Universidade Federal de Santa Catarina
Centro Tecnológico
Departamento de Informática e Estatística
Curso de Bacharelado em Ciências da Computação

SESTAT.NET

ENSINO DE ESTATÍSTICA MEDIADO POR COMPUTADOR

CARLOS ALBERTO NAKAZAWA

MÁRCIO JULIANDREI MARAFON

Florianópolis, Fevereiro de 2003

Universidade Federal de Santa Catarina
Centro Tecnológico
Departamento de Informática e Estatística
Curso de Bacharelado em Ciências da Computação

SESTAT.NET

ENSINO DE ESTATÍSTICA MEDIADO POR COMPUTADOR

Trabalho de conclusão de curso de graduação apresentado a Universidade Federal de Santa Catarina para obtenção do grau de Bacharel em Ciências da Computação.

Autores: **Carlos Alberto Nakazawa**
Márcio Juliandrei Marafon
Orientadora: **Prof. Silvia Modesto Nassar**
Co-orientador: **Prof. Masanao Ohira**
Banca Examinadora: **Prof. Araci Hack Catapan**
Prof. Fernando A. O. Gauthier
Prof. Rogério Cid Bastos

Florianópolis, Fevereiro de 2003

Dedicamos este trabalho a Deus,
a nossos pais e
a todos os nossos amigos
que sempre nos apoiaram.

"A sabedoria da vida
não está em fazer aquilo que se gosta,
mas em gostar daquilo que se faz."
Leonardo da Vinci

RESUMO

O presente projeto de pesquisa tenta buscar uma solução para a implementação e utilização de um software estatístico abordando o conceito de ensino a distância. Esse sistema desenvolvido foi denominado de SEstat.Net por se tratar de uma adaptação do SEstat, um software de ensino de estatística utilizado hoje no Centro Tecnológico da Universidade Federal de Santa Catarina para alunos de graduação. Atualmente o ensino de estatística é realizado utilizando o SEstat [CECHINEL-97] no ensino presencial.

A primeira versão do SEstat.Net no ensino a distância oferece somente páginas Web como interface com o aluno. Essas páginas são dotadas de características especiais que fazem com que se tornem dinâmicas (dependendo do tipo da interação escolhida, a mesma página gera resultados diferentes). A versão aqui desenvolvida do SEstat.Net oferece ao aluno tanto os conceitos estatísticos na forma de páginas *HTML*, quanto os resultados numéricos e gráficos aplicados aos dados de cada aluno conforme o objetivo da analise em questão. Assim o SEstat.Net apóia a aprendizagem de estatística trabalhando simultaneamente com diferentes alunos, que possuem diferentes bases de dados e realizam diferentes análises estatísticas.

O desafio computacional de desenvolvimento desta versão é gerenciar processos, executar diferentes algoritmos, armazenar dados e minimizar a transferência de dados entre cliente(alunos)-servidor.

O SEstat.Net ainda não está totalmente validado, seu uso está sendo aplicado em aulas de estatística de forma experimental, abrangendo somente algumas turmas do Centro Tecnológico. Após concluída a fase de análise e verificada a integridade e eficácia do software, esse será liberado para que alunos da Universidade Federal de Santa Catarina possam acessá-lo fora do domínio da faculdade, se estabelecendo como um software de ensino a distância.

Palavras-Chave

Ensino a distância, Sistema especialista, Software Educacional.

ABSTRACT

The present survey project tries to find a solution to implement and use of a statiscal software dealing with conceit of a long distance teaching. This developed system was named SEstat.Net since this is an adaptation from SEstat, a software for teaching statistics used nowadays at the Centro Tecnológico from the Universidade Federal de Santa Catarina for graduating students. Presently statistics teaching is done using the SEstat at the presential teaching.

The first version of SEstat.Net in long distance teaching offers only web pages as interface with the student. This pages are provided with special characteristics make them become dynamic (depending on the type of chosen interaction, the same page produces different results). The version here developed of SEstat.Net offers the student as the statistical of conceit in the way of HTML pages, as the numerical results and graphs applied for each student data according the aim of the related analysis. So this SEstat.Net supports the learning of statistics working simultaneously with different students, that have different basis of data and produce different statistical analysis.

The computational challenge of development from this version is managing processes, executing different algorithms, storing data and minimizing the transference of data between client (students)-server.

The SEstat.Net is still not totally validated, its use is being applied in statistics classes as an experiment, including only some groups from Centro Tecnológico. After concluding the analysis phase and checked the integrity and effectiveness of the software, this will be released so students from Universidade Federal de Santa Catarina can access it outside of the university, setting this way as a software of long distance teaching.

Key-words

Education in the distance, System specialist, Educational Software.

Sumário

1 Objetivos 2
1.1 Objetivos específicos 2
2 Sistemas Especialistas 3
2.1 Conceituação 3
2.2 Componentes 3
2.2.1 Base de Conhecimento 4
2.2.2 Máquina de Inferência 4
2.2.3 Interface com Usuário 5
2.3 Representação do Conhecimento 5
2.3.1 Representação Através de Regras 6
2.3.2 Representação Através de Frames 7
2.4 Sistemas Especialistas no Ensino 7
2.5 Ciclo de Vida de um Sistema Especialista 8
2.6 Vantagens e desvantagens de sistemas especialistas 9
2.7 Limitações dos sistemas especialistas 10
2.8 Conclusões 10
3 Informática na Educação 12
3.1 Classificação 12
3.2 Avaliação 13
3.3 Aspectos Ergonômicos de um Software Educacional 13
3.4 Conclusão 14
3.5 Educação a Distância 14
3.5.1 Regulamentação em EAD no Brasil 15
4 Estatística 17
4.1 Etapas de uma Pesquisa 17
4.2 Variáveis 17
4.2.1 Variáveis Quantitativas 18
4.2.2 Variáveis Qualitativas 18
4.3 Coleta de Dados 19
4.4 Análise Descritiva 19
4.5 Inferência Estatística 21
4.5.1 Estimação de Parâmetros 22
4.5.2 Testes de Hipóteses 22
4.5.2.1 Testes de Hipóteses Paramétricos 22
4.5.2.1.1.1 Testes T 23
4.5.2.1.1.2 Teste T – Duas Amostras Emparelhadas (Dependentes) 23
4.5.2.1.1.3 Teste T – Duas Amostras Independentes e Grandes (n>30) 24
4.5.2.1.1.4 Teste T – Duas Amostras Independentes, Pequenas (n<=30) e as Variâncias Populacionais são conhecidas 24
4.5.2.1.1.5 Teste T – Duas Amostras Independentes, Pequenas (n<=30) e Existe Igualdade de Variâncias (Com Homocedasticidade) 25
4.5.2.1.1.6 Teste T – Duas Amostras Independentes, Pequenas (n<=30) e Não Existe Igualdade de Variâncias (Sem Homocedasticidade) 25
4.5.2.1.2 Análise da Variância – ANOVA (de Um Critério) 26
4.5.2.2 Testes de Hipóteses Não Paramétricos 28
4.5.2.2.1 Teste de Kruskal-Wallis 28
4.5.2.2.2 Teste U de Mann-Whitney 30

4.5.2.2.3 30
4.5.2.2.4 Teste de Wilcoxon 31
4.5.3 Análise de Correlação e Regressão 32
5 SEstat.Net 33
5.1 Descrição do Sistema 33
5.2 Desenvolvimento 33
5.2.1 Java Server Pages 34
5.2.2 JavaBeans 35
5.2.3 JDBC 36
5.2.4 Java Servlet 36
5.2.5 Classes do SEstat.Net 37
5.2.5.1 Administrador 38
5.2.5.1.1 Funcionamento Básico: 39
5.2.5.2 Estatística 40
5.2.5.3 Base de Dados 40
5.2.5.4 Leitor DBF 40
5.2.5.5 Gráfico 41
5.3 Interface do sistema 42
5.4 Requisitos para o funcionamento do SEstat.Net 49
6 Conclusões 50
6.1 Trabalhos Futuros 50
7 Referências Bibliográficas 51
Anexo I: Diagrama de seqüência: DescricaoUnivariadaMens(boolean) 53
Anexo II: Diagrama de seqüência: DescricaoUnivariadaQuantiCont(string) 54
Anexo III: Fontes do Sistema 55

LISTA DE ILUSTRAÇÕES

Figura 1 - Componentes básicos de um Sistema Especialista 3
Figura 2 - Representação de conhecimento através de regras. 6
Figura 3 - Ciclo de vida de um sistema especialista 9
Figura 4 - Tipos de Variáveis. 18
Figura 5 - Exemplo de gráfico tipo histograma. 20
Figura 6 - Exemplo de gráfico pie-chart 21
Figura 7 - Requisição de uma página JSP. 34
Figura 8 - Núcleo do sistema. 38
Figura 9 - Entrada do sistema. 42
Figura 10 - Escolha da base de dados do aluno. 43
Figura 11 - Seleção do procedimento estatístico. 44
Figura 12 - Escolha da variável de trabalho do aluno. 45
Figura 13 - Seleção do tipo da variável escolhida. 46
Figura 14 - Caso onde o aluno escolheu um tipo que o SEstat. Net julga ser incorreto. 46
Figura 15 - Escolha da mensuração da variável quantitativa. 47
Figura 16 - Resultados estatísticos obtidos na Descrição Univariada da variável TAM (quantitativa discreta) 48

SÍMBOLOS, ABREVIATURAS, SIGLAS E CONVENÇÕES

API	Application Program(ming) Interface
DBF	Database File
EAD	Educação a Distância
HTML	Hypertext Markup Language
J2EE	Java 2 Enterprise Edition
JSP	Java Server Pages
PAP	Programa de Alimentação Popular
PNG	Portable Network Graphics
WWW	World Wide Web
XML	Extensible Markup Language

Introdução

A utilização da informática no ensino é uma das novas ferramentas pedagógicas que vem sendo aplicada com grande sucesso [CATAPAN-01]. Isso pode ser facilmente compreendido se observado o fato do computador interagir com o usuário, prender sua atenção e despertar a sua curiosidade sobre o assunto abordado.

A Estatística é utilizada para transformar dados em informações sobre determinada realidade para resolver um problema ou tomar uma decisão. O Departamento de Informática e de Estatística (INE) da Universidade Federal de Santa Catarina (UFSC), através do seu Laboratório de Estatística Aplicada (LEA), desenvolveu um software, com técnicas de Inteligência Artificial, denominado SEstat (Sistema Especialista de Apoio ao Ensino de Estatística). Esse software está sendo usado desde o primeiro semestre de 1998 para o ensino de estatística nos cursos do Centro Tecnológico (CTC). As atividades didático-pedagógicas são realizadas no Laboratório Integrado de Informática do Centro Tecnológico (LIICT), com a presença dos professores da disciplina.

Com o sucesso alcançado pelo SEstat em mudar a posição do aluno de um mero telespectador passivo e transformá-lo no seu próprio agente de aprendizado, ampliando a abrangência desse software. Para isso foi realizada uma mudança na maneira de acessá-lo, transformando de um software local, onde necessita um grande espaço físico para a administração das aulas, para um software que pode ser utilizado via rede.

O primeiro grande desafio desse projeto foi estudar o funcionamento do programa original, obtendo desse todas as características lógicas de interação com o aluno e adaptando-as para seu funcionamento via web. O segundo desafio foi tentar tornar a aplicação o mais leve possível, diminuindo assim o tempo gasto entre gerar e visualizar os resultados. O terceiro grande problema foi adotar uma arquitetura onde todo processamento do software fosse realizado na máquina servidora, sem sobrecarregá-la, já que haverá vários alunos interagindo com o programa ao mesmo tempo.

1 Objetivos

O principal objetivo desse projeto é construir o SEstat.Net, um sistema especialista para o apoio ao ensino-aprendizagem de estatística na modalidade do *e-learning* para o ensino a distância.

1.1 Objetivos específicos

- Manter no SEstat.Net as mesmas características didático-pedagógicas obtidas no SEstat [DIAS-01].
- Fazer com que o servidor se encarregue de todo o processamento dos dados requerido pelo SEstat.Net
- Desenvolver uma modelagem que permitisse um alto desempenho do software, deixando seu tempo computacional relativamente baixo a fim de não ocorrerem problemas com o servidor devido ao grande número de alunos interagindo com o sistema ao mesmo tempo.
- Tornar a interação cliente/servidor a mais rápida possível, realizando para isso uma troca mínima de dados, não sobrecarregando o número de informações passadas do cliente para o servidor ou do servidor para o cliente.

2 Sistemas Especialistas

2.1 Conceituação

Um especialista é uma pessoa que tem um vasto conhecimento relacionado a um domínio específico dentro de uma certa área. Devido a esse conhecimento, o especialista tem maior aptidão para solucionar problemas relacionados a essa área.

Sistemas especialistas, uma aplicação da inteligência artificial, são programas de computador planejados para adquirir e disponibilizar o conhecimento operacional de um especialista humano [CHAIBEN-99]. São tradicionalmente vistos como sistemas de suporte à decisão, pois são capazes de tomar decisões como especialistas na área de domínio. Sua estrutura reflete a maneira como o especialista humano arranja e faz inferência sobre o seu conhecimento.

2.2 Componentes

Um modelo básico da arquitetura dos sistemas especialistas pode ser apresentado como na figura 1, com três componentes básicos: a base de conhecimento, a máquina de inferência, e a interface com usuário [CHAIBEN-99].

Essa estrutura muda ligeiramente de acordo com o autor. Alguns acrescentam outros componentes (Mecanismo de Explicação, etc.), mas em qualquer abordagem esses três componentes básicos estão presentes.

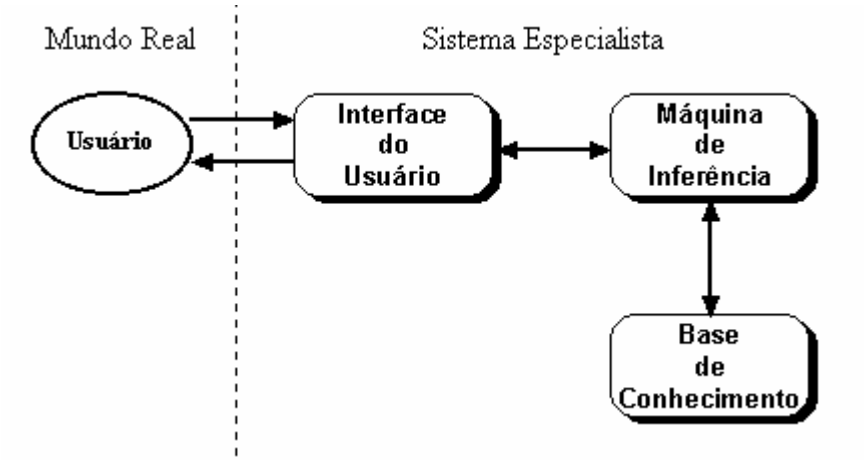


Figura 1 - Componentes básicos de um Sistema Especialista

2.2.1 Base de Conhecimento

O termo "base de conhecimento" é utilizado para designar a coleção de conhecimento do domínio, ou seja, as informações, ao nível de especialista, necessárias para resolver problemas de um domínio específico. Ela é, talvez, o mais importante componente do sistema especialista.

Esse conhecimento precisa ser organizado de uma maneira adequada para que a máquina de inferência consiga tratá-lo convenientemente. O conhecimento de um sistema especialista consiste de fatos, informações que estarão sempre disponíveis para serem compartilhadas e atualizadas, e heurísticas, regras práticas que caracterizam ao nível de tomada de decisão.

A base de conhecimento é, portanto, um agrupamento de conhecimento representado mediante uma técnica adequada ao sistema em questão. Os principais mecanismos de representação do conhecimento, tratados mais à frente, são: a lógica de primeira ordem, os frames, redes semânticas e regras de produção [RAMOS-95]. Este último é o mais utilizado em sistemas especialistas, o que dá origem ao termo, baseado em regras.

A base de conhecimento pode ser classificada em:

- Estática: permanece-se constante à medida que o sistema é utilizado;
- Dinâmica: onde novos conhecimentos são acrescentados do sistema. Esse acréscimo de informações caracteriza o aprendizado do Sistema Especialista.

2.2.2 Máquina de Inferência

Segundo Minsky, "... o conhecimento é útil somente quando podemos explorá-lo para ajudar a alcançarmos nossos objetivos" [MINSKY-86]. Nos sistemas especialistas, a máquina de inferência cumpre este papel, representando o meio pelo qual o conhecimento é manipulado, utilizando-se das informações armazenadas na base de conhecimento, para resolver problemas. Para isto, deve haver uma linguagem ou um formato específico no qual o conhecimento possa ser expresso para permitir o "raciocínio" e a inferência. Métodos de inferência são necessários para fazer uso apropriado e eficiente dos itens em uma base de conhecimento para alcançar alguns propósitos tal como o diagnóstico de doenças.

A máquina de inferência, de certo modo, tenta imitar os tipos de pensamento que o especialista humano emprega quando resolve um problema, ou seja, ele pode começar com

uma conclusão e procurar uma evidência que a comprove, ou pode iniciar com uma evidência para chegar a uma conclusão. Em sistemas especialistas, estes dois métodos são chamados de *"backward chaining"* e *"forward chaining"* respectivamente. Nem todos os sistemas utilizam a mesma abordagem para a representação do seu conhecimento, portanto, a máquina de inferência deve ser projetada para trabalhar com a representação de conhecimento específica utilizada.

2.2.3 Interface com Usuário

É a parte do Sistema Especialista que faz a interação entre o usuário e o sistema. Para que as informações obtidas do usuário sejam realmente consistentes, as questões que a interface apresentará a este devem ser bem formuladas, de modo a não dar margem a interpretações ambíguas (ou errôneas).

Essa "conversa" entre o usuário e o sistema deve fluir de uma maneira natural. Para isso, é necessário que se utilize uma linguagem simples, que possibilite informações a respeito do que está sendo perguntado, e que explique o porquê da necessidade de se obter estas informações. O usuário deve ter a possibilidade de mudar de opinião a respeito de uma resposta anteriormente fornecida ao sistema.

A interface, quando questionada pelo usuário, terá que fornecer informações de como o sistema chegou a certas conclusões. Qualquer informação inconsistente fornecida pelo usuário poderá acarretar em conclusões erradas para o caso em questão.

2.3 Representação do Conhecimento

É o método usado para codificar o conhecimento na base de conhecimentos do Sistema Especialista. A seguir serão descritas duas formas utilizadas para a representação do conhecimento.

2.3.1 Representação Através de Regras

Representa um conhecimento procedural, ou seja, descreve como se deve proceder para resolver o problema. É formado por estruturas do tipo CONDIÇÃO ANTECEDENTE -> CONSEQUÊNCIA. O Sistema Especialista procura em sua base de conhecimentos uma regra que tenha CONDIÇÃO ANTECEDENTE igual a que foi obtida na memória de trabalho, quando encontrada assume-se que a CONSEQUÊNCIA daquela condição é verdadeira, e a mesma é adicionada na memória de trabalho.

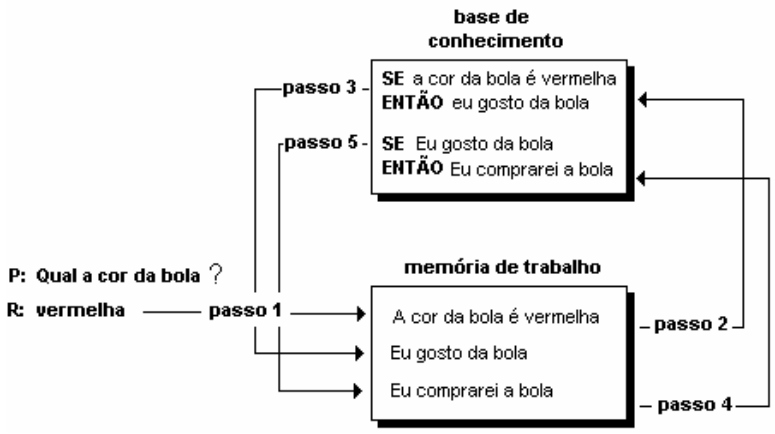


Figura 2 - Representação de conhecimento através de regras.

Uma regra pode possuir uma ou mais CONDIÇÕES ANTECEDENTES.

Ex: **SE** $A > B$ **E** $B \neq 0$

ENTÃO $A \leftarrow B$

Da mesma forma pode encadear mais de uma CONSEQUÊNCIA.

Ex: **SE** $A > B$

ENTÃO $A \leftarrow B$ **E** $B \leftarrow A + B$

Pode haver regras que disparem uma CONSEQUÊNCIA caso seja verificada falsa a sua CONDIÇÃO ANTECEDENTE; sendo que para isso seja utilizada a cláusula SENÃO.

Ex: **SE** $A > B$

ENTÃO $A \leftarrow B$

SENÃO B ← A (conseqüência que será disparada para o caso em que A é menor ou igual a B)

2.3.2 Representação Através de Frames

Pode representar tanto um conhecimento procedural quanto declarativo. O conhecimento declarativo descreve o que é conhecido sobre o problema.

A representação do conhecimento por *frames* é similar ao paradigma de orientação a objetos, onde são definidos objetos da vida real com determinadas características e ações:

Por exemplo:

Objeto	**Características**	**Ações**
Pessoa	altura, idade, sexo, peso	falar, andar
Carro	cor, ano, modelo, fábrica	ligar motor, desligar motor
Impressora	marca, modelo	Imprimir

Nessa representação, os objetos passam a ser conhecidos por *frames* e as características por *slots***.** O paradigma de orientação a objeto proporciona que nessa representação os *frames* possuam uma estrutura hierárquica e herdem características e ações de outros *frames***.** Ao longo da interação com o usuário, o Sistema Especialista irá preenchendo os *slots* respectivos ao *frame* em questão. À medida que esses *slots* são preenchidos o Sistema Especialista, através de ações associadas a ele, definirá melhor o problema e restringirá a solução a um número menor de regras.

2.4 Sistemas Especialistas no Ensino

Do ponto de vista educacional, a maioria dos sistemas especialistas tem pouca utilidade direta, porque não foram projetados para ensinar. Isto pode ser explicado principalmente com relação à [TROLLIP-91]:

- ausência de qualquer estratégia educacional.
- incapacidade de comparar o que o estudante conhece com o conhecimento do especialista.

- incapacidade de determinar o que fazer quando o conhecimento do estudante difere do conhecimento do especialista.

Entretanto, a estrutura do sistema especialista serve perfeitamente para ser adaptada para a construção de **sistemas tutoriais**, proporcionando um grande potencial para a criação de novos ambientes educacionais.

Um sistema tutorial não necessita somente do conhecimento de seu domínio, mas também da perspectiva sobre este conhecimento que permita transmiti-lo ao estudante adequadamente [RICKEL-89].

Uma característica importante nos sistemas especialistas é a separação do conhecimento dos métodos gerais que são usados para manipular este conhecimento. Esta característica é relevante se a aplicação é voltada ao ensino, uma vez que cada domínio (área de aplicação) tem sua própria terminologia, relações e procedimentos. Se os aspectos relacionados ao domínio podem ser formulados independentemente, então o desenvolvimento completo de um sistema de ensino pode ser bastante simplificado [KEMP-92].

2.5 Ciclo de Vida de um Sistema Especialista

O primeiro passo ao se desenvolver um sistema especialista, é avaliar a necessidade e a viabilidade da implementação do mesmo para o problema proposto. Sabendo-se que a implementação é viável e conveniente, dá-se início ao processo de aquisição do conhecimento, que é seguido pelo desenvolvimento propriamente dito.

O desenvolvimento engloba a escolha e a implementação de como o conhecimento será estruturado, qual o algoritmo da máquina de inferência e de como será a interface com o usuário.

Após a implementação o sistema é testado, e de acordo com o resultado dos testes, ele é ou não validado. No caso de não ser validado, volta-se as etapas anteriores e faz-se as mudanças necessárias.

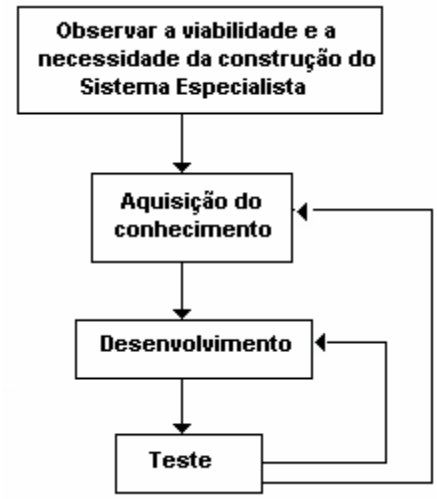


Figura 3 - Ciclo de vida de um sistema especialista

2.6 Vantagens e desvantagens de sistemas especialistas

Dentre as vantagens dos sistemas especialistas podemos citar:

- armazenamento permanente do conhecimento especializado;
- facilidade de transferência e reprodução do conhecimento especializado;
- facilidade de documentação;
- resultados mais consistentes e reprodutíveis sem influência de fatores emocionais, stress ou pressões;
- baixo custo operacional.

Relacionando com outras técnicas computacionais, os sistemas especialistas têm ainda as vantagens de serem bem aceitos por usuários finais que vêem computadores solucionar problemas do mundo real, trabalhar com conhecimento adquirido com a experiência do especialista, efetuar processamento de forma semelhante à humana, facilitar a transformação de conhecimento em código e tratar incertezas [BASDEN-84].

Como principais desvantagens tem-se:

- os sistemas especialistas tendem a trabalhar de uma forma rotineira, sem inspiração ou criatividade onde não é sintetizado novo conhecimento;
- dificuldade de lidar com situações inesperadas - entrada de informação e representação simbólica e não sensória;

- não possuir conhecimento do senso comum [WATERMAN-96].

2.7 Limitações dos sistemas especialistas

A primeira limitação que podemos observar nos sistemas especialistas tem a ver com o processo de aquisição do conhecimento. É relativamente difícil a implementação de um sistema que aprenda com a experiência. Há também problemas com relação a explosões combinatórias em espaços de buscas, dificuldades de manutenção de sistemas com uma grande quantidade de conhecimento e receber e integrar conhecimento de vários especialistas [LIEBOWITZ-88].

Há também uma gama de situações para as quais seria insensato aplicar os sistemas especialistas [BASDEN-84]:

- problemas muito simples que o homem pode manusear adequadamente;
- problemas muito complexos (mais de 10.000 regras) pela busca excessivamente longa e dificuldade do hardware de manuseio de tais dados;
- problemas que não requerem nenhuma das vantagens dos sistemas especialistas tais como problemas numéricos bem estruturados;
- problemas que dependem de informações mais adequadas ao processamento do cérebro humano do que ao computador.

Sistemas especialistas não são bons para: representar conhecimento temporal, representar conhecimento espacial, realizar raciocínio do senso comum, reconhecer os limites de sua habilidade, ou manusear conhecimento inconsistente [WATERMAN-86].

2.8 Conclusões

Nos tempos atuais, as organizações têm no conhecimento acumulado de seu quadro de pessoal, um patrimônio muito importante para seus empreendimentos futuros e para sua própria sobrevivência face às constantes concorrências de mercado. Processos como "downsizing", reengenharia e aposentadorias prematuras têm sido utilizados constantemente pelas grandes instituições, comprometendo a memória institucional da empresa [LIEBOWITZ-95]. Funcionários experientes aposentam-se ou deixam as organizações,

resultando em uma perda incalculável. O conhecimento de vários anos de experiência e treinamentos destes indivíduos será perdido, a menos que haja uma maneira de captar e preservar estas experiências dentro da empresa. A tecnologia dos sistemas especialistas é ideal para atingir estes propósitos. Os processos e experiências funcionais da organização podem ser documentados através do uso de sistemas especialistas. A tecnologia do conhecimento poderá ser utilizada para capitalizar o conhecimento como um produto estratégico para as corporações.

As pesquisas em sistemas especialistas têm ainda um longo caminho a percorrer, particularmente avançando o estado da arte em aprendizagem de máquina, programação automatizada, e padronização. Mesmo assim, várias organizações já estão fazendo grandes economias e estão obtendo muitos benefícios na utilização desta tecnologia.

3 Informática na Educação

O século XX é o século marcado pelos grandes avanços tecnológicos. Se prestarmos atenção nas grandes invenções deste século (televisão**,** vídeo-cassete**,** computador, internet), notaremos que elas giram em torno da informação. Devido ao uso dessas tecnologias, a busca pela informação, que durante toda a existência da humanidade sempre foi um grande obstáculo, tornou-se algo simples e rápido.

A busca por informações deixou de ser um grande problema, o próximo passo agora é que essas informações sejam organizadas e distribuídas de forma adequada.

As imagens, os sons e os movimentos já demonstraram que são as formas de representação que têm mais chance de sobreviver à memória humana. Sabendo disso, novas técnicas pedagógicas vêm sendo aplicadas, utilizando-se das mesmas tecnologias que antes apenas disponibilizavam informações. Essas novas técnicas conseguiram atingir grupos que antes demonstravam dificuldades no aprendizado pelos métodos tradicionais. [RAMOS-91]

A utilização da informática no ensino é uma das novas técnicas pedagógicas que vem sendo aplicada com grande sucesso. Isso pode ser facilmente compreendido se observarmos o fato do computador interagir com o usuário, prender sua atenção e despertar a sua curiosidade sobre o assunto abordado.

3.1 Classificação

Thomas Dwyer, levando em consideração a atividade do aprendiz, propôs uma classificação na qual os *softwares* educacionais estão divididos em dois grandes grupos: software com enfoque do tipo algorítmico e *software* com enfoque do tipo heurístico. [RAMOS-91].

No enfoque do tipo algorítmico predomina a transmissão do conhecimento da pessoa que sabe para a pessoa que deseja aprender. A maneira como a pessoa que sabe irá transmitir o conhecimento ao aprendiz, o ritmo e a direção tomada nesse ensino são previamente determinados, visando o objetivo de que seja utilizada uma seqüência ideal de ensino. Este enfoque é criticado principalmente no sentido de ser muito rígido e inflexível (sendo que o aluno está limitado às vontades do professor), pois mesmo que o professor produza uma

seqüência ideal de ensino, e que o aprendiz tenha entendido os conceitos, ainda assim o controle da atividade do aprendiz continuará nas mãos do professor.

O outro enfoque citado, é o enfoque do tipo heurístico, onde a característica que predomina é o aprendizado por descoberta. Cria-se para o aprendiz um ambiente rico em informações e suficientemente capaz de possibilitar que este "navegue" pelo ambiente de maneira fácil. Neste enfoque não se estipula ao aprendiz um caminho a ser seguido no seu processo de aprendizado, ele possui livre arbítrio para escolher e buscar as informações que julgar necessárias ou que possui interesse em aprender. Uma das grandes vantagens desse enfoque, é que ele permite ao aprendiz ter uma visão global dos conceitos que estão sendo apresentados.

Uma outra maneira de classificar os *softwares* educacionais, é de acordo com as concepções educativas. Neste caso são considerados dois paradigmas: paradigma comportamentalista (enfoque algorítmico) e paradigma construtivista (enfoque heurístico).

3.2 Avaliação

A dificuldade na avaliação de um *software* educacional está diretamente ligada a concepção educativa do mesmo, ou seja, o paradigma sob o qual ele foi desenvolvido.

Segundo o paradigma comportamentalista (*softwares* do tipo tutoriais e exercícios e prática) as questões referentes à sua avaliação dizem respeito principalmente a: motivação para o aprendizado e apresentação dos conteúdos, aplicação dos conhecimentos e retro-alimentação, facilidade na operação do *software*, etc.

No paradigma construtivista a avaliação é mais difícil, pois, sob este paradigma os objetivos a serem alcançados no processo educacional, não se expressam através de comportamentos que devam ser obrigatoriamente mensuráveis [RAMOS-91]. Neste paradigma se encaixam *softwares* de simulação e os jogos educativos.

3.3 Aspectos Ergonômicos de um Software Educacional

A interface de um *software* educacional é indiscutivelmente um dos pontos chave no aprendizado almejado ao aluno. Para que a interface esteja dentro das expectativas do usuário,

é recomendável que sejam obedecidos critérios de ergonomia na construção dessa interface. Alguns desses critérios são mostrados abaixo:

- Possibilitar ao usuário poder sair, anular ou interromper uma transação a qualquer momento da interação;
- Empregar sempre que for possível a linguagem que é utilizada pelo usuário;
- Utilizar códigos mnemônicos, testando sua eficácia, e modificando-os no caso de ineficácia;
- Não deixar o usuário "perdido" dentro do sistema. É interessante que o usuário saiba onde pode encontrar algo que esteja procurando. Não se deve, por exemplo, mudar a posição do botão de "OK" a cada passo que é dado no sistema;
- Disponibilizar ao usuário algum tipo de guia, como, por exemplo, *help* que responda "o que deve ser feito em tal situação";
- Não dar margem a interpretações ambíguas, elas podem ocasionar erros no aprendizado, erros esses que são de detecção mais difícil, e, portanto, mais dispendiosa.

3.4 Conclusão

Lembramos que o objetivo de alguns *softwares* educacionais não é substituir o trabalho do professor, e sim facilitar esse trabalho servindo como ferramenta de apoio ao ensino. O uso da Informática na Educação é uma atividade em expansão, e que está atraindo a atenção de muitos pesquisadores no sentido de avaliar onde e de que maneira deve ser aplicada.

3.5 Educação a Distância

Educação a distância é o processo de ensino-aprendizagem, mediado por tecnologias, onde professores e alunos estão separados espacial e/ou temporalmente.

É ensino-aprendizagem onde professores e alunos não estão normalmente juntos, fisicamente, mas podem estar conectados, interligados por tecnologias, principalmente as telemáticas, como a Internet. Mas também podem ser utilizados o correio, o rádio, a televisão, o vídeo, o CD-ROM, o telefone, o fax e tecnologias similares.

Hoje existem três modalidades de educação:

- Educação presencial: é a dos cursos regulares, em qualquer nível, onde professores e alunos se encontram sempre num local físico, chamado sala de aula. É o ensino convencional.
- Semi-presencial (parte presencial/parte virtual ou a distância): acontece em parte na sala de aula e outra parte a distância, através do uso de tecnologias.
- Educação a distância (ou virtual). A educação a distância pode ter ou não momentos presenciais, mas acontece fundamentalmente com professores e alunos separados fisicamente no espaço e/ou no tempo, mas podendo estar atual através de tecnologias de comunicação.

As tecnologias interativas, sobretudo, vêm evidenciando, na educação a distância, o que deveria ser o cerne de qualquer processo de educação: a interação e a interlocução entre todos os que estão envolvidos nesse processo.

As possibilidades educacionais que se abrem com a educação a distância são fantásticas. Com o alargamento da banda de transmissão, como acontece na TV a cabo e internet de banda larga, torna-se mais fácil ver e ouvir a distância. Muitos cursos poderão ser realizados a distância com áudio e vídeo, principalmente na educação continuada. As possibilidades de interação serão diretamente proporcionais ao número de pessoas envolvidas. Teremos aulas a distância com possibilidade de interação *on line* (ao vivo) e aulas presenciais com interação a distância.

3.5.1 Regulamentação em EAD no Brasil

As bases legais da educação a distância no Brasil foram estabelecidas pelas seguintes leis e portarias:

- LEI Nº 9.394, DE 20 DE DEZEMBRO DE 1996
  - Lei de Diretrizes e Bases da Educação Nacional
- DECRETO Nº 2.494, DE 10 DE FEVEREIRO DE 1998
  - Decreto que regulamenta o Art. 80 do LDB nº 939/96 e normatiza a Educação a Distância no Brasil.
- PORTARIA Nº 2.253, DE 18 DE OUTUBRO DE 2001

- Portaria que normatiza a oferta de disciplinas na modalidade de educação a distância em instituições de ensino superior, com base no art. 81 da Lei nº 9.394, de 1.996.

4 Estatística

A Estatística é uma ferramenta muito útil em pesquisas, pois oferece recursos desde o planejamento da coleta de dados até a obtenção de resultados estatísticos que servem de informações sobre o comportamento das variáveis observadas no domínio pesquisado. Os resultados estatísticos podem ser obtidos com o apoio de *softwares* estatísticos a partir de uma base de dados e da aplicação adequada de procedimentos estatísticos de análise de dados. A seleção de procedimentos estatísticos depende dos objetivos da pesquisa, das características dos dados e do atendimento de suposições do próprio procedimento estatístico.

4.1 Etapas de uma Pesquisa

A elaboração de uma pesquisa pode ser resumida pelas seguintes etapas:

- Especificação do objetivo de forma clara
- Definir a população que será estudada
- Avaliar quais as características da população que serão analisadas
- Estudar e definir a forma de coleta dos dados (censo ou amostragem) mais conveniente
- Coletar os dados
- Aplicar os métodos estatísticos adequados para o estudo em questão

4.2 Variáveis

O estudo de uma população é feito através da avaliação das variáveis e da relação entre as variáveis dessa população. Variáveis são características que são mensuradas, observadas ou obtidas da combinação de outras em uma pesquisa. Diferem em muitos aspectos, principalmente no papel que a elas é dado em uma pesquisa e na forma como podem ser medidas [STAT-94].

Segundo as formas de mensuração das variáveis, elas podem ser classificadas entre quantitativas e qualitativas.

4.2.1 Variáveis Quantitativas

Uma variável é considerada quantitativa quando sua medida tem sentido de quantidade. A estatura de uma pessoa, por exemplo, é uma variável quantitativa, quando avaliada como numericamente e é qualitativa se avaliada como alto, baixo, etc.

As variáveis quantitativas se dividem em **contínuas** e **discretas**. As contínuas são aquelas que podem assumir teoricamente qualquer valor dentro de um intervalo contínuo de valores. Como exemplo, podemos citar o peso de uma pessoa. Já as discretas podem assumir apenas alguns valores do dominio, e em geral esses valores são números inteiros. O número de pessoas numa família é um exemplo de uma variável quantitativa discreta.

4.2.2 Variáveis Qualitativas

Nesta forma de mensuração/observação as variáveis não possuem um valor de medida com sentido de quantidade, sendo assim atribuídas categorias com atributos. Um exemplo é o caso da variável sexo, que possui duas categorias (masculino ou feminino).

As variáveis qualitativas se dividem em **nominais** e **ordinais**. São consideradas nominais as variáveis qualitativas nas quais suas categorias não possuem uma ordem entre si, simplesmente um rõtulo ou um nome. A variável sexo é um exemplo.

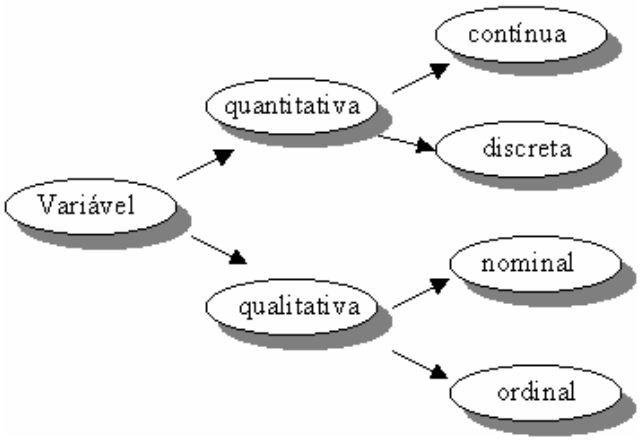


Figura 4 - Tipos de Variáveis.

Classificam-se como variáveis qualitativas ordinais aquelas que possuem uma ordem entre as suas categorias. Por exemplo, uma variável que represente o grau de escolaridade de uma pessoa. Supondo que as categorias fossem 1o grau, 2o grau e Superior, existe uma certa ordem (hierarquia) entre elas.

4.3 Coleta de Dados

Há dois tipos básicos de coleta de dados: **censo** e **amostragem**.

Em um censo são coletados dados de toda população. O censo é indicado quando a população em estudo é pequena e há a necessidade de informação com alta precisão.

Já numa amostragem coleta-se um subconjunto finito da população (amostra), que seja representativa. Sua utilização é interessante quando se necessita de um processamento rápido da informação, no caso de não se possuir recursos para fazer um censo ou este seja inviável. Quando se opta por amostragem, é importante cuidar para que a amostra represente toda a população e não um grupo específico dentro dela. As técnicas de amostragem são:

Probabilísticas:

a) Amostragem aleatória simples
b) Amostragem sistemática
c) Amostragem estratificada
d) Amostragem por conglomerados

Não-Probabilísticas:

a) Amostragem a esmo
b) Amostragem por julgamento

4.4 Análise Descritiva

Após a coleta de dados é feita a análise descritiva. A estatística oferece técnicas para organização dos dados coletados resumindo-os através de gráficos, tabelas e/ou medidas de síntese para uma primeira interpretação. Essas técnicas utilizadas facilitam a visualização desses dados, pois mostram uma visão global (em conjunto) e não uma visão de cada caso de

forma particular. Com essa visão global, formulamos as hipóteses para as variáveis de estudo. Essas hipóteses serão validadas ou não numa fase seguinte chamada inferência estatística.

Para o caso de variáveis quantitativas, utilizam-se as medidas descritivas, tais como: média, mediana, moda, desvio padrão, valores de mínimo e máximo e também gráficos do tipo histogramas. A figura 5 mostra um histograma que descreve uma variável particular TAM. Essa variável representa o número de pessoas que compõem cada família entrevistada. O eixo horizontal mostra o número de pessoas em cada família, sendo cada número representando uma classe, enquanto o eixo vertical indica o número de ocorrências para cada classe. A descrição é dita univariada porque descreve somente uma variável.

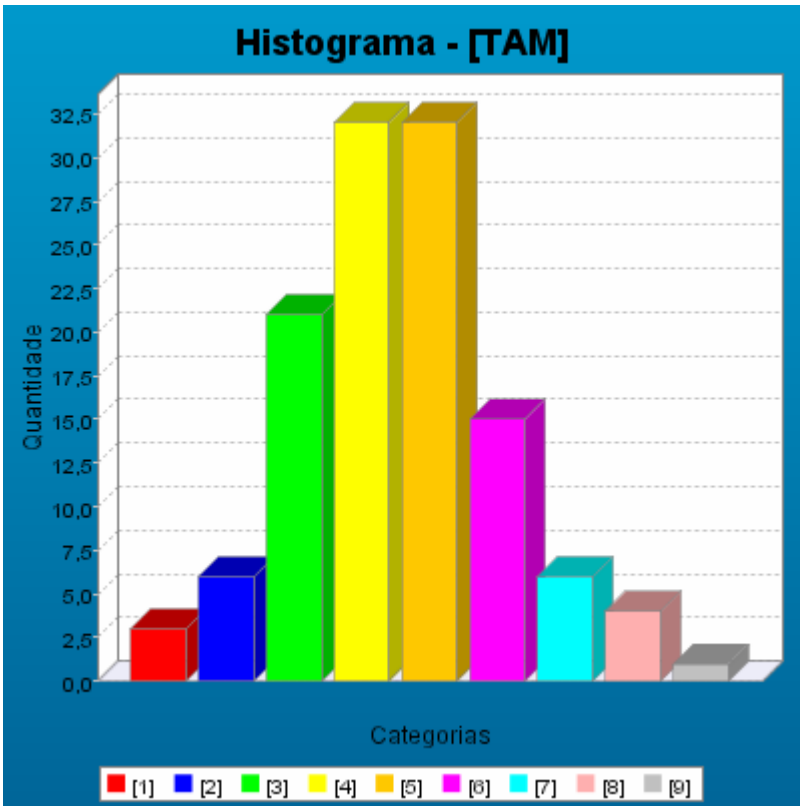


Figura 5 - Exemplo de gráfico tipo histograma.

Nos casos de variáveis qualitativas são utilizadas as tabelas de freqüência e gráficos *pie chart* (também conhecidos como gráfico em setores ou pizza). A figura 6 ilustra um gráfico desse tipo, descrevendo uma variável que indica se a família participa ou não do programa de alimentação popular (PAP) [BARBETTA-94].

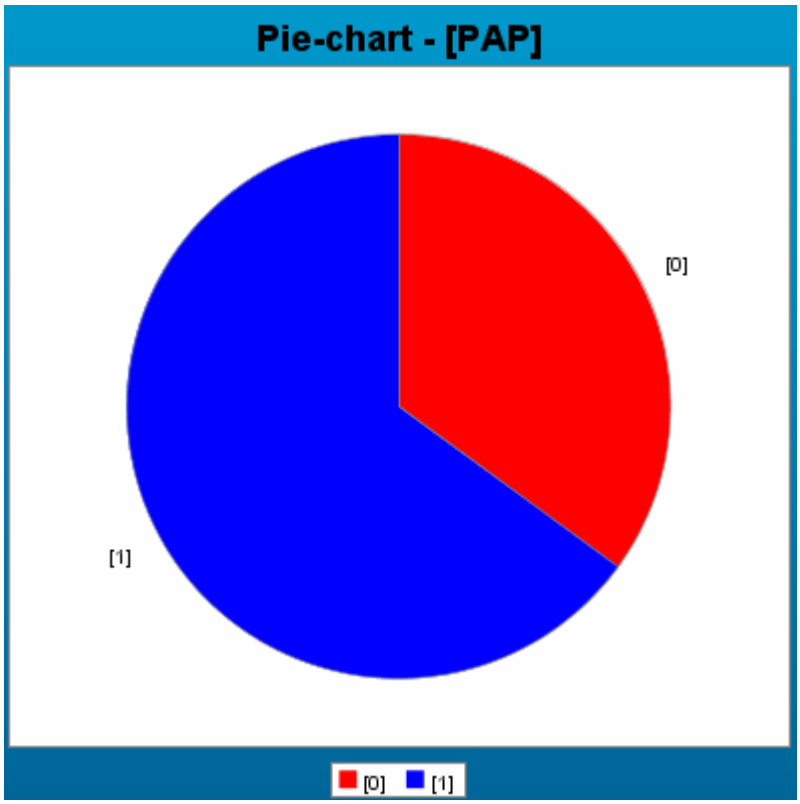


Figura 6 - Exemplo de gráfico pie-chart

4.5 Inferência Estatística

Quando é feito um censo, a descrição por si só consegue resumir todas as características da população. No caso de uma amostra, a descrição nos dá idéias de algumas características que a população pode ou não ter, idéias que serão confirmadas ou não na etapa de inferência estatística. Em outras palavras, esta etapa da inferência serve para formalizar se os dados obtidos fornecem evidências suficientes para que se possa generalizá-los para a população em estudo. Basicamente são estudados Estimação de Parâmetros, Testes de Hipóteses e Análise de Correlação e Regressão.

4.5.1 Estimação de Parâmetros

Tem por objetivo caracterizar os parâmetros de uma distribuição populacional a partir dos dados de uma amostra, com uma probabilidade de erro.

Há dois tipos básicos:

- Estimação por ponto: fornece a melhor estimativa para um parâmetro populacional através de um valor único, porém o erro é grande.
- Estimação por intervalo: Na estimação por intervalo é obtido por intervalo onde é arbitrado o nível de confiança desejado, que contenha o parâmetro. O intervalo de confiança conterá o parâmetro com uma probabilidade $(1-\alpha)$ chamado nível de confiança.

4.5.2 Testes de Hipóteses

Consistem em testes cujo objetivo é decidirem aceitar ou rejeitar hipóteses formuladas, com uma probabilidade de erro.

Os testes de hipóteses exigem a elaboração de duas hipóteses estatísticas que são complementares: a **hipótese nula (H0)** e a **hipótese alternativa (H1)**.

A hipótese nula, geralmente representada por H0, é uma afirmação sobre uma característica da população aceita como verdadeira até que se prove estatisticamente o contrário.

A hipótese alternativa (H1) é o complemento da hipótese nula H0. Supondo que a hipótese nula seja “As médias são iguais”, a hipótese alternativa poderá ser “As médias são diferentes”.

Os testes de hipóteses se dividem em dois grandes grupos: os Paramétricos e os Não-Paramétricos.

4.5.2.1 Testes de Hipóteses Paramétricos

Aplicáveis a variáveis quantitativas que seguem algum modelo probabilístico de distribuição (geralmente distribuição normal). Os Testes de Hipóteses Paramétricos se referem a testes sobre os parâmetros da população (média, proporção) do qual é retirada a amostra.

4.5.2.1.1.1 Testes T

Utilizados para se fazer afirmações sobre médias de duas populações. Dependendo de as amostras serem ou não dependentes, dos tamanhos das amostras, e de os dados possuírem ou não variâncias iguais (homocedasticidade), um dos três tipos de teste-t a seguir deve ser usado (todos os três testes assumem que os dados possuem distribuição normal, isto é, existe Normalidade) [TRIOLA-99].

Normalidade e Homocedasticidade são suposições verificadas através de testes de hipóteses específicos.

4.5.2.1.1.2 Teste T – Duas Amostras Emparelhadas (Dependentes)

Utilizado quando as duas amostras são dependentes, isto é, emparelhadas.

Para cada par de valores, calcula-se a diferença *d* entre eles. Com as *n* diferenças em mão, calcula-se a média $d_{médio}$ e o desvio-padrão s_d dessas diferenças. Sendo μ_d a média das diferenças para a *população* de dados em questão, isto é, o valor que supomos ser verdadeiro, calcula-se a estatística t conforme a Fórmula 4.1:

$$t = \frac{d_{médio} - \mu_d}{\frac{s_d}{\sqrt{n}}}$$

Fórmula 4.1 - Teste-T para duas amostras dependentes

Vamos supor, por exemplo, que temos um par de valores de Renda para cada indivíduo. Podemos utilizar esse teste para fazer hipóteses sobre as médias dos dois pares. Se quisermos testar se as duas médias são iguais, faríamos $\mu_d=0$. Se acharmos que as rendas cresceram 5 unidades, faríamos $\mu_d=5$. Se, ao contrário, a hipótese fosse de que elas diminuíram 5 unidades, faríamos $\mu_d=-5$. A estatística t tem distribuição t de student, com número de graus de liberdade n-1, onde n é o número de pares de dados. Se n é grande (n>30), utiliza-se a distribuição normal ao invés da student.

O p-valor é sempre calculado usando-se a distribuição t-student, mesmo se o número de pares for maior que 30.

4.5.2.1.1.3 Teste T – Duas Amostras Independentes e Grandes (n>30)

Utilizado quando as duas amostras são independentes, e ambos os tamanhos das duas amostras são maiores que 30 (n1>30 e n2>30).

A estatística calculada tem distribuição normal, razão pela qual é mais comum utilizarmos o termo Teste z ao invés de Teste t, neste caso [TRIOLA-99].

$$z = \frac{(m_1 - m_2) - (\mu_1 - \mu_2)}{\sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}}$$

Fórmula 4.2 – Teste z para duas amostras independentes e grandes

Na Fórmula 4.2, m_1 e m_2 referem-se às médias das duas amostras, e n1 e n2, o tamanho delas. Os valores μ_1 e μ_2 são os valores testados (isto é, os supostos valores das médias populacionais). Os valores σ_1^2 e σ_2^2 representam as variâncias das duas populações. Como é muito raro conhecermos tais valores se não conhecemos nem as médias, utilizamos os valores s_1^2 e s_2^2, as variâncias das duas amostras.

4.5.2.1.1.4 Teste T – Duas Amostras Independentes, Pequenas (n<=30) e as Variâncias Populacionais são conhecidas

Utilizado quando as duas amostras são independentes, pelo menos uma das duas amostras é pequena (n<=30) e as variâncias populacionais são conhecidas (o que torna rara a aplicação desse teste) [TRIOLA-99].

A estatística desse teste é exatamente igual a do teste anterior (Fórmula 4.2).

4.5.2.1.1.5 Teste T – Duas Amostras Independentes, Pequenas (n<=30) e Existe Igualdade de Variâncias (Com Homocedasticidade)

Utilizado quando as duas amostras são independentes, pelo menos uma das duas amostras é pequena (n<=30) e existe igualdade de variâncias (homocedasticidade).

Nesse caso, calculamos uma estimativa combinada de σ^2, comum a ambas as populações. Essa estimativa, s_p^2 é, na verdade, uma média ponderada de s_1^2 e s_2^2.

A estatística é calculada conforme a Fórmula 4.3, tem distribuição t-student, com graus de liberdade $n_1 + n_2 - 2$.

$$z = \frac{(m_1 - m_2) - (\mu_1 - \mu_2)}{\sqrt{\frac{s_p^2}{n_1} + \frac{s_p^2}{n_2}}}$$

$$onde \quad s_p^2 = \frac{(n_1 - 1)s_1^2 + (n_2 - 1)s_2^2}{(n_1 - 1) + (n_2 - 1)}$$

$$e\ o\ grau\ de\ liberdade\ gl = n_1 + n_2 - 2.$$

Fórmula 4.3 - Teste T para amostras independentes, pequenas e com variâncias iguais.

Na Fórmula 4.3, m_1 e m_2 referem-se às médias das duas amostras, n1 e n2, o tamanho delas, e os valores s_1^2 e s_2^2, as variâncias das duas amostras. Os valores μ_1 e μ_2 são os valores testados (isto é, os supostos valores das médias populacionais).

4.5.2.1.1.6 Teste T – Duas Amostras Independentes, Pequenas (n<=30) e Não Existe Igualdade de Variâncias (Sem Homocedasticidade)

Utilizado quando as duas amostras são independentes, pelo menos uma das duas amostras é pequena (n<=30) e não existe igualdade de variâncias (homocedasticidade). É um método aproximado, visto que não há um método exato nesses casos [TRIOLA-99].

$$t = \frac{(m_1 - m_2) - (\mu_1 - \mu_2)}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}}$$

onde o grau de liberdade é o menor entre $n_1 - 1$ *e* $n_2 - 1$.

Fórmula 4.4 - Teste T para amostras independentes, pequenas e sem homocedasticidade.

Na Fórmula 4.4, m_1 e m_2 referem-se às médias das duas amostras, n1 e n2, o tamanho delas, e os valores s_1^2 e s_2^2, as variâncias das duas amostras. Os valores μ_1 e μ_2 são os valores testados (isto é, os supostos valores das médias populacionais).

Uma alternativa mais conservativa e simplificada para o cálculo do número de graus de liberdade é dada pela Fórmula 4.5. Obtém-se resultados mais exatos, mas eles continuam a ser apenas aproximados [TRIOLA-99].

$$gl = \frac{(A+B)^2}{\frac{A^2}{n_1 - 1} + \frac{B^2}{n_2 - 1}}$$

$$onde \quad A = \frac{s_1^2}{n_1} \quad e \quad B = \frac{s_2^2}{n_2}$$

Fórmula 4.5 – Outra alternativa para o cálculo do número de graus de liberdade.

4.5.2.1.2 Análise da Variância – ANOVA (de Um Critério)

A análise da variância (ANOVA) é um método parar testar a igualdade de três ou mais médias populacionais, baseado na análise de variâncias amostrais. Aqui será abordado somente a ANOVA de um critério (isto é, existe apenas uma característica que nos permite distinguir diferentes populações umas das outras).

Esse teste exige que as populações tenham distribuições normais (Normalidade) e também a mesma variância (Homocedasticidade).

Para executar esse teste, devemos calcular primeiro os seguintes valores:

$$\bar{\bar{x}} = \textit{médias de todos os valores amostrais combinados}$$
$$k = \textit{número de amostras}$$
$$n_i = \textit{número de valores na } i-\textit{ésima amostra}$$
$$N = \textit{número total de valores}$$
$$\bar{x}_i = \textit{média dos valores da } i-\textit{ésima amostra}$$
$$s_i^2 = \text{var}\textit{iância dos valores da } i-\textit{ésima amostra}$$

Fórmula 4.6 - Simbologia do teste ANOVA

Com esses valores em mão, calculamos as seguintes estatísticas:

$$SQ(tratamento) = n_1(\bar{x}_1 - \bar{\bar{x}})^2 + n_2(\bar{x}_2 - \bar{\bar{x}})^2 + \cdots + n_k(\bar{x}_k - \bar{\bar{x}})^2 = \sum n_i(\bar{x}_i - \bar{\bar{x}})^2$$
$$SQ(erro) = (n_1 - 1)s_1^2 + (n_2 - 1)s_2^2 + \cdots + (n_k - 1)s_k^2 = \sum (n_i - 1)s_i^2$$
$$SQ(total) = \sum (x - \bar{\bar{x}})^2 = SQ(tratamento) + SQ(erro)$$

Fórmula 4.7 - As diferentes Somas de Quadrados do ANOVA

$$QM(tratamento) = \frac{SQ(tratamento)}{k-1}$$
$$QM(erro) = \frac{SQ(erro)}{N-k}$$
$$QM(total) = \frac{SQ(erro)}{N-1} = QM(tratamento) + QM(erro)$$

Fórmula 4.8 - Os Quadrados Médios do ANOVA

Após todos esse cálculos, podemos finalmente calcular a estatística de teste F conforme a Fórmula 4.9. Essa estatística tem distribuição F.

$$F = \frac{QM(tratamento)}{QM(erro)}$$

$$graus\ de\ liberdade\ do\ numerador = k - 1$$
$$graus\ de\ liberdade\ do\ deno\min ador = N - k$$

Fórmula 4.9 - Estatística de Teste para ANOVA

4.5.2.2 Testes de Hipóteses Não Paramétricos

Esses testes são uma alternativa aos testes paramétricos, pois não exigem que a população obedeça a alguma distribuição, são aplicáveis a pequenas amostras e podem lidar com dados que não estejam expressos numericamente (variáveis qualitativas).

Os testes não paramétricos também possibilitam comparar duas ou mais populações, sendo estas dependentes ou não; avaliam medidas centrais, bem como outros aspectos; verificam a aderência a uma distribuição e a independência entre populações.

4.5.2.2.1 Teste de Kruskal-Wallis

O teste de Kruskal-Wallis, também chamado de teste H, é um teste unilateral à direita usado para testar se diferentes amostras provêm da mesma população. Esse teste **exige variâncias iguais** (Homocedasticidade), não devendo ser usado caso as diferentes amostras possuam variâncias muito diferentes. Por outro lado, **não há exigência de normalidade**.

Como a estatística desse teste é calculada por meio de postos, esse teste pode ser utilizado com dados no nível de mensuração ordinal.

O teste de Kruskal-Wallis é calculado da seguinte forma [TRIOLA-99]:

1. Considerando todas as observações combinadas, atribua um posto a cada uma, do mais baixo para o mais alto, e em caso de empate, atribua a cada observação a média dos postos envolvidos.

2. Para cada amostra k, determine a soma dos postos (R_k) e o tamanho (n_k).
3. Com a soma dos postos e o tamanho de cada amostra, calcule a estatística de teste H conforme a Fórmula 4.10, onde N é o número total de observações em todas as amostras combinadas.

$$H = \frac{12}{N(N+1)}\left(\frac{R_1^2}{n_1} + \frac{R_2^2}{n_2} + \ldots + \frac{R_K^2}{n_K}\right) - 3(N+1)$$

Fórmula 4.10 - Estatística de Teste para o Teste de Kruskal-Wallis

A Estatística H pode ser aproximada pela distribuição qui-quadrado, desde que cada amostra contenha ao menos cinco observações (Se as amostras têm menos de cinco observações, recorre-se a tabelas especiais de valores críticos). O número de graus de liberdade é k-1, onde k é o número de amostras.

Ao aplicarmos o teste, há um fator de correção que deve ser aplicado sempre que há muitos empates. Esse fator de correção é dado pela Fórmula 4.11, onde t é número de observações que são empatadas para um grupo de escores empatados. A Estatística H deve ser dividida por este fator de correção (o sistema sempre aplica esse fator, independentemente do número de empates).

Para exemplificar, suponhamos que os postos combinados fossem os seguintes: 1, 3, 3, 3, 5, 6, 8, 8, 8, 10. Temos três empates com posto 3, daí $T_1=3^3 - 1=27-1=26$; também há três empates com o posto 8, dando $T_2=3^3 - 1=26$. Assim, $\Sigma T = 26 + 26 = 52$, sendo que N igual a 10 observações.

$$1 - \frac{\sum T}{N^3 - N}$$
$$onde \quad T = t^3 - t$$

Fórmula 4.11 - Fator de Correção para o Teste de Kruskal-Wallis

4.5.2.2.2 Teste U de Mann-Whitney

Esse teste, também chamado de soma de postos de Wilcoxon, testa a hipótese de que duas amostras independentes provêm da mesma população. O teste **não exige que as populações sejam distribuídas normalmente**.

Como a estatística desse teste é calculada por meio de postos, esse teste pode ser utilizado com dados no nível de mensuração ordinal.

A base do processo utilizado é o princípio de que, se duas amostras são extraídas de populações idênticas e os escores são dispostos em uma seqüência única, combinada, de valores, então os postos altos e os postos baixos devem situar-se eqüitativamente entre as duas amostras. Se os postos baixos predominam em uma amostra e os altos na outra, então se suspeita que as populações não sejam idênticas.

O teste de Mann-Whitney começa com os seguintes passos [TRIOLA-99]:

1. Ordenar todos os dados amostrais combinados;
2. Achar a soma dos postos para uma das duas amostras;
3. Com a soma dos postos de uma das amostras (R), seu tamanho (n_1) e o tamanho da outra amostra (n_2), calcular a média μ_R e o desvio-padrão σ_R, conforme a Fórmula 4.12.

$$z = \frac{R - \mu_R}{\sigma_R}$$

$$onde \quad \mu_R = \frac{n_1(n_1 + n_2 + 1)}{2}$$

$$\sigma_R = \sqrt{\frac{n_1 n_2 (n_1 + n_2 + 1)}{12}}$$

$n_1 = tamanho\ de\ uma\ amostra.$

$n_2 = tamanho\ da\ outra\ amostra.$

$R = soma\ de\ postos\ da\ amostra\ de\ tamanho\ n_1.$

Fórmula 4.12 - Estatística de Teste para o Teste de Mann-Whitney.

A distribuição amostral de R é aproximadamente normal com média μ_R e desvio-padrão σ_R, desde que ambas as amostras tenham tamanho superior a 10 (Amostras com 10 ou menos valores, recorre-se a tabelas especiais).

4.5.2.2.3 Teste de Wilcoxon

Esse teste, também chamado de teste de postos com sinais de Wilcoxon, testa a hipótese de que duas amostras dependentes (emparelhadas) provêm de populações com a mesma distribuição. O teste **não exige que os dados tenham distribuição normal**.

O teste de Wilcoxon começa com os seguintes passos [TRIOLA-99]:

1. Para cada par de dados, determine a diferença d subtraindo o segundo escore do primeiro. Mantenha os sinais, mas descarte os pares onde d=0.
2. Ignore os sinais das diferenças e ordene-as da menor para a maior. Quando aparecerem diferenças com o mesmo valor numérico, atribua-lhes a média dos postos envolvidos no empate.
3. Atribua a cada posto o sinal da diferença de onde proveio, isto é, insira os sinais que foram ignorados no segundo passo.
4. Ache a soma dos valores absolutos dos postos negativos, e a soma dos postos positivos.
5. Seja T a menor das duas somas obtidas no passo anterior e n o número de pares de dados para os quais a diferença d não é 0, determine a estatística de teste e os valores críticos baseados no tamanho da amostra conforme as Fórmulas 4.13 e 4.14.

$$Para\ n \le 30:\ T$$

$$Para\ n > 30:\ z = \frac{T - \frac{n(n+1)}{4}}{\sqrt{\frac{n(n+1)(2n+1)}{24}}}$$

Fórmula 4.13 - Estatística de Teste para o Teste de Wilcoxon.

$$\begin{array}{|l|}\hline \textit{Para } n \leq 30\text{: Recorre} - \textit{se a tabelas especiais;} \\ \\ \textit{Para } n > 30\text{: Tabela de distribuição Normal.} \\ \hline \end{array}$$

Fórmula 4.14 - Valores Críticos para o Teste de Wilcoxon.

4.5.3 Análise de Correlação e Regressão

A Análise de Correlação e Regressão verifica se duas ou mais variáveis quantitativas se relacionam, e como elas se relacionam numa população.

A Análise de Correlação fornecerá um número que expressa a direção e a magnitude do relacionamento entre as variáveis. Já a Análise de Regressão consiste em tentar encontrar uma equação matemática que descreve o relacionamento entre as variáveis.

5 SEstat.Net

5.1 Descrição do Sistema

O SEstat.Net é uma ferramenta de análise estatística de dados usando base de dados flexível, que disponibiliza aos alunos a realização de inferências estatísticas ou análises descritivas com uma ou duas variáveis, oferecendo os resultados estatísticos da análise, os possíveis caminhos de raciocínio e a linha de aprendizagem efetuada pelo usuário bem como a base de dados do aluno.

O software é composto por um "núcleo" que funciona como a parte inteligente do sistema, possuindo todas as funcionalidades necessárias para a administração dos dados e geração dos resultados. Esse "núcleo" é formado por classes Java, utilizando como interface para o usuário páginas JSP [FIELDS-00].

Por herdar a estrutura didático-pedagógica do SEstat, o SEstat.Net também possui um sistema de *help* sensível ao contexto [MELO-03], onde conforme o ponto que o aluno se encontra são disponibilizados tópicos de ajuda que o auxiliam num melhor entendimento do contexto atual.

Os dados referentes aos alunos (login, senha, etc.) são armazenados num Banco de Dados *MySQL* no servidor. Cada aluno possui uma conta no servidor onde ficam armazenadas todas as suas bases de dados. Estas bases de dados são arquivos criados pelo próprio aluno no padrão *dBase* (.dbf), a partir de dados coletados de pesquisas.

5.2 Desenvolvimento

O software SEstat.Net foi desenvolvido utilizando a plataforma Java 2 [POTTS-96], mais especificamente utilizando J2EE™.

A tecnologia J2EE™ tem seu modelo baseado em componentes, simplificando assim seu desenvolvimento. Ela controla a infra-estrutura e a interoperabilidade das aplicações, suportando serviços WEB com um desenvolvimento seguro e robusto.

Dentre os vários componentes que formam essa tecnologia, utilizamos apenas JSP [VIS-02], JavaBeans, JDBC e Java Servlets para o desenvolvimento do SEstat.Net.

5.2.1 Java Server Pages

Dentre as diversas tecnologias existentes que permitem a elaboração de *websites* com conteúdo dinâmico (PHP – *Personal Home Page*, ASP – *Active Server Pages*, Delphi, CGI-*Commom Gateway Interface*), a equipe de desenvolvimento do SEstat.Net optou pelo componente JSP [JSP-02] por se tratar de uma tecnologia para desenvolvimento de aplicações WEB que possui grande portabilidade de plataforma, podendo ser executado em diversos sistemas operacionais. Ela permite o desenvolvimento de aplicações que acessem banco de dados, arquivos-texto, capte informações a partir de formulários, informações sobre o visitante e sobre o servidor, a geração de gráficos dinâmicos, o uso de variáveis e *loops* entre outras coisas.

Além disso, essa tecnologia permite separar a programação lógica (parte dinâmica) da programação visual (parte estática), facilitando o desenvolvimento de aplicações mais robustas.

As páginas JSP [JSP-03] são páginas HTML (*Hypertext Markup Language*) com algumas *tags* embutidas, que podem conter código Java. A primeira vez que uma página JSP é processada, a JSP *engine* cria um código-fonte para um Java *Servlet* e então esse código é compilado. Após essa compilação, dentro do servidor JSP existirá um Java *Servlet* que poderá ser acessado toda vez em que for requisitado.

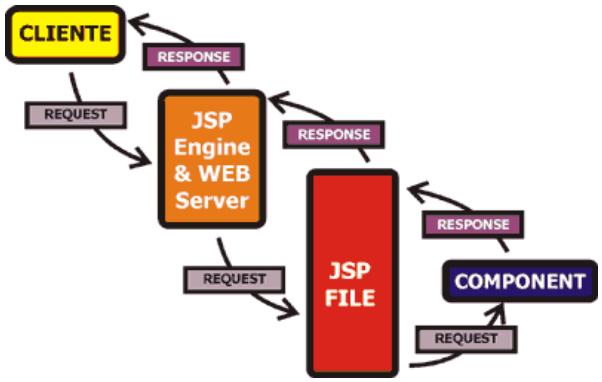


Figura 7 - Requisição de uma página JSP.

O processamento da requisição de uma página JSP funciona como a figura 7. O cliente faz a solicitação de um arquivo JSP, é enviado um *object request* para a JSP *engine*. A JSP *engine* envia a solicitação de um *JavaBean component* especificado no arquivo. O componente controla a requisição possibilitando a obtenção de um gráfico, geração de dados como resultado de uma análise feita ou acesso a um arquivo em banco de dados, em seguida, passa o objeto *response* de volta para a JSP *engine*. A JSP *engine* e o WEB *server* enviam a página JSP

revisada de volta para o cliente, onde o usuário pode visualizar os resultados através do WEB *browser*.

Todas as páginas do SEstat.Net que trocam informações com os alunos na análise estatística dos dados são JSP e realizam a comunicação direta com os *JavaBeans*, atuando como uma interface para os resultados estatísticos gerados por esses. Algumas páginas fazem uso de *JavaScript*[3], que é responsável por encaminhar para uma determinada página dependendo da escolha do usuário. A seqüência de aprendizado que o aluno segue se dá por meio de uma série de páginas, onde cada uma é *linkada* em uma ou mais páginas, formando assim os diversos caminhos de aprendizagem e que são armazenados no servidor.

5.2.2 JavaBeans

JavaBeans (também conhecidos simplesmente como *Beans*) são componentes de software projetados para serem unidades reutilizáveis, onde uma vez criados podem ser reutilizados sem modificação de código, estendendo o conceito de "*Write Once, Run Anywhere*™". Na prática os *Beans* são instâncias de classes Java, e o desenvolvimento de tais classes é semelhante à implementação de uma classe Java comum, com as seguintes diferenças:

- Deve-se declarar um construtor sem argumentos;
- Para acessar os atributos das classes Java através das *tags* jsp:setProperty e jsp:getProperty, deve-se usar uma convenção específica, utilizando os métodos *set* (método para modificar o valor) e *get* (método para retornar o valor) de cada atributo;

A forma de acesso utilizada pelas páginas JSP as classes Java que formam o "núcleo" (figura 8) do SEstat.Net é feita através de *Beans*. As classes "Administrador" e as classes responsáveis pela geração dos dados dos gráficos são instanciadas como *Beans* e são acessadas de forma direta pelas páginas JSP. As classes onde seus objetos são compartilhados entre todos os usuários, como as classes "Gráfico", "base de dados", "leitor DBF", "estatística", "Setip", "educador" são instanciadas como *Beans*, porém são encapsuladas dentro do *Bean* "Administrador" pois não necessitam de acesso direto com as páginas JSP.

Os *Beans* são declarados dentro da página JSP (fazendo uso da *tag* jsp:useBean) e cada um deve definir qual é o seu escopo. O escopo define o ciclo de vida de um objeto, que pode ser definido como:

- **page**: Objetos definidos com o escopo *page* são vistos apenas pela página onde ele foi criado. As referências para objetos no contexto *page* são armazenados no objeto *pageContext*.
- **request**: Objetos são vistos nas páginas usadas para responder a requisição do usuário. Se uma página é redirecionada para outra os objetos de escopo request são preservados uma vez que fazem parte da mesma requisição. As referências são armazenadas no objeto implícito *request*.
- **session**: Objetos que são acessíveis por páginas que fazem parte da mesma sessão (Uma sessão é iniciada no logon do usuário ao sistema e finalizada em sua saída) . As referências são armazenadas no objeto *session*.
- **application**: Objetos que são acessíveis por toda aplicação JSP em questão e são armazenadas no objeto implícito *application*.

5.2.3 JDBC

A tecnologia JDBC™ [SUNJ-03] é uma API que possibilita qualquer aplicação Java acessar qualquer base de dados.

No projeto SEstat.Net foi fundamental o uso do componente JDBC™, já que há a necessidade da utilização de um banco de dados para armazenar os caminhos das bases dos alunos, como também armazenar as senhas e logins, a fim de distinguir o acesso ao software entre alunos e convidados.

Também sua utilização foi essencial para realizar a conexão e iterações com os arquivos do formato Dbase (DBF), já que nesse primeiro instante as bases dos alunos serão fornecidas, ou armazenadas no servidor, no formato DBF, mas o próximo passo é fazer com que o SEstat.Net também possa obter como entrada base em formatos diferentes, como XLS do excel.

5.2.4 Java Servlet

A tecnologia Java Servlet [SUNS-03] fornece um mecanismo simples de estender a funcionalidade de servidores web para um sistema que proporciona um acesso mais rápido. Um servlet possibilita abranger um maior número de tipos de aplicações WEB, estendendo e realçando assim as funcionalidades do servidor.

Basicamente Servlets são módulos de código em Java que rodam em uma aplicação no servidor para responder pedidos do cliente. Eles não são direcionados para um tipo de protocolo cliente-servidor, embora sejam mais comumente usados com HTTP e a palavra "Servlet" seja frequentemente encontrada com o significado "HTTP Servlet".

Como são escritos em Java, eles criam extensões sofisticadas em um servidor, independentemente do sistema operacional, e são comumente usados para acessar conteúdos dinâmicos, processar e armazenar dados mandados de formulários e gerenciar o estado de informação sobre uma conexão de estado HTTP.

No projeto SEstat.Net Servlets possuem uma participação marcante. Como nesse projeto existem diversas páginas JSP funcionando como uma interface para com o aluno, e para cada tipo de análise realizada são processadas diversas páginas no servidor, significando que a cada página acessada, sem o uso de servlets, seria preciso compilá-la novamente. Com o uso dos servlets, apenas no primeiro acesso a página será compilada, fazendo com que os próximos acessos se tornem mais rápidos e eficientes.

5.2.5 Classes do SEstat.Net

Neste artigo, será dado foco maior a descrição das classes consideradas mais importantes no sistema (figura 8).

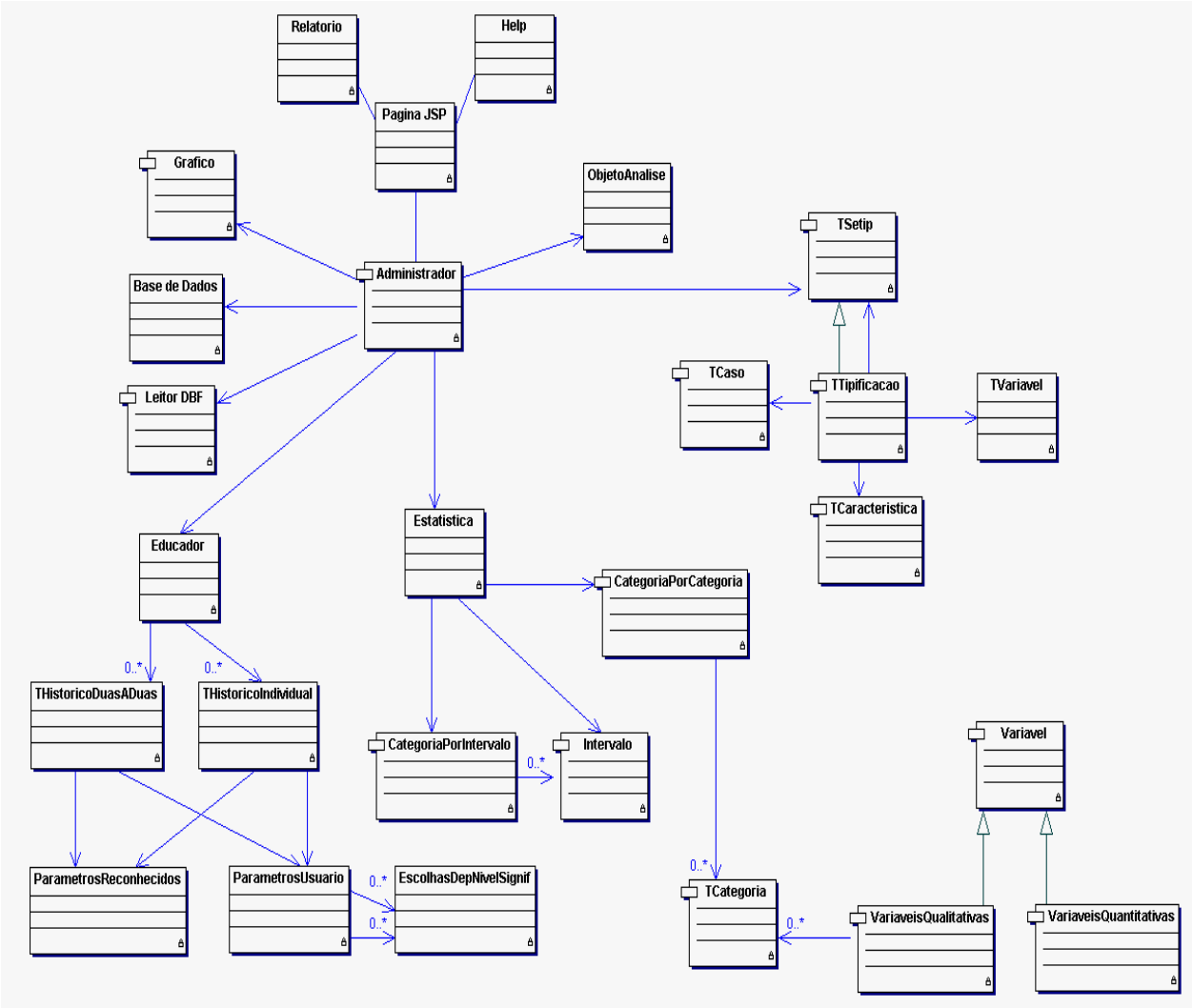


Figura 8 - Núcleo do sistema.

5.2.5.1 Administrador

Ponto central de todo o sistema, é a classe responsável pela comunicação entre as páginas e as outras classes. Cada objeto desta classe é responsável pela gerência dos dados de um aluno específico, e por isso o seu *Bean* é declarado no escopo *session*, fazendo com que cada aluno tenha uma instância particular dessa classe para se relacionar com o sistema.

5.2.5.1.1 Funcionamento Básico:

Todas as requisições das páginas JSP serão feitas diretamente para um *Bean* desta classe, o qual se encarregará de tratar tal requisição e realizar chamadas de funções das outras classes caso sejam necessárias.

Exemplo de uma chamada para esse *Bean*:

- Uma requisição da página JSP é enviada para um *Bean* administrador (Anexo I), informando por exemplo que o aluno escolheu a mensuração "Contínua" para uma descrição Univariada Quantitativa.
- O Administrador verifica se a variável selecionada é verdadeiramente "Contínua", repassando uma mensagem para o aluno caso o sistema tenha reconhecido essa variável como "Discreta". Importante verificar aqui, caso o sistema não concorde com a escolha do aluno, que o tipo de caminho escolhido pelo aluno fica em aberto, deixando para esse a oportunidade para continuar a sua análise ou voltar e escolher essa variável como "Discreta".
- Verificado que o aluno escolheu a variável corretamente, ou que esse deseja continuar sua análise, será então executado o método "DescricaoUnivariadaQuantiCont" (Anexo II).

Método DescricaoUnivariadaQuantiCont:

- Primeiramente essa função requere alguns dados iniciais, como carregar do leitorDBF a variável Quantitativa escolhida pelo aluno, e informar essa variável para um *Bean* da classe Estatística para que esse possa analisá-la, informando assim as suas categorias.
- A partir dessas categorias, são retiras informações que serão repassadas para um *Bean* gráfico.
- Também são gerados tabelas-resultados a partir de dados recolhidos do *Bean* Estatística, que futuramente irão ser passadas para a página JSP para ser exposto ao aluno.

Importante verificarmos nessa seqüência de operações, que todas as informações passadas pelas páginas serão utilizadas apenas para processar os dados e armazená-los nos *Beans*, notando aqui que a classe Administrador age como um verdadeiro gerente do sistema. Todas as informações requeridas pelas páginas JSP serão encaminhadas, também, somente para essa classe, como a requisição de uma tabela, ou dos dados para geração de um gráfico.

5.2.5.2 Estatística

Essa classe é responsável principalmente pela geração e cálculo de todos os procedimentos estatísticos presentes no programa.

Os procedimentos estatísticos de inferência são: TesteT, TesteTSeparate, TesteTPares, Anova, QuiQuadrado, KruskalWallis, Wilcoxon, Spearman, Pearson, MannWhitney. Os procedimentos de avaliação de suposições estatísticas são: freqüência esperada, Homocedasticidade e Normalidade. Os procedimentos descritivos são: tabelas de freqüências, gráficos e medidas descritivas.

Um objeto dessa classe é instanciado pela página JSP como sendo um *Bean* com escopo *application*, e encapsulado em seguida pelo *Bean* Administrador. Esse encapsulamento é realizado para podermos disponibilizar o uso de uma mesma instância para diversos alunos a fim de economizarmos memória e processamento no servidor, ao mesmo tempo tornando a aplicação mais rápida pelo simples fato de não necessitarmos da geração dessa nova instância.

5.2.5.3 Base de Dados

Essa classe faz uso do Driver "Connector/J" para realizar a conexão do JDBC™ ao banco de dados MySQL. Ela é responsável por todas as consultas e atualizações feitas ao banco de dados.

No banco de dados são armazenadas as informações dos alunos, como login e senha, e o caminho onde se encontram os arquivos DBF gerados por esse aluno, como também algumas informações sobre esses arquivos.

Assim como a classe "estatística", uma instância dessa classe é gerada na forma de um *Bean* pela página JSP e em seguida é encapsulada pelo *Bean* "administrador".

5.2.5.4 Leitor DBF

Essa classe faz uso do Driver *Type 4 DBF JDBC 2.0* (obtido no endereço http://hxtt.net/jdbc) para realizar a conexão com do JDBC™ com o arquivo DBF requerido pelo aluno. Ela é responsável pela obtenção dos dados contidos dentro desses arquivos.

Optou-se pela utilização desse formato de arquivo pela facilidade de criação (pode ser gerado em qualquer programa de planilha eletrônica) e para mantermos o raciocínio de elaboração e armazenamento dos dados obtido pelas versões anteriores do SEstat.Net.

Uma instância dessa classe é gerada pela página JSP como sendo um *Bean* com escopo *session* (isso significa que cada usuário terá um *Bean* "leitorDBF" diferente), sendo também encapsulado no *Bean* "administrador". Isso se faz necessário pois cada instância dessa classe necessita guardar os dados da base DBF escolhida por cada usuário.

5.2.5.5 Gráfico

A geração de gráficos é parte fundamental do sistema e tem por finalidade mostrar de uma forma visual as análises estatísticas feitas sobre os dados do aluno. A parte do SEstat.Net responsável pela geração de gráficos se subdivide em 4 partes:

- Classe Gráfico: responsável unicamente por obter os dados oriundos das análises estatísticas e transformá-los para um determinado padrão que será utilizado pelas classes *DataProducer*. Dentro das páginas JSP esta classe é declarada como um *Bean* com escopo *application*.
- Classes *DataProducer*: uma série de classes responsáveis pela organização e armazenamento dos dados de uma forma legível para as bibliotecas geradoras de gráficos, adaptando os tipos dos dados vindos da classe Gráfico para o formato que o gerador de cada tipo de gráfico reconheça. Dentro das páginas JSP estas classes são instanciadas como *Beans* com escopo *page*, sendo declaradas somente nas páginas onde se visualizarão os resultados.
- Biblioteca *Cewolf*: biblioteca *open-source*, desenvolvida em Java, que facilita a utilização das classes geradoras do gráfico (*JFreeChart*), fazendo uma espécie de interface entre a página JSP e tais classes. Sua utilização se dá por meio de *tags* XML utilizadas dentro da página JSP.
- Biblioteca *JFreeChart*: responsável pela própria geração de gráficos. Esta biblioteca, após receber os dados provenientes das classes e bibliotecas anteriores, gera dinamicamente um arquivo gráfico (formato PNG) dentro do servidor e retorna para a página um *link* onde foi armazenado este arquivo. A partir deste *link* a página JSP pode mostrar o gráfico gerado. Esta biblioteca também é *open-source* e desenvolvida em Java.

5.3 Interface do sistema

Nesse capítulo mostraremos a interface do SEstat.Net. A seqüência de figuras mostra um dos possíveis caminhos que o aluno pode seguir para a aprendizagem dos conceitos estatísticos.

A Figura 9 mostra a entrada do sistema, onde o aluno deve informar o seu *login* e sua respectiva senha.

Figura 9 - Entrada do sistema.

Após a entrada no sistema, são mostradas todas as bases de dados referentes ao aluno. Na figura 10 são mostradas as bases pap.dbf e teste.dbf que estão armazenadas no servidor e que podem ser utilizadas pelo aluno e este deve escolher uma delas para o seu trabalho.

Figura 10 - Escolha da base de dados do aluno.

Em seguida o aluno tem a opção de escolher qual o procedimento estatístico ele quer aplicar nos seus dados. Nesse exemplo foi escolhida a Descrição Univariada.

Figura 11 - Seleção do procedimento estatístico.

Aqui o aluno deve escolher qual a variável de sua base de dados que ele irá trabalhar nessa descrição. A figura 12 mostra a seleção da variável TAM.

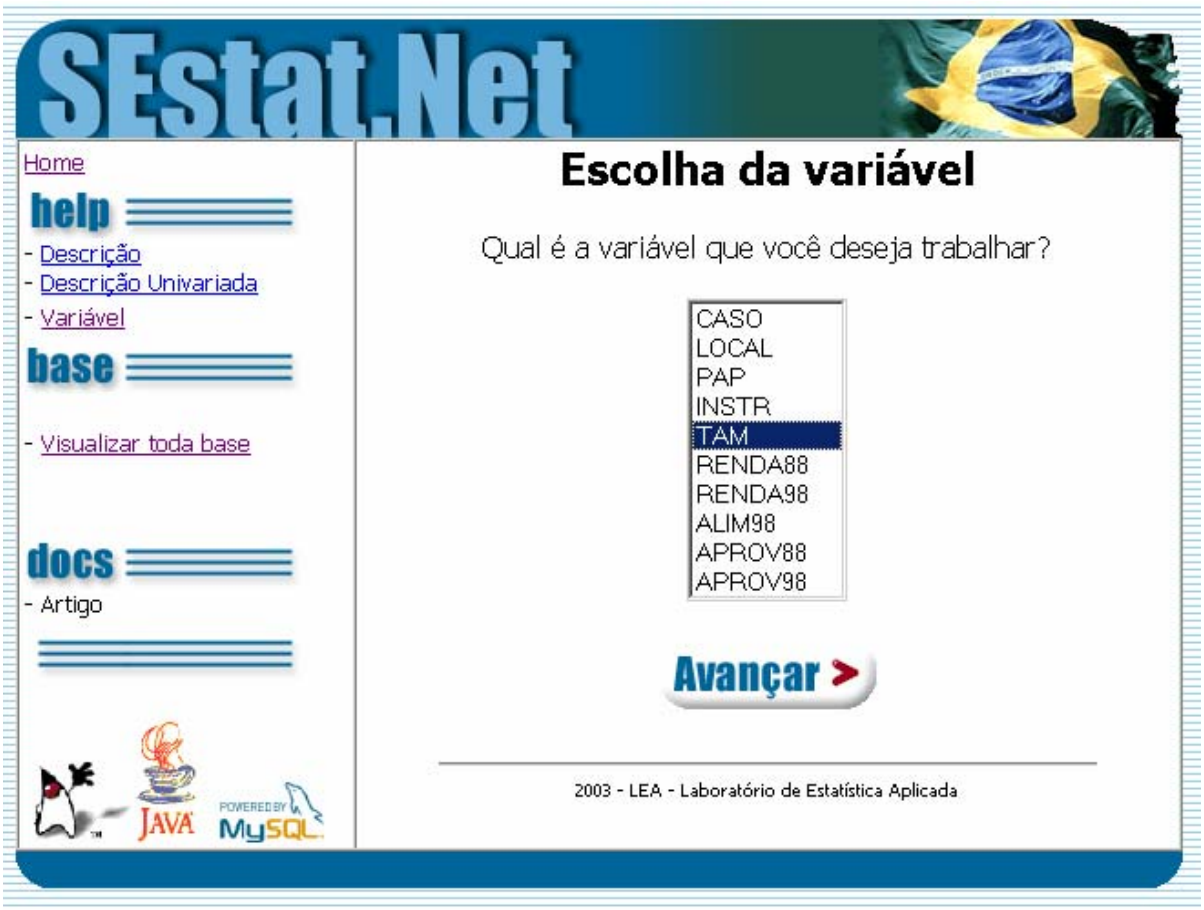


Figura 12 - Escolha da variável de trabalho do aluno.

Nesse ponto do sistema, o aluno deve indicar qual é o tipo da variável escolhida.

Figura 13 - Seleção do tipo da variável escolhida.

Mostra-se aqui um exemplo da utilização da classe Setip, classe responsável por analisar toda a base de dados e tipifica-la. A variável TAM é referente ao tamanho de uma família, portanto esta variável é quantitativa. Como o aluno escolheu Qualitativa, o sistema indica que o tipo escolhido da variável é incorreto, porém dá a possibilidade ao aluno de continuar com a sua análise.

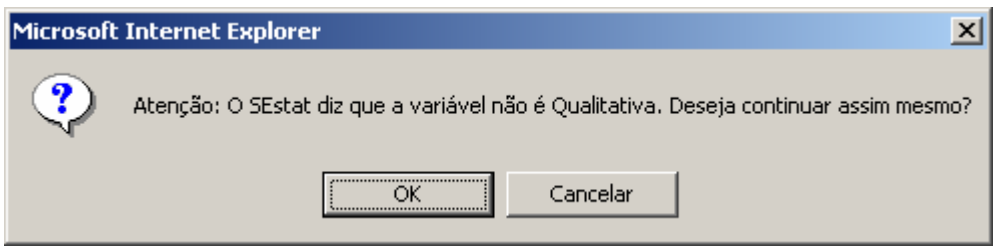


Figura 14 - Caso onde o aluno escolheu um tipo que o SEstat. Net julga ser incorreto.

A figura 15 mostra a seleção da mensuração da variável quantitativa escolhida.

Figura 15 - Escolha da mensuração da variável quantitativa.

Como resultado desse caminho de aprendizagem é mostrada uma tela semelhante a figura 16, onde podem ser visualizados as tabelas de freqüências e de medidas descritivas, além do histograma referente a variável. Vale salientar que durante todo esse processo o aluno tem a disposição um *help* sensível ao contexto, como também pode visualizar os valores da base de dados.

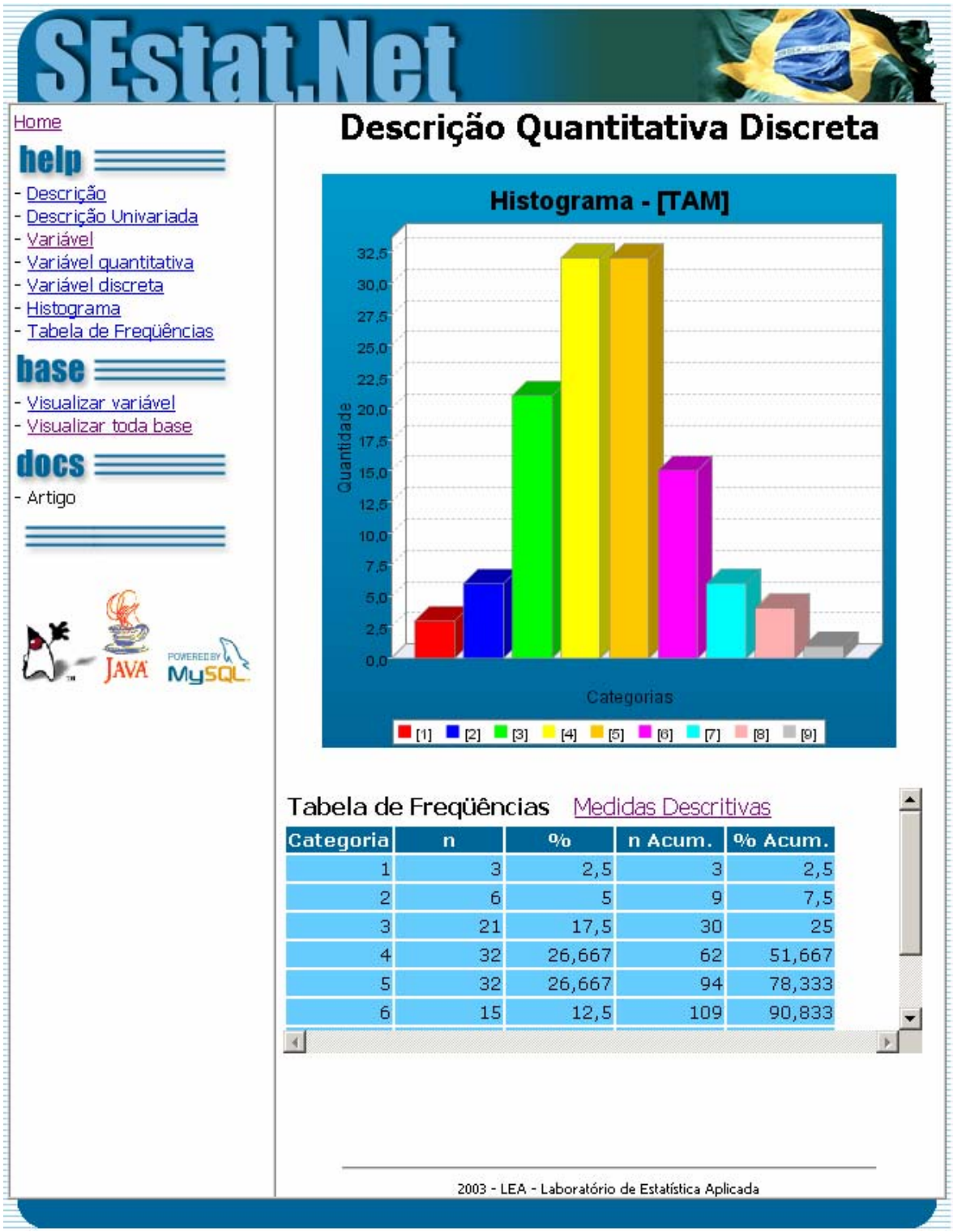


Figura 16 - Resultados estatísticos.

5.4 Requisitos para o funcionamento do SEstat.Net

Como o SEstat.Net utiliza tecnologias como JSP, JavaBeans e Servlets é necessário que se tenha um servidor especifico que compreenda essas tecnologias utilizadas. Para esse projeto, o servidor escolhido e utilizado foi o RESIN Enterprise 2.1.6 (http://www.caucho.com), devido a sua ótima indicação por *sites* especializados, simplicidade de configuração e ótima performance obtida nos testes realizados.

Também para esse projeto foi necessária a utilização de um servidor de banco de dados. O servidor escolhido foi o MySQL (http://www.mysql.com), por se tratar de um banco de dados com performance comprovada para aplicações Web e de fácil utilização.Um ponto importante para essa escolha foi por se tratar de um software sem custo e *open-source*.

Além disso, dentro do servidor JSP devem estar instaladas as bibliotecas *Cewolf* (http://cewolf.sourceforge.net) e *JFreeChart* (http://www.object-refinery.com/jfreechart) e os drivers *Connector/J 2.0* (http://www.mysql.com/downloads/api-jdbc-stable.html) e *Type 4 DBF JDBC 2.0* (http://hxtt.net/jdbc).

6 Conclusões

Nesse projeto, a principal e primeira meta alcançada foi fazer com que o SEstat.Net possuísse as mesmas características didático-pedagógicas do SEstat. Além de preenchermos esse requisito, conseguimos também deixar o software relativamente com baixa taxa computacional, o bastante para que as iterações cliente / servidor não se tornassem prejudicadas pelo alto número de processamento requerido no servidor.

Outro grande êxito na pesquisa foi utilizar uma tecnologia (Java Server Pages) que fizesse com que o servidor da aplicação realizasse todas as operações requeridas, passando para o cliente os resultados já em forma de uma página HTML estática.

Um ponto importante na avaliação do SEstat.Net no ensino a distância é testar seu comportamento quando houver o acesso de vários alunos interagindo com o sistema ao mesmo tempo. Neste projeto está prejudicado, pois agora se inicia o uso pelos alunos de graduação. No entanto as questões são: qual será o comportamento da máquina servidora (uso de memória e processamento), da plataforma Java sobre a eliminação dos objetos que já foram inutilizados e do servidor JSP, se conseguirá suportar tantos acessos quanto esperamos que esse tenha.

6.1 Trabalhos Futuros

Visa ser interessante que o SEstat.Net seja integrado com o ambiente de sala de aula virtual [NOAL-02], para formarem, juntos, um sistema mais completo de ambiente de ensino a distância, simulando perfeitamente um ambiente de sala de aula tradicional.

7 Referências Bibliográficas

[BARBETTA-94] BARBETTA, Pedro A. **Estatística Aplicada às Ciências Sociais**, Editora da UFSC, Florianópolis, 1994.

[BASDEN–84] BASDEN, Andrew. **On the Application of Expert Systems**. Developments in Expert Systems. London: Academic Press Inc., 1984.

[CATAPAN-01]CATAPAN, Araci H.; FIALHO, Francisco A. P.. **Tertium**: o novo modo do ser, do saber e do aprender (construindo uma taxionomia para medição pedagógica em tecnologia de comunicação digital). Florianópolis, 2001. [240] f. Tese apresentada ao programa de pós-graduação em Engenharia de produção e Sistemas - Universidade Federal de Santa Catarina.

[CECHINEL-97]CECHINEL, Cristian; AMORIM, Lidiane. **Sistema especialista de apoio ao Ensino de Estatística**. Florianópolis, 1997. Trabalho de conclusão de curso de graduação em ciências da computação - Universidade Federal de Santa Catarina.

[CHAIBEN-99] CHAIBEN, Hamilton. **Inteligência Artificial na Educação**, http://www.cce.ufpr.br/~hamilton/iaed/iaed.htm.

[FIELDS-00]FIELDS, Duane K.; KOLB, Mark A. **Web Development with JavaServer Pages.** 2. ed. Greenwich: Manning, 2000. 584 p.

[DIAS-01]DIAS, Kirliam M.. **Sistema especialista para auxílio ao Ensino de Estatística**. Florianópolis, 2000. [168] f. Trabalho de conclusão de curso de graduação em ciências da computação, Universidade Federal de Santa Catarina.

[JSP-02]JSP Tutorial. **JSP Tutorial.** Disponível em: < http://www.jsptut.com>. Acesso em: 11 dezembro 2002.

[JSP-03]JSPBrasil. **Tutorial JSP**. Disponível em: <http://www.jspbrasil.com.br>. Acesso em: 15 janeiro 2003.

[KEMP-92] KEMP, R. **Intelligent Computer Assisted Instruction: A Knowledge-Based Perspective**, The Australian Computer Journal, Vol. 24 N. 3, pp.121-129, 1992.

[LIEBOWITZ-88] LIEBOWITZ, Jay. **Introduction to Expert Systems**, Santa Cruz, California, USA, Mitchell Publishing Inc., 1988.

[LIEBOWITZ-95] LIEBOWITZ, J. **Expert Systems: Dead or Alive?**, Educational Technology, pp. 53-55, March-April 1995.

[MELO-03]MELO, Angelo dos S. **Help SEstat**. Disponível em: <http://www.jspbrasil.com.br>. Acesso em: 15 janeiro 2003.

[MINSKY-86] MINSKY, M. **The Society of Mind**, Simon & Schuster, Inc, New York, 1986.

[NOAL-02] NOAL, Renato Bica. **Ambiente colaborativo para ensino de estatística com o SESTAT**. Florianópolis, 2002. [63] f. Dissertação apresentada ao programa de pós-graduação em Ciências da Computação - Universidade Federal de Santa Catarina.

[POTTS-96]POTTS, Anthony; FRIEDEL, Jr., David H. **Java: Programming Language Handbook.** 1. ed. Scottsdale: Coriolis Group Books, 1996. 143 p.

[RAMOS-91] RAMOS, Edla. **O Fundamental na Avaliação da Qualidade do Software Educacional**, Departamento de Informática e Estatística, UFSC, 1991.

[RAMOS-95] RAMOS, Ronaldo F. **Sistemas Especialistas – Uma Abordagem Baseada em Objetos com Prototipagem de um Selecionador de Processos de Soldagem**, Departamento de Engenharia de Produção e Sistemas, UFSC, 1995.

[RICKEL-89] RICKEL, J. W. **Intelligent Computer-Aided Instruction: A Survey Organized Around System Components**, IEEE Transactions on Systems, Man, and Cybernetics, Vol. 19, N. 1, pp. 40-57.

[STAT-94] STATISTICA, **Manual do Software Statistica**, volume 3, 1994.

[SUNS-03]SUN Microsystems. **JAVA™ Servlet Technology**. Disponível em: <http://java.sun.com/products/servlet/ >. Acesso em: 10 fevereiro 2003.

[SUNJ-03]SUN Microsystems. **JDBC™ Data Acess API**. Disponível em: <http://java.sun.com/products/jdbc/ >. Acesso em: 10 fevereiro 2003.

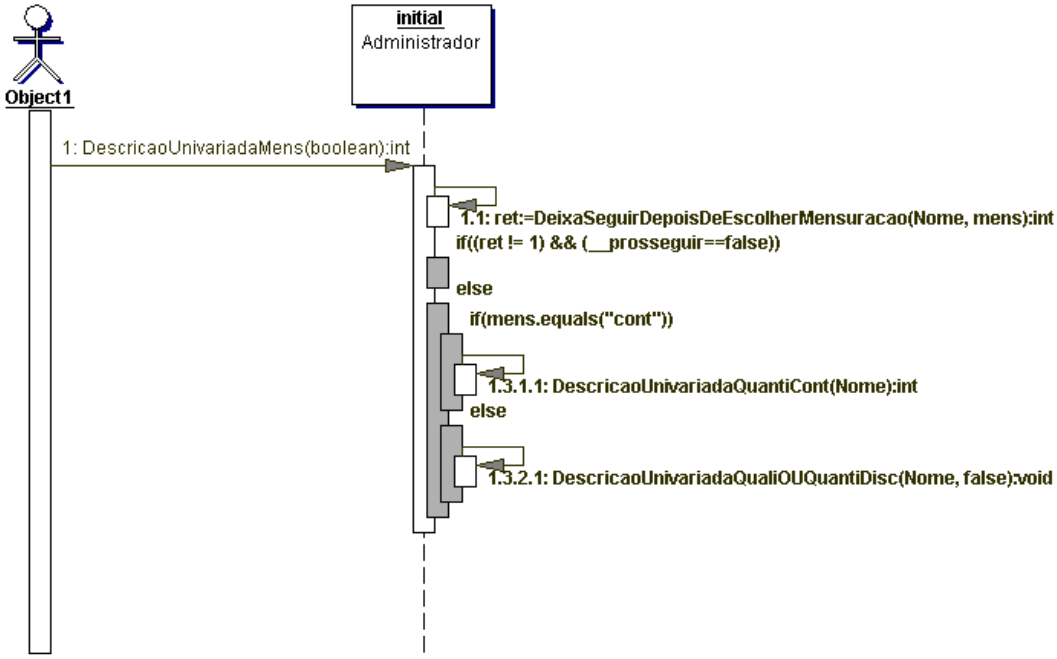
[TRIOLA-99] TRIOLA, Mario F. **Introdução à Estatística**, 7ª edição, Livros Técnicos e Científicos Editora S.A., Rio de Janeiro, 1999.

[TROLLIP-91] TROLLIP, S. R. & ALESSI, S. M. **Computer-Based Instruction, Methods and Development**, Second Edition, Prentice Hall, INC, 1991.

[VIS-02]Visualbuilder.com. **JSP Tutorial.** Disponível em: http://visualbuilder.com. Acesso em: 10 dezembro 2002.

[WATERMAN-85] WATERMAN, Donald Arthur. **A Guide to Expert Systems**, USA, Addison Wesley Publishing Company, Inc.,1985.

Anexo I: Diagrama de seqüência: DescricaoUnivariadaMens(boolean)



Anexo II: Diagrama de seqüência: DescricaoUnivariadaQuantiCont(string)

Anexo III: Fontes do Sistema

Administrador.java

```
import java.text.NumberFormat;

/**
 * Será o gerenciador para a obtenção dos dados estatisticos. Ele comandará o
acesso a base de dados, reconhecimento e criacao das variáveis, geraçao de
gráficos e chamadas das funções estatisticas.
 * (Um para cada usuario)
 */
public class Administrador {
    /*
     * @stereotype constructor
     */
    public Administrador() {
        mensVar = new String[2];
        tipoVar = new String[2];
        nCat = new int[2];
        id_usuario = 0;
        login = "";
        senha = "";
    }

    public void setEducador(Educador __educador) {
        educador = __educador;
        double[] temp1 = new double[7];
        double[] temp2 = new double[7];
        for(int i=0; i<7; i++){
            temp1[i] = 90;
            temp2[i] = 70;
        }
        educador.init(temp1, temp2);
    }

    //Armazena e Fornece os objetos que serao tratados no Administrador
    public void setSetip(TSetip __setip) {
        this.setip = __setip;
    }

    public void setLeitorDBF(Leitor_DBF __leitor) {
        leitorDBF = __leitor;
    }

    public Leitor_DBF getLeitorDBF() {
        return leitorDBF;
    }

    public void setGrafico(Grafico __grafico) {
        this.grafico = __grafico;
    }

    public String[] getGraphPieSeriesNames() {
        return grafico.getSeriesNames();
    }

    public String[] getGraphSeriesNames() {
       return grafico.getSeriesNames();
    }
```

```
public String[] getGraphCategoriesNames() {
    return grafico.getCategoriesNames();
}

public Integer[][] getGraphBarValues() {
    return grafico.getBarValues();
}

public Integer[][][] getXyValues() {
    return grafico.getXyValues();
}


public Integer[] getGraphPieValues() {
    return grafico.getPieValues();
}

public  void setBase(Base_de_Dados __base) {
    this.base = __base;
}

public String[][] getFObservadas() {
    return fObservadas;
}

public String[][] getFEsperadas() {
    return fEsperadas;
}
public String[][] getTabelaContribuicaoQQ() {
    return TabelaContribuicaoQQ;
}

public void setEstatistica(Estatistica __estatistica) {
    this.estatistica = __estatistica;
}

public void setLogin(String __login) {
    this.login = __login;
}

public void setSenha(String __senha) {
    this.senha = __senha;
}

public void setNome(String __nome) {
    Nome = __nome;
}

public String getNomeE() {
    return NomeE;
}

public void setNomeE(String __nomeE) {
    NomeE = __nomeE;
}

public String getNomeD() {
    return NomeD;
}
```

```
public void setNomeD(String __nomeD) {
    NomeD = __nomeD;
}

public void setTipo(String __tipo) {
    tipo = __tipo;
}

public void setTipoE(String __tipoE) {
    tipoE = __tipoE;
}

public void setTipoD(String __tipoD) {
    tipoD = __tipoD;
}

public void setMens(String __mens){
    mens = __mens;
}

public void setMensE(String __mensE){
    mensE = __mensE;
}

public void setMensD(String __mensD){
    mensD = __mensD;
}

public void setSignificancia(double __significancia){
    significancia = __significancia;
}

public void setcatE(int __catE){
    catE = __catE;
}

public void setcatD(int __catD){
    catD = __catD;
}

public void setParidade(String __paridade){
    if(__paridade.equals("sim")){
        paridade = 1;
    }else{
        paridade = 0;
    }
}

public void setNormalidade(String __normalidade){
   if(__normalidade.equals("sim")){
        normalidade = 1;
    }else{
        normalidade = 0;
    }
}

public void sethomocedasticidade(String __homocedasticidade){
    if(__homocedasticidade.equals("sim")){
        homocedasticidade = 1;
    }else{
        homocedasticidade = 0;
```

```
    }
}

public void setFrequenciaEsperada(String __FrequenciaEsperada){
    if(__FrequenciaEsperada.equals("sim")){
       frequenciaEsperada = 1;
    }else{
       frequenciaEsperada = 0;
    }
}

public String getNome() {
    return Nome;
}

public String getTipo() {
    return tipo;
}

public String getLogin() {
    return this.login;
}

public String getSenha() {
    return this.senha;
}

public TSetip getSetip() {
    return this.setip;
}

public Grafico getGrafico() {
    return this.grafico;
}

public Base_de_Dados getBase() {
    return this.base;
}

public Estatistica getEstatistica() {
    return this.estatistica;
}

public void setArquivoDBF(String __arquivo){
    leitorDBF.setArquivoDBF(__arquivo);
}

public String getArquivoDBF(){
    return leitorDBF.getArquivoDBF();
}

public boolean connectDBF() {
    boolean _retorno = leitorDBF.ConectaDBF(login);
    if(_retorno == false){
        return false;
    }else{
        leitorDBF.iniciaVariaveis();
        _retorno = educadorReconheceBase();
        return _retorno;
    }
}
```

```
    //Trabalha com a base de dados e o arquivo DBF
    public boolean verificaSenhaLogin() {
       id_usuario = base.ConsultaUsuario(this.login, this.senha);
       if(id_usuario == -1){
           return false;
       }else{
           return true;
       }
    }

    public String[][] getVisualizacaoArquivoDBF(){
        String[] labels = leitorDBF.RetornaLabelColunas();
        String[][] colunasTemp = new
String[labels.length][leitorDBF.RetornaNumeroValores()];
        for(int i=0; i<labels.length; i++){
            colunasTemp[i] = leitorDBF.RetornaDadosVariavel(labels[i]);
        }
        String[][] colunas = new
String[labels.length][leitorDBF.RetornaNumeroValores()+1];
        for(int i=0; i<labels.length;i++){
            colunas[i][0] = labels[i];
            for(int j=1;j<leitorDBF.RetornaNumeroValores()+1;j++){
                colunas[i][j] = colunasTemp[i][j-1];
            }
        }
        return colunas;

    }

    public String[] getNomeBaseArquivos(){
        //caso ouve erro retornará null ou "", verificar isso.
        return base.RetornaNomeArquivo(id_usuario);
    }

    public String[] getDescricaoArquivos(){
        return base.RetornaDescricao(id_usuario);
    }

    public String getCriacao(String __arquivo){
       return base.RetornaCriacao(id_usuario, __arquivo);
    }

    public String getUltimaVizualizacao(String __arquivo){
        return base.RetornaUltimaVizualizacao(id_usuario, __arquivo);
    }

    public String getEndereco(){
        return base.RetornaEndereco(id_usuario);
    }

    public String[] getLabelsDBF(){
        return leitorDBF.RetornaLabelColunas();
    }

     public String[] getLabelsDBF(String __nome_arquivo){
        return leitorDBF.RetornaLabelColunas(__nome_arquivo);
    }

    public String[] getDadosColunaDBF(String __coluna){
        return leitorDBF.RetornaDadosVariavel(__coluna);
```

```
    }

    public String[] getDadosColunaDBF(String __coluna, String
__nome_arquivo){
        return leitorDBF.RetornaDadosVariavel(__coluna,__nome_arquivo);
    }

    public int getId_usuario(){
        return id_usuario;
    }

    public String[] getValoresColuna(int __coluna) {
       return leitorDBF.RetornaDadosVariavelPorColuna(__coluna);
    }

    //Funcoes para os métodos estatisticos.
    public String[] getMensuracaoVariavel(){
        return mensVar;
    }

    public String[] getTipoVariavel(){
        return tipoVar;
    }

    public int[] getNCat(){
        return nCat;
    }


//**************************************************************************
**//
    //                              DESCRICAO UNIVARIADA
//

//**************************************************************************
**//

    public boolean DescricaoUnivariada(){
        Objeto Tipo_Automatico = new Objeto("quali");
        Tipo_Automatico.setBoolean(false);
        if(educador.AutomatizaTipo(Nome, Tipo_Automatico,
Tipo_Automatico)==false){
          //Erro no automatizaTipo.
          return false;
        }else{
          mensVar[0] = null;
          mensVar[1] = null;
          tipoVar[0] = null;
          tipoVar[1] = null;
          if(Tipo_Automatico.booleanValue()){
            tipoVar[0] = Tipo_Automatico.getString();
            tipoVar[1] = null;
            if("quanti".equals(tipoVar[0])){
                Tipo_Automatico.setBoolean(false);
                Objeto mensuracao = new Objeto("nom");
                if(educador.AutomatizaMensuracao(Nome, mensuracao,
Tipo_Automatico)==false){
                    //Erro no automatizaMensuracao
                    return false;
                }else{
                    if(Tipo_Automatico.booleanValue()){
```

```
                        mensVar[0] = mensuracao.getString();
                        mensVar[1] = null;
                    }
                }
            }
          }
        }
        return true;
    }

//*************************************************************************
*******************
    //  Verificar os tipos de retornos e os tipos de mensagens que serao
dadas!!!
    //   0-> Nao pode realizar a analise
    //   1-> Passou certo.
    //   2-> 'Atenção: O SEstat diz que a variável '+Nome+' não é '+auxstr+'.
Deseja continuar assim mesmo?
    //   3-> 'Atenção: essa variável não pode ser tratada como '+auxstr+'!'
    //   4-> Gráfico e tabelas gerados (TabelaFrequencias,
TabelaDescritivas);
    //   5 -> Erro no automatizarMensuracao.

//*************************************************************************
*******************
    public int DescricaoUnivariadaTipo(boolean __prosseguir){
      int ret = DeixaSeguirDepoisDeEscolherTipo(Nome, tipo);
      if((ret != 1)&&(__prosseguir == false)){
          return ret;
      }else{
          if(tipo.equals("quali")){
             DescricaoUnivariadaQualiOUQuantiDisc(Nome,true);
             return 4; //Gráfico e tabelas gerados (TabelaFrequencias,
TabelaDescritivas);
          }else{
              Objeto mensuracao = new Objeto("nom");
              Objeto automatizar = new Objeto(false);

if(educador.AutomatizaMensuracao(Nome,mensuracao,automatizar)==false){
                 return 5; //Erro no automatizarMensuracao.
              }else{
                 if(automatizar.booleanValue()){
                    mensVar[0] = mensuracao.getString();
                    mensVar[1] = null;
                    return 1; //Passou certo.
                 }
              }
          }
      }
      return 1; //Passou certo!!!
    }


//*************************************************************************
*******************
    //  Verificar os tipos de retornos e os tipos de mensagens que serao
dadas!!!
    //   0-> Nao pode realizar a analise
    //   1-> Passou certo (Gráfico e tabelas gerados (TabelaFrequencias,
TabelaDescritivas).
```

```
    //   2-> 'Atenção: O SEstat diz que a variável '+Nome+' não é '+auxstr+'.
Deseja continuar assim mesmo?

//**************************************************************************
*********************
    public int DescricaoUnivariadaMens(boolean __prosseguir){
      int ret = DeixaSeguirDepoisDeEscolherMensuracao(Nome, mens);
      if((ret != 1) && (__prosseguir==false)){
          return ret;
      }else{
          if(mens.equals("cont")){
            return DescricaoUnivariadaQuantiCont(Nome);
          }else{
            DescricaoUnivariadaQualiOUQuantiDisc(Nome,false);
            return 1;
          }
      }
    }


//**************************************************************************
**//
    //                               DESCRICAO BIVARIADA
//

//**************************************************************************
**//
    public boolean DescricaoBivariada(){
          Objeto Tipo_AutomaticoE = new Objeto("quali");
          Tipo_AutomaticoE.setBoolean(false);

if(educador.AutomatizaTipo(NomeE,Tipo_AutomaticoE,Tipo_AutomaticoE)==false){
              //Erro no AutomatizaTipo;
              return false;
          }
          Objeto Tipo_AutomaticoD = new Objeto("quali");
          Tipo_AutomaticoD.setBoolean(false);

if(educador.AutomatizaTipo(NomeD,Tipo_AutomaticoD,Tipo_AutomaticoD)==false){
              //Erro no AutomatizaTipo;
              return false;
          }
          tipoVar[0] = null;
          tipoVar[1] = null;
          mensVar[0] = null;
          mensVar[1] = null;
          if(Tipo_AutomaticoE.booleanValue()){
            tipoVar[0] = Tipo_AutomaticoE.getString();
          }
          if(Tipo_AutomaticoD.booleanValue()){
            tipoVar[1] = Tipo_AutomaticoD.getString();
          }
          return true;
    }

//**************************************************************************
*******************
    //  Verificar os tipos de retornos e os tipos de mensagens que serao
dadas!!!
    //   0-> Nao pode realizar a analise
    //   1-> Passou certo.
```

```
    //   2-> 'Atenção: O SEstat diz que a variável '+Nome+' não é '+auxstr+'.
Deseja continuar assim mesmo?
    //   3-> 'Atenção: essa variável não pode ser tratada como '+auxstr+'!'
    //   4-> Gráfico e tabelas gerados (TabelaFrequencias,
TabelaDescritivas);
    //   5 -> Erro no automatizarMensuracao.
    //   6 -> Erro ao tentar criar os Gráficos e as Tabelas

//**************************************************************************
********************

    public int DescricaoBivariadaTipo(boolean __prosseguir){
       int retE = DeixaSeguirDepoisDeEscolherTipo(NomeE,tipoE);
       int retD = DeixaSeguirDepoisDeEscolherTipo(NomeD, tipoD);
       if(((retE != 1)||(retD!=1))&&(__prosseguir==false)){
           if(retE !=1){
               return retE;
           }else{
               return retD;
           }
       }else{
         if(("quali".equals(tipoE)) &&("quali".equals(tipoD))){
           //'Será aplicado o método Crosstabulation para descrever as
variáveis'
           tipoAtual = "QualiQuali";
           return DescricaoBivariadaQualiQuali(NomeE, NomeD);
         }
         mensVar[0] = null;
         mensVar[1] = null;
         if(tipoE.equals("quanti")){
             Objeto Mens_Automatico = new Objeto(false);
             Mens_Automatico.setString("nom");

if(educador.AutomatizaMensuracao(NomeE,Mens_Automatico,Mens_Automatico)==fals
e){
                 return 5; //Erro no automatizarMensuracao.
             }else{
               if(Mens_Automatico.booleanValue()){
                  mensVar[0] = Mens_Automatico.getString();
               }
             }
         }
         if(tipoD.equals("quanti")){
             Objeto Mens_Automatico = new Objeto(false);
             Mens_Automatico.setString("nom");

if(educador.AutomatizaMensuracao(NomeD,Mens_Automatico,Mens_Automatico)==fals
e){
                 return 5; //Erro no automatizarMensuracao.
             }else{
               if(Mens_Automatico.booleanValue()){
                  mensVar[1] = Mens_Automatico.getString();
               }
             }
         }
         return 1;
       }
    }
```

```
//**************************************************************************
********************
    //  Verificar os tipos de retornos e os tipos de mensagens que serao
dadas!!!
    //   0-> Nao pode realizar a analise
    //   1-> Passou certo.
    //   2-> 'Atenção: O SEstat diz que a variável '+Nome+' não é '+auxstr+'.
Deseja continuar assim mesmo?
    //   3-> 'Atenção: essa variável não pode ser tratada como '+auxstr+'!'
    //   4-> Gráfico e tabelas gerados (TabelaFrequencias,
TabelaDescritivas);
    //   5 -> Erro no automatizarMensuracao.
    //   6 -> Erro ao tentar criar os Gráficos e as Tabelas

//**************************************************************************
********************

    public int DescricaoBivariadaMens(boolean __prosseguir){
      if(mensE.equals("")){
          int ret = DeixaSeguirDepoisDeEscolherMensuracao(NomeD,mensD);
          if((ret != 1)&&(__prosseguir == false)){
              return ret;
          }else{
              if(mensD.equals("cont")){
                  //'Será aplicado o método BreakDown para descrever as
Variáveis'
                  tipoAtual = "QualiQuanti";
                  return
DescricaoBivariadaQuantiQualiOUQualiQuanti(NomeE,NomeD);
              }else{
                  //'Será aplicado o método BreakDown para descrever as
Variáveis'
                  tipoAtual = "QualiQuanti";
                  return
DescricaoBivariadaQuantiQualiOUQualiQuanti(NomeE,NomeD);
              }
          }
      }
      if(mensD.equals("")){
          int ret = DeixaSeguirDepoisDeEscolherMensuracao(NomeE,mensE);
          if((ret != 1)&&(__prosseguir == false)){
              return ret;
          }else{
              if(mensE.equals("cont")){
                  //'Será aplicado o método BreakDown para descrever as
Variáveis'
                  tipoAtual = "QualiQuanti";
                  return
DescricaoBivariadaQuantiQualiOUQualiQuanti(NomeE,NomeD);
              }else{
                  //'Será aplicado o método BreakDown para descrever as
Variáveis'
                  tipoAtual = "QualiQuanti";
                  return
DescricaoBivariadaQuantiQualiOUQualiQuanti(NomeE,NomeD);
              }
          }
      }
      int ret = DeixaSeguirDepoisDeEscolherMensuracao(NomeE,mensE);
      int ret1 = DeixaSeguirDepoisDeEscolherMensuracao(NomeD,mensD);
```

```
      if(((ret != 1)||(ret1 != 1))&&(__prosseguir == false)){
          return ret;
      }else{
          if(mensE.equals("cont")){
              if(mensD.equals("cont")){
                  //'Será aplicado uma Descrição das Variáveis Contínuas'
                  tipoAtual = "QuantiQuanti";
                  return DescricaoBivariadaQuantiQuanti(NomeE, NomeD);
              }else{
                  //'Será aplicado o método BreakDown para descrever as
Variáveis'
                  tipoAtual = "QuantiQuanti";
                  return DescricaoBivariadaQuantiQuanti(NomeE, NomeD);
              }
          }else{
              if(mensD.equals("cont")){
                  //'Será aplicado o método BreakDown para descrever as
Variáveis'
                  tipoAtual = "QuantiQuanti";
                  return DescricaoBivariadaQuantiQuanti(NomeE, NomeD);
              }else{
                  //'Será aplicado o método CrossTabulation para descrever as
Variáveis'
                  tipoAtual = "QuantiQuanti";
                  return DescricaoBivariadaQuantiQuanti(NomeE, NomeD);
              }
          }
      }
    }


//*************************************************************************
**//
    //                                INFERENCIA
//

//*************************************************************************
**//


//*************************************************************************
*******************
    //  Verificar os tipos de retornos e os tipos de mensagens que serao
dadas!!!
    //  * 0-> Nao pode realizar a analise
    //  * 1-> Passou certo.

//*************************************************************************
********************

    public int InferenciaSignificancia(){
        mensVar[0] = null;
        mensVar[1] = null;
        tipoVar[0] = null;
        tipoVar[1] = null;
        nCat[0] = 2;
        nCat[1] = 2;
        inicializaObjetoAnalise();
        return ConsulteGerenciadorUsandoObjetoAnalise();
    }
```

```
//*************************************************************************
********************
    //  Verificar os tipos de retornos e os tipos de mensagens que serao
dadas!!!
    //  * 0-> Nao pode realizar a analise
    //  * 1-> Passou certo.

//*************************************************************************
********************

    public int InferenciaNomesVariaveis(){
       minhaAnalise.NomeVI = NomeE;
       minhaAnalise.NomeVD = NomeD;
       return ConsulteGerenciadorUsandoObjetoAnalise();
    }


//*************************************************************************
********************
    //  Verificar os tipos de retornos e os tipos de mensagens que serao
dadas!!!
    //  * 0-> Nao pode realizar a analise
    //  * 1-> Passou certo.
    //  * 2-> 'Atenção: O SEstat diz que a variável '+Nome+' não é
'+auxstr+'. Deseja continuar assim mesmo?
    //  * 3->  'Atenção: essa variável não pode ser tratada como '+auxstr+'!'

//*************************************************************************
********************

    public int InferenciaTipo(boolean __prosseguir1, boolean __prosseguir2){
           if(tipoE.equals("quanti")){
                int ret =
DeixaSeguirDepoisDeEscolherTipo(minhaAnalise.NomeVI,tipoE);
                if((ret != 1)&&(__prosseguir1 == false)){
                    return ret;
                }else{
                  minhaAnalise.TipoVI = "quanti";
                }
                if(tipoD.equals("quanti")){
                     ret =
DeixaSeguirDepoisDeEscolherTipo(minhaAnalise.NomeVD,tipoD);
                     if((ret != 1)&&(__prosseguir2 == false)){
                        return ret;
                     }else{
                       minhaAnalise.TipoVD = "quanti";
                     }
                }else{
                     ret =
DeixaSeguirDepoisDeEscolherTipo(minhaAnalise.NomeVD,"quali");
                     if((ret != 1)&&(__prosseguir2 == false)){
                         return ret;
                     }else{
                       minhaAnalise.TipoVD = "quali";
                     }
                }
           }
           if(tipoE.equals("quali")){
                int ret =
DeixaSeguirDepoisDeEscolherTipo(minhaAnalise.NomeVI,"quali");
```

```
                if((ret != 1)&&(__prosseguir1 == false)){
                    return ret;
                }else{
                   minhaAnalise.TipoVI = "quali";
                }
                if(tipoD.equals("quanti")){
                     ret =
DeixaSeguirDepoisDeEscolherTipo(minhaAnalise.NomeVD,"quanti");
                     if((ret != 1)&&(__prosseguir2 == false)){
                       return ret;
                     }else{
                       minhaAnalise.TipoVD = "quanti";
                     }
                }else{
                     ret =
DeixaSeguirDepoisDeEscolherTipo(minhaAnalise.NomeVD,"quali");
                     if((ret != 1)&&(__prosseguir2 == false)){
                       return ret;
                     }else{
                       minhaAnalise.TipoVD = "quali";
                     }
                }
           }
      return ConsulteGerenciadorUsandoObjetoAnalise();
    }

//**************************************************************************
********************
    //  Verificar os tipos de retornos e os tipos de mensagens que serao
dadas!!!
    //  * 0-> Nao pode realizar a analise
    //  * 1-> Passou certo.
    //  * 2->  'Atenção: O SEstat diz que a variável '+Nome+' não é
'+auxstr+'. Deseja continuar assim mesmo?
    //    8 ->  Erro ao calcular frequencia esperada

//**************************************************************************
********************
    public  int InferenciaMensuracao(boolean __prosseguir1,boolean
__prosseguir2 ){
          int ret =
DeixaSeguirDepoisDeEscolherMensuracao(minhaAnalise.NomeVI,mensE);
          if((ret != 1)&&(__prosseguir1 == false)){
              return ret;
          }else{
              minhaAnalise.MensuracaoVI = mensE;
              ret =
DeixaSeguirDepoisDeEscolherMensuracao(minhaAnalise.NomeVD,mensD);
              if((ret != 1)&&(__prosseguir2 == false)){
                return ret;
              }else{
                  minhaAnalise.MensuracaoVD = mensD;
              }
          }
          return ConsulteGerenciadorUsandoObjetoAnalise();
    }

//**************************************************************************
********************
    //  PARA ENTRAR COM UM VALOR NÃO EXISTENTE, ENTRA-SE COM O VALOR 0
```

```
    //  Verificar os tipos de retornos e os tipos de mensagens que serao
dadas!!!
    //   0-> Nao pode realizar a analise
    //   1-> Passou certo.
    //   2-> 'Atenção: O SEstat diz que a variável '+Nome+' não é '+auxstr+'.
Deseja continuar assim mesmo?
    //   3-> Erro ao executar teste de Brown-Forsythe
    //   4-> Erro ao executar teste de Levene

//**************************************************************************
*********************

    public int InferenciaNumeroCategorias(boolean __prosseguir, boolean
__prosseguir2){
        if(catE != 0){
            int ret =
DeixaSeguirDepoisDeEscolherNumCategorias(minhaAnalise.NomeVI,catE);
            if((ret != 1)&&(__prosseguir == false)){
                return ret;
            }else{
               minhaAnalise.nCategoriasVI = catE;
            }
        }
        if(catD != 0){
            int ret =
DeixaSeguirDepoisDeEscolherNumCategorias(minhaAnalise.NomeVD,catD);
            if((ret != 1)&&(__prosseguir2 == false)){
               return ret;
            }else{
               minhaAnalise.nCategoriasVD = catD;
            }
        }
        return ConsulteGerenciadorUsandoObjetoAnalise();

    }

//**************************************************************************
*********************
    //  Verificar os tipos de retornos e os tipos de mensagens que serao
dadas!!!
    //  * 0-> Nao pode realizar a analise
    //  * 1-> Passou certo.
    //    5-> Erro ao achar Intervalos
    //    6-> Erro ao achar valores esperados
    //    7-> Erro ao executar teste Qui-Quadrado Normal
    //  * 10 -> Erro no ExecutaMetodo()

//**************************************************************************
*********************

    public int InferenciaParidade(){
        minhaAnalise.Paridade = paridade;
        return ConsulteGerenciadorUsandoObjetoAnalise();
    }

//**************************************************************************
*********************
    //  Verificar os tipos de retornos e os tipos de mensagens que serao
dadas!!!
    //  * 0-> Nao pode realizar a analise
    //  * 1-> Passou certo.
```

```
    //  * 10 -> Erro no ExecutaMetodo()

//*************************************************************************
*********************

    public int InferenciaNormalidade(){
        minhaAnalise.Normalidade = normalidade;
        return ConsulteGerenciadorUsandoObjetoAnalise();
    }


//*************************************************************************
********************
    //  Verificar os tipos de retornos e os tipos de mensagens que serao
dadas!!!
    //  * 0-> Nao pode realizar a analise
    //  * 1-> Passou certo.
    //    5-> Erro ao achar Intervalos
    //    6-> Erro ao achar valores esperados
    //    7-> Erro ao executar teste Qui-Quadrado Normal
    //  * 10 -> Erro no ExecutaMetodo()

//*************************************************************************
********************
    public int InferenciaHomocedasticidade(){
        minhaAnalise.Homocedasticidade = homocedasticidade;
        return ConsulteGerenciadorUsandoObjetoAnalise();
    }

//*************************************************************************
********************
    //  Verificar os tipos de retornos e os tipos de mensagens que serao
dadas!!!
    //  * 0-> Nao pode realizar a analise
    //  * 1-> Passou certo. (Ou o método nao será executado)
    //    5-> Erro ao achar Intervalos
    //    6-> Erro ao achar valores esperados
    //    7-> Erro ao executar teste Qui-Quadrado Normal
    //  * 10 -> Erro no ExecutaMetodo()

//*************************************************************************
*********************

    public int InferenciaFrequenciaEsperada(){
        minhaAnalise.FreqEsperadasMaiores5 = frequenciaEsperada;
        return ConsulteGerenciadorUsandoObjetoAnalise();
    }

    public boolean getValorCalculoHomocedasticidadeLevene(Objeto __nomeE,
Objeto __nomeD, Objeto __GL1, Objeto __GL2, Objeto __valorTeste){
        __nomeE.setString(NomeE);
        __nomeD.setString(NomeD);
        __GL1.setDouble(GL1L);
        __GL2.setDouble(GL2L);
        __valorTeste.setDouble(ValorTesteL);
        return true;
    }

    public boolean getValorCalculoHomocedasticidadeBrownForsythe(Objeto
__nomeE, Objeto __nomeD, Objeto __GL1, Objeto __GL2, Objeto __valorTeste){
        __nomeE.setString(NomeE);
```

```
        __nomeD.setString(NomeD);
        __GL1.setDouble(GL1BF);
        __GL2.setDouble(GL2BF);
        __valorTeste.setDouble(ValorTesteBF);
        return true;
    }

    public boolean getValoresNormalidade(Objeto __nomeQuanti, Objeto
__nomeQuali, Objeto __valoresQuanti, ObjetoIntervalo __intervalos, Objeto
__catQuali, Objeto __ValoreL, Objeto __PL, Objeto __GL){
        __nomeQuanti.setString(NomeQuanti);
        __nomeQuali.setString(NomeQuali);
        __valoresQuanti.__string = valoresQuanti;
        __intervalos.intervalo = intervalos;
        __catQuali.setDouble(CatQuali);
        __ValoreL.setDouble(ValorL);
        __PL.setDouble(PL);
        __GL.setDouble(GL);
        return true;
    }

    public boolean getFrequenciaEsperada(ObjetoRefVariaveisQualitativas
__vQuali, ObjetoRefVariaveisQualitativas __vQuali2, Objeto __FObservadas,
Objeto __FEsperadas){
        __vQuali.variaveisQualitativas = vQuali;
        __vQuali2.variaveisQualitativas = vQuali2;
        __FObservadas.___string = fObservadas;
        __FEsperadas.___string = fEsperadas;
        return true;
    }

    public boolean getValoresKruskalWallis(Objeto __p, Objeto __GL, Objeto
__H, Objeto __Somas, Objeto __Ns){
       __p.setDouble(KruskalWallisP);
       __GL.setLong(KruskalWallisGL);
       __Somas.__double = KruskalWallisSomas;
       __Ns.__long = KruskalWallisNs;
       __H.setDouble(KruskalWallisH);
       return true;
    }

    public boolean getValoresWilcoxon(Objeto __SomaMinima, Objeto __P, Objeto
__Z){
        __SomaMinima.setDouble(WilcoxonSomaMinima);
        __P.setDouble(WilcoxonP);
        __Z.setDouble(WilcoxonZ);
        return true;
    }

    public boolean getSpearman(Objeto __SpearmanT, Objeto __SpearmanP, Objeto
__SpearmanRs){
      __SpearmanT.setDouble(SpearmanT);
      __SpearmanP.setDouble(SpearmanP);
      __SpearmanRs.setDouble(SpearmanRs);
      return true;
    }

    public boolean getPearson(Objeto __Pearson){
        __Pearson.setDouble(PearsonR);
        return true;
    }
```

```
    public boolean getTesteTPares(Objeto __TesteTParesT, Objeto
__TesteTParesGL, Objeto __TesteTParesP){
      __TesteTParesT.setDouble(TesteTParesT);
      __TesteTParesGL.setDouble(TesteTParesGL);
      __TesteTParesP.setDouble(TesteTParesP);
      return true;
    }

    public boolean getTesteT(Objeto __TesteTT,Objeto __TesteTGL,Objeto
__TesteTP){
      __TesteTT.setDouble(TesteTT);
      __TesteTGL.setDouble(TesteTGL);
      __TesteTP.setDouble(TesteTP);
      return true;
    }

    public boolean getMannWhitney(Objeto __MannWhitneySoma1,Objeto
__MannWhitneySoma2,Objeto __MannWhitneyn1,Objeto __MannWhitneyn2,Objeto
__MannWhitneyZ,Objeto __MannWhitneyP){
       __MannWhitneySoma1.setDouble(MannWhitneySoma1);
       __MannWhitneySoma2.setDouble(MannWhitneySoma2);
       __MannWhitneyn1.setDouble(MannWhitneyn1);
       __MannWhitneyn2.setDouble(MannWhitneyn2);
       __MannWhitneyZ.setDouble(MannWhitneyZ);
       __MannWhitneyP.setDouble(MannWhitneyP);
       return true;
    }

    public boolean getTesteTSeparate(Objeto __TesteTSeparateT, Objeto
__TesteTSeparateGL,Objeto __TesteTSeparateP){
       __TesteTSeparateT.setDouble(TesteTSeparateT);
       __TesteTSeparateGL.setDouble(TesteTSeparateGL);
       __TesteTSeparateP.setDouble(TesteTSeparateP);
       return true;
    }

    public boolean getAnova(Objeto __AnovaF, Objeto __AnovaP,Objeto
__AnovaGLT,Objeto __AnovaGLE,Objeto __AnovaSQT,Objeto __AnovaSQE,Objeto
__AnovaQMT,Objeto __AnovaQME){
      __AnovaF.setDouble(AnovaF);
      __AnovaP.setDouble(AnovaP);
      __AnovaGLT.setDouble(AnovaGLT);
      __AnovaGLE.setDouble(AnovaGLE);
      __AnovaSQT.setDouble(AnovaSQT);
      __AnovaSQE.setDouble(AnovaSQE);
      __AnovaQMT.setDouble(AnovaQMT);
      __AnovaQME.setDouble(AnovaQME);
      return true;
    }
    //Para Anova

    public boolean getQuiquadrado(Objeto __pValor, Objeto __gl, Objeto
__valorTeste) {
      NumberFormat _nf =  NumberFormat.getInstance();
      _nf.setMaximumFractionDigits(3);
      __pValor.setString(_nf.format(QuiquadradoPValor));
      __gl.setString(_nf.format(QuiQuadradoGL));
      __valorTeste.setString(_nf.format(QuiQuadradoValorTeste));
      return true;
    }
```

```
    public String[][] getTabelaFrequencias(){
      return TabelaFrequencias;
    }

    public String[][] getTabelaDescritivas(){
       return TabelaDescritivas;
    }


//**************************************************************************
**//
    //                               FUNCOES INTERNAS
//

//**************************************************************************
**//
    private boolean educadorReconheceBase(){
          boolean ret,PSQ;
          Variavel auxVariavel;
          String tiporec;
          ObjetoRefVariaveisQualitativas vquali = new
ObjetoRefVariaveisQualitativas();
          ObjetoRefVariaveisQuantitativas vquanti = new
ObjetoRefVariaveisQuantitativas();
          educador.ZeraHistoricoIndividual();
          educador.ZeraHistorico2a2();
          auxVariavel = new Variavel();
          int numeroVariaveis = leitorDBF.RetornaLabelColunas().length;
          TTipificacao Obj_TTipificacao = new TTipificacao();
          for(int i=0; i< numeroVariaveis; i++){

leitorDBF.FornecaVariavelQualitativa(leitorDBF.FornecaNomeVariavel(i),vquali)
;
             auxVariavel.setNome(vquali.variaveisQualitativas.getNome());

auxVariavel.setMissing(vquali.variaveisQualitativas.getMissing());
             auxVariavel.setN(vquali.variaveisQualitativas.getN());
             auxVariavel.setValores(vquali.variaveisQualitativas.valores);
             if(auxVariavel==null){
               return false;
             }
             if(auxVariavel.getValores()==null){
               return false;
             }

Obj_TTipificacao.executar(auxVariavel.getValores(),leitorDBF.RetornaNumeroVal
ores());
             if("quali".equals(Obj_TTipificacao.getTipo())){
               tiporec = "quali";
             }else{
               if("quanti".equals(Obj_TTipificacao.getTipo())){
                 tiporec = "quanti";
               }else{
                 tiporec = "naosabe";
               }
             }
```

```
leitorDBF.FornecaVariavelQuantitativa(leitorDBF.FornecaNomeVariavel(i),vquant
i);
             if(vquanti==null){
                PSQ = false;
             }else{
                PSQ = true;
             }
             ret =
educador.InsereNoHistoricoIndividual(auxVariavel.getNome(),PSQ,tiporec,"nenhu
ma",vquali.variaveisQualitativas.getNumCategorias(),-1);
             if(ret==false){
                auxVariavel = null;
                return false;
             }
             for(int j=0; j< numeroVariaveis; j++){
               if(j!=i){

leitorDBF.FornecaVariavelQualitativa(leitorDBF.FornecaNomeVariavel(j),vquali)
;
                 ret =
educador.InsereNoHistorico2a2(auxVariavel.getNome(),vquali.variaveisQualitati
vas.getNome(),naosei,naosei,-1,-1);
                 if(ret==false){
                    auxVariavel = null;
                    return false;
                 }
               }
             }
          }
          auxVariavel = null;
          return true;
        }

    private int DeixaSeguirDepoisDeEscolherTipo(String __nome, String
__tipoVar){
      if(__tipoVar.equals("naosabe")){
         return 0; //Nao pode realizar a analise
      }
      if(educador.ArmazenaTipo(__nome,__tipoVar)==false){
         return 0; //Nao pode realizar a analise
      }
      Objeto Corr = new Objeto(false);
      Objeto Perm = new Objeto(false);
      if(educador.ConsultaTipo(__nome,__tipoVar,Corr,Perm)==false){
         return 0; //Nao pode realizar a analise
      }
      if(!(Corr.booleanValue())){
         if(Perm.booleanValue()){
             return 2; //'Atenção: O SEstat diz que a variável '+Nome+' não é
'+auxstr+'. Deseja continuar assim mesmo?
         }else{
             return 3; //'Atenção: essa variável não pode ser tratada como
'+auxstr+'!'
         }
      }
      return 1; //Passou certo.
    }

    private int DeixaSeguirDepoisDeEscolherMensuracao(String __nome, String
__mensuracaoEscolhida){
```

```
      Objeto Corr = new Objeto(false);
      if(__mensuracaoEscolhida.equals("nenhuma")){
         return 0; //Nao pode realizar a analise;
      }
      if(educador.ArmazenaMensuracao(__nome,__mensuracaoEscolhida)==false){
         return 0; //Nao pode realizar a analise;
      }

if(educador.ConsultaMensuracao(__nome,__mensuracaoEscolhida,Corr)==false){
         return 0; //Nao pode realizar a analise;
      }
      if(!(Corr.booleanValue())){
         return 2; //'Atenção: O SEstat diz que a variável '+Nome+' não é
'+auxstr+'. Deseja continuar assim mesmo?',
      }
      return 1; //Passou certo!!!
    }

    private int DeixaSeguirDepoisDeEscolherNumCategorias(String __Nome, long
numCatEscolhida){
      Objeto Corr = new Objeto(false);
      if(educador.ArmazenaNumCategorias(__Nome,numCatEscolhida)==false){
         return 0; //Nao pode realizar a analise;
      }
      if(educador.ConsultaNumCategorias(__Nome,numCatEscolhida,Corr)==false){
         return 0;
      }
      if(!(Corr.booleanValue())){
           return 2; //Atenção: O SEstat diz que a variável '+Nome+' não
possui '+auxstr+' categorias. Deseja continuar assim mesmo?',
      }
      return 1;
    }

    public void DescricaoUnivariadaQualiOUQuantiDisc(String __nome, boolean
__quali){
      long nAcum,total,Tamanho;
      double pAcum;
      Objeto VMediana = new Objeto(0);
      Objeto VQ75 = new Objeto(0);
      Objeto VQ25 = new Objeto(0);
      Objeto lixon = new Objeto(0);
      Objeto VMin = new Objeto(0);
      Objeto VMax = new Objeto(0);
      Objeto VMedia = new Objeto(0);
      Objeto VDP = new Objeto(0);
      ObjetoRefVariaveisQualitativas Vquali;
      double[] auxMediana;
      ObjetoIntervalo Intervalos = new ObjetoIntervalo();
      total = 0;
      Vquali = new ObjetoRefVariaveisQualitativas();
      leitorDBF.FornecaVariavelQualitativa(__nome,Vquali);

estatistica.AcheIntervalos(Vquali.variaveisQualitativas.getValores(),Interval
os,lixon,VMin,VMax,VMedia,VDP);
      // Prepara o gráfico
      if(!(__quali)){
           String Title = "Histograma: " + __nome;
           int[] valores = new
int[Vquali.variaveisQualitativas.getCategorias().length];
```

```
            String[] labels = new
String[Vquali.variaveisQualitativas.getCategorias().length];
            for(int i = 0; i<=
Vquali.variaveisQualitativas.getCategorias().length-1; i++){
                  total = total +
Vquali.variaveisQualitativas.getCategorias()[i].getQuantidade();
                  valores[i] =
(int)Vquali.variaveisQualitativas.getCategorias()[i].getQuantidade();
                  labels[i] = "[" +
Vquali.variaveisQualitativas.getCategorias()[i].getValor() + "]";
            }
            grafico.geraGraficoBarra3DSingle(Title, valores, labels);
      }else{
            String Title = "Grafico de Setores: " + __nome;
            int[] valores = new
int[Vquali.variaveisQualitativas.getCategorias().length];
            String[] labels = new
String[Vquali.variaveisQualitativas.getCategorias().length];
            for(int i=0; i<=
Vquali.variaveisQualitativas.getCategorias().length-1; i++){
                  total =
total+Vquali.variaveisQualitativas.getCategorias()[i].getQuantidade();
                  valores[i] =
(int)Vquali.variaveisQualitativas.getCategorias()[i].getQuantidade();
                  labels[i] = "[" +
Vquali.variaveisQualitativas.getCategorias()[i].getValor() + "]";
            }
            grafico.setPieCharts(Title, labels, valores);
      }
      // Termina de preparar o gráfico
      // Começa a preparar a tabela
      NumberFormat _nf =  NumberFormat.getInstance();
      _nf.setMaximumFractionDigits(3);
      TabelaFrequencias = new
String[5][Vquali.variaveisQualitativas.getCategorias().length+1];
      TabelaFrequencias[0][0] = "Categoria";
      TabelaFrequencias[1][0] = "    n    ";
      TabelaFrequencias[2][0] = "    %    ";
      TabelaFrequencias[3][0] = " n Acum. ";
      TabelaFrequencias[4][0] = " % Acum. ";
      nAcum = 0;
      pAcum = 0;
      for(int i=0;
i<Vquali.variaveisQualitativas.getCategorias().length;i++){
         TabelaFrequencias[0][i+1] =
Vquali.variaveisQualitativas.getCategorias()[i].getValor();
         TabelaFrequencias[1][i+1] =
String.valueOf(Vquali.variaveisQualitativas.getCategorias()[i].getQuantidade(
));
         nAcum =
nAcum+Vquali.variaveisQualitativas.getCategorias()[i].getQuantidade();
         pAcum =
pAcum+((double)(Vquali.variaveisQualitativas.getCategorias()[i].getQuantidade
()*100)/total);
         TabelaFrequencias[3][i+1] = String.valueOf(nAcum);
         TabelaFrequencias[2][i+1] =
_nf.format((double)(Vquali.variaveisQualitativas.getCategorias()[i].getQuanti
dade()*100)/total);
         TabelaFrequencias[4][i+1] = _nf.format(pAcum);
      }
      // Termina de preparar a tabela
```

```
      Tamanho = Vquali.variaveisQualitativas.getValores().length;
      auxMediana = new double[(int)Tamanho];
      for(int i=0; i< Tamanho; i++){
        if(!"".equals(Vquali.variaveisQualitativas.getValores()[i])){
          auxMediana[i] =
Double.valueOf(Vquali.variaveisQualitativas.getValores()[i]).doubleValue();
        }
      }
      estatistica.CalculeQ25(auxMediana,VQ25);
      estatistica.CalculeMediana(auxMediana,VMediana);
      estatistica.CalculeQ75(auxMediana,VQ75);
      //comeca a preparara a tabela de Descricao
      //Fixa Labels
      TabelaDescritivas = new String[9][2];
      TabelaDescritivas[0][1] = Vquali.variaveisQualitativas.getNome();
      TabelaDescritivas[0][0] = "Variável";
      TabelaDescritivas[1][0] = "    N    ";
      TabelaDescritivas[2][0] = "  Media  ";
      TabelaDescritivas[3][0] = " Desvio  ";
      TabelaDescritivas[4][0] = " Minimo  ";
      TabelaDescritivas[5][0] = "   Q1    ";
      TabelaDescritivas[6][0] = " Mediana ";
      TabelaDescritivas[7][0] = "   Q3    ";
      TabelaDescritivas[8][0] = " Maximo  ";
      //Fixa Valores
      TabelaDescritivas[1][1] = String.valueOf(nAcum);
      TabelaDescritivas[2][1] = _nf.format(VMedia.doubleValue());
      TabelaDescritivas[3][1] = _nf.format(VDP.doubleValue());
      TabelaDescritivas[4][1] = _nf.format(VMin.doubleValue());
      TabelaDescritivas[5][1] = _nf.format(VQ25.doubleValue());
      TabelaDescritivas[6][1] = _nf.format(VMediana.doubleValue());
      TabelaDescritivas[7][1] = _nf.format(VQ75.doubleValue());
      TabelaDescritivas[8][1] = _nf.format(VMax.doubleValue());
      //Termina de preparar a tabela
    }

    private int DescricaoUnivariadaQuantiCont(String __nome){
      long nAcum,total,tamanho;
      double pAcum;
      double[] auxMediana;
      ObjetoIntervalo Intervalos = new ObjetoIntervalo();
      Objeto VMediana = new Objeto(0);
      Objeto VQ25 = new Objeto(0);
      Objeto VQ75 = new Objeto(0);
      Objeto lixon = new Objeto(0);
      Objeto VMin = new Objeto(0);
      Objeto VMax = new Objeto(0);
      Objeto VMedia = new Objeto(0);
      Objeto VDP = new Objeto(0);
      ObjetoRefVariaveisQuantitativas VQuanti = new
ObjetoRefVariaveisQuantitativas();
      leitorDBF.FornecaVariavelQuantitativa(__nome,VQuanti);
      if(VQuanti.variaveisQuantitativas==null){
         return 0; //Nao foi possivel completar a analise.
      }

estatistica.AcheIntervalos(VQuanti.variaveisQuantitativas.getValores(),Interv
alos,lixon,VMin,VMax,VMedia,VDP);
      total = 0;
      //USADO PRO GRÁFICO
      String Title = "Histograma: " + __nome;
```

```
      int[] valores = new int[Intervalos.intervalo.length];
      String[] labels = new String[Intervalos.intervalo.length];
      for(int i=0; i< Intervalos.intervalo.length;i++){
          valores[i] = (int)Intervalos.intervalo[i].getQtdade();
          labels[i] =  "[" + String.valueOf(Intervalos.intervalo[i].getMax())
+ "-" + String.valueOf(Intervalos.intervalo[i].getMax()) + "]";
          total = total+Intervalos.intervalo[i].getQtdade();
      }
      // Prepara o gráfico
      grafico.geraGraficoBarra3DSingle(Title, valores,labels);
      // Termina de preparar o gráfico  */

      // Começa a preparar a tabela de freqeuncias
      NumberFormat _nf =  NumberFormat.getInstance();
      _nf.setMaximumFractionDigits(3);
      TabelaFrequencias = new String[5][Intervalos.intervalo.length+1];
      TabelaFrequencias[0][0] =" Faixa   ";
      TabelaFrequencias[1][0] ="    n    ";
      TabelaFrequencias[2][0] ="    %    ";
      TabelaFrequencias[3][0] =" n Acum. ";
      TabelaFrequencias[4][0] =" % Acum. ";
      nAcum = 0;
      pAcum = 0;
      for(int i=0; i< Intervalos.intervalo.length;i++){
          TabelaFrequencias[0][i+1] = "[" +
String.valueOf(Intervalos.intervalo[i].getMin()) + " - " +
String.valueOf(Intervalos.intervalo[i].getMax()) + "]" ;
          TabelaFrequencias[1][i+1] =
String.valueOf(Intervalos.intervalo[i].getQtdade());
          nAcum = nAcum+Intervalos.intervalo[i].getQtdade();
          pAcum += ((double)(Intervalos.intervalo[i].getQtdade()*100) /
total);
          TabelaFrequencias[3][i+1] = String.valueOf(nAcum);
          TabelaFrequencias[2][i+1] =
_nf.format((double)(Intervalos.intervalo[i].getQtdade()*100)/total);
          TabelaFrequencias[4][i+1] = _nf.format(pAcum);
      }
      // Termina de preparar a tabela
      tamanho = VQuanti.variaveisQuantitativas.getValores().length;
      auxMediana = new double[(int)tamanho];
      for(int i=0; i< tamanho; i++){
        if(!"".equals(VQuanti.variaveisQuantitativas.getValores()[i])){
          auxMediana[i] =
Float.valueOf(VQuanti.variaveisQuantitativas.getValores()[i]).floatValue();
        }
      }
      estatistica.CalculeQ25(auxMediana,VQ25);
      estatistica.CalculeMediana(auxMediana,VMediana);
      estatistica.CalculeQ75(auxMediana,VQ75);
      //comeca a preparara a tabela de Descricao
      TabelaDescritivas = new String[9][2];
      //Fixa Labels
      TabelaDescritivas[0][1] = VQuanti.variaveisQuantitativas.getNome();
      TabelaDescritivas[0][0] = "Variável";
      TabelaDescritivas[1][0] ="    N    ";
      TabelaDescritivas[2][0] ="  Media  ";
      TabelaDescritivas[3][0] =" Desvio  ";
      TabelaDescritivas[4][0] =" Minimo  ";
      TabelaDescritivas[5][0] ="   Q1    ";
      TabelaDescritivas[6][0] =" Mediana ";
      TabelaDescritivas[7][0] ="   Q3    ";
```

```
      TabelaDescritivas[8][0] =" Maximo  ";
      //Fixa Valores
      TabelaDescritivas[1][1] = String.valueOf(nAcum);
      TabelaDescritivas[2][1] = _nf.format(VMedia.doubleValue());
      TabelaDescritivas[3][1] = _nf.format(VDP.doubleValue());
      TabelaDescritivas[4][1] = _nf.format(VMin.doubleValue());
      TabelaDescritivas[5][1] = _nf.format(VQ25.doubleValue());
      TabelaDescritivas[6][1] = _nf.format(VMediana.doubleValue());
      TabelaDescritivas[7][1] = _nf.format(VQ75.doubleValue());
      TabelaDescritivas[8][1] = _nf.format(VMax.doubleValue());
      //Termina de preparar a tabela
      return 1; //Passou certo!!!
    }

    private int DescricaoBivariadaQualiQuali(String __nomeE, String __nomeD){
      long total,TotalN,ultimoVI,ultimoVD;
      double TotalP;
      long[] TotalCategorias;
      double[] TotalCategoriasP;
      ObjetoRefVariaveisQualitativas qualiI = new
ObjetoRefVariaveisQualitativas();
      ObjetoRefVariaveisQualitativas qualiD = new
ObjetoRefVariaveisQualitativas();
      if(leitorDBF.FornecaVariavelQualitativa(__nomeE,qualiI)==false){
         return 6;
      }
      if(leitorDBF.FornecaVariavelQualitativa(__nomeD,qualiD)==false){
         return 6;
      }
      ObjetoRefCategoriaPorCategoria CategoriasPorCategorias = new
ObjetoRefCategoriaPorCategoria();

if(estatistica.AcheCategoriasPorCategorias(qualiI.variaveisQualitativas,quali
D.variaveisQualitativas,CategoriasPorCategorias) != 0){
         return 6;
      }
      ultimoVI = CategoriasPorCategorias.categoriaPorCategoria.length;
      ultimoVD =
CategoriasPorCategorias.categoriaPorCategoria[0].Categorias.length;
      String NomeVI = __nomeE;
      String NomeVD = __nomeD;
      // Prepara o Gráfico
      String Title;
      int[][] valores = new int[(int)ultimoVI][(int)ultimoVD];
      String[] labels  = new String[(int)ultimoVD];
      String[] categ = new String[(int)ultimoVI];
      if((tipoE.equals("quali")) && (tipoD.equals("quali"))){
        Title = "Diagrama em Colunas: " + NomeVI + "=" +
CategoriasPorCategorias.categoriaPorCategoria[0].getValor() + " X " + NomeVD;
        for(int k=0; k<= ultimoVI-1; k++){
          for(int i=0; i<= ultimoVD-1; i++){
              valores[k][i] =
(int)CategoriasPorCategorias.categoriaPorCategoria[k].Categorias[i].getQuanti
dade();
              labels[i] = "[" +
CategoriasPorCategorias.categoriaPorCategoria[k].Categorias[i].getValor() +
"]";
          }
          categ[k] = NomeE + "=" +
qualiI.variaveisQualitativas.getCategorias()[k].getValor();
        }
```

```
        grafico.geraGraficoBarra3D(Title, valores, labels, categ);
      }else{
        Title = "Diagrama em Setores: " + NomeVI + "=" +
CategoriasPorCategorias.categoriaPorCategoria[0].getValor() + " X " + NomeVD;
        int[] valoresPie = new int[(int)ultimoVD];
        labels  = new String[(int)ultimoVD];
        for(int i=0; i<= ultimoVD-1; i++){
            valoresPie[i] =
(int)CategoriasPorCategorias.categoriaPorCategoria[0].Categorias[i].getQuanti
dade();
            labels[i] = "[" +
CategoriasPorCategorias.categoriaPorCategoria[0].Categorias[i].getValor() +
"]";
        }
        grafico.setPieCharts(Title, labels, valoresPie);
      }
      // Termina o Gráfico

      // Prepara a Tabela de Frequencias
      NumberFormat _nf =  NumberFormat.getInstance();
      _nf.setMaximumFractionDigits(3);
      TabelaFrequencias = new String[(int)ultimoVI + 2][(int)ultimoVD*2 + 3];
      TabelaFrequencias[0][0] = "Categoria";
      for(int i=1; i<= ultimoVI; i++){
          TabelaFrequencias[i][0] = NomeVI + "=" +
CategoriasPorCategorias.categoriaPorCategoria[i-1].getValor();
      }
      int j =0;
      int k = 0;
      for(int i=1; i<= ultimoVD; i++){
            j =j+2;
            k =k+2;
            TabelaFrequencias[0][j-1] = NomeVD + "=" +
CategoriasPorCategorias.categoriaPorCategoria[0].Categorias[i-1].getValor();
            TabelaFrequencias[0][k] = "% do Total";
      }
      TabelaFrequencias[0][(int)ultimoVD*2+1] = "Todas";
      TabelaFrequencias[0][(int)ultimoVD*2+2] = "% do Total";
      TabelaFrequencias[(int)ultimoVI+1][0] = "Total";
      total = 0;
      for(int i=0; i<=ultimoVI-1; i++){
          for(j=0; j<=ultimoVD-1;j++){
              total = total +
CategoriasPorCategorias.categoriaPorCategoria[i].Categorias[j].getQuantidade(
);
          }
      }
      k = 0;
      int l = 0;
      TotalCategorias = new long[(int)ultimoVI];
      TotalCategoriasP = new double[(int)ultimoVI];
      for(int i=1; i<= ultimoVD; i++){ // Para cada linha
            TotalN =0;
            TotalP =0;
            k =k+2;
            l =l+2;
            for(j=1; j<=ultimoVI;j++){ // Para cada coluna
                  TabelaFrequencias[j][k-1] =
_nf.format(CategoriasPorCategorias.categoriaPorCategoria[j-1].Categorias[i-
1].getQuantidade());
```

```
                  TabelaFrequencias[j][l] =
_nf.format(CategoriasPorCategorias.categoriaPorCategoria[j-1].Categorias[i-
1].getQuantidade()*100/total);
                  TotalN = TotalN +
CategoriasPorCategorias.categoriaPorCategoria[j-1].Categorias[i-
1].getQuantidade();
                  TotalP = TotalP +
(CategoriasPorCategorias.categoriaPorCategoria[j-1].Categorias[i-
1].getQuantidade()*100/total);
                  TotalCategorias[j-1] = TotalCategorias[j-1] +
CategoriasPorCategorias.categoriaPorCategoria[j-1].Categorias[i-
1].getQuantidade();
                  TotalCategoriasP[j-1] = TotalCategoriasP[j-1] +
(double)(CategoriasPorCategorias.categoriaPorCategoria[j-1].Categorias[i-
1].getQuantidade()*100)/(double)total;
            }
            TabelaFrequencias[(int)ultimoVI+1][k-1] = _nf.format(TotalN);
            TabelaFrequencias[(int)ultimoVI+1][l] = _nf.format(TotalP);
      }
      for(int i=1; i<= ultimoVI; i++){
            TabelaFrequencias[i][(int)ultimoVD*2+1] =
_nf.format(TotalCategorias[i-1]);
            TabelaFrequencias[i][(int)ultimoVD*2+2] =
_nf.format(TotalCategoriasP[i-1]);
      }
      TabelaFrequencias[(int)ultimoVI+1][(int)ultimoVD*2+1] =
String.valueOf(total);
      TabelaFrequencias[(int)ultimoVI+1][(int)ultimoVD*2+2] = "100";
      return 4;
    }

    public int DescricaoBivariadaQuantiQuanti(String __nomeE, String
__nomeD){
      long ultimoVI,tamanho,ultimoVD,TotalVI,TotalVD;
      double[] auxMediana,auxMedianaD;
      Objeto lixo = new Objeto(0);
      Objeto MinimoVI = new Objeto(0);
      Objeto MaximoVI = new Objeto(0);
      Objeto MediaVI = new Objeto(0);
      Objeto DPVI = new Objeto(0);
      Objeto MinimoVD = new Objeto(0);
      Objeto MaximoVD = new Objeto(0);
      Objeto MediaVD = new Objeto(0);
      Objeto DPVD = new Objeto(0);
      Objeto VMediana = new Objeto(0);
      Objeto VQ25 = new Objeto(0);
      Objeto VQ75 = new Objeto(0);
      Objeto VMedianaD = new Objeto(0);
      Objeto VQ25D = new Objeto(0);
      Objeto VQ75D = new Objeto(0);
      ObjetoRefVariaveisQuantitativas quantiE = new
ObjetoRefVariaveisQuantitativas();
      ObjetoRefVariaveisQuantitativas quantiD = new
ObjetoRefVariaveisQuantitativas();
      ObjetoIntervalo Intervalos = new ObjetoIntervalo();
      ObjetoIntervalo IntervalosVD = new ObjetoIntervalo();
      int NCategorias;
      TotalVI =0;
      TotalVD = 0;
      String NomeVI = __nomeE;
      String NomeVD = __nomeD;
```

```
      if(leitorDBF.FornecaVariavelQuantitativa(NomeVI,quantiE)==false){
         return 6;
      }
      if(leitorDBF.FornecaVariavelQuantitativa(NomeVD,quantiD)==false){
         return 6;
      }

if(estatistica.AcheIntervalos(quantiE.variaveisQuantitativas.valores,Interval
os,lixo,MinimoVI,MaximoVI,MediaVI,DPVI) != 0){
         return 6;
      }

if(estatistica.AcheIntervalos(quantiD.variaveisQuantitativas.valores,Interval
osVD,lixo,MinimoVD,MaximoVD,MediaVD,DPVD) != 0){
         return 6;
      }
      ultimoVI = quantiE.variaveisQuantitativas.valores.length;
      ultimoVD = quantiD.variaveisQuantitativas.valores.length;
      //MenorValor.Text:=floattostr(Intervalos[0].Min);
      //NIntervalos.Text:=inttostr(ultimoVI);
      //TIntervalos.Text:=floattostr(Intervalos[0].Max-Intervalos[0].Min);

//Gerenciador.FornecaEstatistico.ColoqueValoresNosIntervalos(NomeVI,Intervalo
s,Intervalos);

//Gerenciador.FornecaEstatistico.ColoqueValoresNosIntervalos(NomeVD,Intervalo
sVD,IntervalosVD);
      NCategorias = 0;
      if((mensD.equals("disc")) || (mensE.equals("disc"))){
        if((mensE.equals("cont")) && (mensD.equals("disc"))){
            NCategorias =
estatistica.AcheNumeroDeCategorias(quantiD.variaveisQuantitativas);
        }else{
            if((mensE.equals("disc")) && (mensD.equals("cont"))){
              NCategorias =
estatistica.AcheNumeroDeCategorias(quantiE.variaveisQuantitativas);
            }
        }
      }else{
         NCategorias  = 5;   //continua x continua
      }
      if(NCategorias >=5){
           // Prepara o gráfico
           String Title = "Diagrama de Dispersão: " + NomeVI;
           //**Continua x Discreta**
           if(mensE.equals("cont")){
              TotalVI = 0;
              TotalVD = 0;
              double[] X = new double[(int)ultimoVI];
              double[] Y = new double[(int)ultimoVI];
              for(int i=0; i<= ultimoVI-1; i++){
                  //Series3.Add(IntervalosVD[i].Qtdade, '[' +
floattostr(IntervalosVD[i].Min) + ' - ' + floattostr(IntervalosVD[i].Max) +
']' ,Cores[cor]);
                  X[i] =
Double.valueOf(quantiD.variaveisQuantitativas.valores[i]).doubleValue();
                  Y[i] =
Double.valueOf(quantiE.variaveisQuantitativas.valores[i]).doubleValue();
              }
              for(int i= 0; i< IntervalosVD.intervalo.length; i++){
                TotalVD = TotalVD + IntervalosVD.intervalo[i].getQtdade();
```

```
                }
                for(int i= 0; i< Intervalos.intervalo.length; i++){
                  TotalVI = TotalVI + Intervalos.intervalo[i].getQtdade();
                }
                grafico.geraScatterPlot(Title, X, Y);
             }
             //**Discreta x Continua**
             if(mensE.equals("disc")){
                TotalVI = 0;
                TotalVD = 0;
                double[] X = new double[(int)ultimoVD];
                double[] Y = new double[(int)ultimoVD];
                for(int i=0; i<= ultimoVD-1; i++){
                    X[i] =
Double.valueOf(quantiE.variaveisQuantitativas.valores[i]).doubleValue();
                    Y[i] =
Double.valueOf(quantiD.variaveisQuantitativas.valores[i]).doubleValue();
                }
                for(int j=0; j<= ultimoVI-1; j++){
                  TotalVI = TotalVI + Intervalos.intervalo[j].getQtdade();
                  TotalVD = TotalVD + IntervalosVD.intervalo[j].getQtdade();
                }
                grafico.geraScatterPlot(Title, X, Y);
             }
       // Termina de preparar o gráfico
       }else{
          // Prepara o gráfico
          String Title = "Histograma: " + NomeVI;
          int[][] valores = new int[2][Intervalos.intervalo.length];
          String[] labels = new String[Intervalos.intervalo.length];
          String[] categ = new String[2];
          categ[0] = "Um";
          categ[1] = "Dois";
          TotalVI = 0;
          TotalVD = 0;
          for(int i=0; i<Intervalos.intervalo.length;i++){
             valores[0][i] = (int)Intervalos.intervalo[i].getQtdade();
             labels[i] = "[" +
String.valueOf(Intervalos.intervalo[i].getMin()) + "-" +
String.valueOf(Intervalos.intervalo[i].getMax()) + "]";
             TotalVI = TotalVI + Intervalos.intervalo[i].getQtdade();
          }
//          grafico.armazenaDadosGrafico(Title, valores, labels, 0);
          TotalVD = 0;
          //for(int i=0; i<= ultimoVD-1; i++){
  //          valores = new int[IntervalosVD.intervalo.length];
          labels = new String[IntervalosVD.intervalo.length];
          for(int i=0; i< IntervalosVD.intervalo.length; i++){
            valores[1][i] = (int)Intervalos.intervalo[i].getQtdade();
            labels[i] = "[" + String.valueOf(Intervalos.intervalo[i].getMin())
+ "-" + String.valueOf(Intervalos.intervalo[i].getMax()) + "]";
            TotalVD = TotalVD + IntervalosVD.intervalo[i].getQtdade();
          }
          grafico.geraGraficoBarra3D(Title, valores, labels, categ);
          //grafico.geraGraficoBarra3D(Title, valores, labels);
          // Termina de preparar o gráfico
       }
       // Prepara a Tabela de Frequencias
       //variavel Independente
      NumberFormat _nf =  NumberFormat.getInstance();
      _nf.setMaximumFractionDigits(3);
```

```
      tamanho = quantiE.variaveisQuantitativas.valores.length;
      auxMediana = new double[(int)tamanho];
      for(int i=0; i<= tamanho-1; i++){
        if(! "".equals(quantiE.variaveisQuantitativas.valores[i])){
          auxMediana[i] =
Float.valueOf(quantiE.variaveisQuantitativas.valores[i]).floatValue();
        }
      }
      estatistica.CalculeQ25(auxMediana,VQ25);
      estatistica.CalculeMediana(auxMediana,VMediana);
      estatistica.CalculeQ75(auxMediana,VQ75);
      //variavel dependente
      tamanho = quantiD.variaveisQuantitativas.valores.length;
      auxMedianaD = new double[(int)tamanho];
      for(int i=0; i<= tamanho-1; i++){
        if (!"".equals(quantiD.variaveisQuantitativas.valores[i])){
          auxMedianaD[i] =
Float.valueOf(quantiD.variaveisQuantitativas.valores[i]).floatValue();
        }
      }
      estatistica.CalculeQ25(auxMedianaD,VQ25D);
      estatistica.CalculeMediana(auxMedianaD,VMedianaD);
      estatistica.CalculeQ75(auxMedianaD,VQ75D);
      //termino
      //TabelaFrequencias.caption := 'Medidas Descritivas';
      TabelaFrequencias = new String[9][3];
      TabelaFrequencias[0][0] = "Variável ";
      TabelaFrequencias[1][0] = "    n    ";
      TabelaFrequencias[2][0] = "  Média  ";
      TabelaFrequencias[3][0] = "D. Padrão";
      TabelaFrequencias[4][0] = "  Mínimo ";
      TabelaFrequencias[5][0] = "   Q1    "; //
      TabelaFrequencias[6][0] = " Mediana "; //
      TabelaFrequencias[7][0] = "   Q3    "; //
      TabelaFrequencias[8][0] = "  Máximo ";

      TabelaFrequencias[0][1] =NomeVI;
      TabelaFrequencias[0][2] =NomeVD;
      TabelaFrequencias[1][1] =String.valueOf(TotalVI);
      TabelaFrequencias[2][1] =_nf.format(MediaVI.doubleValue());
      TabelaFrequencias[3][1] =_nf.format(DPVI.doubleValue());
      TabelaFrequencias[4][1] =_nf.format(MinimoVI.doubleValue());
      TabelaFrequencias[5][1] = _nf.format(VQ25.doubleValue());
      TabelaFrequencias[6][1] = _nf.format(VMediana.doubleValue());
      TabelaFrequencias[7][1] = _nf.format(VQ75.doubleValue());
      TabelaFrequencias[8][1] =_nf.format(MaximoVI.doubleValue());
      TabelaFrequencias[1][2] =_nf.format(TotalVD);
      TabelaFrequencias[2][2] =_nf.format(MediaVD.doubleValue());
      TabelaFrequencias[3][2] =_nf.format(DPVD.doubleValue());
      TabelaFrequencias[4][2] =_nf.format(MinimoVD.doubleValue());
      TabelaFrequencias[5][2] = _nf.format(VQ25D.doubleValue());
      TabelaFrequencias[6][2] = _nf.format(VMedianaD.doubleValue());
      TabelaFrequencias[7][2] = _nf.format(VQ75D.doubleValue());
      TabelaFrequencias[8][2] =_nf.format(MaximoVD.doubleValue());
      // Termina de fazer a Tabela de Frequencias
      return 4;
    }

    public int DescricaoBivariadaQuantiQualiOUQualiQuanti(String __nomeE,
String __nomeD){
      long Total,ultimoVI,ultimoVD;
```

```
      Objeto Media = new Objeto(0);
      Objeto DP = new Objeto(0);
      Objeto DPs = new Objeto();
      Objeto Medias = new Objeto();
      Objeto Ns = new Objeto();
      ObjetoRefVariaveisQualitativas quali = new
ObjetoRefVariaveisQualitativas();
      ObjetoRefVariaveisQuantitativas quanti = new
ObjetoRefVariaveisQuantitativas();
      ObjetoRefCategoriaPorIntervalo CategoriasPorIntervalos = new
ObjetoRefCategoriaPorIntervalo();
      String NomeVI, NomeVD;
      if(tipoAtual.equals("QualiQuanti")){
           NomeVI = __nomeE;
           NomeVD = __nomeD;
      }else{
           NomeVI = __nomeD;
           NomeVD = __nomeE;
      }
      if(leitorDBF.FornecaVariavelQualitativa(NomeVI,quali)==false){
         return 6;
      }
      if(leitorDBF.FornecaVariavelQuantitativa(NomeVD,quanti)==false){
         return 6;
      }

if(estatistica.AcheCategoriasPorIntervalos(quali.variaveisQualitativas,quanti
.variaveisQuantitativas,CategoriasPorIntervalos) != 0){
         return 6;
      }
      ultimoVD = 0;
      // Prepara o Gráfico para A 1a Categoria
      String Title = "Diagrama em colunas: " + NomeVD + " X " + NomeVI + "="
+ CategoriasPorIntervalos.categoriaPorIntervalo[0].getValor();
      ultimoVI =
CategoriasPorIntervalos.categoriaPorIntervalo[0].Intervalos.length;
      ultimoVD = CategoriasPorIntervalos.categoriaPorIntervalo.length;
      int[][] valores = new
int[CategoriasPorIntervalos.categoriaPorIntervalo.length][(int)ultimoVI];
      String[] labels = new String[(int)ultimoVI];
      String[] categ = new String[(int)ultimoVI];
//
grafico.createListaDados(CategoriasPorIntervalos.categoriaPorIntervalo.length
);
      for(int k=0;
k<CategoriasPorIntervalos.categoriaPorIntervalo.length;k++){
        if(ultimoVI <
CategoriasPorIntervalos.categoriaPorIntervalo[k].Intervalos.length){
            ultimoVI =
CategoriasPorIntervalos.categoriaPorIntervalo[k].Intervalos.length;
//            valores = new int[(int)ultimoVI];
            labels = new String[(int)ultimoVI];
        }
        for(int i=0; i<
CategoriasPorIntervalos.categoriaPorIntervalo[k].Intervalos.length; i++){
            valores[k][i] =
(int)CategoriasPorIntervalos.categoriaPorIntervalo[k].Intervalos[i].getQtdade
();
            labels[i] = "[" +
String.valueOf(CategoriasPorIntervalos.categoriaPorIntervalo[k].Intervalos[i]
.getMin()) + " - " +
```

```
String.valueOf(CategoriasPorIntervalos.categoriaPorIntervalo[k].Intervalos[i]
.getMax()) + "]";
        }
        categ[k] = NomeE + "=" +
quali.variaveisQualitativas.getCategorias()[k].getValor();
//        grafico.armazenaDadosGrafico(Title, valores, labels, k);
      }
      grafico.geraGraficoBarra3D(Title, valores, labels, categ);

      // Termina o Gráfico

      // Prepara a Tabela de Frequencias
      //TabFrequencias.caption := 'Medidas Descritivas';
      NumberFormat _nf =  NumberFormat.getInstance();
      _nf.setMaximumFractionDigits(3);
      TabelaFrequencias = new String[4][(int)ultimoVD + 2];
      TabelaFrequencias[0][0] = "Categoria";
      TabelaFrequencias[1][0] = "  Média  ";
      TabelaFrequencias[2][0] = "    n    ";
      TabelaFrequencias[3][0] = "D. Padrão";
      TabelaFrequencias[0][(int)ultimoVD+1] = "Todas";

if(estatistica.CalculeMediasPorCategorias(quali.variaveisQualitativas,quanti.
variaveisQuantitativas,CategoriasPorIntervalos.categoriaPorIntervalo,Medias,N
s,Media) != 0){
         return 6;
      }

if(estatistica.CalculeDPsPorCategorias(quali.variaveisQualitativas,quanti.var
iaveisQuantitativas,CategoriasPorIntervalos.categoriaPorIntervalo,Medias.__do
uble,Media.doubleValue(),DPs,DP) != 0){
         return 6;
      }
      Total = 0;
      for(int i=1; i<= ultimoVD; i++){
            TabelaFrequencias[0][i] = NomeVI + "="
+CategoriasPorIntervalos.categoriaPorIntervalo[i-1].getValor();
            TabelaFrequencias[1][i] = _nf.format(Medias.__double[i-1]);
            TabelaFrequencias[2][i] = String.valueOf(Ns.__long);
            TabelaFrequencias[3][i] = _nf.format(DPs.__double[i-1]);
            Total = Total + Ns.__long[i-1];
      }
      TabelaFrequencias[2][(int)ultimoVD+1] =
_nf.format(Media.doubleValue());
      TabelaFrequencias[1][(int)ultimoVD+1] = String.valueOf(Total);
      TabelaFrequencias[3][(int)ultimoVD+1] = _nf.format(DP.doubleValue());
      // Termina de fazer a Tabela de Frequencias
      return 4;
    }

    private void inicializaObjetoAnalise(){
      minhaAnalise = new ObjetoAnalise();
      minhaAnalise.NomeVI = "";
      minhaAnalise.NomeVD = "";
      minhaAnalise.TipoVI = "naosabe";
      minhaAnalise.TipoVD = "naosabe";
      minhaAnalise.MensuracaoVI = "nenhuma";
      minhaAnalise.MensuracaoVD ="nenhuma";
      minhaAnalise.nCategoriasVI =0;
      minhaAnalise.nCategoriasVD =0;
      minhaAnalise.Homocedasticidade = naosei;
```

```
      minhaAnalise.Normalidade = naosei;
      minhaAnalise.Paridade = naosei;
      minhaAnalise.FreqEsperadasMaiores5 = naosei;
    }

    private int ConsulteGerenciadorUsandoObjetoAnalise(){
      ObjetoRefVariaveisQualitativas Vquali,Vquali2;
      ObjetoRefVariaveisQuantitativas Vquanti;
      int ultimo,ret2;
      ObjetoIntervalo Intervalos;
      String[] Valores;
      Objeto vmens1 = new Objeto("nom");
      Objeto vmens2 = new Objeto("nom");
      Objeto vtipoVI = new Objeto("quali");
      Objeto vtipoVD = new Objeto("quali");
      Objeto ncat1 = new Objeto(0);
      Objeto ncat2 = new Objeto(0);
      Objeto auto = new Objeto(false);
      Objeto auto1 = new Objeto(false);
      Objeto auto2 = new Objeto(false);
      Objeto acao = new Objeto(0);
      Objeto metodo = new Objeto(0);

      educador.VerificaObjetoAnalise(minhaAnalise,acao,metodo);
      if(acao.intValue()==0){
           // Executa um Método
           int ret3 = ExecutaMetodo(metodo.intValue());
           if(ret3 != 1){
               return ret3;
           }else{
              inicializaObjetoAnalise();
           }
      }else{
           // Pede algum dado
           if(acao.intValue()==1){
           }
           if(acao.intValue()==2){

if(educador.AutomatizaTipo(minhaAnalise.NomeVI,vtipoVI,auto)==false){
                    return 0;
                }
                if(auto.booleanValue()){
                    tipoVar[0] = vtipoVI.getString();
                }
                auto.setBoolean(false);

if(educador.AutomatizaTipo(minhaAnalise.NomeVD,vtipoVD,auto)==false){
                   return 0;
                }
                if(auto.booleanValue()){
                     tipoVar[1] = vtipoVD.getString();
                }
           }
           if(acao.intValue()==3){

if(educador.AutomatizaMensuracao(minhaAnalise.NomeVI,vmens1,auto1)==false){
                   return 0;
                }

if(educador.AutomatizaMensuracao(minhaAnalise.NomeVD,vmens2,auto2)==false){
                   return 0;
```

```
                }
                if(auto1.booleanValue()){
                    mensVar[0] = vmens1.getString();
                }
                if(auto2.booleanValue()){
                     mensVar[1] = vmens2.getString();
                }
            }
            if(acao.intValue()==4){
                nCat[0] = 2;
                nCat[1] = 2;

if(educador.AutomatizaNumCategorias(minhaAnalise.NomeVI,ncat1,auto1)==false){
                   return 0;
                }

if(educador.AutomatizaNumCategorias(minhaAnalise.NomeVD,ncat2,auto2)==false){
                   return 0;
                }

                if(auto1.booleanValue()){
                    nCat[0] = ncat1.intValue();
                }
                if(auto2.booleanValue()){
                   nCat[1] = ncat2.intValue();
                }
            }
            if(acao.intValue()==5){
                if (("quali".equals(minhaAnalise.TipoVI)) &&
("quali".equals(minhaAnalise.TipoVD))){
                     nCat[0] = 2;
                     nCat[1] = 2;

if(educador.AutomatizaNumCategorias(minhaAnalise.NomeVI,ncat1,auto1)==false){
                         return 0;
                     }

if(educador.AutomatizaNumCategorias(minhaAnalise.NomeVD,ncat2,auto2)==false){
                        return 0;
                     }
                     if(auto1.booleanValue()){
                        nCat[0] = ncat1.intValue();
                     }
                     if(auto2.booleanValue()){
                        nCat[1] = ncat2.intValue();
                     }
                   }
                if(("quali".equals(minhaAnalise.TipoVI)) &&
("quanti".equals(minhaAnalise.TipoVD))){
                     nCat[0] = 2;
                     nCat[1] = 2;

if(educador.AutomatizaNumCategorias(minhaAnalise.NomeVI,ncat1,auto1)==false){
                        return 0;
                     }
                     if(auto1.booleanValue()){
                        nCat[0] = ncat1.intValue();
                     }
                }
            }
            if(acao.intValue()==6){
```

```
                if(("quali".equals(minhaAnalise.TipoVI)) &&
("quali".equals(minhaAnalise.TipoVD))){
                     nCat[0] = 2;
                     nCat[1] = 2;

if(educador.AutomatizaNumCategorias(minhaAnalise.NomeVI,ncat1,auto1)==false){
                        return 0;
                     }

if(educador.AutomatizaNumCategorias(minhaAnalise.NomeVD,ncat2,auto2)==false){
                        return 0;
                     }
                     if(auto1.booleanValue()){
                        nCat[0] = ncat1.intValue();
                     }
                     if(auto2.booleanValue()){
                        nCat[1] = ncat2.intValue();
                     }
                }
                if(("quanti".equals(minhaAnalise.TipoVI)) &&
("quali".equals(minhaAnalise.TipoVD))){
                     nCat[0] = 2;
                     nCat[1] = 2;

if(educador.AutomatizaNumCategorias(minhaAnalise.NomeVD,ncat2,auto2)==false){
                        return 0;
                     }
                     if(auto2.booleanValue()){
                        nCat[1] = ncat2.intValue();
                     }
                }
           }
           if(acao.intValue()==7){}
           if(acao.intValue()==8){
                Vquali = new ObjetoRefVariaveisQualitativas();
                Vquanti = new ObjetoRefVariaveisQuantitativas();
                if(("quali".equals(minhaAnalise.TipoVI)) &&
("quanti".equals(minhaAnalise.TipoVD))){

if(leitorDBF.FornecaVariavelQualitativa(minhaAnalise.NomeVI,Vquali)==false){
                        return 0;
                     }

if(leitorDBF.FornecaVariavelQuantitativa(minhaAnalise.NomeVD,Vquanti)==false)
{
                       return 0;
                     }
                    NomeE = minhaAnalise.NomeVI;  //Usado para
Homocesdaticidade
                     NomeD = minhaAnalise.NomeVD;  //Usado para
Homocesdaticidade
                }
                if(("quanti".equals(minhaAnalise.TipoVI)) &&
("quali".equals(minhaAnalise.TipoVD))){

if(leitorDBF.FornecaVariavelQualitativa(minhaAnalise.NomeVD,Vquali)==false){
                        return 0;
                     }

if(leitorDBF.FornecaVariavelQuantitativa(minhaAnalise.NomeVI,Vquanti)==false)
{
```

```
                        return 0;
                    }
                    NomeE = minhaAnalise.NomeVI;
                    NomeD = minhaAnalise.NomeVD;
               }
               Objeto gl1BF = new Objeto(0);
               Objeto gl2BF = new Objeto(0);
               Objeto valorBF = new Objeto(0);
               Objeto pBF = new Objeto(0);
               Objeto gl1L = new Objeto(0);
               Objeto gl2L = new Objeto(0);
               Objeto valorL = new Objeto(0);
               Objeto pL = new Objeto(0);
               ret2 =
estatistica.TesteLevene(Vquali.variaveisQualitativas,Vquanti.variaveisQuantit
ativas,gl1L,gl2L,valorL,pL);
               if(ret2 != 0){
                    return 4; //Erro ao executar teste de Levene
               }
               ret2 =
estatistica.TesteBrownForsythe(Vquali.variaveisQualitativas,Vquanti.variaveis
Quantitativas,gl1BF,gl2BF,valorBF,pBF);
               if(ret2 != 0){
                    return 3; //Erro ao executar teste de Brown-Forsythe
               }
               //Para Homcedasticidade
               GL1L = gl1L.doubleValue();
               GL2L = gl2L.doubleValue();
               GL1BF = gl1BF.doubleValue();
               GL2BF = gl2BF.doubleValue();
               ValorTesteL = valorL.doubleValue();
               ValorTesteBF = valorBF.doubleValue();
               //Fhomo.Show;
          }
          if(acao.intValue()==9){
               Vquali = new ObjetoRefVariaveisQualitativas();
               Vquanti = new ObjetoRefVariaveisQuantitativas();
               if((("quali".equals(minhaAnalise.TipoVI)) &&
("quanti".equals(minhaAnalise.TipoVD))) ||
                  (("quanti".equals(minhaAnalise.TipoVI)) &&
("quali".equals(minhaAnalise.TipoVD)))){
                    if(("quali".equals(minhaAnalise.TipoVI))){

leitorDBF.FornecaVariavelQualitativa(minhaAnalise.NomeVI,Vquali);

leitorDBF.FornecaVariavelQuantitativa(minhaAnalise.NomeVD,Vquanti);
                    }else{

leitorDBF.FornecaVariavelQualitativa(minhaAnalise.NomeVD,Vquali);

leitorDBF.FornecaVariavelQuantitativa(minhaAnalise.NomeVI,Vquanti);
                    }
                    for(int i=0; i<=
Vquali.variaveisQualitativas.getNumCategorias()-1; i++){
                         Intervalos = new ObjetoIntervalo();
                         Objeto Esperados = new Objeto(0);
                         int tamanhoBase =
leitorDBF.RetornaNumeroValores();
                         Valores = new String[tamanhoBase];
                         ultimo = 0;
                         for(int j=0; j<= tamanhoBase-1; j++){
```

```
if((!"".equals(Vquanti.variaveisQuantitativas.valores[j])) &&
(Vquali.variaveisQualitativas.valores[j].equals(Vquali.variaveisQualitativas.
getCategorias()[i].getValor()))){
                                    Valores[ultimo] =
Vquanti.variaveisQuantitativas.valores[j];
                                    ultimo = ultimo+1;
                                }
                            }
                            String[] tempValores = new String[ultimo];
                            for(int j=0; j< ultimo; j++){
                                tempValores[j] = Valores[j];
                            }
                            Valores = tempValores;
                            Objeto lixo2 = new Objeto(0);
                            Objeto lixo = new Objeto(0);
                            Objeto media = new Objeto(0);
                            Objeto dp = new Objeto(0);
                            Objeto gl1L = new Objeto(0);
                            Objeto valorL = new Objeto(0);
                            Objeto pL = new Objeto(0);
                            ret2 =
estatistica.AcheIntervalos(Valores,Intervalos,lixo2,lixo,lixo,media,dp);
                            if(ret2!=0){
                                return 5; //Erro ao achar Intervalos
                            }
                            ret2 =
estatistica.ValoresEsperados_ParaTesteNormalidadeQuiQuadradro(Intervalos.inte
rvalo,media.doubleValue(),dp.doubleValue(),Esperados);
                            if(ret2!=0){
                               return 6; //Erro ao achar valores esperados
                            }
                            ret2 =
estatistica.TesteQuiQuadradoNormal(Intervalos.intervalo,Esperados.__double,gl
1L,valorL,pL);
                            if(ret2!=0){
                                 return 7; //Erro ao executar teste Qui-
Quadrado Normal
                            }
                            NomeQuanti =
Vquanti.variaveisQuantitativas.getNome();
                            NomeQuali =
Vquali.variaveisQualitativas.getNome();
                            CatQuali =
Integer.valueOf(Vquali.variaveisQualitativas.getCategorias()[i].getValor()).i
ntValue();
                            ValorL = valorL.doubleValue();
                            PL = pL.doubleValue();
                            gl1L.setDouble(Intervalos.intervalo.length);
                            GL = gl1L.doubleValue()-1;
                            valoresQuanti = Valores;
                            intervalos = Intervalos.intervalo;
                            esperados = Esperados.__double;;
                     }
                }
                if(("quanti".equals(minhaAnalise.TipoVI)) &&
("quanti".equals(minhaAnalise.TipoVD))){

leitorDBF.FornecaVariavelQuantitativa(minhaAnalise.NomeVI,Vquanti);
                    Intervalos = new ObjetoIntervalo();
                    Objeto Esperados = new Objeto(0);
```

```
                    int tamanhoBase = leitorDBF.RetornaNumeroValores();
                    Valores = new String[tamanhoBase];
                    ultimo = 0;
                    for(int j=0; j<= tamanhoBase-1;j++){

if(!"".equals(Vquanti.variaveisQuantitativas.valores[j])){
                            Valores[ultimo] =
Vquanti.variaveisQuantitativas.valores[j];
                            ultimo = ultimo+1;
                        }
                    }
                    String[] tempValores = new String[ultimo];
                    for(int j = 0; j< ultimo; j++){
                        tempValores[j] = Valores[j];
                    }
                    Valores = tempValores;
                    Objeto lixo2 = new Objeto(0);
                    Objeto lixo = new Objeto(0);
                    Objeto media = new Objeto(0);
                    Objeto dp = new Objeto(0);
                    Objeto gl1L = new Objeto(0);
                    Objeto valorL = new Objeto(0);
                    Objeto pL = new Objeto(0);
                    ret2 =
estatistica.AcheIntervalos(Valores,Intervalos,lixo2,lixo,lixo,media,dp);
                    if(ret2!=0){
                         return 5; //Erro ao achar Intervalos
                    }
                    ret2 =
estatistica.ValoresEsperados_ParaTesteNormalidadeQuiQuadradro(Intervalos.inte
rvalo,media.doubleValue(),dp.doubleValue(),Esperados);
                    if(ret2!=0){
                        return 6; //Erro ao achar valores esperados
                    }
                    ret2 =
estatistica.TesteQuiQuadradoNormal(Intervalos.intervalo,Esperados.__double,gl
1L,valorL,pL);
                    if(ret2!=0){
                        return  7; //Erro ao executar teste Qui-Quadrado
Normal
                    }
                    CatQuali = 0;
                    NomeQuanti = Vquanti.variaveisQuantitativas.getNome();
                    NomeQuali = "";
                    ValorL = valorL.doubleValue();
                    PL = pL.doubleValue();
                    gl1L.setDouble(Intervalos.intervalo.length);
                    GL = gl1L.doubleValue()-1;
                    valoresQuanti = Valores;
                    intervalos = Intervalos.intervalo;
                    esperados = Esperados.__double;
                    //////////////

leitorDBF.FornecaVariavelQuantitativa(minhaAnalise.NomeVD,Vquanti);
                    Intervalos = new ObjetoIntervalo();
                    Esperados = new Objeto(0);
                    tamanhoBase = leitorDBF.RetornaNumeroValores();
                    Valores = new String[tamanhoBase];
                    ultimo = 0;
                    for(int j=0; j<= tamanhoBase-1; j++){
```

```
if(!"".equals(Vquanti.variaveisQuantitativas.valores[j])){
                            Valores[ultimo] =
Vquanti.variaveisQuantitativas.valores[j];
                            ultimo = ultimo+1;
                        }
                    }
                    tempValores = new String[ultimo];
                    for(int j=0; j< ultimo; j++){
                        tempValores[j] = Valores[j];
                    }
                    Valores = tempValores;
                    ret2 =
estatistica.AcheIntervalos(Valores,Intervalos,lixo2,lixo,lixo,media,dp);
                    if(ret2!=0){
                        return 5; //Erro ao achar Intervalos
                    }
                    ret2 =
estatistica.ValoresEsperados_ParaTesteNormalidadeQuiQuadradro(Intervalos.inte
rvalo,media.doubleValue(),dp.doubleValue(),Esperados);
                    if(ret2!=0){
                        return 6; //Erro ao achar valores esperados
                    }
                    ret2 =
estatistica.TesteQuiQuadradoNormal(Intervalos.intervalo,Esperados.__double,gl
1L,valorL,pL);
                    if(ret2!=0){
                        return 7; //Erro ao executar teste Qui-Quadrado
Normal
                    }
                    CatQuali = 0;
                    NomeQuanti = Vquanti.variaveisQuantitativas.getNome();
                    NomeQuali = "";
                    ValorL = valorL.doubleValue();
                    PL = pL.doubleValue();
                    gl1L.setDouble(Intervalos.intervalo.length);
                    GL = gl1L.doubleValue()-1;
                    valoresQuanti = Valores;
                    intervalos = Intervalos.intervalo;
                    esperados = Esperados.__double;
                }
            }
            if(acao.intValue()==10){
                Vquali = new ObjetoRefVariaveisQualitativas();
                Vquali2 = new ObjetoRefVariaveisQualitativas();

leitorDBF.FornecaVariavelQualitativa(minhaAnalise.NomeVI,Vquali);

leitorDBF.FornecaVariavelQualitativa(minhaAnalise.NomeVD,Vquali2);
                Objeto Esperadas = new Objeto(0);
                Objeto Observadas = new Objeto(0);
                Objeto TabelaQuiquadrado = new Objeto(0);
                ret2 =
estatistica.CalculeFrequenciasObservadasEEsperadas(Vquali.variaveisQualitativ
as,Vquali2.variaveisQualitativas,Observadas,Esperadas,TabelaQuiquadrado);
                if(ret2!=0){
                    return 8; //Erro ao Calcular Frequências Observadas e
Esperadas
                }
```

```
                fObservadas = new
String[(int)Vquali.variaveisQualitativas.getNumCategorias()+2][(int)Vquali2.v
ariaveisQualitativas.getNumCategorias()+2];
                fEsperadas = new
String[(int)Vquali.variaveisQualitativas.getNumCategorias()+2][(int)Vquali2.v
ariaveisQualitativas.getNumCategorias()+2];
                TabelaContribuicaoQQ = new
String[(int)Vquali.variaveisQualitativas.getNumCategorias()+2][(int)Vquali2.v
ariaveisQualitativas.getNumCategorias()+2];
                fObservadas[0][0] = "";
                fEsperadas[0][0] = "";
                TabelaContribuicaoQQ[0][0] = "";
                NumberFormat _nf =  NumberFormat.getInstance();
                _nf.setMaximumFractionDigits(3);
                for(int i=1; i <=
(int)Vquali.variaveisQualitativas.getNumCategorias(); i++){
                  fObservadas[i][0] = Vquali.variaveisQualitativas.getNome() +
"=" + Vquali.variaveisQualitativas.getCategorias()[i-1].getValor();
                  fEsperadas[i][0] =  Vquali.variaveisQualitativas.getNome() +
"=" + Vquali.variaveisQualitativas.getCategorias()[i-1].getValor();
                  TabelaContribuicaoQQ[i][0] =
Vquali.variaveisQualitativas.getNome() + "=" +
Vquali.variaveisQualitativas.getCategorias()[i-1].getValor();
                }

fObservadas[(int)Vquali.variaveisQualitativas.getNumCategorias()+1][0] =
"Todas";

fEsperadas[(int)Vquali.variaveisQualitativas.getNumCategorias()+1][0] =
"Todas";

TabelaContribuicaoQQ[(int)Vquali.variaveisQualitativas.getNumCategorias()+1][
0] = "Todas";
                for(int j=1; j<= (int)
Vquali2.variaveisQualitativas.getNumCategorias(); j++){
                  fObservadas[0][j] = Vquali2.variaveisQualitativas.getNome()
+ "=" + Vquali2.variaveisQualitativas.getCategorias()[j-1].getValor();
                  fEsperadas[0][j] =  Vquali2.variaveisQualitativas.getNome()
+ "=" + Vquali2.variaveisQualitativas.getCategorias()[j-1].getValor();
                  TabelaContribuicaoQQ[0][j] =
Vquali2.variaveisQualitativas.getNome() + "=" +
Vquali2.variaveisQualitativas.getCategorias()[j-1].getValor();
                }

fObservadas[0][(int)Vquali2.variaveisQualitativas.getNumCategorias()+1] =
"Total Linhas";

fEsperadas[0][(int)Vquali2.variaveisQualitativas.getNumCategorias()+1] =
"Total Linhas";

TabelaContribuicaoQQ[0][(int)Vquali2.variaveisQualitativas.getNumCategorias()
+1] = "Total Linhas";
                for(int i=0; i <=
(int)Vquali.variaveisQualitativas.getNumCategorias(); i++){
                   for(int j=0; j<= (int)
Vquali2.variaveisQualitativas.getNumCategorias(); j++){
                     fObservadas[i+1][j+1] =
_nf.format(Observadas.___double[i][j]);
                     fEsperadas[i+1][j+1] =
_nf.format(Esperadas.___double[i][j]);
```

```
                    TabelaContribuicaoQQ[i+1][j+1] =
_nf.format(TabelaQuiquadrado.___double[i][j]);
                  }
                }
                vQuali = Vquali.variaveisQualitativas;
                vQuali2 = Vquali2.variaveisQualitativas;
            }
      }
      return 1;
    }

    private int ExecutaMetodo(int metodo){
      Objeto ValorTeste,p; //double
      Objeto aux1,aux2,aux3,aux4; //double
      Objeto auxN1,auxN2,gL; //long
      ObjetoRefVariaveisQualitativas quali1,quali2;
      ObjetoRefVariaveisQuantitativas quanti1,quanti2;
      Objeto Ns; //long[]
      Objeto Somas; //double[]
      int ret;
      if(metodo== 0){
             quali1 = new ObjetoRefVariaveisQualitativas();
             quali2 = new ObjetoRefVariaveisQualitativas();

leitorDBF.FornecaVariavelQualitativa(minhaAnalise.NomeVI,quali1);

leitorDBF.FornecaVariavelQualitativa(minhaAnalise.NomeVD,quali2);
             gL = new Objeto(0);
             p = new Objeto(0);
             ValorTeste = new Objeto(0);
             ret =
estatistica.TesteQuiQuadrado(quali1.variaveisQualitativas,quali2.variaveisQua
litativas,gL,ValorTeste,p);
             if(ret!=0){
                 return 10;
             }
             QuiquadradoPValor = p.doubleValue();
             QuiQuadradoGL = gL.doubleValue();
             QuiQuadradoValorTeste = ValorTeste.doubleValue();
             vQuali = quali1.variaveisQualitativas;
             vQuali2 = quali2.variaveisQualitativas;
             Objeto FObservadas = new Objeto();
             Objeto FEsperadas = new Objeto();
             Objeto TabelaQuiquadrado = new Objeto();

estatistica.CalculeFrequenciasObservadasEEsperadas(quali1.variaveisQualitativ
as,quali2.variaveisQualitativas,FObservadas,FEsperadas,TabelaQuiquadrado);
             fObservadas = new
String[(int)quali1.variaveisQualitativas.getNumCategorias()+2][(int)quali2.va
riaveisQualitativas.getNumCategorias()+2];
             fEsperadas = new
String[(int)quali1.variaveisQualitativas.getNumCategorias()+2][(int)quali2.va
riaveisQualitativas.getNumCategorias()+2];
             TabelaContribuicaoQQ = new
String[(int)quali1.variaveisQualitativas.getNumCategorias()+2][(int)quali2.va
riaveisQualitativas.getNumCategorias()+2];
             fObservadas[0][0] = "";
               fEsperadas[0][0] = "";
               TabelaContribuicaoQQ[0][0] = "";
               NumberFormat _nf =  NumberFormat.getInstance();
               _nf.setMaximumFractionDigits(3);
```

```
               for(int i=1; i <=
(int)quali1.variaveisQualitativas.getNumCategorias(); i++){
                  fObservadas[i][0] = quali1.variaveisQualitativas.getNome() +
"=" + quali1.variaveisQualitativas.getCategorias()[i-1].getValor();
                  fEsperadas[i][0] =  quali1.variaveisQualitativas.getNome() +
"=" + quali1.variaveisQualitativas.getCategorias()[i-1].getValor();
                  TabelaContribuicaoQQ[i][0] =
quali1.variaveisQualitativas.getNome() + "=" +
quali1.variaveisQualitativas.getCategorias()[i-1].getValor();
               }

fObservadas[(int)quali1.variaveisQualitativas.getNumCategorias()+1][0] =
"Todas";

fEsperadas[(int)quali1.variaveisQualitativas.getNumCategorias()+1][0] =
"Todas";

TabelaContribuicaoQQ[(int)quali1.variaveisQualitativas.getNumCategorias()+1][
0] = "Todas";
               for(int j=1; j<= (int)
quali2.variaveisQualitativas.getNumCategorias(); j++){
                  fObservadas[0][j] = quali2.variaveisQualitativas.getNome() +
"=" + quali2.variaveisQualitativas.getCategorias()[j-1].getValor();
                  fEsperadas[0][j] =  quali2.variaveisQualitativas.getNome() +
"=" + quali2.variaveisQualitativas.getCategorias()[j-1].getValor();
                  TabelaContribuicaoQQ[0][j] =
quali2.variaveisQualitativas.getNome() + "=" +
quali2.variaveisQualitativas.getCategorias()[j-1].getValor();
               }

fObservadas[0][(int)quali2.variaveisQualitativas.getNumCategorias()+1] =
"Total Linhas";

fEsperadas[0][(int)quali2.variaveisQualitativas.getNumCategorias()+1] =
"Total Linhas";

TabelaContribuicaoQQ[0][(int)quali2.variaveisQualitativas.getNumCategorias()+
1] = "Total Linhas";
               for(int i=0; i <=
(int)quali1.variaveisQualitativas.getNumCategorias(); i++){
                   for(int j=0; j<= (int)
quali2.variaveisQualitativas.getNumCategorias(); j++){
                     fObservadas[i+1][j+1] =
_nf.format(FObservadas.___double[i][j]);
                     fEsperadas[i+1][j+1] =
_nf.format(FEsperadas.___double[i][j]);
                     TabelaContribuicaoQQ[i+1][j+1] =
_nf.format(TabelaQuiquadrado.___double[i][j]);
                   }
               }
             return 1;
      }
      if(metodo == 1){
            quali1 = new ObjetoRefVariaveisQualitativas();
            quali2 = new ObjetoRefVariaveisQualitativas();
             if(("quali".equals(minhaAnalise.TipoVI)) &&
("quali".equals(minhaAnalise.TipoVD))){
                  if("ord".equals(minhaAnalise.MensuracaoVI)){

leitorDBF.FornecaVariavelQualitativa(minhaAnalise.NomeVI,quali1);
```

```
leitorDBF.FornecaVariavelQualitativa(minhaAnalise.NomeVD,quali2);
                    }else{

leitorDBF.FornecaVariavelQualitativa(minhaAnalise.NomeVD,quali1);

leitorDBF.FornecaVariavelQualitativa(minhaAnalise.NomeVI,quali2);
                    }
                    Ns = new Objeto();
                    Somas = new Objeto();
                    gL = new Objeto(0);
                    ValorTeste = new Objeto(0);
                    p = new Objeto(0);
                    ret =
estatistica.TesteKruskalWallis(quali1.variaveisQualitativas,quali2.variaveisQ
ualitativas,Ns,Somas,gL,ValorTeste,p);
                    if(ret!=0){
                         return 10;
                    }
               }else{
                    quanti1 = new ObjetoRefVariaveisQuantitativas();
                    if("quali".equals(minhaAnalise.TipoVI)){

leitorDBF.FornecaVariavelQualitativa(minhaAnalise.NomeVI,quali1);

leitorDBF.FornecaVariavelQuantitativa(minhaAnalise.NomeVD,quanti1);
                    }else{

leitorDBF.FornecaVariavelQualitativa(minhaAnalise.NomeVD,quali1);

leitorDBF.FornecaVariavelQuantitativa(minhaAnalise.NomeVI,quanti1);
                    }
                    Ns = new Objeto();
                    Somas = new Objeto();
                    gL = new Objeto(0);
                    ValorTeste = new Objeto(0);
                    p = new Objeto(0);
                    ret =
estatistica.TesteKruskalWallis(quanti1.variaveisQuantitativas,quali1.variavei
sQualitativas,Ns,Somas,gL,ValorTeste,p);
                    if(ret!=0){
                         return 10;
                    }
               }
               KruskalWallisP = p.doubleValue();
               KruskalWallisH = ValorTeste.doubleValue();
               KruskalWallisGL = gL.longValue();
               KruskalWallisNs = Ns.__long;
               KruskalWallisSomas = Somas.__double;
               return 1;
      }
      if(metodo == 2){
             quanti1 = new ObjetoRefVariaveisQuantitativas();
             quanti2 = new ObjetoRefVariaveisQuantitativas();

leitorDBF.FornecaVariavelQuantitativa(minhaAnalise.NomeVI,quanti1);

leitorDBF.FornecaVariavelQuantitativa(minhaAnalise.NomeVD,quanti2);
             aux1 = new Objeto(0);
             ValorTeste = new Objeto(0);
             p = new Objeto(0);
```

```
            ret =
estatistica.TesteWilcoxon(quanti1.variaveisQuantitativas,quanti2.variaveisQua
ntitativas,aux1,ValorTeste,p);
            if(ret!=0){
                 return 10;
            }
            WilcoxonSomaMinima = aux1.doubleValue();
            WilcoxonP =  p.doubleValue();
            // WilcoxonGL = gL;
            WilcoxonZ = ValorTeste.doubleValue();
            return 1;
      }
      if(metodo == 3){
            quanti1 = new ObjetoRefVariaveisQuantitativas();
            quanti2 = new ObjetoRefVariaveisQuantitativas();

leitorDBF.FornecaVariavelQuantitativa(minhaAnalise.NomeVI,quanti1);

leitorDBF.FornecaVariavelQuantitativa(minhaAnalise.NomeVD,quanti2);
            auxN1 = new Objeto(0);
            aux1 = new Objeto(0);
            ValorTeste = new Objeto(0);
            p = new Objeto(0);
            ret =
estatistica.CorrelacaoSpearman(quanti1.variaveisQuantitativas,quanti2.variave
isQuantitativas,auxN1,aux1,ValorTeste,p);
            if(ret!=0){
               return 10;
            }
            SpearmanT = aux1.doubleValue();
            SpearmanP = p.doubleValue();
            SpearmanRs = ValorTeste.doubleValue();
            return 1;
      }
      if(metodo == 4){
            quanti1 = new ObjetoRefVariaveisQuantitativas();
            quanti2 = new ObjetoRefVariaveisQuantitativas();

leitorDBF.FornecaVariavelQuantitativa(minhaAnalise.NomeVI,quanti1);

leitorDBF.FornecaVariavelQuantitativa(minhaAnalise.NomeVD,quanti2);
            ValorTeste = new Objeto(0);
            ret =
estatistica.CorrelacaoPearson(quanti1.variaveisQuantitativas,quanti2.variavei
sQuantitativas,ValorTeste);
            if(ret!=0){
               return 10;
            }
            PearsonR = ValorTeste.doubleValue();
            return 1;
      }
      if(metodo == 5){
            quanti1 = new ObjetoRefVariaveisQuantitativas();
            quanti2 = new ObjetoRefVariaveisQuantitativas();

leitorDBF.FornecaVariavelQuantitativa(minhaAnalise.NomeVI,quanti1);

leitorDBF.FornecaVariavelQuantitativa(minhaAnalise.NomeVD,quanti2);
            gL = new Objeto(0);
            ValorTeste = new Objeto(0);
            p = new Objeto(0);
```

```
            ret =
estatistica.TesteTPares(quanti1.variaveisQuantitativas,quanti2.variaveisQuant
itativas,0,gL,ValorTeste,p);
            if(ret!=0){
                return 10;
            }
            TesteTParesT = ValorTeste.doubleValue();
            TesteTParesGL= gL.doubleValue();
            TesteTParesP = p.doubleValue();
            return 1;
      }
      if(metodo == 6) {
            quali1 = new ObjetoRefVariaveisQualitativas();
            quanti1 = new ObjetoRefVariaveisQuantitativas();
            if("quali".equals(minhaAnalise.TipoVI)){

leitorDBF.FornecaVariavelQualitativa(minhaAnalise.NomeVI,quali1);

leitorDBF.FornecaVariavelQuantitativa(minhaAnalise.NomeVD,quanti1);
            }else{

leitorDBF.FornecaVariavelQualitativa(minhaAnalise.NomeVD,quali1);

leitorDBF.FornecaVariavelQuantitativa(minhaAnalise.NomeVI,quanti1);
            }
            gL = new Objeto(0);
            ValorTeste = new Objeto(0);
            p = new Objeto(0);
            ret =
estatistica.TesteT(quali1.variaveisQualitativas,quanti1.variaveisQuantitativa
s,gL,ValorTeste,p);
            if(ret!=0){
                return 10;
            }
            TesteTT = ValorTeste.doubleValue();
            TesteTGL = gL.doubleValue();
            TesteTP = p.doubleValue();
            return 1;
      }
      if(metodo == 7){
            quali1 = new ObjetoRefVariaveisQualitativas();
            quali2 = new ObjetoRefVariaveisQualitativas();
            if(("quali".equals(minhaAnalise.TipoVI)) &&
("quali".equals(minhaAnalise.TipoVD))){
                  if("ord".equals(minhaAnalise.MensuracaoVI)){

leitorDBF.FornecaVariavelQualitativa(minhaAnalise.NomeVI,quali1);

leitorDBF.FornecaVariavelQualitativa(minhaAnalise.NomeVD,quali2);
                  }else{

leitorDBF.FornecaVariavelQualitativa(minhaAnalise.NomeVD,quali1);

leitorDBF.FornecaVariavelQualitativa(minhaAnalise.NomeVI,quali2);
                  }
                  aux1 = new Objeto(0);
                  aux2 = new Objeto(0);
                  auxN1 = new Objeto(0);
                  auxN2 = new Objeto(0);
                  ValorTeste = new Objeto(0);
                  p = new Objeto(0);
```

```
                    ret =
estatistica.TesteMannWhitney(quali1.variaveisQualitativas,quali2.variaveisQua
litativas,aux1,aux2,auxN1,auxN2,ValorTeste,p);
                    if(ret!=0){
                       return 10;
                    }
               }else{
                    quanti1 = new ObjetoRefVariaveisQuantitativas();
                    if("quali".equals(minhaAnalise.TipoVI)){

leitorDBF.FornecaVariavelQualitativa(minhaAnalise.NomeVI,quali1);

leitorDBF.FornecaVariavelQuantitativa(minhaAnalise.NomeVD,quanti1);
                    }else{

leitorDBF.FornecaVariavelQualitativa(minhaAnalise.NomeVD,quali1);

leitorDBF.FornecaVariavelQuantitativa(minhaAnalise.NomeVI,quanti1);
                    }
                    aux1 = new Objeto(0);
                    aux2 = new Objeto(0);
                    auxN1 = new Objeto(0);
                    auxN2 = new Objeto(0);
                    ValorTeste = new Objeto(0);
                    p = new Objeto(0);
                    ret =
estatistica.TesteMannWhitney(quanti1.variaveisQuantitativas,quali1.variaveisQ
ualitativas,aux1,aux2,auxN1,auxN2,ValorTeste,p);
                    if(ret!=0){
                       return 10;
                    }
               }
               MannWhitneySoma1 = aux1.doubleValue();
               MannWhitneySoma2 = aux2.doubleValue();
               MannWhitneyn1 = auxN1.doubleValue();
               MannWhitneyn2 = auxN2.doubleValue();
               MannWhitneyZ = ValorTeste.doubleValue();
               MannWhitneyP = p.doubleValue();
               return 1;
      }
      if(metodo == 8){
               quali1 = new ObjetoRefVariaveisQualitativas();
               quanti1 = new ObjetoRefVariaveisQuantitativas();
               if("quali".equals(minhaAnalise.TipoVI)){

leitorDBF.FornecaVariavelQualitativa(minhaAnalise.NomeVI,quali1);

leitorDBF.FornecaVariavelQuantitativa(minhaAnalise.NomeVD,quanti1);
               }else{

leitorDBF.FornecaVariavelQualitativa(minhaAnalise.NomeVD,quali1);

leitorDBF.FornecaVariavelQuantitativa(minhaAnalise.NomeVI,quanti1);
               }
               gL = new Objeto(0);
               p = new Objeto(0);
               ValorTeste = new Objeto(0);
               ret =
estatistica.TesteTSeparate(quali1.variaveisQualitativas,quanti1.variaveisQuan
titativas,gL,ValorTeste,p);
               if(ret!=0){
```

```
                return 10;
            }
            TesteTSeparateT = ValorTeste.doubleValue();
            TesteTSeparateGL = gL.doubleValue();
            TesteTSeparateP = p.doubleValue();
            return 1;
      }
      if(metodo == 9){
            quali1 = new ObjetoRefVariaveisQualitativas();
            quanti1 = new ObjetoRefVariaveisQuantitativas();
            if("quali".equals(minhaAnalise.TipoVI)){

leitorDBF.FornecaVariavelQualitativa(minhaAnalise.NomeVI,quali1);

leitorDBF.FornecaVariavelQuantitativa(minhaAnalise.NomeVD,quanti1);
            }else{

leitorDBF.FornecaVariavelQualitativa(minhaAnalise.NomeVD,quali1);

leitorDBF.FornecaVariavelQuantitativa(minhaAnalise.NomeVI,quanti1);
            }
            auxN1 = new Objeto(0);
            auxN2 = new Objeto(0);
            aux1 = new Objeto(0);
            aux2 = new Objeto(0);
            aux3 = new Objeto(0);
            aux4 = new Objeto(0);
            ValorTeste = new Objeto(0);
            p = new Objeto(0);
            ret =
estatistica.TesteAnova(quali1.variaveisQualitativas,quanti1.variaveisQuantita
tivas,auxN1,auxN2,aux1,aux2,aux3,aux4,ValorTeste,p);
            if(ret!=0){
                 return 10;
            }
            AnovaF = ValorTeste.doubleValue();
            AnovaP = p.doubleValue();
            AnovaGLT = auxN1.doubleValue();
            AnovaGLE = auxN2.doubleValue();
            AnovaSQT = aux1.doubleValue();
            AnovaSQE = aux2.doubleValue();
            AnovaQMT = aux3.doubleValue();
            AnovaQME = aux4.doubleValue();
            return 1;
      }
      if(metodo == 10){
            return 1;
      }
      return 10;
    }



    //Variaveis para guardar o nome das variaveis escolhido pelo usuario
    private String NomeE; //Nome da esquerda no Bivariada
    private String NomeD; //Nome da direita no Bivariada
    private String Nome; //Nome no Univariada
    //Variaveis para guardar o tipo das variaveis escolhido pelo usuario
    private String tipoE; //Tipo da esquerda no Bivariada
    private String tipoD; //Tipo da direita no Bivariada
```

```
    private String tipo; //Tipo no Univariada
    //Variaveis para guardar a Mensuracao das variaveis escolhido pelo
usuario
    private String mensE; //Mensuracao da esquerda no Bivariada
    private String mensD; //Mensuracao da direita no Bivariada
    private String mens; //Mensuracao no Univariada
    //Variaveis para guardar a Categoria das variaveis escolhido pelo usuario
    private int catE;
    private int catD;
    private int paridade;
    private int normalidade;
    private int homocedasticidade;
    private int frequenciaEsperada;
    //Variaveis de controle para as opcoes do usuario
    private String[] tipoVar;  //0-> Esquerda,  1-> Direita
    private String[] mensVar;  //0-> Esquerda,  1-> Direita
    private int[] nCat;     //0-> Esquerda,  1-> Direita
  //  private int nao = 0;
  //  private int sim = 1;
    private int naosei = 2;
    private double significancia;
    private String tipoAtual;
    //Tabelas
    private String[][] TabelaFrequencias, TabelaDescritivas;
    private TSetip setip;
  //  private Variavel variavel;
    private Estatistica estatistica;
    private Base_de_Dados base;
    private int id_usuario;
    private String login;
    private String senha;
    private Grafico grafico;
    private Leitor_DBF leitorDBF;
    private Educador educador;
    private ObjetoAnalise minhaAnalise;
    //Para Homcedasticidade
    private double GL1L,GL2L,GL1BF,GL2BF,ValorTesteL,ValorTesteBF;
    //Para Normalidade
    String NomeQuanti,NomeQuali;
    double CatQuali,ValorL,PL,GL;
    String[] valoresQuanti;
    Intervalo[] intervalos;
    double[] esperados;
    //Para Frequencia Esperada
    VariaveisQualitativas vQuali, vQuali2;
    String[][] fObservadas;
    String[][] fEsperadas;
    String[][] TabelaContribuicaoQQ;
    //Para Quiquadrado
    double QuiquadradoPValor,QuiQuadradoGL,QuiQuadradoValorTeste;
    //Para KruskalWallis
    double KruskalWallisP,KruskalWallisH;
    long KruskalWallisGL;
    double[] KruskalWallisSomas;
    long[] KruskalWallisNs;
    //Para Wilcoxon
    double WilcoxonSomaMinima,WilcoxonP, WilcoxonZ; //,WilcoxonGL,
    //*************Fazere Funcoes de Retorno*************
    //Para Spearman
    double SpearmanT,SpearmanP,SpearmanRs;
    //Para Pearson
```

```
    double PearsonR;
    //Para TesteTPares
    double TesteTParesT,TesteTParesGL,TesteTParesP;
    //Para TesteT
    double TesteTT, TesteTGL, TesteTP;
    //Para MannWhitney
    double
MannWhitneySoma1,MannWhitneySoma2,MannWhitneyn1,MannWhitneyn2,MannWhitneyZ,Ma
nnWhitneyP;
    //Para TesteTSeparate
    double TesteTSeparateT, TesteTSeparateGL,TesteTSeparateP;
    //Para Anova
    double AnovaF,
AnovaP,AnovaGLT,AnovaGLE,AnovaSQT,AnovaSQE,AnovaQMT,AnovaQME;
}
```

Base_de_Dados.java

```
import java.sql.*;

public class Base_de_Dados {
    public Base_de_Dados(){
        connection = null;
        try
        {
        // Carregando o JDBC Driver
        String driverName = "com.mysql.jdbc.Driver"; // MySQL MM JDBC driver
        Class.forName(driverName);

        // Criando a conexão com o Banco de Dados
        String serverName = "localhost";
        String mydatabase = "sestatnet";
        String url = "jdbc:mysql://" + serverName + "/" + mydatabase; // a
JDBC url
        String username = "root";
        String password = "";
        connection = DriverManager.getConnection(url, username, password);
        } catch (ClassNotFoundException e)
        {
        //Driver não encontrado
        System.out.println("O driver expecificado não foi encontrado.");
        } catch (SQLException e)
        {
        //Não está conseguindo se conectar ao banco
        System.out.println("Não foi possível conectar ao Banco de Dados");
        }
    }

    public  String RetornaUltimaVizualizacao(int __id, String __nome_arquivo)
{
        try {
            ExecutaSql("SELECT ultima_visualizacao from tabela_arquivo WHERE
id_usuario = \"" +__id  + "\" AND nome_arquivo = \"" + __nome_arquivo +
"\"");
            table.first();
            return table.getString(1);
        } catch (SQLException e) {
            e.printStackTrace();  //To change body of catch statement use
Options | File Templates.
            return "";
        }
    }

    public  String RetornaCriacao(int __id, String __nome_arquivo) {
        try {
            ExecutaSql("SELECT criacao from tabela_arquivo WHERE id_usuario =
\"" + __id + "\" AND nome_arquivo = \"" + __nome_arquivo + "\"");
            table.first();
            return table.getString(1);
        } catch (SQLException e) {
            e.printStackTrace();  //To change body of catch statement use
Options | File Templates.
            return null;
        }
    }

    public  String[] RetornaNomeArquivo(int __id) {
```

```
        try {
            ExecutaSql("SELECT nome_arquivo from tabela_arquivo WHERE
id_usuario = \"" + __id + "\"");
            int rowCount = table.last() ? table.getRow() : 0;
            table.first();
            String[] temp = new String[rowCount];
            temp[0] = table.getString(1);
            int i = 1;
            while(table.next()){
                temp[i] = table.getString(1);
                i++;
            }
            return temp;
        } catch (SQLException e) {
            e.printStackTrace();  //To change body of catch statement use
Options | File Templates.
            return null;
        }
    }

    public  String RetornaEndereco(int __id) {
        try {
            ExecutaSql("SELECT endereco from tabela_arquivo WHERE id_usuario
= \""+__id+"\"");
            table.first();
            return table.getString(1);
        } catch (SQLException e) {
            e.printStackTrace();  //To change body of catch statement use
Options | File Templates.
            return null;
        }
    }

    public  String[] RetornaDescricao(int __id) {
        try {
            ExecutaSql("SELECT descricao from tabela_arquivo WHERE id_usuario
= \""+__id+"\"");
            int rowCount = table.last() ? table.getRow() : 0;
            table.first();
            String[] temp = new String[rowCount];
            temp[0] = table.getString(1);
            int i = 1;
            while(table.next()){
               temp[i] = table.getString(1);
               i++;
            }
            return temp;
        } catch (SQLException e) {
            e.printStackTrace();  //To change body of catch statement use
Options | File Templates.
            return null;
        }
    }

    public  boolean InsereDadosArquivo(int __id, String __endereco, String
__nome_arquivo, String __descricao, String __criacao, String
__ultimaVizualizacao){
        try {
            ExecutaSql("insert into
tabela_arquivo(id_usuario,endereco,nome_arquivo,descricao,criacao,ultima_visu
```

```
alizacao) values ("+__id+",\""+ __endereco+"\",\""+ __nome_arquivo+"\",\""+
__descricao+"\","+ __criacao + "," + __ultimaVizualizacao + ")");
            return true;
        } catch (SQLException e) {
            e.printStackTrace();  //To change body of catch statement use
Options | File Templates.
            return false;
        }
    }

    /**
     * Caso Volte -1 é que o usuário nao consta ou a senha está incorreta.
     * Caso volte algum valor ideferente deste, esse valor indica o id do
usuario.
     */
    public  int ConsultaUsuario(String __login, String __senha) {
        try {
            ExecutaSql("select id_usuario from tabela_usuario where login =
\"" + __login + "\" and senha = \"" + __senha + "\"");
            table.first();
            return table.getInt(1);
        } catch (SQLException e) {
          //  e.printStackTrace();  //To change body of catch statement use
Options | File Templates.
            return -1;
        }
    }

    public  boolean InsereUsuario(String __login, String __senha){
        int i = 0;
        try {
            ExecutaSql("select id_usuario from tabela_usuario");
            while(table.next()){
               i = table.getInt(1);
            }
            i++;
        } catch (SQLException e) {
            e.printStackTrace();  //To change body of catch statement use
Options | File Templates.
            return false;
        }
        try {
            ExecutaSql("insert into tabela_usuario(login,senha,id_usuario)
values (\"" + __login + "\",\"" + __senha + "\", " + i + ")");
            return true;
        } catch (SQLException e) {
            e.printStackTrace();  //To change body of catch statement use
Options | File Templates.
            return false;
        }
    }

    protected  void ExecutaSql(String __sql) throws SQLException {
        Statement s = connection.createStatement();
           table = s.executeQuery(__sql);
    }

    private Connection connection;
    private ResultSet table;
}
```

Educador.java

```
import java.util.Enumeration;

public class Educador {

    public Educador() {

    }

    public void destrua_se() {
    }

    public void init(double[] __PGlobais, double[] __PIndividuais) {
        HistoricosIndividuais = new THistoricoIndividual[0];
        HistoricosDuasADuas = new THistoricoDuasADuas[0];
        PorcentagensMinimasGlobais = new double[8];
        PorcentagensMinimasIndividuais = new double[8];
        Acertos = new long[8];
        Erros = new long[8];
        for(int i=1; i<= 7; i++){
            Acertos[i] = 0;
            Erros[i] = 0;
            PorcentagensMinimasGlobais[i] = __PGlobais[i-1];
            PorcentagensMinimasIndividuais[i] = __PIndividuais[i-1];
         }
    }


    public boolean ConsultaTipo(String __nome, String
__TipoVariavel_TipoEscolhido, Objeto __Correto, Objeto __Permitido) {
        Objeto j = new Objeto(0);
        if(ExisteNoIndividual(__nome,j)==false){
            return false;
        }

if(("naosabe".equals(HistoricosIndividuais[j.intValue()].parametrosReconhecid
os.Tipo)) ||

(HistoricosIndividuais[j.intValue()].parametrosReconhecidos.Tipo.equals(__Tip
oVariavel_TipoEscolhido))){
             __Correto.setBoolean(true);
             __Permitido.setBoolean(true);
        }else{
            __Correto.setBoolean(false);
            if(("quanti".equals(__TipoVariavel_TipoEscolhido)) && (!
HistoricosIndividuais[j.intValue()].PodeSerQuanti)){
                __Permitido.setBoolean(false);
            }else{
                __Permitido.setBoolean(true);
            }
        }
          return true;
    }

    public void operation1() {
    }

    public boolean ConsultaNormalidade(String __Nome, double __Nivel, int
__simnaonaosei_Norm, Objeto __Correto) {
        Objeto j = new Objeto(0);
```

```
        if(ExisteNoIndividual(__Nome,j)==false){
           return false;
        }

if(HistoricosIndividuais[j.intValue()].parametrosReconhecidos.Normalidade<0){
           __Correto.setBoolean(false);
        }else{

if(((HistoricosIndividuais[j.intValue()].parametrosReconhecidos.Normalidade>_
_Nivel) && (__simnaonaosei_Norm==sim)) ||

((HistoricosIndividuais[j.intValue()].parametrosReconhecidos.Normalidade<=__N
ivel) && (__simnaonaosei_Norm==nao))){
              __Correto.setBoolean(true);
           }else{
              __Correto.setBoolean(false);
           }
        }
        return true;
    }

    public boolean ConsultaNumCategorias(String __Nome, long __NumCategorias,
Objeto __Correto) {
        Objeto j = new Objeto(0);
        if(ExisteNoIndividual(__Nome,j)==false){
           return false;
        }

if((HistoricosIndividuais[j.intValue()].parametrosReconhecidos.NumeroCategori
as==0) ||

(HistoricosIndividuais[j.intValue()].parametrosReconhecidos.NumeroCategorias=
= __NumCategorias)){
           __Correto.setBoolean(true);
        }else{
           __Correto.setBoolean(false);
        }
        return true;
    }

    public boolean ConsultaMensuracao(String __Nome, String
__MensuracaoVariavel_MensuracaoEscolhida, Objeto __Correto) {
        Objeto j = new Objeto(0);
        if(ExisteNoIndividual(__Nome,j)==false){
            return false;
        }

if(("nenhuma".equals(HistoricosIndividuais[j.intValue()].parametrosReconhecid
os.Mensuracao)) ||

(HistoricosIndividuais[j.intValue()].parametrosReconhecidos.Mensuracao.equals
( __MensuracaoVariavel_MensuracaoEscolhida))){
           __Correto.setBoolean(true);
        }else{
            __Correto.setBoolean(false);
        }
        return true;
    }

    public boolean ModifiquePorcentagensMinimasIndividuais(double[]
__PIndividuais) {
```

```
       if(__PIndividuais.length !=7){
          return false;
       }else{
          for(int i=1; i<=7;i++){
             PorcentagensMinimasIndividuais[i] = __PIndividuais[i-1];
          }
          return true;
       }
    }

    public boolean ModifiquePorcentagensMinimasGlobais(double[] __PGlobais) {
      if(__PGlobais.length !=7){
         return false;
      }else{
         for(int i=1; i<= 7; i++){
           PorcentagensMinimasGlobais[i] = __PGlobais[i-1];
         }
         return true;
      }
    }


    public boolean ArmazenaTipo(String __Nome, String
__TipoVariavel_TipoEscolhido){
        long Tamanho;
        Objeto j = new Objeto(0);
        if(ExisteNoIndividual(__Nome,j)==false){
           return false;
        }
        Tamanho =
HistoricosIndividuais[j.intValue()].parametrosUsuario.Tipo.length;
        String[] temp = new String[(int)Tamanho+1];
        for(int i=0; i<Tamanho; i++){
            temp[i] =
HistoricosIndividuais[j.intValue()].parametrosUsuario.Tipo[i];
        }
        HistoricosIndividuais[j.intValue()].parametrosUsuario.Tipo = temp;

//SetLength(HistoricosIndividuais[j].ParametrosUsuario.Tipo,Tamanho+1);

HistoricosIndividuais[j.intValue()].parametrosUsuario.Tipo[(int)Tamanho] =
__TipoVariavel_TipoEscolhido;

if(!"naosabe".equals(HistoricosIndividuais[j.intValue()].parametrosReconhecid
os.Tipo)){

if(HistoricosIndividuais[j.intValue()].parametrosReconhecidos.Tipo.equals(__T
ipoVariavel_TipoEscolhido)){
             Acertos[1] = Acertos[1]+1;
             HistoricosIndividuais[j.intValue()].Acertos[1] =
HistoricosIndividuais[j.intValue()].Acertos[1]+1;
             if(HistoricosIndividuais[j.intValue()].TipoAcumulados[1]==sim){
                HistoricosIndividuais[j.intValue()].Acumulados[1] =
HistoricosIndividuais[j.intValue()].Acumulados[1]+1;
             }else{
                HistoricosIndividuais[j.intValue()].Acumulados[1] = 1;
                HistoricosIndividuais[j.intValue()].TipoAcumulados[1] = sim;
             }
          }else{
             Erros[1] = Erros[1]+1;
```

```
              HistoricosIndividuais[j.intValue()].Erros[1] =
HistoricosIndividuais[j.intValue()].Erros[1]+1;
              if(HistoricosIndividuais[j.intValue()].TipoAcumulados[1]==nao){
                 HistoricosIndividuais[j.intValue()].Acumulados[1] =
HistoricosIndividuais[j.intValue()].Acumulados[1]+1;
              }else{
                 HistoricosIndividuais[j.intValue()].Acumulados[1] = 1;
                 HistoricosIndividuais[j.intValue()].TipoAcumulados[1] = nao;
              }
           }
        }
        return true;
    }

    public boolean ArmazenaMensuracao(String __Nome, String
__MensuracaoVariavel_MensuracaoEscolhida){
        long Tamanho;
        Objeto j = new Objeto(0);
        if(ExisteNoIndividual(__Nome,j)==false){
            return false;
        }
        Tamanho =
HistoricosIndividuais[j.intValue()].parametrosUsuario.Mensuracao.length;
        String[] temp = new String[(int)Tamanho+1];
        for(int i=0; i< Tamanho; i++){
            temp[i] =
HistoricosIndividuais[j.intValue()].parametrosUsuario.Mensuracao[i];
        }
        HistoricosIndividuais[j.intValue()].parametrosUsuario.Mensuracao =
temp;

//SetLength(HistoricosIndividuais[j.intValue()].ParametrosUsuario.Mensuracao,
Tamanho+1);

HistoricosIndividuais[j.intValue()].parametrosUsuario.Mensuracao[(int)Tamanho
] = __MensuracaoVariavel_MensuracaoEscolhida;

if(!"nenhuma".equals(HistoricosIndividuais[j.intValue()].parametrosReconhecid
os.Mensuracao)){

if(HistoricosIndividuais[j.intValue()].parametrosReconhecidos.Mensuracao.equa
ls(__MensuracaoVariavel_MensuracaoEscolhida)){
              Acertos[2] = Acertos[2]+1;
              HistoricosIndividuais[j.intValue()].Acertos[2] =
HistoricosIndividuais[j.intValue()].Acertos[2]+1;
              if(HistoricosIndividuais[j.intValue()].TipoAcumulados[2]==sim){
                 HistoricosIndividuais[j.intValue()].Acumulados[2] =
HistoricosIndividuais[j.intValue()].Acumulados[2]+1;
              }else{
                 HistoricosIndividuais[j.intValue()].Acumulados[2] = 1;
                 HistoricosIndividuais[j.intValue()].TipoAcumulados[2] = sim;
              }
           }else{
              Erros[2] = Erros[2]+1;
              HistoricosIndividuais[j.intValue()].Erros[2] =
HistoricosIndividuais[j.intValue()].Erros[2]+1;
              if(HistoricosIndividuais[j.intValue()].TipoAcumulados[2]==nao){
                 HistoricosIndividuais[j.intValue()].Acumulados[2] =
HistoricosIndividuais[j.intValue()].Acumulados[2]+1;
              }else{
                 HistoricosIndividuais[j.intValue()].Acumulados[2] = 1;
```

```
                HistoricosIndividuais[j.intValue()].TipoAcumulados[2] = nao;
             }
          }
       }
       return true;
    }

    public boolean ArmazenaNumCategorias(String __Nome, long
__NumCategorias){
       long Tamanho;
       Objeto j = new Objeto(0);
       if(ExisteNoIndividual(__Nome,j)==false){
          return false;
       }
       Tamanho =
HistoricosIndividuais[j.intValue()].parametrosUsuario.NumeroCategorias.length
;
       long[] temp = new long[(int)Tamanho+1];
       for(int i=0; i< Tamanho; i++){
           temp[i] =
HistoricosIndividuais[j.intValue()].parametrosUsuario.NumeroCategorias[i];
       }

HistoricosIndividuais[j.intValue()].parametrosUsuario.NumeroCategorias =
temp;

//SetLength(HistoricosIndividuais[j.intValue()].ParametrosUsuario.NumeroCateg
orias,Tamanho+1);

HistoricosIndividuais[j.intValue()].parametrosUsuario.NumeroCategorias[(int)T
amanho] = __NumCategorias;

if(HistoricosIndividuais[j.intValue()].parametrosReconhecidos.NumeroCategoria
s !=0){

if(HistoricosIndividuais[j.intValue()].parametrosReconhecidos.NumeroCategoria
s==__NumCategorias){
               Acertos[3] = Acertos[3]+1;
               HistoricosIndividuais[j.intValue()].Acertos[3] =
HistoricosIndividuais[j.intValue()].Acertos[3]+1;

if(HistoricosIndividuais[j.intValue()].TipoAcumulados[3]==sim){
                  HistoricosIndividuais[j.intValue()].Acumulados[3] =
HistoricosIndividuais[j.intValue()].Acumulados[3]+1;
               }else{
                  HistoricosIndividuais[j.intValue()].Acumulados[3] = 1;
                  HistoricosIndividuais[j.intValue()].TipoAcumulados[3] =
sim;
               }
           }else{
              Erros[3] = Erros[3]+1;
              HistoricosIndividuais[j.intValue()].Erros[3] =
HistoricosIndividuais[j.intValue()].Erros[3]+1;
              if(HistoricosIndividuais[j.intValue()].TipoAcumulados[3]==nao){

HistoricosIndividuais[j.intValue()].Acumulados[3]=HistoricosIndividuais[j.int
Value()].Acumulados[3]+1;
               }else{
                  HistoricosIndividuais[j.intValue()].Acumulados[3] = 1;
                  HistoricosIndividuais[j.intValue()].TipoAcumulados[3] = nao;
               }
```

```
            }
        }
        return true;
    }

    public boolean ArmazenaNormalidade(String __Nome, double __Nivel, int
__simnaonaosei_Norm){
        long Tamanho;
        Objeto j = new Objeto(0);
        if(ExisteNoIndividual(__Nome,j)==false){
            return false;
        }
        Tamanho =
HistoricosIndividuais[j.intValue()].parametrosUsuario.Normalidade.length;
        EscolhasDepNivelSignif[] temp = new
EscolhasDepNivelSignif[(int)Tamanho+1];
        temp[(int)Tamanho] = new EscolhasDepNivelSignif();
        for(int i=0; i<Tamanho; i++){
            temp[i] =
HistoricosIndividuais[j.intValue()].parametrosUsuario.Normalidade[i];
        }

//SetLength(HistoricosIndividuais[j.intValue()].ParametrosUsuario.Normalidade
,Tamanho+1);
        HistoricosIndividuais[j.intValue()].parametrosUsuario.Normalidade =
temp;

HistoricosIndividuais[j.intValue()].parametrosUsuario.Normalidade[(int)Tamanh
o].Significancia = __Nivel;

HistoricosIndividuais[j.intValue()].parametrosUsuario.Normalidade[(int)Tamanh
o].opcao = __simnaonaosei_Norm;

if(HistoricosIndividuais[j.intValue()].parametrosReconhecidos.Normalidade>=0)
{

if(((HistoricosIndividuais[j.intValue()].parametrosReconhecidos.Normalidade>_
_Nivel) && (__simnaonaosei_Norm==sim)) ||

((HistoricosIndividuais[j.intValue()].parametrosReconhecidos.Normalidade<=__N
ivel) && (__simnaonaosei_Norm==nao))){
                Acertos[4] = Acertos[4]+1;
                HistoricosIndividuais[j.intValue()].Acertos[4] =
HistoricosIndividuais[j.intValue()].Acertos[4]+1;

if(HistoricosIndividuais[j.intValue()].TipoAcumulados[4]==sim){
                    HistoricosIndividuais[j.intValue()].Acumulados[4] =
HistoricosIndividuais[j.intValue()].Acumulados[4]+1;
                }else{
                    HistoricosIndividuais[j.intValue()].Acumulados[4] = 1;
                    HistoricosIndividuais[j.intValue()].TipoAcumulados[4] =
sim;
                }
            }else{
                Erros[4] = Erros[4]+1;
                HistoricosIndividuais[j.intValue()].Erros[4] =
HistoricosIndividuais[j.intValue()].Erros[4]+1;

if(HistoricosIndividuais[j.intValue()].TipoAcumulados[4]==nao){
                    HistoricosIndividuais[j.intValue()].Acumulados[4] =
HistoricosIndividuais[j.intValue()].Acumulados[4]+1;
```

```
                }else{
                   HistoricosIndividuais[j.intValue()].Acumulados[4] = 1;
                   HistoricosIndividuais[j.intValue()].TipoAcumulados[4] =
nao;
                }
             }
          }
          return true;
     }

     public boolean AutomatizaTipo(String __Nome, Objeto
__TipoVariavel_TipoUsuario, Objeto __Automatizar){
          //  {Contador,Tamanho,i,}j:Longword;
          //  {ultimoTipo:TipoVariavel;
          //  fim:boolean;}
          Objeto j = new Objeto(0);
          double pglo,pind;
          __TipoVariavel_TipoUsuario.setString("naosabe");
          if(ExisteNoIndividual(__Nome,j)==false){
             return false;
          }
          if(((Acertos[1]+Erros[1])==0) ||

(HistoricosIndividuais[j.intValue()].Acertos[1]+HistoricosIndividuais[j.intVa
lue()].Erros[1]==0)){
               __Automatizar.setBoolean(false);
               return true;
          }
          pglo = ((Acertos[1]*100)/(Acertos[1]+Erros[1]));
          pind =
(HistoricosIndividuais[j.intValue()].Acertos[1]*100)/(HistoricosIndividuais[j
.intValue()].Acertos[1]+HistoricosIndividuais[j.intValue()].Erros[1]);
          if((pglo>=PorcentagensMinimasGlobais[1]) &&
             (pind>=PorcentagensMinimasIndividuais[1])){
               __Automatizar.setBoolean(true);

__TipoVariavel_TipoUsuario.setString(HistoricosIndividuais[j.intValue()].para
metrosReconhecidos.Tipo);
          }else{
             __Automatizar.setBoolean(false);
          }
          return true;
}

     //Verificar se está retornando _mensuracaoVariavel!!!!
     public boolean AutomatizaMensuracao(String __Nome, Objeto
__MensuracaoVariavel_MensuracaoUsuario, Objeto __Automatizar){
          //  {Contador,Tamanho,i,}j:Longword;
          //  {ultimaMens:MensuracaoVariavel;
          //  fim:boolean;}
          Objeto j = new Objeto(0);
          double pglo,pind;
          __MensuracaoVariavel_MensuracaoUsuario.setString("nenhuma");
          if(ExisteNoIndividual(__Nome,j)==false){
             return false;
          }
          if((Acertos[2]+Erros[2]==0) ||

(HistoricosIndividuais[j.intValue()].Acertos[2]+HistoricosIndividuais[j.intVa
lue()].Erros[2]==0)){
               __Automatizar.setBoolean(false);
```

```
            return true;
        }
        pglo = (Acertos[2]*100)/(Acertos[2]+Erros[2]);
        pind =
(HistoricosIndividuais[j.intValue()].Acertos[2]*100)/(HistoricosIndividuais[j
.intValue()].Acertos[2]+HistoricosIndividuais[j.intValue()].Erros[2]);
        if((pglo>=PorcentagensMinimasGlobais[2]) &&
           (pind>=PorcentagensMinimasIndividuais[2])){
             __Automatizar.setBoolean(true);

__MensuracaoVariavel_MensuracaoUsuario.setString(HistoricosIndividuais[j.intV
alue()].parametrosReconhecidos.Mensuracao);
        }else{
           __Automatizar.setBoolean(false);
        }
        return true;
   }

    public boolean AutomatizaNumCategorias(String __Nome, Objeto
__NumCategorias, Objeto __Automatizar){
        Objeto j = new Objeto(0);
        double pglo,pind;
          __NumCategorias.setLong(0);
          if(ExisteNoIndividual(__Nome,j)==false){
             return false;
          }
          if((Acertos[3]+Erros[3]==0) ||

(HistoricosIndividuais[j.intValue()].Acertos[3]+HistoricosIndividuais[j.intVa
lue()].Erros[3]==0)){
               __Automatizar.setBoolean(false);
               return true;
          }
          pglo = (Acertos[3]*100)/(Acertos[3]+Erros[3]);
          pind =
(HistoricosIndividuais[j.intValue()].Acertos[3]*100)/(HistoricosIndividuais[j
.intValue()].Acertos[3]+HistoricosIndividuais[j.intValue()].Erros[3]);
          if((pglo>=PorcentagensMinimasGlobais[3]) &&
             (pind>=PorcentagensMinimasIndividuais[3])){
               __Automatizar.setBoolean(true);

__NumCategorias.setLong(HistoricosIndividuais[j.intValue()].parametrosReconhe
cidos.NumeroCategorias);
          }else{
             __Automatizar.setBoolean(false);
          }
          return true;

    }

    public boolean AutomatizaNormalidade(String __Nome, double Nivel, Objeto
__simnaonaosei_Normalidade, Objeto __Automatizar){
        Objeto j = new Objeto(0);
        double pglo,pind;
        //  {Contador,Tamanho,i,}j:Longword;
        //  {fim,existe:boolean;}

          __simnaonaosei_Normalidade.setInt(naosei);
          if(ExisteNoIndividual(__Nome,j)==false){
              return false;
          }
```

```
          if((Acertos[4]+Erros[4]==0) ||

(HistoricosIndividuais[j.intValue()].Acertos[4]+HistoricosIndividuais[j.intVa
lue()].Erros[4]==0)){
               __Automatizar.setBoolean(false);
               return true;
          }
          pglo = (Acertos[4]*100)/(Acertos[4]+Erros[4]);
          pind =
(HistoricosIndividuais[j.intValue()].Acertos[4]*100)/(HistoricosIndividuais[j
.intValue()].Acertos[4]+HistoricosIndividuais[j.intValue()].Erros[4]);
          if((pglo>=PorcentagensMinimasGlobais[4]) &&
             (pind>=PorcentagensMinimasIndividuais[4])){

if(HistoricosIndividuais[j.intValue()].parametrosReconhecidos.Normalidade<0){
                  __Automatizar.setBoolean(false);
               }else{
                   __Automatizar.setBoolean(true);

if((HistoricosIndividuais[j.intValue()].parametrosReconhecidos.Normalidade>Ni
vel)){
                      __simnaonaosei_Normalidade.setInt(sim);
                  }else{
                     __simnaonaosei_Normalidade.setInt(nao);
                  }
               }
          }else{
             __Automatizar.setBoolean(false);
          }
          return true;
   }


    /*
     * @stereotype Funcões que Consultam a Funções de auxílio (Zerar e
Inserir)
    */
    public boolean InsereNoHistoricoIndividual(String __Nome, boolean __PSQ,
String __TipoVariavel_Tipo, String __mensuracaoVariavel__Mens, long __numCat,
double __Norm){
        long Tamanho;
        boolean Achou;
        Tamanho = HistoricosIndividuais.length;
        Achou = false;
        if(Tamanho>0){
          for(int i=0; i<=Tamanho-1; i++){
              if(HistoricosIndividuais[i].Nome.equals(__Nome)){
                 Achou=true;
              }
          }
          if(Achou){
             return false;
          }
        }
        THistoricoIndividual[] tempHistoricosIndividuais = new
THistoricoIndividual[(int)Tamanho+1];
        tempHistoricosIndividuais[(int)Tamanho] = new THistoricoIndividual();
        for(int i=0; i<HistoricosIndividuais.length; i++){
            tempHistoricosIndividuais[i] = HistoricosIndividuais[i];
        }
        HistoricosIndividuais = tempHistoricosIndividuais;
```

```
        HistoricosIndividuais[(int)Tamanho].Nome = __Nome;
        HistoricosIndividuais[(int)Tamanho].PodeSerQuanti = __PSQ;
        HistoricosIndividuais[(int)Tamanho].parametrosReconhecidos.Tipo =
__TipoVariavel_Tipo;
        HistoricosIndividuais[(int)Tamanho].parametrosReconhecidos.Mensuracao
= __mensuracaoVariavel__Mens;

HistoricosIndividuais[(int)Tamanho].parametrosReconhecidos.NumeroCategorias =
__numCat;

HistoricosIndividuais[(int)Tamanho].parametrosReconhecidos.Normalidade =
__Norm;
        HistoricosIndividuais[(int)Tamanho].parametrosUsuario.Tipo = new
String[0];
        HistoricosIndividuais[(int)Tamanho].parametrosUsuario.Mensuracao =
new String[0];

HistoricosIndividuais[(int)Tamanho].parametrosUsuario.NumeroCategorias = new
long[0];
        HistoricosIndividuais[(int)Tamanho].parametrosUsuario.Normalidade =
new EscolhasDepNivelSignif[0];
        for(int i=1; i<= 4; i++){
          HistoricosIndividuais[(int)Tamanho].Acertos[i] =0;
          HistoricosIndividuais[(int)Tamanho].Erros[i] = 0;
          HistoricosIndividuais[(int)Tamanho].Acumulados[i] = 0;
          HistoricosIndividuais[(int)Tamanho].TipoAcumulados[i] = naosei;
        }
        return true;
    }

    public void ZeraHistoricoIndividual(){
        HistoricosIndividuais = new THistoricoIndividual[0];
    }


  // 2) Históricos Duas a Duas
      // 2.1) Simplesmente consultam
    public boolean Consulta2a2Dependencia(String __NomeVI, String __NomeVD,
int __SimNaoNaoSei_dep, Objeto __Correto){
      Objeto j = new Objeto(0);
      if(ExisteNo2a2(__NomeVI,__NomeVD,j)==false){
         return false;
      }

if((HistoricosDuasADuas[j.intValue()].parametrosReconhecidos.Dependencia==nao
sei) ||

(HistoricosDuasADuas[j.intValue()].parametrosReconhecidos.Dependencia==
__SimNaoNaoSei_dep)){
         __Correto.setBoolean(true);
      }else{
         __Correto.setBoolean(false);
      }
      return true;
    }

    public boolean Consulta2a2Paridade(String __NomeVI, String __NomeVD, int
__SimNaoNaoSei_par, Objeto __Correto){
        Objeto j = new Objeto(0);
        if(ExisteNo2a2(__NomeVI,__NomeVD,j)==false){
           return false;
```

```
        }

if((HistoricosDuasADuas[j.intValue()].parametrosReconhecidos.Paridade==naosei
) ||

(HistoricosDuasADuas[j.intValue()].parametrosReconhecidos.Paridade==__SimNaoN
aoSei_par)){
            __Correto.setBoolean(true);
        }else{
            __Correto.setBoolean(false);
        }
        return true;
    }

    public boolean Consulta2a2Homocedasticidade(String __NomeVI, String
__NomeVD, double __Nivel, int __SimNaoNaoSei_hom, Objeto __Correto){
        Objeto j = new Objeto(0);
        if(ExisteNo2a2(__NomeVI,__NomeVD,j)==false){
            return false;
        }

if(((HistoricosDuasADuas[j.intValue()].parametrosReconhecidos.Homocedasticida
de>__Nivel) && (__SimNaoNaoSei_hom==sim)) ||

((HistoricosDuasADuas[j.intValue()].parametrosReconhecidos.Homocedasticidade<
=__Nivel) && (__SimNaoNaoSei_hom==nao)) ||

(HistoricosDuasADuas[j.intValue()].parametrosReconhecidos.Homocedasticidade<0
)){
            __Correto.setBoolean(true);
        }
        else{
            __Correto.setBoolean(false);
        }
        return true;
    }

    public boolean Consulta2a2Normalidade(String __NomeVI, String __NomeVD,
double __Nivel, int __SimNaoNaoSei_norm, Objeto __Correto){
        Objeto j = new Objeto(0);
        if( ExisteNo2a2(__NomeVI,__NomeVD,j)==false){
           return false;
        }

if(((HistoricosDuasADuas[j.intValue()].parametrosReconhecidos.Normalidade>__N
ivel) && (__SimNaoNaoSei_norm == sim)) ||

((HistoricosDuasADuas[j.intValue()].parametrosReconhecidos.Normalidade<=__Niv
el) && (__SimNaoNaoSei_norm == nao)) ||

(HistoricosDuasADuas[j.intValue()].parametrosReconhecidos.Normalidade<0)){
            __Correto.setBoolean(true);
        }else{
            __Correto.setBoolean(false);
        }
        return true;
    }

    // 2.2) Armazenam uma escolha do usuário
    public boolean Armazena2a2Dependencia(String __NomeVI, String __NomeVD,
int __SimNaoNaoSei_dep){
```

```
        long Tamanho;
        Objeto j = new Objeto(0);
        if(ExisteNo2a2(__NomeVI,__NomeVD,j)==false){
           return false;
        }
        Tamanho =
HistoricosDuasADuas[j.intValue()].parametrosUsuario.Dependencia.length;
        int[] temp = new int[(int)Tamanho+1];
        for(int i=0; i< Tamanho; i++){
            temp[i] =
HistoricosDuasADuas[j.intValue()].parametrosUsuario.Dependencia[i];
        }
        HistoricosDuasADuas[j.intValue()].parametrosUsuario.Dependencia =
temp;

//SetLength(HistoricosDuasADuas[j].ParametrosUsuario.Dependencia,Tamanho+1);

HistoricosDuasADuas[j.intValue()].parametrosUsuario.Dependencia[(int)Tamanho]
= __SimNaoNaoSei_dep;

if(HistoricosDuasADuas[j.intValue()].parametrosReconhecidos.Dependencia !=
naosei){

if(HistoricosDuasADuas[j.intValue()].parametrosReconhecidos.Dependencia==__Si
mNaoNaoSei_dep){
              Acertos[5] = Acertos[5]+1;
           }else{
                Erros[5] = Erros[5]+1;
           }
        }
        return true;
    }

    public boolean Armazena2a2Paridade(String __NomeVI, String __NomeVD, int
__SimNaoNaoSei_par){
        long Tamanho;
        Objeto j = new Objeto(0);
        if(ExisteNo2a2(__NomeVI,__NomeVD,j)==false){
            return false;
        }
        Tamanho =
HistoricosDuasADuas[j.intValue()].parametrosUsuario.Paridade.length;
        int[] temp = new int[(int)Tamanho+1];
        for(int i =0; i< Tamanho; i++){
            temp[i] =
HistoricosDuasADuas[j.intValue()].parametrosUsuario.Paridade[i];
        }
        HistoricosDuasADuas[j.intValue()].parametrosUsuario.Paridade = temp;

//SetLength(HistoricosDuasADuas[j].ParametrosUsuario.Paridade,Tamanho+1);

HistoricosDuasADuas[j.intValue()].parametrosUsuario.Paridade[(int)Tamanho] =
__SimNaoNaoSei_par;
        if(HistoricosDuasADuas[j.intValue()].parametrosReconhecidos.Paridade
!= naosei){

if(HistoricosDuasADuas[j.intValue()].parametrosReconhecidos.Paridade==__SimNa
oNaoSei_par){
              Acertos[6] = Acertos[6]+1;
           }else{
              Erros[6] = Erros[6]+1;
```

```
            }
        }
        return true;
    }

    public boolean Armazena2a2Homocedasticidade(String __NomeVI, String
__NomeVD, double __Nivel, int __SimNaoNaoSei_hom){
        long Tamanho;
        Objeto j = new Objeto(0);
        if(ExisteNo2a2(__NomeVI,__NomeVD,j)==false){
           return false;
        }
        Tamanho =
HistoricosDuasADuas[j.intValue()].parametrosUsuario.Homocedasticidade.length;
        EscolhasDepNivelSignif[] temp = new
EscolhasDepNivelSignif[(int)Tamanho+1];
        temp[(int)Tamanho] = new EscolhasDepNivelSignif();
        for(int i=0; i< Tamanho; i++){
            temp[i] =
HistoricosDuasADuas[j.intValue()].parametrosUsuario.Homocedasticidade[i];
        }
        HistoricosDuasADuas[j.intValue()].parametrosUsuario.Homocedasticidade
= temp;

//SetLength(HistoricosDuasADuas[j].ParametrosUsuario.Homocedasticidade,Tamanh
o+1);

HistoricosDuasADuas[j.intValue()].parametrosUsuario.Homocedasticidade[(int)Ta
manho].Significancia =__Nivel;

HistoricosDuasADuas[j.intValue()].parametrosUsuario.Homocedasticidade[(int)Ta
manho].opcao = __SimNaoNaoSei_hom;

if(HistoricosDuasADuas[j.intValue()].parametrosReconhecidos.Homocedasticidade
>=0){

if(((HistoricosDuasADuas[j.intValue()].parametrosReconhecidos.Homocedasticida
de>__Nivel) && (__SimNaoNaoSei_hom==sim)) ||

((HistoricosDuasADuas[j.intValue()].parametrosReconhecidos.Homocedasticidade<
=__Nivel) && (__SimNaoNaoSei_hom==nao))){
              Acertos[7] = Acertos[7]+1;
           }else{
              Erros[7] = Erros[7]+1;
           }
        }
        return true;
    }

    public boolean Armazena2a2Normalidade(String __NomeVI, String __NomeVD,
double __Nivel, int __SimNaoNaoSei_norm){
        long Tamanho;
        Objeto j = new Objeto(0);
        if(ExisteNo2a2(__NomeVI,__NomeVD,j)==false){
           return false;
        }
        Tamanho =
HistoricosDuasADuas[j.intValue()].parametrosUsuario.Normalidade.length;
        EscolhasDepNivelSignif[] temp = new
EscolhasDepNivelSignif[(int)Tamanho+1];
        temp[(int)Tamanho] = new EscolhasDepNivelSignif();
```

```
        for(int i=0; i< Tamanho; i++){
            temp[i] =
HistoricosDuasADuas[j.intValue()].parametrosUsuario.Normalidade[i];
        }
        HistoricosDuasADuas[j.intValue()].parametrosUsuario.Normalidade =
temp;

//SetLength(HistoricosDuasADuas[j.intValue()].parametrosUsuario.Normalidade,T
amanho+1);

HistoricosDuasADuas[j.intValue()].parametrosUsuario.Normalidade[(int)Tamanho]
.Significancia = __Nivel;

HistoricosDuasADuas[j.intValue()].parametrosUsuario.Normalidade[(int)Tamanho]
.opcao = __SimNaoNaoSei_norm;

if(HistoricosDuasADuas[j.intValue()].parametrosReconhecidos.Normalidade>=0){

if(((HistoricosDuasADuas[j.intValue()].parametrosReconhecidos.Normalidade>__N
ivel) && (__SimNaoNaoSei_norm==sim)) ||

((HistoricosDuasADuas[j.intValue()].parametrosReconhecidos.Normalidade<=__Niv
el) && (__SimNaoNaoSei_norm==nao))){
              Acertos[4] = Acertos[4]+1;
           }else{
              Erros[4] = Erros[4]+1;
           }
        }
        return true;
    }

    // 2.3) Verifica se a escolha pode ser automatizada
    public boolean Automatiza2a2Dependencia(String __NomeVI, String __NomeVD,
Objeto __SimNaoNaoSei_dep, Objeto __Automatizar){
        Objeto j = new Objeto(0);
        double pglo,pind;
          __SimNaoNaoSei_dep.setInt(naosei);
          if(ExisteNo2a2(__NomeVI,__NomeVD,j)==false){
             return false;
          }
          if((Acertos[5]+Erros[5]==0) ||

(HistoricosDuasADuas[j.intValue()].Acertos[5]+HistoricosDuasADuas[j.intValue(
)].Erros[5]==0)){
               __Automatizar.setBoolean(false);
               return true;
          }
          pglo = (Acertos[5]*100)/(Acertos[5]+Erros[5]);
          pind =
(HistoricosDuasADuas[j.intValue()].Acertos[5]*100)/(HistoricosDuasADuas[j.int
Value()].Acertos[5]+HistoricosDuasADuas[j.intValue()].Erros[5]);
          if((pglo>=PorcentagensMinimasGlobais[5]) &&
             (pind>=PorcentagensMinimasIndividuais[5])){
               __Automatizar.setBoolean(true);

__SimNaoNaoSei_dep.setInt(HistoricosDuasADuas[j.intValue()].parametrosReconhe
cidos.Dependencia);
          }else{
             __Automatizar.setBoolean(false);
          }
          return true;
```

```
/*Tamanho:=Length(HistoricosDuasADuas[j].ParametrosUsuario.Dependencia);
          if (Tamanho<InsistenciaDependencia) or (Tamanho=0) then
             begin
               Automatizar:=false;
               Automatiza2a2Dependencia:=true;
               exit;
             end;
          i:=Tamanho;

ultimoValor:=HistoricosDuasADuas[j].ParametrosUsuario.Dependencia[i-1];
          contador:=0;
          fim:=false;
          while not(fim) do
            begin
              if HistoricosDuasADuas[j].ParametrosUsuario.Dependencia[i-
1]=ultimoValor then
                 contador:=contador+1
               else
                 fim:=true;
               i:=i-1;
               if i=0 then
                 fim:=true;
            end;
          if contador>=InsistenciaDependencia then
             begin
               Automatizar:=true;
               Insist:=contador;
               dep:=ultimoValor;
             end
          else
             Automatizar:=false;
          Automatiza2a2Dependencia:=true */
    }

    public boolean Automatiza2a2Paridade(String __NomeVI, String __NomeVD,
Objeto __SimNaoNaoSei_par, Objeto __Automatizar){
        Objeto j = new Objeto(0);
        double pglo,pind;
        // {ultimoValor:SimNaoNaoSei;
        //  fim:boolean;}
         //{Contador,Tamanho,i,}j:Longword;
          __SimNaoNaoSei_par.setInt(naosei);
          if(ExisteNo2a2(__NomeVI,__NomeVD,j)==false){
             return false;
          }
          if((Acertos[6]+Erros[6]==0) ||

(HistoricosDuasADuas[j.intValue()].Acertos[6]+HistoricosDuasADuas[j.intValue(
)].Erros[6]==0)){
               __Automatizar.setBoolean(false);
               return true;
          }
          pglo = (Acertos[6]*100)/(Acertos[6]+Erros[6]);
          pind =
(HistoricosDuasADuas[j.intValue()].Acertos[6]*100)/(HistoricosDuasADuas[j.int
Value()].Acertos[6]+HistoricosDuasADuas[j.intValue()].Erros[6]);
          if((pglo>=PorcentagensMinimasGlobais[6]) &&
             (pind>=PorcentagensMinimasIndividuais[6])){
               __Automatizar.setBoolean(true);
```

```
__SimNaoNaoSei_par.setInt(HistoricosDuasADuas[j.intValue()].parametrosReconhe
cidos.Paridade);
          }else{
             __Automatizar.setBoolean(false);
          }
          return true;

/*Tamanho:=Length(HistoricosDuasADuas[j].ParametrosUsuario.Paridade);
          if (Tamanho<InsistenciaParidade) or (Tamanho=0) then
             begin
               Automatizar:=false;
               Automatiza2a2Paridade:=true;
               exit;
             end;
          i:=Tamanho;
          ultimoValor:=HistoricosDuasADuas[j].ParametrosUsuario.Paridade[i-
1];
          contador:=0;
          fim:=false;
          while not(fim) do
            begin
              if HistoricosDuasADuas[j].ParametrosUsuario.Paridade[i-
1]=ultimoValor then
                 contador:=contador+1
               else
                 fim:=true;
               i:=i-1;
               if i=0 then
                 fim:=true;
            end;
          if contador>=InsistenciaParidade then
             begin
               Automatizar:=true;
               Insist:=contador;
               par:=ultimoValor;
             end
          else
             Automatizar:=false;
          Automatiza2a2Paridade:=true*/
    }

    public boolean Automatiza2a2Homocedasticidade(String __NomeVI,String
__NomeVD, double __Nivel, Objeto __SimNaoNaoSei_hom, Objeto __Automatizar){
        Objeto j = new Objeto(0);
        double pglo,pind;
        //{Contador,Tamanho,i,}j:Longword;
        //{ultimoValor:SimNaoNaoSei;
        //fim:boolean;}
        __SimNaoNaoSei_hom.setInt(naosei);
        if(ExisteNo2a2(__NomeVI,__NomeVD,j)==false){
           return false;
        }
        if((Acertos[7]+Erros[7]==0) ||

(HistoricosDuasADuas[j.intValue()].Acertos[7]+HistoricosDuasADuas[j.intValue(
)].Erros[7]==0)){
             __Automatizar.setBoolean(false);
             return true;
        }
        pglo = (Acertos[7]*100)/(Acertos[7]+Erros[7]);
```

```
        pind =
(HistoricosDuasADuas[j.intValue()].Acertos[7]*100)/(HistoricosDuasADuas[j.int
Value()].Acertos[7]+HistoricosDuasADuas[j.intValue()].Erros[7]);
        if((pglo>=PorcentagensMinimasGlobais[7]) &&
           (pind>=PorcentagensMinimasIndividuais[7])){

if(HistoricosDuasADuas[j.intValue()].parametrosReconhecidos.Homocedasticidade
<0){
                __Automatizar.setBoolean(false);
             }else{
                  __Automatizar.setBoolean(true);

if((HistoricosDuasADuas[j.intValue()].parametrosReconhecidos.Homocedasticidad
e>__Nivel)){
                      __SimNaoNaoSei_hom.setInt(sim);
                  }else{
                     __SimNaoNaoSei_hom.setInt(nao);
                  }
             }
        }else{
           __Automatizar.setBoolean(false);
        }
        return true;

/*Tamanho:=Length(HistoricosDuasADuas[j].ParametrosUsuario.Homocedasticidade)
;
          if (Tamanho<InsistenciaHomocedasticidade) or (Tamanho=0) then
             begin
               Automatizar:=false;
               Automatiza2a2Homocedasticidade:=true;
               exit;
             end;
          i:=Tamanho;

ultimoValor:=HistoricosDuasADuas[j].ParametrosUsuario.Homocedasticidade[i-1];
          contador:=0;
          fim:=false;
          while not(fim) do
            begin
              if
HistoricosDuasADuas[j].ParametrosUsuario.Homocedasticidade[i-1]=ultimoValor
then
                 contador:=contador+1
               else
                 fim:=true;
               i:=i-1;
               if i=0 then
                 fim:=true;
            end;
          if contador>=InsistenciaHomocedasticidade then
             begin
               Automatizar:=true;
               Insist:=contador;
               hom:=ultimoValor;
             end
          else
             Automatizar:=false;
          Automatiza2a2Homocedasticidade:=true;*/
    }
```

```
    public boolean Automatiza2a2Normalidade(String __NomeVI,String __NomeVD,
double __Nivel, Objeto __SimNaoNaoSei_norm, Objeto __Automatizar){
        Objeto j = new Objeto(0);
        double pglo,pind;
        //  {Contador,Tamanho,i,}j:Longword;
        //  {ultimoValor:SimNaoNaoSei;
        //  fim:boolean;}
        __SimNaoNaoSei_norm.setInt(naosei);
        if(ExisteNo2a2(__NomeVI,__NomeVD,j)==false){
           return false;
        }
        if((Acertos[4]+Erros[4]==0) ||

(HistoricosDuasADuas[j.intValue()].Acertos[8]+HistoricosDuasADuas[j.intValue(
)].Erros[8]==0)){
             __Automatizar.setBoolean(false);
            return true;
        }
        pglo = (Acertos[4]*100)/(Acertos[4]+Erros[4]);
        pind =
(HistoricosDuasADuas[j.intValue()].Acertos[8]*100)/(HistoricosDuasADuas[j.int
Value()].Acertos[8]+HistoricosDuasADuas[j.intValue()].Erros[8]);
        if((pglo>=PorcentagensMinimasGlobais[4]) &&
           (pind>=PorcentagensMinimasIndividuais[4])){

if(HistoricosDuasADuas[j.intValue()].parametrosReconhecidos.Normalidade<0){
                __Automatizar.setBoolean(false);
             }else{
                __Automatizar.setBoolean(true);

if((HistoricosDuasADuas[j.intValue()].parametrosReconhecidos.Normalidade>__Ni
vel)){
                    __SimNaoNaoSei_norm.setInt(sim);
                }else{
                    __SimNaoNaoSei_norm.setInt(nao);
                }
             }
        }else{
           __Automatizar.setBoolean(false);
        }
        return true;
        /*
{Tamanho:=Length(HistoricosDuasADuas[j].ParametrosUsuario.Normalidade);
          if (Tamanho<InsistenciaNormalidade) or (Tamanho=0) then
             begin
               Automatizar:=false;
               Automatiza2a2Normalidade:=true;
               exit;
             end;
          i:=Tamanho;

ultimoValor:=HistoricosDuasADuas[j].ParametrosUsuario.Normalidade[i-1];
          contador:=0;
          fim:=false;
          while not(fim) do
            begin
              if HistoricosDuasADuas[j].ParametrosUsuario.Normalidade[i-
1]=ultimoValor then
                 contador:=contador+1
              else
                 fim:=true;
```

```
              i:=i-1;
              if i=0 then
                 fim:=true;
            end;
          if contador>=InsistenciaNormalidade then
             begin
               Automatizar:=true;
               Insist:=contador;
               norm:=ultimoValor;
             end
          else
             Automatizar:=false;
          Automatiza2a2Normalidade:=true} */


    }

    // 2.4) Funções de auxílio (Zerar e Inserir)
    public boolean InsereNoHistorico2a2(String __NomeVI,String __NomeVD, int
__SimNaoNaoSei_dep, int __SimNaoNaoSei_par, double __hom, double __norm){
        long Tamanho;
        boolean Achou;
        Tamanho = HistoricosDuasADuas.length;
        Achou = false;
        if(Tamanho>0){
          for(int i=0; i<=Tamanho-1;i++){
             if((HistoricosDuasADuas[i].NomeVI.equals(__NomeVI)) &&
                (HistoricosDuasADuas[i].NomeVD.equals(__NomeVD))){
                 Achou=true;
             }
          }
           if(Achou){
            return false;
           }
        }
        THistoricoDuasADuas[] tempHistoricosDuasADuas = new
THistoricoDuasADuas[(int)Tamanho+1];
        tempHistoricosDuasADuas[(int)Tamanho] = new THistoricoDuasADuas();
        for(int i=0; i< HistoricosDuasADuas.length; i++){
            tempHistoricosDuasADuas[i] = HistoricosDuasADuas[i];
        }
        HistoricosDuasADuas = tempHistoricosDuasADuas;
        //SetLength(HistoricosDuasADuas,Tamanho+1);
        HistoricosDuasADuas[(int)Tamanho].NomeVI = __NomeVI;
        HistoricosDuasADuas[(int)Tamanho].NomeVD = __NomeVD;
        HistoricosDuasADuas[(int)Tamanho].parametrosReconhecidos.Dependencia
= __SimNaoNaoSei_dep;
        HistoricosDuasADuas[(int)Tamanho].parametrosReconhecidos.Paridade =
__SimNaoNaoSei_par;

HistoricosDuasADuas[(int)Tamanho].parametrosReconhecidos.Homocedasticidade =
__hom;
        HistoricosDuasADuas[(int)Tamanho].parametrosReconhecidos.Normalidade
= __norm;
        HistoricosDuasADuas[(int)Tamanho].parametrosUsuario.Dependencia = new
int[0];
        HistoricosDuasADuas[(int)Tamanho].parametrosUsuario.Paridade = new
int[0];
        HistoricosDuasADuas[(int)Tamanho].parametrosUsuario.Homocedasticidade
= new EscolhasDepNivelSignif[0];
```

```
        HistoricosDuasADuas[(int)Tamanho].parametrosUsuario.Normalidade = new
EscolhasDepNivelSignif[0];
        for(int i=5; i<= 8; i++){
            HistoricosDuasADuas[(int)Tamanho].Acertos[i] = 0;
            HistoricosDuasADuas[(int)Tamanho].Erros[i] = 0;
            HistoricosDuasADuas[(int)Tamanho].Acumulados[i] = 0;
            HistoricosDuasADuas[(int)Tamanho].TipoAcumulados[i] = naosei;
        }
        return true;
    }

    public void ZeraHistorico2a2(){
        HistoricosDuasADuas = new THistoricoDuasADuas[0];

    }

    // Diretivas e Estratégias para uma Análise de duas variáveis
    public boolean VerificaObjetoAnalise(ObjetoAnalise __objAnalise, Objeto
__acao, Objeto __metodo){
        String mensQuali; // {,mensQuanti}:MensuracaoVariavel;
        long ncatQuali;
            __metodo.setInt(-1);
            if(("".equals(__objAnalise.NomeVI)) ||
("".equals(__objAnalise.NomeVD))){
                __acao.setInt(1);
                return true;
            }
            if(("naosabe".equals(__objAnalise.TipoVI)) ||
("naosabe".equals(__objAnalise.TipoVD))){
                __acao.setInt(2);
                return true;
            }
            if(("nenhuma".equals(__objAnalise.MensuracaoVI)) ||
("nenhuma".equals(__objAnalise.MensuracaoVD))){
                __acao.setInt(3);
               return true;
            }
            if((("quali".equals(__objAnalise.TipoVI)) &&
("nom".equals(__objAnalise.MensuracaoVI))) &&
              (("quali".equals(__objAnalise.TipoVD)) &&
("nom".equals(__objAnalise.MensuracaoVD)))){
                if(__objAnalise.FreqEsperadasMaiores5==naosei){
                    __acao.setInt(10);
                    return true;
                }
                if(__objAnalise.FreqEsperadasMaiores5==sim){
                    __metodo.setInt(0);
                }else{
                    __metodo.setInt(10);
                }
                __acao.setInt(0);
                return true;
            }
            if((("quali".equals(__objAnalise.TipoVI)) &&
("quali".equals(__objAnalise.TipoVD)))){
              if((("ord".equals(__objAnalise.MensuracaoVI)) &&
("ord".equals(__objAnalise.MensuracaoVD))) &&
                  ((__objAnalise.nCategoriasVI==0) &&
(__objAnalise.nCategoriasVD==0))){
                  __acao.setInt(4);
                  return true;
```

```
                }
                if((("ord".equals(__objAnalise.MensuracaoVI)) &&
("nom".equals(__objAnalise.MensuracaoVD))) &&
                    (__objAnalise.nCategoriasVI==0)){
                      __acao.setInt(5);
                      return true;
                }
                if((("nom".equals(__objAnalise.MensuracaoVI)) &&
("ord".equals(__objAnalise.MensuracaoVD))) &&
                    (__objAnalise.nCategoriasVD==0)){
                      __acao.setInt(6);
                      return true;
                }
            }
            if((("quali".equals(__objAnalise.TipoVI)) &&
("quanti".equals(__objAnalise.TipoVD))) &&
                (__objAnalise.nCategoriasVI==0)){
                 __acao.setInt(5);
                 return true;
            }
            if((("quanti".equals(__objAnalise.TipoVI)) &&
("quali".equals(__objAnalise.TipoVD))) &&
                (__objAnalise.nCategoriasVD==0)){
                  __acao.setInt(6);
                  return true;
            }
                      //Analise Quali-Quali
            if(("quali".equals(__objAnalise.TipoVI)) &&
("quali".equals(__objAnalise.TipoVD))){
                if(("nom".equals(__objAnalise.MensuracaoVI)) ||
                    (("ord".equals(__objAnalise.MensuracaoVI)) &&
(__objAnalise.nCategoriasVI<=4))){
                    if("nom".equals(__objAnalise.MensuracaoVD)){
                        if(__objAnalise.FreqEsperadasMaiores5==naosei){
                            __acao.setInt(10);
                            return true;
                        }
                        if(__objAnalise.FreqEsperadasMaiores5==sim){
                            __metodo.setInt(0);
                        }else{
                            __metodo.setInt(10);
                        }
                        __acao.setInt(0);
                        return true;
                    }else{
                        if(__objAnalise.nCategoriasVD<=4){
                            if(__objAnalise.FreqEsperadasMaiores5==naosei){
                                __acao.setInt(10);
                                 return true;
                            }
                            if(__objAnalise.FreqEsperadasMaiores5==sim){
                                __metodo.setInt(0);
                            }else{
                                __metodo.setInt(10);
                            }
                            __acao.setInt(0);
                            return true;
                        }else{ // >4
                            __acao.setInt(0);
                            __metodo.setInt(1);
                            return true;
```

```
                    }
                }
            }
            if(("ord".equals(__objAnalise.MensuracaoVI)) &&
(__objAnalise.nCategoriasVI>4)){
                if("nom".equals(__objAnalise.MensuracaoVD)){
                    __acao.setInt(0);
                    __metodo.setInt(1);
                   return true;
                }else{
                    if(__objAnalise.nCategoriasVD<=4){
                        __acao.setInt(0);
                        __metodo.setInt(1);
                       return true;
                    }else{ // >4
                        if(__objAnalise.Paridade==naosei){
                            __acao.setInt(7);
                            return true;
                        }
                        if(__objAnalise.Paridade==sim){
                            __acao.setInt(0);
                            __metodo.setInt(2);
                            return true;
                        }
                        if(__objAnalise.Paridade==nao){
                           __acao.setInt(0);
                           __metodo.setInt(3);
                           return true;
                        }
                    }
                }
            }
        }
          // Fim da Analise Quali-Quali

          // Analise Quanti-Quanti
        if(("quanti".equals(__objAnalise.TipoVI)) &&
("quanti".equals(__objAnalise.TipoVD))){
            if(__objAnalise.Paridade==naosei){
                __acao.setInt(7);
                return true;
            }
            if(__objAnalise.Normalidade==naosei){
               __acao.setInt(9);
               return true;
            }
            if(__objAnalise.Paridade==sim){
                if(__objAnalise.Normalidade==sim){
                    __metodo.setInt(5);
                }else{
                    __metodo.setInt(2);
                }
                __acao.setInt(0);
                return true;
            }
            if(__objAnalise.Paridade==nao){
                if(__objAnalise.Normalidade==sim){
                    __metodo.setInt(4);
                }else{
                    __metodo.setInt(3);
                }
```

```
                    __acao.setInt(0);
                    return true;
                }
            }
              // Fim da Analise Quanti-Quanti

            if((("quali".equals(__objAnalise.TipoVI)) &&
("quanti".equals(__objAnalise.TipoVD))) ||
                (("quanti".equals(__objAnalise.TipoVI)) &&
("quali".equals(__objAnalise.TipoVD)))){
                mensQuali = "nenhuma";
                ncatQuali = 0;
                if(("quali".equals(__objAnalise.TipoVI)) &&
("quanti".equals(__objAnalise.TipoVD))){
                    mensQuali = __objAnalise.MensuracaoVI;
                    //mensQuanti:=MensuracaoVD;
                    ncatQuali = __objAnalise.nCategoriasVI;
                }
                if(("quanti".equals(__objAnalise.TipoVI)) &&
("quali".equals(__objAnalise.TipoVD))){
                   mensQuali =  __objAnalise.MensuracaoVD;
                   //mensQuanti:MensuracaoVI;
                   ncatQuali = __objAnalise.nCategoriasVD;
                }

                if((mensQuali.equals("ord")) && (ncatQuali>4)){
                   __acao.setInt(0);
                   __metodo.setInt(3);
                   return true;
                }
                if((mensQuali.equals("nom")) ||
                   ((mensQuali.equals("ord")) && (ncatQuali<=4))){
                     if(__objAnalise.Homocedasticidade==naosei){
                          __acao.setInt(8);
                         return true;
                     }
                     if(ncatQuali==2){
                          if(__objAnalise.Normalidade==naosei){
                             __acao.setInt(9);
                             return true;
                          }
                          if(__objAnalise.Homocedasticidade==sim){
                              if(__objAnalise.Normalidade==sim){
                                  __metodo.setInt(6);
                              }else{
                                  __metodo.setInt(7);
                              }
                          }else{
                              if(__objAnalise.Normalidade==sim){
                                 __metodo.setInt(8);
                              }else{
                                 __metodo.setInt(7);
                              }
                          }
                          __acao.setInt(0);
                         return true;
                     }
                     if(ncatQuali>2){
                          if(__objAnalise.Homocedasticidade==nao){
                             __acao.setInt(0);
                             __metodo.setInt(1);
```

```
                                return true;
                            }else{
                                 if(__objAnalise.Normalidade==naosei){
                                     __acao.setInt(9);
                                     return true;
                                 }
                                 if(__objAnalise.Normalidade==sim){
                                     __metodo.setInt(9);
                                 }else{
                                    __metodo.setInt(1);
                                 }
                                 __acao.setInt(0);
                                 return true;
                            }
                        }
                    }
                }
            return true;
        }


        private boolean ExisteNo2a2(String __NomeVI, String __NomeVD, Objeto
    __posicao) {
            boolean Achou,Repetida;
            long j;
            Achou = false;
            Repetida = false;
            j=0;
            for(int i=0; i<= HistoricosDuasADuas.length-1; i++){
                if((HistoricosDuasADuas[i].NomeVI.equals(__NomeVI)) &&
                   (HistoricosDuasADuas[i].NomeVD.equals(__NomeVD))){
                     if(Achou==true){
                        Repetida = true;
                     }
                     Achou = true;
                     j = i;
                }
            }
            if((Repetida) || (!Achou)){
               return false;
            }else{
               __posicao.setLong(j);
               return true;
            }
        }

        private boolean ExisteNoIndividual(String __nome, Objeto __posicao) {
            boolean Achou,Repetida;
            long j;
            Achou = false;
            Repetida = false;
            j = 0;
            for(int i=0; i<= HistoricosIndividuais.length-1; i++){
                if(__nome.equals(HistoricosIndividuais[i].Nome)){
                   if(Achou==true){
                      Repetida=true;
                   }
                   Achou=true;
                   j=i;
                }
            }
```

```
            if((Repetida) || (!Achou)){
               return false;
            }
            __posicao.setLong(j);
            return true;
        }

        /**
         * { 1: Tipo
         *                  2: Mensuração
         *                  3: Categorias
         *                  4: Normalidade
         *                  5: Dependencia
         *                  6: Paridade
         *                  7: Homocedasticidade}
         */
        private long[] Erros;
        private long[] Acertos;
        private THistoricoDuasADuas[] HistoricosDuasADuas;
        private THistoricoIndividual[] HistoricosIndividuais;
        private double[] PorcentagensMinimasIndividuais;
        private double[] PorcentagensMinimasGlobais;
        private int nao = 0;
        private int sim = 1;
        private int naosei = 2;
        private String TipoVariavel;
}
```

Estatística.java

```
import javax.print.attribute.standard.Media;
import java.util.Random;

/**
 * Possui e fornece todos os métodos estatisticos.
 * (Um para todos os usuários).
 */
public class Estatistica {
    /*
     * @stereotype constructor
     */
    public Estatistica() {
    }

    private  int VerificaParametrosEntrada(VariaveisQualitativas __quali1,
VariaveisQualitativas __quali2) {
        long tamanho;
        if((__quali1 == null) || (__quali2 == null)){
             return(9);
        }else{
            //temp = __quali1.getValores();
            tamanho = __quali1.getValores().length;
            if(__quali1.getValores().length != __quali2.getValores().length){
                return(10);
            }else{
                if(tamanho==0){
                  return(1);
                }else{
                  tamanho = __quali1.getNumCategorias();
                  if(tamanho != __quali1.getCategorias().length){
                    return(18);
                  }else{
                    if(tamanho==0){
                        return(12);
                    }else{
                        tamanho = __quali2.getNumCategorias();
                        if(tamanho != __quali2.getCategorias().length){
                            return(18);
                        }else{
                            if(tamanho==0){
                                return(12);
                            }else{
                               return(0);
                            }
                        }
                    }
                  }
                }
            }
        }
    }

    private  int VerificaParametrosEntrada(VariaveisQuantitativas __quanti1,
VariaveisQuantitativas __quanti2) {
        long tamanho;
        //String temp;
        if((__quanti1== null) || (__quanti2 == null)){
           return(9);
        }else{
```

```
          tamanho = __quanti1.getValores().length;
          if(tamanho != __quanti2.getValores().length){
             return(10);
          }else{
             if(tamanho==0){
               return(1);
             }else{
               return(0);
             }
          }
        }
    }

    private  int VerificaParametrosEntrada(VariaveisQualitativas __quali,
VariaveisQuantitativas __quanti) {
        long tamanho;
        if((__quali==null)||(__quanti==null)){
          return(9);
        }else{
            tamanho = __quanti.getValores().length;
            if(tamanho != __quali.getValores().length){
               return(10);
            }else{
               if(tamanho==0){
                 return(1);
               }else{
                 tamanho = __quali.getNumCategorias();
                 if(tamanho != (__quali.getCategorias().length)){
                     return(18);
                 }else{
                     if(tamanho==0){
                         return(12);
                     }else{
                         return(0);
                     }
                 }
               }
            }
        }
    }

    private  int ColoqueValoresNosIntervalos(String[] __valores,
ObjetoIntervalo __intervalo) {
       int j,k;
       boolean passou;
       double aux = 0;
       long tamanhoIntervalo,tamanhoValores;
       tamanhoIntervalo = __intervalo.intervalo.length;
       tamanhoValores = __valores.length;
       passou = false;
       if(tamanhoIntervalo==0){
           return(4);
       }else{
           if(tamanhoValores==0){
             return(1);
           }else{
               for(int i=0; i <= tamanhoIntervalo-1; i++){
                   __intervalo.intervalo[i].setQtdade(0);
               }
               for(int i=0; i<= tamanhoValores-1; i++){
                   if(!__valores[i].equals("")){
```

```
                    try{
                      j = 0;
                      k = 0;
                      try{
                        aux = Double.valueOf(__valores[i]).doubleValue();
//verificar
                      }catch(NumberFormatException e){/* {ocoreu um erro de
conversao de string*/ }
                      passou = true;
                      while(j < tamanhoIntervalo){
                        if(aux > __intervalo.intervalo[j].getMax()){
                           j++;
                        }else{
                           k = j;
                           j = (int)tamanhoIntervalo+1;
                        }
                      }
                      if(j==tamanhoIntervalo){
                           return(5);
                      }else{

__intervalo.intervalo[k].setQtdade(__intervalo.intervalo[k].getQtdade()+1);
                      }
                    }catch(NumberFormatException e){
                      return(3);
                    }
                  }
               }
            if(!(passou)){
               return(2);
            }else{
                return(0);
            }
           }
       }
    }

    private  int OrdeneListaTriplaPelaSegundaLista(Objeto __lista) {
        long i,j,k,ultimoret,tamanho;
        double[][] retorno;
        tamanho = __lista.___double[0].length;
        i = __lista.___double[1].length;
        j = __lista.___double[2].length;
        if((tamanho != i) || (tamanho != j)){
             return(10);
        }else{
          if(tamanho==0){
             return(7);
          }else{
            ultimoret = 0;
            retorno = new double[3][(int)tamanho];
            retorno[0][0] = __lista.___double[0][0];
            retorno[1][0] = __lista.___double[1][0];
            retorno[2][0] = __lista.___double[2][0];
            for(i = 1; i<= tamanho-1; i++){
              j = 0;
              while((retorno[1][(int)j] < __lista.___double[1][(int)i]) && (j
<= ultimoret)){
                j++;
              }
              ultimoret = ultimoret+1;
```

```
              for(k = ultimoret; k >= j+1; k--){
                 retorno[0][(int)k] = retorno[0][(int)k-1];
                 retorno[1][(int)k] = retorno[1][(int)k-1];
                 retorno[2][(int)k] = retorno[2][(int)k-1];
              }
              retorno[0][(int)j] = __lista.___double[0][(int)i];
              retorno[1][(int)j] = __lista.___double[1][(int)i];
              retorno[2][(int)j] = __lista.___double[2][(int)i];
            }
            __lista.___double = retorno;
            return(0);
          }
        }
    }

    public int OrdeneValores(Objeto __valores) {
      long n,i,j,k,ultimoret;
      double[] retorno;
      n = __valores.__double.length;
      if(n==0){
        return(8);
      }else{
        retorno = new double[(int)n];
        ultimoret = 0;
        retorno[0] = __valores.__double[0];
        for(i = 1; i<= n-1; i++){
           j = 0;
           while((retorno[(int)j] < __valores.__double[(int)i]) && (j <=
ultimoret)){
             j++;
           }
           ultimoret++;
           for(k = ultimoret; k >= j+1; k--){
              retorno[(int)k] = retorno[(int)k-1];
           }
           retorno[(int)j] = __valores.__double[(int)i];
        }
        __valores.__double = retorno;
        return(0);
      }
    }

    public int CalculeQ75(double[] __valores, Objeto __Q75) {
      long n;
      int ret;
      Objeto _valores = new Objeto();
      _valores.__double = __valores;
      n = _valores.__double.length;
      if(n==0){
        return(8);
      }else{
        ret = OrdeneValores(_valores);
        if(ret != 0){
          return(ret);
        }else{
          __Q75.setDouble(_valores.__double[Math.round(3*(n+1)/4)-1]);
          return(0);
        }
      }
    }
```

```
    public int CalculeQ25(double[] __valores, Objeto __Q25){
        long n;
        int ret;
        Objeto _valores = new Objeto();
        _valores.__double = __valores;
        n = _valores.__double.length;
        if(n==0){
           return(8);
        }else{
          ret = OrdeneValores(_valores);
          if(ret != 0){
            return(ret);
          }else{
            __Q25.setDouble(_valores.__double[Math.round((n+1)/4)-1]);
            return(0);
          }
        }
    }

    public int CalculeMediana(double[] __valores, Objeto __mediana) {
       long n;
       int ret;
       Objeto _valores = new Objeto();
       _valores.__double = __valores;
       n = _valores.__double.length;
       if(n==0){
         return(8);
       }else{
          ret = OrdeneValores(_valores);
          if(ret != 0){
            return(ret);
          }else{
            if(n%2 == 0){  // É Par
                __mediana.setDouble((_valores.__double[(int)(n/2)-
1]+__valores[(int)(n/2)])/2);
            }else{
               __mediana.setDouble(_valores.__double[(int)((n+1)/2)-1]);
            }
            return(0);
          }
       }
    }

      /**
     * Passado como referencia o valor de __valores
     */
    public int ValoresEsperados_ParaTesteNormalidadeQuiQuadradro(Intervalo[]
__intervalos, double __media, double __dp, Objeto __valores) {
       long tamanho,i,N;
       double z1,z2;
       tamanho = __intervalos.length;
       if(tamanho==0){
         return(4);
       }else{
         if((__dp==0) || (tamanho==1)){
            return(6);
         }else{
           __valores.__double = new double[(int)tamanho];
           N = 0;
           for(i =0; i<= tamanho-1; i++){
                N = N + __intervalos[(int)i].getQtdade();
```

```
            }
            z1 = (__intervalos[0].getMax() - __media)/__dp;
            __valores.__double[0] = N*DistribuicaoNormal(z1);
            for(i =1; i<= tamanho-2; i++){
               z1 = (__intervalos[(int)i].getMin() - __media)/__dp;
               z2 = (__intervalos[(int)i].getMax() - __media)/__dp;
               __valores.__double[(int)i] = N*(DistribuicaoNormal(z2)-
DistribuicaoNormal(z1));
            }
            z1 = (__intervalos[(int)tamanho-1].getMin() - __media)/__dp;
            __valores.__double[(int)tamanho-1] = N*(1-DistribuicaoNormal(z1));
            return(0);
          }
        }
    }

    public int AcheIntervalos(String[] __valoresQuanti, ObjetoIntervalo
__intervalos, Objeto __n, Objeto __minimo, Objeto __maximo, Objeto __media,
Objeto __DP) {
        long NumeroClasses,i;
        double tamanhoIntervalo;
        int ret;
        // Acha o Número de Classes pela regra de Sturges
        Objeto vMin = new Objeto(0); // será passado para referencia.
        ret = AcheMenorValor(__valoresQuanti, __n, __media, vMin);
        if(ret != 0){
           return(ret);
        }else{
          Objeto vMax = new Objeto(0);
          ret = AcheMaiorValor(__valoresQuanti,__media.doubleValue(), __n,
__DP, vMax);
          if(ret != 0){
             return(ret);
          }else{
            NumeroClasses = Math.round(1 +
3.3*(Math.log(__valoresQuanti.length) * (1/Math.log(10))));
            __intervalos.intervalo = new Intervalo[(int)NumeroClasses];
            for(int p=0; p<(int)NumeroClasses; p++ ){
              __intervalos.intervalo[p] = new Intervalo();
            }
            for(i = 0; i<= NumeroClasses-1; i++){
                __intervalos.intervalo[(int)i].setQtdade(0);
            }
            tamanhoIntervalo = (vMax.doubleValue()-
vMin.doubleValue())/NumeroClasses;
             if(tamanhoIntervalo==0){   // array constante - Minimo e Maximo
são iguais
               __intervalos.intervalo = new Intervalo[1];
               __intervalos.intervalo[0] = new Intervalo();
               __intervalos.intervalo[0].setMin(vMin.doubleValue());
               __intervalos.intervalo[0].setMax(vMin.doubleValue());
             }else{
               __intervalos.intervalo[0].setMin(vMin.doubleValue());

__intervalos.intervalo[0].setMax(vMin.doubleValue()+tamanhoIntervalo);
               for(i=1; i<=NumeroClasses-1; i++){

__intervalos.intervalo[(int)i].setMin(__intervalos.intervalo[(int)i-
1].getMax());
```

```
__intervalos.intervalo[(int)i].setMax(__intervalos.intervalo[(int)i-
1].getMax()+tamanhoIntervalo);
              }
            }
            __intervalos.intervalo[(int)NumeroClasses-
1].setMax(vMax.doubleValue());
            ret = ColoqueValoresNosIntervalos(__valoresQuanti,__intervalos);
            if(ret != 0){
              return(ret);
            }else{
              __minimo.setDouble(vMin.doubleValue());
              __maximo.setDouble(vMax.doubleValue());
              return(0);
            }
          }
        }
    }

    public int AchePostosEGuardeNaTerceiraLista(Objeto __lista, Objeto
__empates) {
      long auxTam,i,j,inicio;
      double valor,posto,emp;
      double[][] retorno;
      int ret;
      if(__lista.___double.length==0){
          return(7);
      }else{
        ret = OrdeneListaTriplaPelaSegundaLista(__lista);
        if(ret !=0){
            return(ret);
        }else{
          auxTam = __lista.___double[0].length;
          retorno = new double[3][(int)auxTam];
          for(i=0; i<= auxTam-1; i++){
             retorno[0][(int)i] =__lista.___double[0][(int)i];
             retorno[1][(int)i] =__lista.___double[1][(int)i];
          }
          valor = __lista.___double[1][0];
          inicio = 1;
          emp = 0;
          for(i=1; i<=auxTam; i++){
             if(__lista.___double[1][(int)i-1] !=valor){
                posto = (inicio+i-1)/2;
                for(j=inicio; j<=i-1; j++){
                    retorno[2][(int)j-1] = posto;
                }
                if(inicio !=i-1){
                   emp = emp+((i-inicio)*(i-inicio)*(i-inicio)) - (i-inicio);
                }
                inicio = i;
                valor = __lista.___double[1][(int)i-1];
             }
             if(i==auxTam){
                 posto = (inicio+i)/2;
                for(j=inicio; j<=i; j++){
                   retorno[2][(int)j-1] = posto;
                }
                if(inicio !=i){
                   emp = emp+((i-inicio+1)*(i-inicio+1)*(i-inicio+1)) - (i-
inicio+1);
```

```
                }
              }
            }
              __empates.setDouble(emp);
              __lista.___double = retorno;
              return(0);
          }
        }
    }

    public int AcheMaiorValor(String[] __valores, double __media, Objeto __n,
Objeto __DP, Objeto __maximo) {
        long i,tamanho;
        double aux2,soma;
        int ret;
        tamanho = __valores.length;
        Objeto aux = new Objeto(0);
        Objeto j = new Objeto(0);
        ret = PrimeiroValor(__valores,aux,j);
        if(ret !=0){
           return(ret);
        }else{
           // aux contém o primeiro (não nulo) valor do array
           __n.setLong(1);
           soma = Math.pow(aux.doubleValue() -__media,2);
           for(i=(int)j.doubleValue()+1; i<=tamanho-1;i++){
               if(!__valores[(int)i].equals("")){
                  try{
                    aux2 = Double.valueOf(__valores[(int)i]).doubleValue();
                    if(aux.doubleValue() < aux2){
                       aux.setDouble(aux2);
                    }
                    soma = soma+Math.pow(aux2 -__media,2);
                    __n.setLong((long)__n.doubleValue()+1);
                  }catch(NumberFormatException e){
                    return(3);
                  }
               }
           }
           if(__n.doubleValue()==1){
               __DP.setDouble(0);
           }else{
              __DP.setDouble(Math.sqrt(soma/(__n.doubleValue()-1)));
           }
           __maximo.setDouble(aux.doubleValue());
           return(0);
        }
    }

    public int AcheMenorValor(String[] __valores, double __media, Objeto __n,
Objeto __DP, Objeto __minimo) {
         long i,tamanho;
         double aux2,soma;
         int ret;
         tamanho = __valores.length;
         Objeto aux = new Objeto(0); //double
         Objeto j = new Objeto(0); //long
         ret = PrimeiroValor(__valores,aux,j);
         if(ret != 0){
           return(ret);
         }else{
```

```
            // aux contém o primeiro (não nulo) valor do array
            __n.setLong(1);
            soma = Math.pow(aux.doubleValue() - __media,2);
            for(i= (long)j.doubleValue()+1; i<= tamanho-1; i++){
               if(!__valores[(int)i].equals("")){
                  try{
                    aux2 = Double.valueOf(__valores[(int)i]).doubleValue();
                    if(aux.doubleValue() > aux2){
                        aux.setDouble(aux2);
                    }
                    soma = soma + Math.pow(aux2 -__media,2);
                    __n.setLong(__n.longValue() + 1);
                  }catch(NumberFormatException e){
                     return(3);
                  }
               }
            }
            if(__n.intValue()==1){
               __DP.setDouble(0);
            }else{
               __DP.setDouble(Math.sqrt(soma/(__n.intValue()-1)));
            }
            __minimo.setDouble(aux.doubleValue());
            return(0);
          }
      }

      public int AcheMaiorValor(String[] __valores, Objeto __n, Objeto __media,
Objeto __maximo) {
            long i,tamanho;
            double aux2,soma;
            int ret;
            tamanho = __valores.length;
            Objeto aux = new Objeto(0);
            Objeto j = new Objeto(0);
            ret = PrimeiroValor(__valores,aux,j);
            if(ret !=0){
               return(ret);
            }else{
               // aux contém o primeiro (não nulo) valor do array
               __n.setLong(1);
               soma = aux.doubleValue();
               for(i=j.intValue()+1; i<=tamanho-1;i++){
                   if(!__valores[(int)i].equals("")){
                      try{
                        aux2 = Double.valueOf(__valores[(int)i]).doubleValue();
                        if(aux.doubleValue() < aux2){
                           aux.setDouble(aux2);
                        }
                        soma = soma + aux2;
                        __n.setLong(__n.longValue()+1);
                      }catch(NumberFormatException e){
                        return(3);
                      }
                   }
               }
               __media.setDouble(soma/__n.doubleValue());
               __maximo.setDouble(aux.doubleValue());
               return(0);
            }
      }
```

```
    public int AcheMenorValor(String[] __valores, Objeto __n, Objeto __media,
Objeto __minimo) {
        long i,tamanho;
        double aux2,soma;
        int ret;
        tamanho = __valores.length;
        Objeto aux = new Objeto(0);
        Objeto j = new Objeto(0);
        ret = PrimeiroValor(__valores,aux,j);
        if(ret != 0){
          return(ret);
        }else{
          // aux contém o primeiro (não nulo) valor do array
          __n.setLong(1);
          soma = aux.doubleValue();
          for(i=j.intValue()+1; i<=tamanho-1;i++){
             if(!__valores[(int)i].equals("")){
                try{
                  aux2 = Double.valueOf(__valores[(int)i]).doubleValue();
                  if(aux.doubleValue() > aux2){
                     aux.setDouble(aux2);
                  }
                  soma = soma + aux2;
                  __n.setLong(__n.longValue() + 1);
                }catch(NumberFormatException e){
                  return(3);
                }
             }
          }
          __media.setDouble(soma/__n.doubleValue());
          __minimo.setDouble(aux.doubleValue());
          return(0);
        }
    }

    public  int PrimeiroValor(String[] __valores, Objeto __primeiro, Objeto
__posicao) {
        long tamanho,i;
        tamanho = __valores.length;
        i = 0;
        while( i<tamanho){
          if(__valores[(int)i].equals("")){
             i=i+1;
          }else{
             __posicao.setLong(i);
             i = tamanho+1;
          }
        }
        if(i==tamanho){   // array tem Tamanho zero ou só tem valores nulos
(='')
           if(tamanho==0){
              return(1);
           }else{
              return(2);
           }
        }
        try{

__primeiro.setDouble(Double.valueOf(__valores[(int)__posicao.longValue()]).do
ubleValue());
```

```
        }catch(NumberFormatException e){
          return(3);
        }
        return(0);
    }

    public double DistribuicaoF(double __valor, int __gl1, int __gl2) {
       /* Utiliza a funcao FDistriIntegral(valor,gl1,gl2) do SDl do
delphi!!!!*/
       double v=0.5;
       double dv=0.5;
       double f=0;
       while(dv>1e-10){
           f=1/v-1;
           dv=dv/2;
           if(FishF(f,__gl1,__gl2)>__valor){
               v=v-dv;
           }else{ v=v+dv;
           }
       }
       return f;
    }


    public double DistribuicaoQuiQuadrado(double __valor, int __gl) {
       /* Utiliza a funcao Chi2DistriIntegral(valor,gl); do SDl do Delphi!!!
*/
       double v=0.5;
       double dv=0.5;
       double x=0;
       while(dv>1e-10){
           x=1/v-1;
           dv=dv/2;
           if(ChiSq(x,__gl)>__valor){
               v=v-dv;
           }else{
               v=v+dv;
           }
       }
       return x;
    }

    public double DistribuicaoTStudent(double __valor, int __gl) {
        /* Utiliza a funcao tDistriIntegral(valor,gl); do SDL do delphi!!! */
         double v=0.5;
         double dv=0.5;
         double t=0;
         while(dv>1e-6){
             t=1/v-1;
             dv=dv/2;
             if(StudenT(t,__gl)>__valor){
                 v=v-dv;
             }else{
                 v=v+dv;
             }
         }
         return t;
    }


    public  double DistribuicaoNormal(double __valor) {
```

```
        /*Utiliza a funcao nDistriIntegral(valor); do SDl do delphi!!!*/
         double v = 0.5;
         double dv=0.5;
         double z=0;
         while(dv>1e-6){
             z=1/v-1;
             dv=dv/2;
             if(Normal(z)>__valor){
                 v=v-dv;
             }else{
                 v=v+dv;
             }
         }
         return z;
    }

    public int CalculeFrequenciasObservadasEEsperadas(VariaveisQualitativas
__quali1, VariaveisQualitativas __quali2, Objeto __FObservadas, Objeto
__FEsperadas, Objeto __TabelaQuiquadrado) {
        int ret;
        long tamanho,i,j,k;
        double soma,soma2,soma3;
        ret = VerificaParametrosEntrada(__quali1,__quali2);
        if(ret != 0){
          return(ret);
        }else{
          tamanho = __quali1.getValores().length;
          __FObservadas.___double = new
double[(int)__quali1.getNumCategorias()+1][(int)__quali2.getNumCategorias()+1
];
          __FEsperadas.___double = new
double[(int)__quali1.getNumCategorias()+1][(int)__quali2.getNumCategorias()+1
];
          __TabelaQuiquadrado.___double = new
double[(int)__quali1.getNumCategorias()+1][(int)__quali2.getNumCategorias()+1
];
          for(i=0; i<= __quali1.getNumCategorias()-1; i++){
            soma = 0;
            for(j=0; j<= __quali2.getNumCategorias()-1; j++){
                __FObservadas.___double[(int)i][(int)j] = 0;
               for(k=0; k<= tamanho-1; k++){

if((__quali1.getValores()[(int)k].equals(__quali1.getCategorias()[(int)i].get
Valor())) &&

(__quali2.getValores()[(int)k].equals(__quali2.getCategorias()[(int)j].getVal
or()))){
                          __FObservadas.___double[(int)i][(int)j] =
__FObservadas.___double[(int)i][(int)j]+1;
                        soma = soma+1;
                   }
               }
            }
            __FObservadas.___double[(int)i][(int)__quali2.getNumCategorias()]
= soma;
          }
          for(i=0; i<= __quali2.getNumCategorias(); i++){
             soma = 0;
             for(j=0; j<= __quali1.getNumCategorias()-1; j++){
                soma = soma + __FObservadas.___double[(int)j][(int)i];
             }
```

```
__FObservadas.___double[(int)__quali1.getNumCategorias()][(int)i] = soma;
          }

if(__FObservadas.___double[(int)__quali1.getNumCategorias()][(int)__quali2.ge
tNumCategorias()]==0){
             return(13);
          }else{
            double temp =0;
            for(i=0; i <= (int)__quali1.getNumCategorias()-1; i++){
               soma2 = 0;
               soma3 = 0;
               for(j=0; j<= (int) __quali2.getNumCategorias()-1; j++){
                   __FEsperadas.___double[(int)i][(int)j] =
(__FObservadas.___double[(int)i][(int)__quali2.getNumCategorias()]*

__FObservadas.___double[(int)__quali1.getNumCategorias()][(int)j])/

__FObservadas.___double[(int)__quali1.getNumCategorias()][(int)__quali2.getNu
mCategorias()];
                   temp = __FObservadas.___double[(int)i][(int)j] -
__FEsperadas.___double[(int)i][(int)j];
                   __TabelaQuiquadrado.___double[(int)i][(int)j] =
(temp*temp)/__FEsperadas.___double[(int)i][(int)j];
                   soma3 = soma3 +
__TabelaQuiquadrado.___double[(int)i][(int)j];
                   soma2 = soma2 + __FEsperadas.___double[(int)i][(int)j];
               }

__FEsperadas.___double[(int)i][(int)__quali2.getNumCategorias()] = soma2;

__TabelaQuiquadrado.___double[(int)i][(int)__quali2.getNumCategorias()] =
soma3;
            }
            for(i=0; i<=(int)__quali2.getNumCategorias(); i++){
                soma2 = 0;
                soma3 = 0;
                for(j=0; j<=(int) __quali1.getNumCategorias()-1;j++){
                    soma2 = soma2 + __FEsperadas.___double[(int)j][(int)i];
                    soma3 = soma3 +
__TabelaQuiquadrado.___double[(int)j][(int)i];
                }

__FEsperadas.___double[(int)__quali1.getNumCategorias()][(int)i] = soma2;

__TabelaQuiquadrado.___double[(int)__quali1.getNumCategorias()][(int)i] =
soma3;
            }
            return(0);
          }
        }
    }

    public int TesteKolmogorovSmirnov(Intervalo[] __intervalos, double[]
__esperados, Objeto __valorTeste) {
        double SEsp, SFreq, maior;
        long i,N,ultimo;
        double[] KS;
        ultimo = __intervalos.length;
        if(ultimo==0){
           return(4);
```

```
        }else{
          i = __esperados.length;
          if(ultimo !=i){
            return(10);
          }else{
            if(ultimo < 2){
               return(16);
            }else{
              SEsp = 0;
              SFreq = 0;
              N = 0;
              ultimo = ultimo-1;
              for(i=0; i<=(int)ultimo; i++){
                  N = N + __intervalos[(int)i].getQtdade();
              }
              KS = new double[(int)ultimo+1];
              for(i=0; i<=(int) ultimo; i++){
                 SEsp = SEsp + __intervalos[(int)i].getQtdade();
                 SFreq = SFreq + __esperados[(int)i];
                 KS[(int)i] = Math.abs(SEsp/N-SFreq/N);
              }
              maior = KS[0];
              for(i=1; i<=(int)ultimo;i++){
                if(KS[(int)i] > maior){
                   maior = KS[(int)i];
                }
              }
              __valorTeste.setDouble(maior);
              return(0);
            }
          }
        }
    }

    public int TesteQuiQuadradoNormal(Intervalo[] __intervalos, double[]
__esperados, Objeto __df, Objeto __ValorTeste, Objeto __p) {
        double soma, somaEsperado, somaObservado, somaEsperadoAnt,
somaObservadoAnt;
        long i,n,ultimo;
        boolean fim;
        ultimo = __intervalos.length;
        if(ultimo==0){
           return(4);
        }else{
          i = __esperados.length;
          if(ultimo != i){
            return(10);
          }else{
            if((int)ultimo < 2){
              return(16);
            }else{
              soma = 0;
              fim = false;
              i = 0;
              n = 0;
              somaEsperado = 0;
              somaObservado = 0;
              somaEsperadoAnt = 0;
              somaObservadoAnt = 0;
              ultimo = ultimo-1;
              while(!(fim)){
```

```
                somaEsperado = __esperados[(int)i];
                somaObservado = __intervalos[(int)i].getQtdade();
                while(((int)somaEsperado < 5) && (!fim)){
                    i++;
                    if(!(i > ultimo)){
                       somaEsperado = somaEsperado + __esperados[(int)i];
                       somaObservado = somaObservado +
__intervalos[(int)i].getQtdade();
                    }
                    if(i >= ultimo){
                       fim = true;
                    }
                }
                if(!fim){
                  somaEsperadoAnt = somaEsperado;
                  somaObservadoAnt = somaObservado;
                  soma = soma+Math.pow(somaObservado-
somaEsperado,2)/somaEsperado;
                   n = n+1;
                }
                i = i+1;
                if(i>ultimo){
                   fim = true;
                }
              }
              if(somaEsperado < 5){
                 soma = soma - Math.pow(somaObservadoAnt-
somaEsperadoAnt,2)/somaEsperadoAnt;
                 somaEsperado = somaEsperado+somaEsperadoAnt;
                 somaObservado = somaObservado + somaObservadoAnt;
                 soma = soma + Math.pow(somaObservado-
somaEsperado,2)/somaEsperado;
              }else{
                 soma = soma+Math.pow(somaObservado-
somaEsperado,2)/somaEsperado;
                 n = n+1;
              }
              __ValorTeste.setDouble(soma);
              __df.setLong(n-3);
              __p.setDouble(1-
DistribuicaoQuiQuadrado(__ValorTeste.doubleValue(),__df.intValue()));
              return(0);
            }
          }
        }
    }

    public int TesteQuiQuadrado(VariaveisQualitativas __quali1,
VariaveisQualitativas __quali2, Objeto __gl, Objeto __valorTeste, Objeto __p)
{
        int ret;
        Objeto FObservadas = new Objeto();
        Objeto FEsperadas = new Objeto();
        long i,j;
        double soma;
        ret = VerificaParametrosEntrada(__quali1,__quali2);
          if(ret != 0){
            return(ret);
          }else{
            if((__quali1.getNumCategorias()<=1) ||
(__quali2.getNumCategorias() <=1)){
```

```
                return(17);
              }else{
                Objeto TabelaQuiquadrado = new Objeto();
                ret =
CalculeFrequenciasObservadasEEsperadas(__quali1,__quali2,FObservadas,FEsperad
as,TabelaQuiquadrado);
                if(ret !=0){
                  return(ret);
                }else{
                  soma = 0;
                  for(i=0; i<=(int)__quali1.getNumCategorias()-1; i++){
                    for(j=0; j<= (int)__quali2.getNumCategorias()-1; j++){
                       if(FEsperadas.___double[(int)i][(int)j] < 5){
                          return(20);
                       }
                       soma =
soma+Math.pow(FEsperadas.___double[(int)i][(int)j] -
FObservadas.___double[(int)i][(int)j],2)/FEsperadas.___double[(int)i][(int)j]
;
                    }
                  }
                  __valorTeste.setDouble(soma);
                  __gl.setLong((__quali1.getNumCategorias()-
1)*(__quali2.getNumCategorias()-1));
                  __p.setDouble(1-
DistribuicaoQuiQuadrado(__valorTeste.doubleValue(),__gl.intValue()));
                  return(0);
                }
              }
            }
        }

    public int CorrelacaoPearson(VariaveisQuantitativas __quanti1,
VariaveisQuantitativas __quanti2, Objeto __valorTeste) {
        long tamanho,ultimo,i;
        double R,SX,SY,SX2,SY2,SXY;
        double[][] Lista;
        int ret;
        double aux1,aux2;
        ret = VerificaParametrosEntrada(__quanti1,__quanti2);
        if(ret != 0){
           return(ret);
        }else{
          tamanho = __quanti1.getValores().length;
          Lista = new double[3][(int)tamanho];
          ultimo = 0;
          for(i=0; i<= tamanho-1; i++){
              if( (!"".equals(__quanti1.getValores()[(int)i])) &&
(!"".equals(__quanti2.getValores()[(int)i]))){
                 try{
                   aux1 =
Double.valueOf(__quanti1.getValores()[(int)i]).doubleValue();
                   aux2 =
Double.valueOf(__quanti2.getValores()[(int)i]).doubleValue();
                 }catch(NumberFormatException e){
                   return(3);
                 }
                 ultimo = ultimo+1;
                 Lista[0][(int)ultimo-1] = ultimo-1;
                 Lista[1][(int)ultimo-1] = aux1;
                 Lista[2][(int)ultimo-1] = aux2;
```

```
            }
          }
          //SetLength(Lista[1],ultimo);
          //SetLength(Lista[2],ultimo);
          //SetLength(Lista[3],ultimo);
          SX = 0;
          SY = 0;
          SX2 = 0;
          SY2 = 0;
          SXY = 0;
          for(i=0; i<=ultimo-1; i++){
             SX = SX+Lista[1][(int)i];
             SY = SY+Lista[2][(int)i];
             SX2 = SX2+Math.pow(Lista[1][(int)i],2);
             SY2 = SY2+Math.pow(Lista[2][(int)i],2);
             SXY = SXY+(Lista[1][(int)i]*Lista[2][(int)i]);
          }
          R = ((ultimo*SXY)-(SX*SY)) / (Math.sqrt((ultimo*SX2)-
Math.pow(SX,2))*Math.sqrt((ultimo*SY2)-Math.pow(SY,2)));
          __valorTeste.setDouble(R);
          return(0);
        }
    }

    public int CorrelacaoSpearman(VariaveisQuantitativas __quanti1,
VariaveisQuantitativas __quanti2, Objeto __dfT, Objeto __T, Objeto
__ValorTeste, Objeto __p) {
        long tamanho,ultimo,i,j,k;
        double R,SX,SY,SX2,SY2,SXY;
        double[][] Lista1_temp, Lista2_temp;
        Objeto Lista1,Lista2;
        int ret;
        double aux1,aux2;
        Lista1 = new Objeto();
        Lista2 = new Objeto();
        Objeto lixo = new Objeto();
        ret = VerificaParametrosEntrada(__quanti1,__quanti2);
        if(ret!=0){
          return(ret);
        }else{
          tamanho = __quanti1.getValores().length;
          Lista1_temp = new double[3][(int)tamanho];
          Lista2_temp = new double[3][(int)tamanho];
          //SetLength(Lista1_temp[1],Tamanho);
          //SetLength(Lista1_temp[2],Tamanho);
          //SetLength(Lista1_temp[3],Tamanho);
          //SetLength(Lista2_temp[1],Tamanho);
          //SetLength(Lista2_temp[2],Tamanho);
          //SetLength(Lista2_temp[3],Tamanho);
          ultimo = 0;
          for(i=0; i<=tamanho-1; i++){
              if((!"".equals(__quanti1.getValores()[(int)i])) &&
(!"".equals(__quanti2.getValores()[(int)i]))){
                  try{
                      aux1 =
Double.valueOf(__quanti1.getValores()[(int)i]).doubleValue();
                      aux2 =
Double.valueOf(__quanti2.getValores()[(int)i]).doubleValue();
                  }catch(NumberFormatException e){
                      return(3);
                  }
```

```
                    ultimo = ultimo+1;
                    Lista1_temp[0][(int)ultimo-1] = ultimo-1;
                    Lista1_temp[1][(int)ultimo-1] = aux1;
                    Lista1_temp[2][(int)ultimo-1] = 0;
                    Lista2_temp[0][(int)ultimo-1] = ultimo-1;
                    Lista2_temp[1][(int)ultimo-1] = aux2;
                    Lista2_temp[2][(int)ultimo-1] = 0;
                  };
              };
            //SetLength(Lista1_temp[1],ultimo);
            //SetLength(Lista1_temp[2],ultimo);
            //SetLength(Lista1_temp[3],ultimo);
            //SetLength(Lista2_temp[1],ultimo);
            //SetLength(Lista2_temp[2],ultimo);
            //SetLength(Lista2_temp[3],ultimo);
            Lista1.___double = new double[3][(int)ultimo];
            Lista2.___double = new double[3][(int)ultimo];
            for(i=0; i<= ultimo-1; i++){
               Lista1.___double[0][(int)i] = Lista1_temp[0][(int)i];
               Lista1.___double[1][(int)i] = Lista1_temp[1][(int)i];
               Lista1.___double[2][(int)i] = Lista1_temp[2][(int)i];
               Lista2.___double[0][(int)i] = Lista2_temp[0][(int)i];
               Lista2.___double[1][(int)i] = Lista2_temp[1][(int)i];
               Lista2.___double[2][(int)i] = Lista2_temp[2][(int)i];
            }
            ret = OrdeneListaTriplaPelaSegundaLista(Lista1);
            if(ret!=0){
                 return(ret);
            }
            ret = OrdeneListaTriplaPelaSegundaLista(Lista2);
            if(ret!=0){
                 return(ret);
            }
            ret = AchePostosEGuardeNaTerceiraLista(Lista1,lixo);
            if(ret!=0){
                 return(ret);
            }
            ret = AchePostosEGuardeNaTerceiraLista(Lista2,lixo);
            if(ret!=0){
                 return(ret);
            }
            SX = 0;
            SY = 0;
            SX2 = 0;
            SY2 = 0;
            SXY = 0;
            for(i=0; i<= ultimo-1;i++){
                  j=0;
                  while(Lista1.___double[0][(int)j]!=i){
                     j = j+1;
                  }
                  k = 0;
                  while(Lista2.___double[0][(int)k]!=i){
                     k=k+1;
                  }
                  SX = SX+Lista1.___double[2][(int)j];
                  SY = SY+Lista2.___double[2][(int)k];
                  SX2 =SX2+Math.pow(Lista1.___double[2][(int)j],2);
                  SY2 =SY2+Math.pow(Lista2.___double[2][(int)k],2);
                  SXY
=SXY+(Lista1.___double[2][(int)j]*Lista2.___double[2][(int)k]);
```

```
            };
          R = ((ultimo*SXY)-(SX*SY)) / (Math.sqrt((ultimo*SX2)-
Math.pow(SX,2))*Math.sqrt((ultimo*SY2)-Math.pow(SY,2)));
          __T.setDouble((R/Math.sqrt(1-Math.pow(R,2)))*Math.sqrt(ultimo-2));
          Long temp2 = new Long(ultimo-2);
          __dfT.setLong(ultimo-2);
          __ValorTeste.setDouble(R);
          __p.setDouble((1-
DistribuicaoTStudent(__T.doubleValue(),__dfT.intValue()))*2);
          return(0);
        }
    }

    public int TesteBrownForsythe(VariaveisQualitativas __quali,
VariaveisQuantitativas __quanti, Objeto __GL1, Objeto __GL2, Objeto
__valorTeste, Objeto __p) {
        long tamanho,i,j,auxN;
        Objeto mediana;
        long[] NI;
        double[] sumI;
        double[] sumSqrI;
        double[] auxMediana;
        double[] auxMediana_2;
        double sumSqr,SST,SSTRT,SSE,somaN,F;
        int ret;
        double aux;
        ret = VerificaParametrosEntrada(__quali,__quanti);
        if(ret != 0){
               return(ret);
        }else{
          if(__quali.getNumCategorias() <= 1){
               return(17);
          }else{
            tamanho = __quali.getValores().length;
            sumSqr = 0;
            SST = 0;
            SSTRT = 0;
            somaN = 0;
            NI = new long[(int)__quali.getNumCategorias()];
            sumI = new double[(int)__quali.getNumCategorias()];
            sumSqrI = new double[(int)__quali.getNumCategorias()];
            auxMediana = new double[(int)tamanho];
            for(i = 0; i<= __quali.getNumCategorias()-1; i++){
                //SetLength(auxMediana,Tamanho);
                auxN = 0;
                for(j = 0; j<= tamanho-1; j++){
                    if((!"".equals(__quanti.getValores()[(int)j])) &&
(__quali.getValores()[(int)j].equals(__quali.getCategorias()[(int)i].getValor
()))){
                       try{
                          aux =
Double.valueOf(__quanti.getValores()[(int)j]).doubleValue();
                       }catch(NumberFormatException e){
                          return(3);
                       }
                       auxMediana[(int)auxN] = aux;
                       auxN = auxN+1;
                    }
                }
                if(auxN<=1){
                   return(16);
```

```
                }
                auxMediana_2 = new double[(int)auxN];
                for(j=0; j<= auxN-1; j++){
                   auxMediana_2[(int)j] = auxMediana[(int)j];
                }
                //SetLength(auxMediana,auxN);
                mediana = new Objeto(0);
                ret = CalculeMediana(auxMediana_2,mediana);
                if(ret != 0){
                     return(ret);
                }
                NI[(int)i] = auxN;
                sumI[(int)i] = 0;
                sumSqrI[(int)i] = 0;
                for(j = 0; j<= tamanho-1; j++){
                    if((!"".equals(__quanti.getValores()[(int)j])) &&
(__quali.getValores()[(int)j].equals(__quali.getCategorias()[(int)i].getValor
())))){
                        try{
                          aux =
Double.valueOf(__quanti.getValores()[(int)j]).doubleValue();
                        }catch(NumberFormatException e){
                          return(3);
                        }
                        sumI[(int)i] = sumI[(int)i]+Math.abs(aux-
mediana.doubleValue());
                        sumSqrI[(int)i] = sumSqrI[(int)i]+Math.pow(aux-
mediana.doubleValue(),2);
                      }
                }
                sumSqr = sumSqr+sumI[(int)i];
                SST = SST+sumSqrI[(int)i];
                SSTRT = SSTRT+Math.pow(sumI[(int)i],2)/NI[(int)i];
                somaN = somaN+NI[(int)i];
              }
            sumSqr = Math.pow(sumSqr,2);
            SST = SST-sumSqr/somaN;
            SSTRT = SSTRT-sumSqr/somaN;
            SSE = SST-SSTRT;
            if(SSE==0){
             // 'Os valores dentro de cada categoria são todos coincidentes
entre si. Provavelmente a variável __qualitativa e a __quantitativa são as
mesmas.'
               return(19);
            }
            F = ((somaN-
__quali.getNumCategorias())*SSTRT)/(SSE*(__quali.getNumCategorias()-1));
            __valorTeste.setDouble(F);
            __GL1.setLong(__quali.getNumCategorias()-1);
            __GL2.setLong(__quanti.getN() - __quali.getNumCategorias());
            __p.setDouble(1-
DistribuicaoF(__valorTeste.doubleValue(),__GL1.intValue(),__GL2.intValue()));
            return(0);
          }
        }
    }

    public int TesteLevene(VariaveisQualitativas __quali,
VariaveisQuantitativas __quanti, Objeto __GL1, Objeto __GL2, Objeto
__valorTeste, Objeto __p) {
        long tamanho,i,j,auxN;
```

```
        long[] NI;
        double[] sumI;
        double[] sumSqrI;
        double sumSqr,SST,SSTRT,SSE,somaN,F,auxSoma,auxMedia;
        int ret;
        double aux;
        ret = VerificaParametrosEntrada(__quali,__quanti);
        if(ret != 0){
          return(ret);
        }else{
          tamanho = __quali.getValores().length;
          if(__quali.getNumCategorias()<=1){
             return(17);
          }else{
            sumSqr = 0;
            SST = 0;
            SSTRT = 0;
            somaN = 0;
            NI = new long[(int)__quali.getNumCategorias()];
            sumI = new double[(int)__quali.getNumCategorias()];
            sumSqrI = new double[(int)__quali.getNumCategorias()];
            for(i = 0; i<= __quali.getNumCategorias()-1; i++){
                auxN = 0;
                auxSoma = 0;
                for(j = 0; j<= tamanho-1; j++){
                  if((!"".equals(__quanti.getValores()[(int)j])) &&
(__quali.getValores()[(int)j].equals(__quali.getCategorias()[(int)i].getValor
())){
                      try{
                        aux =
Double.valueOf(__quanti.getValores()[(int)j]).doubleValue();
                      }catch(NumberFormatException e){
                        return(3);
                      }
                      auxN = auxN+1;
                      auxSoma = auxSoma + aux;
                  }
                }
                if(auxN<=1){
                  return(16);
                }
                auxMedia = auxSoma/auxN;
                NI[(int)i] = auxN;
                sumI[(int)i] = 0;
                sumSqrI[(int)i] = 0;
                for(j = 0; j<=tamanho-1; j++){
                  if((!"".equals(__quanti.getValores()[(int)j])) &&
(__quali.getValores()[(int)j].equals(__quali.getCategorias()[(int)i].getValor
())){
                       try{
                         aux =
Double.valueOf(__quanti.getValores()[(int)j]).doubleValue();
                       }catch(NumberFormatException e){
                         return(3);
                       }
                       sumI[(int)i] = sumI[(int)i]+Math.abs(aux-auxMedia);
                       sumSqrI[(int)i] = sumSqrI[(int)i]+Math.pow(aux-
auxMedia,2);
                  }
                }
                sumSqr = sumSqr+sumI[(int)i];
```

```
                SST = SST+sumSqrI[(int)i];
                SSTRT = SSTRT+Math.pow(sumI[(int)i],2)/NI[(int)i];
                somaN = somaN+NI[(int)i];
            }
            sumSqr = Math.pow(sumSqr,2);
            SST = SST-sumSqr/somaN;
            SSTRT = SSTRT-sumSqr/somaN;
            SSE = SST-SSTRT;
            if(SSE==0){
             // 'Os valores dentro de cada categoria são todos coincidentes
entre si. Provavelmente a variável __qualitativa e a __quantitativa são as
mesmas.'
               return(19);
            }
            F = ((somaN-
__quali.getNumCategorias())*SSTRT)/(SSE*(__quali.getNumCategorias()-1));
            __valorTeste.setDouble(F);
            __GL1.setLong(__quali.getNumCategorias()-1);
            __GL2.setLong(__quanti.getN() - __quali.getNumCategorias());
            __p.setDouble(1-
DistribuicaoF(__valorTeste.doubleValue(),__GL1.intValue(),__GL2.intValue()));
            return(0);
          }
        }
    }

    public int CalculeDPsPorCategorias(VariaveisQualitativas __quali,
VariaveisQuantitativas __quanti, CategoriaPorIntervalo[]
__categoriasPorIntervalos, double[] __medias, double __mediaTotal, Objeto
__DPs, Objeto __DPTotal) {
        double[] retDPs;
        long[] auxNs;
        long auxTam,tamanho,i,j,k,total;
        int ret;
        double aux;
        ret = VerificaParametrosEntrada(__quali,__quanti);
        if(ret != 0){
          return(ret);
        }else{
          tamanho = __quali.getValores().length;
          retDPs = new double[(int)__quali.getNumCategorias()];
          auxNs = new long[(int)__quali.getNumCategorias()];
          for(i = 0; i<= __quali.getNumCategorias()-1; i++){
              retDPs[(int)i] = 0;
              auxNs[(int)i] = 0;
          }
          total = 0;
          k = 0;
          __DPTotal.setDouble(0);
          auxTam = __categoriasPorIntervalos.length;
          for(i = 0;i<= tamanho-1; i++){
             if((!"".equals(__quali.getValores()[(int)i])) &&
(!"".equals(__quanti.getValores()[(int)i]))){
                 j = 0;
                 while(j<auxTam){

if(!__categoriasPorIntervalos[(int)j].getValor().equals(__quali.getValores()[
(int)i])){
                     j = j+1;
                   }else{
                     k = j;
```

```
                    j = auxTam+1;
                  }
                }
                if(j==auxTam){
                    return(11);
                }
                try{
                   aux =
Double.valueOf(__quanti.getValores()[(int)i]).doubleValue();
                }catch(NumberFormatException e){
                   return(3);
                }
                retDPs[(int)k] = retDPs[(int)k]+Math.pow(aux-
__medias[(int)k],2);
                auxNs[(int)k] = auxNs[(int)k]+1;
                total = total+1;
                __DPTotal.setDouble(__DPTotal.doubleValue() + Math.pow(aux-
__mediaTotal,2));
              }
          }
          for(i = 0; i<= __quali.getNumCategorias()-1; i++){
              if(auxNs[(int)i]<=1){
                 retDPs[(int)i] = 0;
              }else
                 retDPs[(int)i] = Math.sqrt(retDPs[(int)i]/(auxNs[(int)i]-
1));
          }
          __DPs.__double = retDPs;
          //retDps = nil;
          //auxNs = nil;
          if(total<=1){
            __DPTotal.setDouble(0);
          }else{
            __DPTotal.setDouble(Math.sqrt(__DPTotal.doubleValue()/(total-
1)));
          }
          return(0);
       }
    }

    public int CalculeMediasPorCategorias(VariaveisQualitativas __quali,
VariaveisQuantitativas __quanti, CategoriaPorIntervalo[]
__categoriasPorIntervalos, Objeto __medias, Objeto __Ns, Objeto __mediaTotal)
{
        double[] retMedias;
        long[] retNs;
        long auxTam,tamanho,i,j,k,total;
        int ret;
        double aux;
        ret = VerificaParametrosEntrada(__quali,__quanti);
        if(ret != 0){
            return(ret);
        }else{
          tamanho = __quali.getValores().length;
          retMedias = new double[(int)__quali.getNumCategorias()];
          retNs = new long[(int)__quali.getNumCategorias()];
          for(i = 0; i<= __quali.getNumCategorias()-1;i++){
             retMedias[(int)i] = 0;
             retNs[(int)i] = 0;
          }
          total = 0;
```

```
          __mediaTotal.setDouble(0);
          auxTam = __categoriasPorIntervalos.length;
          aux =0;
          k = 0;
          for(i = 0; i<= tamanho-1; i++){
            if((!"".equals(__quali.getValores()[(int)i])) &&
(!"".equals(__quanti.getValores()[(int)i] ))){
              j = 0;
              while(j<auxTam){

if(!__categoriasPorIntervalos[(int)j].getValor().equals(__quali.getValores()[
(int)i])){
                  j = j+1;
                }else{
                  k = j;
                  j = auxTam+1;
                }
              }
              if(j==auxTam){
                 return(11);
              }
              try{
                 aux =
Double.valueOf(__quanti.getValores()[(int)i]).doubleValue();
              }catch(NumberFormatException e){
                 return(3);
              }
              retMedias[(int)k] = retMedias[(int)k] + aux;
              retNs[(int)k] = retNs[(int)k]+1;
              total = total+1;
              __mediaTotal.setDouble(__mediaTotal.doubleValue() + aux);
            }
          }
          for(i = 0; i<= __quali.getNumCategorias()-1; i++){
              if(retNs[(int)i]==0){
                 retMedias[(int)i] = 0;
              }else{
                 retMedias[(int)i] = retMedias[(int)i]/retNs[(int)i];
              }
          }
          if(total==0){
             __mediaTotal.setDouble(0);
          }else{
             __mediaTotal.setDouble(__mediaTotal.doubleValue()/total);
          }
          __medias.__double = retMedias;
          __Ns.__long = retNs;
          //retMedias = nil;
          //retNs = nil;
          return(0);
        }
    }

    public int AcheCategoriasPorIntervalos(VariaveisQualitativas __quali,
VariaveisQuantitativas __quanti, ObjetoRefCategoriaPorIntervalo
__categoriaPorIntervalo) {
        CategoriaPorIntervalo[] retorno;
        long auxTam,tamanho,i,j,k,l,m;
        Objeto lixo2;
        ObjetoIntervalo Intervalos;
        double aux;
```

```
        Objeto lixo;
        int ret;
        ret=VerificaParametrosEntrada(__quali,__quanti);
        if(ret != 0){
           return(ret);
        }else{
          Intervalos = new ObjetoIntervalo();
          lixo = new Objeto(0);
          lixo2 = new Objeto(0);

ret=AcheIntervalos(__quanti.getValores(),Intervalos,lixo2,lixo,lixo,lixo,lixo
);
          if(ret != 0){
             return(ret);
          }
          retorno = new
CategoriaPorIntervalo[(int)__quali.getNumCategorias()];
          for(int p =0; p<(int)__quali.getNumCategorias(); p++ ){
            retorno[p] = new CategoriaPorIntervalo();
          }
          tamanho = __quali.getValores().length;
          for(i=0; i<= __quali.getNumCategorias()-1; i++){

retorno[(int)i].setValor(__quali.getCategorias()[(int)i].getValor());
             retorno[(int)i].setQtdade(0);
             retorno[(int)i].Intervalos = new
Intervalo[Intervalos.intervalo.length];
             for(int p=0; p< Intervalos.intervalo.length; p++){
                retorno[(int)i].Intervalos[p] = new Intervalo();
             }
             for(j=0; j<=Intervalos.intervalo.length-1; j++){

retorno[(int)i].Intervalos[(int)j].setMin(Intervalos.intervalo[(int)j].getMin
());

retorno[(int)i].Intervalos[(int)j].setMax(Intervalos.intervalo[(int)j].getMax
());
                retorno[(int)i].Intervalos[(int)j].setQtdade(0);
             }
          }
          Intervalos=null;
          k = 0;
          m = 0;
          for(i=0; i<= tamanho-1; i++){
              if((!"".equals(__quali.getValores()[(int)i])) &&
(!"".equals(__quanti.getValores()[(int)i] ))){
                   j=0;
                   auxTam= retorno.length;
                   while(j<auxTam){

if(!retorno[(int)j].getValor().equals(__quali.getValores()[(int)i])){
                        j=j+1;
                     }else{
                        k=j;
                        j=auxTam+1;
                     }
                   }
                   if(j==auxTam){
                      return(11);
                   }
                   retorno[(int)k].setQtdade(retorno[(int)k].getQtdade()+1);
```

```
                    try{
                      aux=
Double.valueOf(__quanti.getValores()[(int)i]).doubleValue();
                    }catch(NumberFormatException e){
                      return(3);
                    }
                    l=0;
                    auxTam= retorno[(int)k].Intervalos.length;
                    while(l<auxTam){
                      if(aux>retorno[(int)k].Intervalos[(int)l].getMax()){
                         l=l+1;
                      }else{
                         m=l;
                         l=auxTam+1;
                      }
                    }
                    if(l==auxTam){
                         return(5);
                    }

retorno[(int)k].Intervalos[(int)m].setQtdade(retorno[(int)k].Intervalos[(int)
m].getQtdade()+1);
                }
          }
          __categoriaPorIntervalo.categoriaPorIntervalo = retorno;
          //retorno=nil;
          return(0);
       }
    }

    public int AcheCategoriasPorCategorias(VariaveisQualitativas __quali1,
VariaveisQualitativas __quali2, ObjetoRefCategoriaPorCategoria
__categoriaPorCategorias) {
        long auxTam,tamanho,i,j,k,l,m;
        int ret;
        ret = VerificaParametrosEntrada(__quali1,__quali2);
        k = 0;
        m = 0;
        if(ret != 0){
           return(ret);
        }else{
          tamanho = __quali1.getValores().length;
          __categoriaPorCategorias.categoriaPorCategoria = new
CategoriaPorCategoria[(int)__quali1.getNumCategorias()];
          for(int p =0; p< (int)__quali1.getNumCategorias(); p++){
             __categoriaPorCategorias.categoriaPorCategoria[p] = new
CategoriaPorCategoria();
          }
          for(i=0; i<= __quali1.getNumCategorias()-1; i++){

__categoriaPorCategorias.categoriaPorCategoria[(int)i].setValor(__quali1.getC
ategorias()[(int)i].getValor());

__categoriaPorCategorias.categoriaPorCategoria[(int)i].setQtdade(0);

__categoriaPorCategorias.categoriaPorCategoria[(int)i].Categorias = new
TCategoria[(int)__quali2.getNumCategorias()];
             for(int p =0; p< (int)__quali2.getNumCategorias(); p++){

__categoriaPorCategorias.categoriaPorCategoria[(int)i].Categorias[p] = new
TCategoria();
```

```
            }
            for(j=0; j<=  __quali2.getNumCategorias()-1; j++){

__categoriaPorCategorias.categoriaPorCategoria[(int)i].Categorias[(int)j].set
Valor(__quali2.getCategorias()[(int)j].getValor());

__categoriaPorCategorias.categoriaPorCategoria[(int)i].Categorias[(int)j].set
Quantidade(0);
            }
          }
          for(i=0; i<= tamanho-1; i++){
              if((!"".equals(__quali1.getValores()[(int)i])) &&
(!"".equals(__quali2.getValores()[(int)i] ))){
                j=0;
                auxTam =
__categoriaPorCategorias.categoriaPorCategoria.length;
                while(j<auxTam){

if(!__categoriaPorCategorias.categoriaPorCategoria[(int)j].getValor().equals(
__quali1.getValores()[(int)i])){
                    j=j+1;
                  }else{
                    k=j;
                    j=auxTam+1;
                  }
                }
                if(j==auxTam){
                  return(11);
                }

__categoriaPorCategorias.categoriaPorCategoria[(int)k].setQtdade(__categoriaP
orCategorias.categoriaPorCategoria[(int)k].getQtdade()+1);
                l=0;
                auxTam =
__categoriaPorCategorias.categoriaPorCategoria[(int)k].Categorias.length;
                while(l<auxTam){

if(!__quali2.getValores()[(int)i].equals(__categoriaPorCategorias.categoriaPo
rCategoria[(int)k].Categorias[(int)l].getValor())){
                    l=l+1;
                  }else{
                    m=l;
                    l=auxTam+1;
                  }
                }
                if(l==auxTam){
                  return(11);
                }

__categoriaPorCategorias.categoriaPorCategoria[(int)k].Categorias[(int)m].set
Quantidade(__categoriaPorCategorias.categoriaPorCategoria[(int)k].Categorias[
(int)m].getQuantidade()+1);
              }
          }
          return(0);
        }
    }

    public int AcheNumeroDeCategorias(VariaveisQuantitativas __vdiscreta) {
        String[] NCat;
        int i,j,tamanho;
```

```
        int posicao = 0;
        boolean igual;
        tamanho =__vdiscreta.getValores().length;
        NCat = new String[tamanho];
        for(i=0; i<tamanho; i++){
            NCat[i] = "";
        }
        for(i= 0; i<= tamanho-1; i++){
          igual = false;
          for(j= 0; j<= NCat.length-1; j++){
            if(NCat[j].equals(__vdiscreta.getValores()[i])){
                 igual = true;
            }
          }
          if(igual == false){
            NCat[posicao] = __vdiscreta.getValores()[i];
            posicao++;
          }
        }
        return(posicao);
    }

    /*
     * @stereotype Testes Estatisticos
     */
    public int TesteAnova(VariaveisQualitativas __quali,
VariaveisQuantitativas __quanti, Objeto __GLT, Objeto __GLE, Objeto __SQT,
Objeto __SQE, Objeto __QMT, Objeto __QME, Objeto __ValorTeste, Objeto __p) {
        long NTotal,tamanho,i,j,auxN;
        double auxMediaGeral,F,ST,SE,auxSoma,auxMedia,auxDP;
        long[] NI;
        double[] MediaI;
        double[] DPI;
        int ret;
        ret = VerificaParametrosEntrada(__quali,__quanti);
        if(ret != 0){
           return(ret);
        }else{
          if(__quali.getNumCategorias()<=1){
             return(17);
          }else{
            tamanho= __quali.getValores().length;
            NI = new long[(int)__quali.getNumCategorias()];
            MediaI = new double[(int)__quali.getNumCategorias()];
            DPI = new double[(int)__quali.getNumCategorias()];
            auxMediaGeral=0;
            NTotal=0;
            for(i=0; i<= __quali.getNumCategorias()-1; i++){
               auxN=0;
               auxSoma=0;
               for(j=0; j<= tamanho-1; j++){
                    if((!"".equals(__quanti.getValores()[(int)j])) &&
(__quali.getValores()[(int)j].equals(__quali.getCategorias()[(int)i].getValor
()))){
                        try{
                          auxSoma=auxSoma+
Double.valueOf(__quanti.getValores()[(int)j]).doubleValue();
                          auxN=auxN+1;

auxMediaGeral=auxMediaGeral+Double.valueOf(__quanti.getValores()[(int)j]).dou
bleValue();
```

```
                            NTotal=NTotal+1;
                          }catch(NumberFormatException e){
                            return(3);
                          }
                      }
                  }
                  if(auxN<=1){
                    return(16);
                  }
                  auxMedia=auxSoma/auxN;
                  auxN=0;
                  auxSoma=0;
                  for(j=0; j<= tamanho-1; j++){
                    if((!"".equals(__quanti.getValores()[(int)j])) &&
(__quali.getValores()[(int)j].equals(__quali.getCategorias()[(int)i].getValor
())){
                      try{
                        double temp =
auxSoma=auxSoma+Math.pow(Double.valueOf(__quanti.getValores()[(int)j]).double
Value() - auxMedia,2);
                        auxN=auxN+1;
                      }catch(NumberFormatException e){
                        return(3);
                      }
                    }
                }
                if(auxN<=1){
                   return(16);
                }
                auxDP = Math.sqrt(auxSoma/(auxN-1));
                NI[(int)i]=auxN;
                MediaI[(int)i]=auxMedia;
                DPI[(int)i]=auxDP;
              }
              ST=0;
              SE=0;
              auxMediaGeral=auxMediaGeral/NTotal;
              for(i=0; i<= __quali.getNumCategorias()-1; i++){
                  ST=ST+NI[(int)i]*Math.pow(MediaI[(int)i]-auxMediaGeral,2);
                  SE=SE+(NI[(int)i]-1)*Math.pow(DPI[(int)i],2);
                }
              __SQT.setDouble(ST);
              __SQE.setDouble(SE);
              __GLT.setLong(__quali.getNumCategorias()-1);
              __GLE.setLong(NTotal-__quali.getNumCategorias());
              __QMT.setDouble(ST/__GLT.doubleValue());
              __QME.setDouble(SE/__GLE.doubleValue());
              F=__QMT.doubleValue()/__QME.doubleValue();
              __ValorTeste.setDouble(F);
              __p.setDouble(1-
DistribuicaoF(__ValorTeste.doubleValue(),__GLT.intValue(),__GLE.intValue()));
              return(0);
          }
        }
    }

    public int TesteTPares(VariaveisQuantitativas __quanti1,
VariaveisQuantitativas __quanti2, double __mediaTestada, Objeto __df, Objeto
__valorTeste, Objeto __p) {
        double Media,Soma,s,auxSoma;
        long tamanho,i,N,auxN;
```

```
        int ret;
        ret = VerificaParametrosEntrada(__quanti1,__quanti2);
        if(ret != 0){
          return(ret);
        }else{
          tamanho= __quanti1.getValores().length;
          auxN=0;
          auxSoma=0;
          for(i=0; i<= tamanho-1; i++){
             if((!"".equals(__quanti1.getValores()[(int)i])) &&
(!"".equals(__quanti2.getValores()[(int)i]))){
                 try{
                   auxSoma=auxSoma +
Float.valueOf(__quanti2.getValores()[(int)i]).floatValue() -
Float.valueOf(__quanti1.getValores()[(int)i]).floatValue();
                   auxN=auxN+1;
                 }catch(NumberFormatException e){
                   return(3);
                 }
             }
          }
          if(auxN<=1){
             return(16);
          }
          Media=auxSoma/auxN;
          Soma=0;
          N=0;
          for(i=0; i<= tamanho-1; i++){
            if((!"".equals(__quanti1.getValores()[(int)i])) &&
(!"".equals(__quanti2.getValores()[(int)i]))){
                try{

Soma=Soma+Math.pow((Float.valueOf(__quanti2.getValores()[(int)i]).floatValue(
) - Float.valueOf(__quanti1.getValores()[(int)i]).floatValue()) - Media,2);
                  N=N+1;
                }catch(NumberFormatException e){
                  return(3);
                }
             }
          }
          if(N<=1){
            return(16);
          }
          s=Math.sqrt(Soma/(N-1));
          if(s==0){
            return(6);
          }
          __df.setLong(N-1);
          __valorTeste.setDouble(((Media-__mediaTestada)*Math.sqrt(N)) / s);
          //if df>=30 then
          //   p=2*(1-DistribuicaoNormal(abs(ValorTeste)))
          //else
          __p.setDouble(2 * (1-
DistribuicaoTStudent(Math.abs(__valorTeste.doubleValue()),__df.intValue())));
          return(0);
        }
    }

    public int TesteTSeparate(VariaveisQualitativas __quali,
VariaveisQuantitativas __quanti, Objeto __df, Objeto __valorTeste, Objeto
__p) {
```

```
        long tamanho,i,j,auxN;
        double s0,s1,auxSoma;
        double[] media,soma;
        long[] n;
        int ret;
        media = new double[2];
        soma = new double[2];
        n = new long[2];
        ret = VerificaParametrosEntrada(__quali,__quanti);
        if(ret != 0){
          return(ret);
        }else{
          if((int)__quali.getNumCategorias() != 2){
               return(14);
          }
          tamanho= __quali.getValores().length;
          for(i=0; i<= 1; i++){
            auxSoma=0;
            auxN=0;
            for(j=0; j<= (int)tamanho-1;j++){
                if((!"".equals(__quanti.getValores()[(int)j])) &&
(__quali.getValores()[(int)j].equals(__quali.getCategorias()[(int)i].getValor
())))){
                    try{

auxSoma=auxSoma+Float.valueOf(__quanti.getValores()[(int)j]).floatValue();
                      auxN=auxN+1;
                    }catch(NumberFormatException e){
                      return(3);
                    }
                }
            }
            if(auxN<=1){
              return(16);
            }
            media[(int)i]=auxSoma/auxN;
            soma[(int)i]=0;
            n[(int)i]=0;
            for(j=0; j<=(int)tamanho-1;j++){
               if((!"".equals(__quanti.getValores()[(int)j])) &&
(__quali.getValores()[(int)j].equals(__quali.getCategorias()[(int)i].getValor
())))){
                  try{

soma[(int)i]=soma[(int)i]+Math.pow(Float.valueOf(__quanti.getValores()[(int)j
]).floatValue() - media[(int)i],2);
                    n[(int)i]=n[(int)i]+1;
                  }catch(NumberFormatException e){
                    return(3);
                  }
               }
            }
          }
          if((n[0]<=1) || (n[1]<=1)){
               return(16);
          }
          s0=soma[0]/(n[0]-1);
          s1=soma[1]/(n[1]-1);
          if(n[0]<n[1]){
             __df.setLong(n[0]-1);
          }else{
```

```
            __df.setLong(n[1]-1);
          }
          if((s0==0) && (s1==0)){
             return(6);
          }
          __valorTeste.setDouble((media[0]-media[1]) /
(Math.sqrt(s0/n[0]+s1/n[1])));
          __p.setDouble(2*(1-
DistribuicaoTStudent(__valorTeste.doubleValue(),__df.intValue())));
          return(0);
        }
    }

    public int TesteT(VariaveisQualitativas __quali, VariaveisQuantitativas
__quanti, Objeto __df, Objeto __valorTeste, Objeto __p) {
        long tamanho,i,j,auxN;
        double sp,s0,s1,auxSoma;
        double[] media,soma;
        long[] n;
        int ret;
        media = new double[2];
        soma = new double[2];
        n = new long[2];
        ret = VerificaParametrosEntrada(__quali,__quanti);
        if(ret != 0){
           return(ret);
        }else{
          if((int)__quali.getNumCategorias() != 2 ){
             return(14);
          }
          tamanho = __quali.getValores().length;
          for(i=0; i<= 1;i++){
                auxSoma=0;
                auxN=0;
                for(j=0; j<= (int)tamanho-1; j++){
                    if((!"".equals(__quanti.getValores()[(int)j])) &&
(__quali.getValores()[(int)j].equals(__quali.getCategorias()[(int)i].getValor
()))){
                        try{
                          auxSoma= auxSoma +
Float.valueOf(__quanti.getValores()[(int)j]).floatValue();
                          auxN=auxN+1;
                        }catch(NumberFormatException e){
                          return(3);
                        }
                    }
                }
                if(auxN<=1){
                  return(16);
                }
                media[(int)i]=auxSoma/auxN;
                soma[(int)i]=0;
                n[(int)i]=0;
                for(j=0; j<= (int)tamanho-1;j++){
                    if((!"".equals(__quanti.getValores()[(int)j])) &&
(__quali.getValores()[(int)j].equals(__quali.getCategorias()[(int)i].getValor
()))){
                       try{
                         soma[(int)i]=soma[(int)i] +
Math.pow(Float.valueOf(__quanti.getValores()[(int)j]).floatValue() -
media[(int)i],2);
```

```
                        n[(int)i]=n[(int)i]+1;
                       }catch(NumberFormatException e){
                           return(3);
                       }
                    }
                 }
          }
          if((n[0]<=1) || (n[1]<=1)){
               return(16);
          }
          s0=soma[0]/(n[0]-1);
          s1=soma[1]/(n[1]-1);
          if((s0==0) && (s1==0)){
               return(6);
          }
          __df.setLong(n[0]+n[1]-2);
          sp = ((n[0]-1)*s0+(n[1]-1)*s1) / __df.doubleValue();
          __valorTeste.setDouble((media[0]-media[1]) /
(Math.sqrt(sp/n[0]+sp/n[1])));
          __p.setDouble(2*(1-
DistribuicaoTStudent(__valorTeste.doubleValue(),__df.intValue())));
          return(0);
       }
    }

    public int TesteWilcoxon(VariaveisQuantitativas __quanti1,
VariaveisQuantitativas __quanti2, Objeto __somaMinima, Objeto __valorTeste,
Objeto __p) {
        long tamanho,ultimo,i,j,k;
        double aux1,aux2,somaP,somaN;
        double[][]  lista_temp;
        Objeto lista = new Objeto();
        Objeto lixo;
        int ret;
        ret = VerificaParametrosEntrada(__quanti1,__quanti2);
        k = 0;
        if(ret != 0){
          return(ret);
        }else{
          tamanho = __quanti1.getValores().length;
          lista_temp = new double[3][(int)tamanho];
          ultimo=0;
          for(i=0; i<=(int) tamanho-1;i++){
              if((!"".equals(__quanti1.getValores()[(int)i])) &&
(!"".equals(__quanti2.getValores()[(int)i]))){
                  try{
                    aux1 =
Float.valueOf(__quanti1.getValores()[(int)i]).floatValue();
                    aux2 =
Float.valueOf(__quanti2.getValores()[(int)i]).floatValue();
                  }catch(NumberFormatException e){
                    return(3);
                  }
                  ultimo=ultimo+1;
                  lista_temp[1][(int)ultimo-1]=Math.abs(aux1-aux2);
                  if((aux1-aux2)>0){
                     lista_temp[0][(int)ultimo-1]=1;
                  }else{
                     if((aux1-aux2)<0){
                        lista_temp[0][(int)ultimo-1]=-1;
                     }else{
```

```
                lista_temp[0][(int)ultimo-1]=0;
             }
          }
          lista_temp[2][(int)ultimo-1]=0;
     }
}
if(ultimo==0){
  return(13);
}
lista.___double = new double[3][(int)ultimo];
for(i=0; i<= ultimo-1; i++){
  lista.___double[0][(int)i] = lista_temp[0][(int)i];
  lista.___double[1][(int)i] = lista_temp[1][(int)i];
  lista.___double[2][(int)i] = lista_temp[2][(int)i];
}
ret = OrdeneListaTriplaPelaSegundaLista(lista);
if(ret != 0){
   return(ret);
}
i=0;
while(i<ultimo){
  if(lista.___double[1][(int)i]==0){
     i=i+1;
  }else{
     k=i;
     i=ultimo+1;
  }
}
if(i==ultimo){
   return(15);
}
if(k>0){
  ultimo=ultimo-k;
  for(j=0; j<=ultimo-1;j++){
     lista.___double[0][(int)j]=lista.___double[0][(int)k];
     lista.___double[1][(int)j]=lista.___double[1][(int)k];
     lista.___double[2][(int)j]=lista.___double[2][(int)k];
     k=k+1;
  }
  lista_temp = new double[3][(int)ultimo];
  for(j=0;j<=ultimo-1;j++){
    lista_temp[0][(int)j] = lista.___double[0][(int)j];
    lista_temp[1][(int)j] = lista.___double[1][(int)j];
    lista_temp[2][(int)j] = lista.___double[2][(int)j];
  }
  lista.___double = lista_temp;
  //SetLength(Lista[1],ultimo);
  //SetLength(Lista[2],ultimo);
  //SetLength(Lista[3],ultimo);
}
lixo = new Objeto(0);
ret = AchePostosEGuardeNaTerceiraLista(lista,lixo);
if(ret != 0){
   return(ret);
}
somaN=0;
somaP=0;
for(i=0; i<= ultimo-1; i++){
   if(lista.___double[0][(int)i] == 1){
      somaP=somaP+lista.___double[2][(int)i];
   }else{  //  =-1
```

```
                somaN=somaN+lista.___double[2][(int)i];
             }
          }
          //df=ultimo-1;
          if(somaN<somaP){
             __somaMinima.setDouble(somaN);
          }else{
             __somaMinima.setDouble(somaP);
          }
          __valorTeste.setDouble((__somaMinima.doubleValue() -
(ultimo*(ultimo+1)/4)) / Math.sqrt(ultimo*(ultimo+1)*(2*ultimo+1)/24));
          __p.setDouble(DistribuicaoNormal(__valorTeste.doubleValue())*2);
          return(0);
        }
    }

    public int TesteMannWhitney(VariaveisQualitativas __qualiOrd,
VariaveisQualitativas __quali, Objeto __soma1, Objeto __soma2, Objeto __n1,
Objeto __n2, Objeto __valorTeste, Objeto __p) {
        double media,dP,aux;
        Objeto lixo;
        long ultimo,i,j,tamanho;
        double[][] lista_temp;
        Objeto lista = new Objeto();
        int ret;
        ret = VerificaParametrosEntrada(__qualiOrd,__quali);
        if(ret != 0){
           return(ret);
        }else{
          tamanho = __quali.getValores().length;
          lista.___double = new double[3][(int)tamanho];
          if(__quali.getNumCategorias() != 2){
             return(14);
          }
          ultimo=0;
          for(i=0; i<= tamanho-1; i++){
              if((!"".equals(__quali.getValores()[(int)i])) &&
(!"".equals(__qualiOrd.getValores()[(int)i] ))){
                   try{
                      aux =
Float.valueOf(__qualiOrd.getValores()[(int)i]).floatValue();
                   }catch(NumberFormatException e){
                      return(3);
                   }
                   ultimo=ultimo+1;

if(__quali.getCategorias()[0].getValor().equals(__quali.getValores()[(int)i])
){
                      j=1;
                   }else{

if(__quali.getCategorias()[1].getValor().equals(__quali.getValores()[(int)i])
){
                         j=2;
                      }else{
                         return(11);
                      }
                   }
                   lista.___double[0][(int)ultimo-1]=j;
                   lista.___double[1][(int)ultimo-1]=aux;
                   lista.___double[2][(int)ultimo-1]=0;
```

```
            }
          }
          if(ultimo==0){
             return(13);
          }
          lista_temp = new double[3][(int)ultimo];
          for(i=0; i<= ultimo; i++){
            lista_temp[0][(int)i] = lista.___double[0][(int)i];
            lista_temp[1][(int)i] = lista.___double[1][(int)i];
            lista_temp[2][(int)i] = lista.___double[2][(int)i];
          }
          lista.___double = lista_temp;
          ret=OrdeneListaTriplaPelaSegundaLista(lista);
          if(ret != 0){
             return(ret);
          }
          lixo = new Objeto(0);
          ret = AchePostosEGuardeNaTerceiraLista(lista,lixo);
          if(ret != 0){
             return(ret);
          }
          double soma1 =0;
          double soma2 =0;
          long n1 = 0;
          long n2 = 0;
          for(i=0; i<=ultimo-1; i++){
             if(lista.___double[0][(int)i]==1){
                soma1 = soma1 + lista.___double[2][(int)i];
                n1=n1+1;
             }else{  //  =2
                soma2=soma2+lista.___double[2][(int)i];
                n2=n2+1;
             }
          }
          __soma1.setDouble(soma1);
          __soma2.setDouble(soma2);
          __n1.setLong(n1);
          __n2.setLong(n2);
          media = n1*(n1+n2+1)/2;
          dP= Math.sqrt(n1*n2*(n1+n2+1)/12);
          __valorTeste.setDouble(Math.abs((soma1-media)/dP));
          __p.setDouble((1-
DistribuicaoNormal(__valorTeste.doubleValue()))*2);
          return(0);
        }
    }

    public int TesteMannWhitney(VariaveisQuantitativas __quanti,
VariaveisQualitativas __quali, Objeto __soma1, Objeto __soma2, Objeto __n1,
Objeto __n2, Objeto __valorTeste, Objeto __p) {
        double media,dP,aux;
        Objeto lixo;
        long ultimo,i,j,tamanho;
        double[][] lista_temp;
        Objeto lista = new Objeto();
        int ret;
        ret = VerificaParametrosEntrada(__quali,__quanti);
        if(ret != 0){
           return(ret);
        }else{
          tamanho = __quali.getValores().length;
```

```
          lista.___double = new double[3][(int)tamanho];
          if(__quali.getNumCategorias() != 2){
               return(14);
          }
          ultimo=0;
          for(i=0; i<= tamanho-1; i++){
              if((!"".equals(__quali.getValores()[(int)i])) &&
(!"".equals(__quanti.getValores()[(int)i] ))){
                    try{
                       aux =
Float.valueOf(__quanti.getValores()[(int)i]).floatValue();
                    }catch(NumberFormatException e){
                       return(3);
                    }
                    ultimo=ultimo+1;

if(__quali.getCategorias()[0].getValor().equals(__quali.getValores()[(int)i])
){
                       j=1;
                    }else{

if(__quali.getCategorias()[1].getValor().equals(__quali.getValores()[(int)i])
){
                          j=2;
                       }else{
                          return(11);
                       }
                    }
                    lista.___double[0][(int)ultimo-1]=j;
                    lista.___double[1][(int)ultimo-1]=aux;
                    lista.___double[2][(int)ultimo-1]=0;
               }
          }
          if(ultimo==0){
            return(13);
          }
          lista_temp = new double[3][(int)ultimo];
          for(i=0; i<=ultimo-1; i++){
              lista_temp[0][(int)i] = lista.___double[0][(int)i];
              lista_temp[1][(int)i] = lista.___double[1][(int)i];
              lista_temp[2][(int)i] = lista.___double[2][(int)i];
          }
          lista.___double = lista_temp;
          ret=OrdeneListaTriplaPelaSegundaLista(lista);
          if(ret != 0){
            return(ret);
          }
          lixo = new Objeto(0);
          ret = AchePostosEGuardeNaTerceiraLista(lista,lixo);
          if(ret != 0){
            return(ret);
          }
          double soma1=0;
          double soma2=0;
          long n1=0;
          long n2=0 ;
          for(i=0; i<= ultimo-1; i++){
            if(lista.___double[0][(int)i]==1){
               soma1=soma1+lista.___double[2][(int)i];
               n1=n1+1;
           }else{  //  =2
```

```
                soma2=soma2+lista.___double[2][(int)i];
                n2=n2+1;
              }
            }
            __soma1.setDouble(soma1);
            __soma2.setDouble(soma2);
            __n1.setLong(n1);
            __n2.setLong(n2);
            media = n1*(n1+n2+1)/2;
            dP= Math.sqrt(n1*n2*(n1+n2+1)/12);
            __valorTeste.setDouble(Math.abs((soma1-media)/dP));
            __p.setDouble((1-
DistribuicaoNormal(__valorTeste.doubleValue()))*2);
            return(0);
         }
     }

    public int TesteKruskalWallis(VariaveisQualitativas __qualiOrd,
VariaveisQualitativas __quali, Objeto __Ns, Objeto __somas, Objeto __gl,
Objeto __valorTeste, Objeto __p) {
       long auxTam,ultimo,i,j,k,tamanho;
       double aux,H;
       Objeto correcao;
       double[][] lista_temp;
       Objeto lista;
       int ret;
       ret = VerificaParametrosEntrada(__qualiOrd,__quali);
       k =0;
       if(ret != 0){
         return(ret);
       }else{
           tamanho = __quali.getValores().length;
           lista = new Objeto();
           lista.___double = new double[3][(int)tamanho];
           ultimo=0;
           auxTam = __quali.getCategorias().length;
           for(i=0; i<= tamanho-1; i++){
               if((!"".equals(__quali.getValores()[(int)i])) &&
(!"".equals(__qualiOrd.getValores()[(int)i]))){
                    try{

aux=Float.valueOf(__qualiOrd.getValores()[(int)i]).floatValue();
                    }catch(NumberFormatException e){
                       return(3);
                    }
                    ultimo=ultimo+1;
                    j=0;
                    while(j<auxTam){

if(!__quali.getCategorias()[(int)j].getValor().equals(__quali.getValores()[(i
nt)i])){
                         j=j+1;
                      }else{
                         k=j;
                         j=auxTam+1;
                      }
                    }
                    if(j==auxTam){
                      return(11);
                    }
                    lista.___double[0][(int)ultimo-1]=k+1;
```

```
                    lista.___double[1][(int)ultimo-1]=aux;
                    lista.___double[2][(int)ultimo-1]=0;
                }
            }
            if(ultimo==0){
              return(13);
            }
            lista_temp = new double[3][(int)ultimo];
            for(i=0; i<=ultimo;i++){
              lista_temp[0][(int)i] = lista.___double[0][(int)i];
              lista_temp[1][(int)i] = lista.___double[1][(int)i];
              lista_temp[2][(int)i] = lista.___double[2][(int)i];
            }
            lista.___double = lista_temp;
            ret = OrdeneListaTriplaPelaSegundaLista(lista);
            if(ret != 0){
              return(ret);
            }
            correcao = new Objeto(0);
            ret = AchePostosEGuardeNaTerceiraLista(lista,correcao);
            if(ret != 0){
              return(ret);
            }
            __Ns.__long = new  long[(int)__quali.getNumCategorias()];
            __somas.__double = new double[(int)__quali.getNumCategorias()];
            for(i=0; i<= __quali.getNumCategorias()-1; i++){
               __Ns.__long[(int)i]=0;
               __somas.__double[(int)i]=0;
            }
            for(i=0; i<=ultimo-1;i++){
               j=1;
               while(lista.___double[0][(int)i] != j){
                 j=j+1;
               }
               __somas.__double[(int)j-1]=__somas.__double[(int)j-
1]+lista.___double[2][(int)i];
               __Ns.__long[(int)j-1]=__Ns.__long[(int)j-1]+1;
            }
            H=0;
            for(i=0; i<= __quali.getNumCategorias()-1; i++){
                if(__Ns.__long[(int)i]>0){
                   H = H +
Math.pow(__somas.__double[(int)i],2)/__Ns.__long[(int)i];
                }
            }
            H = 12/((ultimo)*(ultimo+1))*H - 3*(ultimo+1);
            correcao.setDouble(1-
correcao.doubleValue()/(((ultimo)*(ultimo)*(ultimo))-(ultimo)));
            __valorTeste.setDouble(H/correcao.doubleValue());
            __p.setDouble(1-
DistribuicaoQuiQuadrado(__valorTeste.doubleValue(),(int)__quali.getNumCategor
ias()-1));
            __gl.setLong(__quali.getNumCategorias()-1);
            return(0);
        }
    }

    public int TesteKruskalWallis(VariaveisQuantitativas __quanti,
VariaveisQualitativas __quali, Objeto __Ns, Objeto __somas, Objeto __gl,
Objeto __valorTeste, Objeto __p) {
        long auxTam,ultimo,i,j,k,tamanho;
```

```
        double aux,H;
        Objeto correcao, lista;
        double[][] lista_temp;
        int ret;
        k = 0;
        ret = VerificaParametrosEntrada(__quali,__quanti);
        if(ret != 0){
           return(ret);
        }else{
          tamanho = __quali.getValores().length;
          lista = new Objeto();
          lista.___double = new double[3][(int)tamanho];
          ultimo=0;
          auxTam = __quali.getCategorias().length;
          for(i=0; i<= tamanho-1; i++){
              if((!"".equals(__quali.getValores()[(int)i])) &&
(!"".equals(__quanti.getValores()[(int)i]))){
                  try{
                     aux =
Float.valueOf(__quanti.getValores()[(int)i]).floatValue();
                  }catch(NumberFormatException e){
                     return(3);
                  }
                  ultimo=ultimo+1;
                  j=0;
                  while(j<auxTam){

if(!__quali.getCategorias()[(int)j].getValor().equals(__quali.getValores()[(i
nt)i])){
                        j=j+1;
                    }else{
                        k=j;
                        j=auxTam+1;
                    }
                  }
                  if(j==auxTam){
                    return(11);
                  }
                  lista.___double[1][(int)ultimo-1]=k+1;
                  lista.___double[2][(int)ultimo-1]=aux;
                  lista.___double[3][(int)ultimo-1]=0;
                }
            }
          if(ultimo==0){
             return(13);
          }
          lista_temp = new double[3][(int)ultimo];
          for(i=0; i<=ultimo; i++){
              lista_temp[0][(int)i] = lista.___double[0][(int)i];
              lista_temp[1][(int)i] = lista.___double[1][(int)i];
              lista_temp[2][(int)i] = lista.___double[2][(int)i];
          }
          lista.___double = lista_temp;
          ret = OrdeneListaTriplaPelaSegundaLista(lista);
          if(ret != 0){
               return(ret);
          }
          correcao = new Objeto(0);
          ret = AchePostosEGuardeNaTerceiraLista(lista,correcao);
          if(ret != 0){
             return(ret);
```

```
          }
          __Ns.__long = new long[(int)__quali.getNumCategorias()];
          __somas.__double = new double[(int)__quali.getNumCategorias()];
          for(i=0; i<=__quali.getNumCategorias()-1; i++){
              __Ns.__long[(int)i]=0;
              __somas.__double[(int)i]=0;
          }
          for(i=0; i<=ultimo-1; i++){
             j=1;
             while(lista.___double[1][(int)i] != j){
               j=j+1;
             }
             __somas.__double[(int)j-1]=__somas.__double[(int)j-
1]+lista.___double[3][(int)i];
             __Ns.__long[(int)j-1]=__Ns.__long[(int)j-1]+1;
          }
          H=0;
          for(i=0; i<= __quali.getNumCategorias()-1; i++){
              if(__Ns.__long[(int)i]>0){
                 H = H+
Math.pow(__somas.__double[(int)i],2)/__Ns.__long[(int)i];
              }
          }
          H= 12/((ultimo)*(ultimo+1))*H - 3*(ultimo+1);
          correcao.setDouble(1-
correcao.doubleValue()/(((ultimo)*(ultimo)*(ultimo))-(ultimo)));
          __valorTeste.setDouble(H/correcao.doubleValue());
          __p.setDouble(1-
DistribuicaoQuiQuadrado(__valorTeste.doubleValue(),(int)__quali.getNumCategor
ias()-1));
          __gl.setLong(__quali.getNumCategorias()-1);
          return(0);
        }
    }

    protected double StatCom(double q, double i,double j, double b){
      double zz=1;
      double z=zz;
      double k=i;
      while(k<=j){
        zz=zz*q*k/(k-b);
        z=z+zz;
        k=k+2;
      }
      return z;
    }

    protected double Normal(double __valor){
            __valor = Math.abs(__valor);
            double p = 1 + __valor*(0.04986735+ __valor*(0.02114101+
__valor*(0.00327763+ __valor*(0.0000380036+ __valor*(0.0000488906+
__valor*0.000005383)))));
            p = p*p;
            p=p*p;
            p=p*p;
            return 1/(p*p);
        }

        protected double StudenT(double __valor, int __gl){
           __valor = Math.abs(__valor);
           double w =__valor/Math.sqrt(__gl);
```

```
            double th = Math.atan(w);
            if(__gl==1){
                return 1-th/(Math.PI/2);
            }
            double sth = Math.sin(th);
            double cth = Math.cos(th);
            if((__gl%2)==1){
                return 1-(th+sth*cth*StatCom(cth*cth,2,__gl-3,-
1))/(Math.PI/2);
            }else{
                return 1-sth*StatCom(cth*cth,1,__gl-3,-1);
            }
        }

        protected double ChiSq(double __valor, int __gl){
            if(__valor==1 & __gl>1000){
                return 0;
            }
            if(__valor>1000 | __gl>1000) {
              double q = DistribuicaoQuiQuadrado((__valor - __gl)*(__valor -
__gl)/(2 * __gl),1)/2;
              if(__valor>__gl){
                  return q;
              }else{
                 return 1-q;
              }
            }
            double p = Math.exp(-0.5 * __valor);
            if((__gl%2)==1){
               p = p * Math.sqrt(2*__valor/Math.PI);
            }
            double k = __gl;
            while( k >= 2){
               p = p * __valor / k;
               k = k - 2;
            }
            double t = p;
            double a = __gl;
            while( t >0.0000000001*p){
               a = a+2;
               t = t*__valor/a;
               p=p+t;
            }
            return 1-p;
        }

        protected double FishF(double __valor, int __gl1, int __gl2){
            double x=__gl2/(__gl1*__valor+__gl2);
            if((__gl1%2)==0){
                return StatCom(1 - x,__gl2,__gl1+__gl2-4,__gl2-2) *
Math.pow(x,__gl2/2);
            }
            if((__gl2%2)==0){
                return 1 - StatCom(x,__gl1,__gl1+__gl2-4,__gl1-2) *
Math.pow(1-x,__gl1/2);
            }
            double th = Math.atan(Math.sqrt(__gl1*__valor/__gl2));
            double a=th/(Math.PI/2);
            double sth = Math.sin(th);
            double cth = Math.cos(th);
            if(__gl2>1) {
```

```
                a=a+sth*cth*StatCom(cth*cth,2,__gl2-3,-1)/(Math.PI/2);
            }
            if(__gl1==1) {
                return 1-a;
            }
            double c = 4*StatCom(sth*sth,__gl2+1,__gl1+__gl2-4,__gl2-
2)*sth*Math.pow(cth,__gl2)/Math.PI;
            if(__gl2==1){
                return 1-a+c/2;
            }
            double k=2;
            while(k<=(__gl2-1)/2){
                c=c*k/(k-.5);
                k=k+1;
            }
            return 1-a+c;
        }


    /*
     * @clientCardinality 1
     * @supplierCardinality 1..*
     */
    private CategoriaPorIntervalo categoriaPorIntervalo;
    private CategoriaPorCategoria categoriaPorCategoria;
    private Intervalo intervalo;
}
```

Leitor_DBF.java

```
import java.sql.*;

public class Leitor_DBF {
    public Leitor_DBF() {
        connection = null;
        try {
            // Carregando o DBF_JDBC Driver
            String driverName = "com.hxtt.sql.dbf.DBFDriver"; // MySQL MM
JDBC driver
            Class.forName(driverName);
        } catch (ClassNotFoundException e) {
            //Driver não encontrado
            System.out.println("O driver expecificado não foi encontrado.");
        }
    }

    // Antes de conectar, deve-se setar o atributo arquivoDBF com um set!
    public  boolean ConectaDBF(String __endereco) {
      String myDB = "jdbc:dbf:/C:/SEstat.Net/Banco de dados/" + __endereco;
      try {
        connection = DriverManager.getConnection(myDB,"","");
        return true;
      } catch (SQLException e) {
        e.printStackTrace();  //To change body of catch statement use Options
| File Templates.
        return false;
      }
    }

    public String[] RetornaDadosVariavel(String __coluna, String
__nome_arquivo){
       arquivoDBF = __nome_arquivo;
       return RetornaDadosVariavel(__coluna);
    }

    public String[] RetornaDadosVariavel(String __coluna) {
       try {
            String[] temp = new String[numeroValores];
            ExecutaSql("SELECT "+__coluna + " from " + arquivoDBF);
            int i = 0;
            while(table.next()){
               temp[i] = table.getString(1);
               i++;
            }
            table.close();
            return temp;
        } catch (SQLException e) {
            e.printStackTrace();  //To change body of catch statement use
Options | File Templates.
            return null;
        }
    }

    public String[] RetornaDadosVariavelPorColuna(int __coluna, String
__nome_arquivo){
       arquivoDBF = __nome_arquivo;
       return RetornaDadosVariavelPorColuna(__coluna);
    }
```

```
    public String[] RetornaDadosVariavelPorColuna(int __coluna) {
       try {
           String[] temp = new String[numeroValores];
           ExecutaSql("SELECT * from " + arquivoDBF);
           table.beforeFirst();
           int i = 0;
           while(table.next()){
               temp[i] = table.getString(__coluna);
               i++;
           }
           table.close();
            return temp;
        } catch (SQLException e) {
            e.printStackTrace();  //To change body of catch statement use
Options | File Templates.
            return null;
        }
    }

    public String[] RetornaLabelColunas(String __nome_arquivo){
       arquivoDBF = __nome_arquivo;
       return RetornaLabelColunas();
    }

    public  String[] RetornaLabelColunas() {
        try {
            ExecutaSql("SELECT * from " + arquivoDBF);
            int colCount = table.getMetaData().getColumnCount();
            String[] temp = new String[colCount];
            for(int i=1; i<= colCount; i++){
                temp[i-1] = table.getMetaData().getColumnName(i);
            }
            table.close();
            return temp;
        } catch (SQLException e) {
            e.printStackTrace();  //To change body of catch statement use
Options | File Templates.
            return null;
        }
    }

    public void setNumeroValores(int __dados){
        numeroValores = __dados;
    }

    public void iniciaVariaveis(){
      //Reconhece Variaveis Qualitativas e Quantitativas
      String[] auxVariaveis;
      boolean ExisteString;
      int ValoresValidos;
      numeroValores = RetornaNumeroValores();
      String[] colunas = RetornaLabelColunas();
      numeroVariaveis = colunas.length;
      auxVariaveis = new String[numeroValores];
      valoresVariaveis = new String[numeroVariaveis][numeroValores];
      nomesVariaveis = new String[numeroVariaveis];
      VariaveisQuali = new VariaveisQualitativas[numeroVariaveis];
      VariaveisQuanti = new VariaveisQuantitativas[numeroVariaveis];
      ObjetoRefVariaveisQualitativas variavelQuali = new
ObjetoRefVariaveisQualitativas();
```

```
      ObjetoRefVariaveisQuantitativas variavelQuanti = new
ObjetoRefVariaveisQuantitativas();
      for(int i=0; i< numeroVariaveis; i++){
          ExisteString = false;
          ValoresValidos = 0;
          //SetLength(auxVariaveis,TamanhoBase);
          //while(j==tamanhoBase-1){
          auxVariaveis = this.RetornaDadosVariavelPorColuna(i+1);
          for(int k=0; k< auxVariaveis.length;k++){
             if(!"".equals(auxVariaveis[k])){
                ValoresValidos++;
                try{
                  Float.valueOf(auxVariaveis[k]).floatValue();
                }catch(NumberFormatException e){
                   ExisteString = true;
                }
             }
          }
          if(ValoresValidos>0){
             valoresVariaveis[i] = auxVariaveis;
             nomesVariaveis[i] = colunas[i];
             auxVariaveis = null;
             variavelQuali.variaveisQualitativas = VariaveisQuali[i];

ConstruaVariavelQualitativa(valoresVariaveis[i],colunas[i],variavelQuali);
             VariaveisQuali[i] = variavelQuali.variaveisQualitativas;
             if(!ExisteString){
                variavelQuanti.variaveisQuantitativas = VariaveisQuanti[i];

ConstruaVariavelQuantitativa(valoresVariaveis[i],colunas[i],variavelQuanti);
                 VariaveisQuanti[i] = variavelQuanti.variaveisQuantitativas;
             }
          }
      }
      auxVariaveis = null;
    }

    public  void DesconectaDBF() {
        try {
            connection.close();
        } catch (SQLException e) {
            e.printStackTrace();  //To change body of catch statement use
Options | File Templates.
        }
        connection = null;
        arquivoDBF = "";
    }

    public boolean FornecaVariavelQualitativa(String __nome,
ObjetoRefVariaveisQualitativas __VQuali){
        int j,passou;
        j = -1;
        passou = 0;
        for(int i=0; i< VariaveisQuali.length; i++){
            if(__nome.equals(VariaveisQuali[i].getNome())){
               j = i;
               passou = passou+1;
            }
        }
        if((j==-1) || (passou>1)){
           return false;
```

```
        }else{
           __VQuali.variaveisQualitativas = VariaveisQuali[j];
           return true;
        }
    }


    public boolean FornecaVariavelQuantitativa(String __nome,
ObjetoRefVariaveisQuantitativas __VQuanti){
        int j,passou;
        j=-1;
        passou=0;
        for(int i =0; i< VariaveisQuali.length; i++){
          if(__nome.equals(VariaveisQuali[i].getNome())){
              j=i;
             passou=passou+1;
           }
        }
        if((j==-1) || (passou>1)){
           return false;
        }else{
           //Acho que está errado esse código!!!!
           __VQuanti.variaveisQuantitativas = VariaveisQuanti[j];
           return true;
        }
    }

    public long getNumeroVariaveis(){
      return numeroVariaveis;
    }

    public String getArquivoDBF(){
      return arquivoDBF;
    }

    public void setArquivoDBF(String __arquivoDBF) {
        arquivoDBF = __arquivoDBF;
    }

    public String FornecaNomeVariavel(int i){
      return nomesVariaveis[i];
    }

    protected  void ExecutaSql(String __sql) throws SQLException {
       Statement s = connection.createStatement();
       table = s.executeQuery(__sql);
       s.close();
    }

    protected void ConstruaVariavelQualitativa(String[] __valores, String
__nome, ObjetoRefVariaveisQualitativas __VariavelQuali){
      boolean igual,TodosNumeros;
      long i, j, k, atual, missing;
      TCategoria[] Cat, tempCat;
      double aux;
      __VariavelQuali.variaveisQualitativas = new VariaveisQualitativas();
      __VariavelQuali.variaveisQualitativas.setNome(__nome);
      __VariavelQuali.variaveisQualitativas.setN(__valores.length);
      __VariavelQuali.variaveisQualitativas.setCategorias(new TCategoria[0]);
      atual = 0;
      missing = 0;
```

```
      while(atual<=__valores.length-1){
          igual = false;
          i = 0;
          if("".equals(__valores[(int)atual])){
             missing = missing+1;
          }else{
               while((!igual) &&
(i<=__VariavelQuali.variaveisQualitativas.getCategorias().length-1)){

if(__valores[(int)atual].equals(__VariavelQuali.variaveisQualitativas.getCate
gorias()[(int)i].getValor())){
                    igual = true;
                }else{
                    i++;
                }
              }

if(i==__VariavelQuali.variaveisQualitativas.getCategorias().length){
                   // Novo Valor encontrado
                    j =
__VariavelQuali.variaveisQualitativas.getCategorias().length;
                    for(i=
__VariavelQuali.variaveisQualitativas.getCategorias().length-1; i>= 0; i--){
                        if(Float.valueOf(__valores[(int)atual]).floatValue()
<
Float.valueOf(__VariavelQuali.variaveisQualitativas.getCategorias()[(int)i].g
etValor()).floatValue()){
                           j = i;
                        }
                    }
                    TCategoria[] temp = new
TCategoria[__VariavelQuali.variaveisQualitativas.getCategorias().length+1];
                    for (int l = 0; l
<__VariavelQuali.variaveisQualitativas.getCategorias().length;l++){
                        temp[l] =
__VariavelQuali.variaveisQualitativas.getCategorias()[l];
                    }
                    for (int m =
__VariavelQuali.variaveisQualitativas.getCategorias().length; m< temp.length;
m++) {
                        temp[m] = new TCategoria();
                    }

__VariavelQuali.variaveisQualitativas.setCategorias(temp);

for(i=__VariavelQuali.variaveisQualitativas.getCategorias().length-1; i>=
j+1; i--){

__VariavelQuali.variaveisQualitativas.getCategorias()[(int)i].setValor(__Vari
avelQuali.variaveisQualitativas.getCategorias()[(int)i-1].getValor());

__VariavelQuali.variaveisQualitativas.getCategorias()[(int)i].setQuantidade(_
_VariavelQuali.variaveisQualitativas.getCategorias()[(int)i-
1].getQuantidade());
                    }

__VariavelQuali.variaveisQualitativas.setCategorias((int)j,
__valores[(int)atual], 1);
               }else{

__VariavelQuali.variaveisQualitativas.getCategorias()[(int)i].setQuantidade(_
```

```
_VariavelQuali.variaveisQualitativas.getCategorias()[(int)i].getQuantidade()+
1);
                }
            }
            atual = atual+1;
        }
        __VariavelQuali.variaveisQualitativas.setMissing(missing);

__VariavelQuali.variaveisQualitativas.setNumCategorias(__VariavelQuali.variav
eisQualitativas.getCategorias().length);
        // Tenta ordenar as categorias como se fossem números
        TodosNumeros = true;
        Cat = null;
        for(i=0; i <=
__VariavelQuali.variaveisQualitativas.getCategorias().length-1; i++){
            try{

Float.valueOf(__VariavelQuali.variaveisQualitativas.getCategorias()[(int)i].g
etValor()).floatValue();
            }catch(NumberFormatException e){
              TodosNumeros = false;
            }
        }
        if(__VariavelQuali.variaveisQualitativas.getCategorias().length==0){
           TodosNumeros = false;
        }
        /*if(TodosNumeros){
             Cat = new TCategoria[1];
             Cat[0] = new TCategoria();

Cat[0].setValor(__VariavelQuali.variaveisQualitativas.getCategorias()[0].getV
alor());

Cat[0].setQuantidade(__VariavelQuali.variaveisQualitativas.getCategorias()[0]
.getQuantidade());
            for(i=1; i<=
__VariavelQuali.variaveisQualitativas.getCategorias().length-1; i++){
                aux =
Float.valueOf(__VariavelQuali.variaveisQualitativas.getCategorias()[(int)i].g
etValor()).floatValue();
                j=0;
                while((j<=Cat.length-1) &&
(Float.valueOf((Cat[(int)j].getValor())).floatValue() < aux)){
                  j++;
                }
                tempCat = new TCategoria[Cat.length+1];
                //SetLength(Cat,Length(Cat)+1);
                for(k = tempCat.length-1; k>= j+1; k--){
                   tempCat[(int)k].setValor(Cat[(int)k-1].getValor());
                   tempCat[(int)k].setQuantidade(Cat[(int)k-
1].getQuantidade());
                }

tempCat[(int)j].setValor(__VariavelQuali.variaveisQualitativas.getCategorias(
)[(int)i].getValor());

tempCat[(int)j].setQuantidade(__VariavelQuali.variaveisQualitativas.getCatego
rias()[(int)i].getQuantidade());
                Cat = tempCat;
            }
            __VariavelQuali.variaveisQualitativas.setCategorias(null);
```

```
            __VariavelQuali.variaveisQualitativas.setCategorias(Cat);
      }      */
      //
      __VariavelQuali.variaveisQualitativas.setValores(__valores);
      }

    protected void ConstruaVariavelQuantitativa(String[] __valores, String
__nome, ObjetoRefVariaveisQuantitativas __VariavelQuanti){
      long missing;
      double Soma;
      __VariavelQuanti.variaveisQuantitativas = new VariaveisQuantitativas();
      __VariavelQuanti.variaveisQuantitativas.setNome(__nome);
      __VariavelQuanti.variaveisQuantitativas.setN(__valores.length);
      Soma = 0;
      missing = 0;
      for(int i=0; i< __valores.length; i++){
        if("".equals(__valores[i])){
           missing++;
        }else{
           Soma = Soma+ Float.valueOf(__valores[i]).floatValue();
        }
      }

__VariavelQuanti.variaveisQuantitativas.setMedia(Soma/(__VariavelQuanti.varia
veisQuantitativas.getN()-missing));
      __VariavelQuanti.variaveisQuantitativas.setMissing(missing);
      Soma = 0;
      for(int i=0; i< __valores.length; i++){
        if(!("".equals(__valores[i]))){
           Soma = Soma+ Math.pow(Float.valueOf(__valores[i]).floatValue() -
__VariavelQuanti.variaveisQuantitativas.getMedia(),2);
        }
      }

__VariavelQuanti.variaveisQuantitativas.setDP(Math.sqrt(Soma/(__VariavelQuant
i.variaveisQuantitativas.getN()-missing-1)));
      __VariavelQuanti.variaveisQuantitativas.setValores(__valores);
    }

    public int RetornaNumeroValores() {
        try {
            ExecutaSql("SELECT COUNT(*) FROM " + arquivoDBF);
            table.next();
            int _numeroValores = table.getInt(1);
            table.close();
            return _numeroValores;
        } catch (SQLException e) {
            e.printStackTrace();  //To change body of catch statement use
Options | File Templates.
            return -1;
        }
    }
    private String[][] valoresVariaveis;
    private String[] nomesVariaveis;
    private int numeroVariaveis;
    private int numeroValores;
    private VariaveisQuantitativas[] VariaveisQuanti;
    private VariaveisQualitativas[] VariaveisQuali;
    private ResultSet table;
    private String arquivoDBF;
    private Connection connection;}
```