

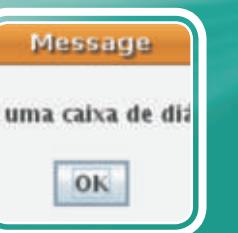


e-Tec Brasil  
Escola Técnica Aberta do Brasil

# Linguagem de Programação Comercial

Robison Cris Brito  
Beatriz Terezinha Borsoi

Curso Técnico em Meio Ambiente



e-Tec Brasil  
Escola Técnica Aberta do Brasil

ISBN:

**UTFPR**  
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

**UFMT**

Ministério da Educação



**e-Tec Brasil**  
*Escola Técnica Aberta do Brasil*

# Linguagem de Programação Comercial

Robison Cris Brito  
Beatriz Terezinha Borsoi



Cuiabá-MT  
2010

**Presidência da República Federativa do Brasil**  
**Ministério da Educação**  
**Secretaria de Educação a Distância**

© Universidade Tecnológica Federal do Paraná

Este caderno foi elaborado em parceria entre a UTFPR e a Universidade Federal de Mato Grosso para o Sistema Escola Técnica Aberta do Brasil – e-Tec Brasil.

**Comissão Editorial**

Profª Drª Maria Lucia Cavalli Neder - UFMT  
Profª Drª Ana Arlinda de Oliveira - UFMT  
Profª Drª Lucia Helena Vendrusculo Possari - UFMT  
Profª Drª Gleyva Maria Simões de Oliveira - UFMT  
Prof. M. Sc. Oreste Preti - UAB/UFMT

**Designer Educacional**

Oreste Preti

**Diagramação**

T. F. Oliveira/UFMT

**Revisão**

Germano Aleixo Filho

**Projeto Gráfico**

e-Tec/MEC

**Ficha Catalográfica**

# Apresentação e-Tec Brasil

Prezado estudante:

Bem-vindo ao e-Tec Brasil!

Você faz parte de uma rede nacional pública de ensino, a Escola Técnica Aberta do Brasil, instituída pelo Decreto nº 6.301, de 12 de dezembro 2007, com o objetivo de democratizar o acesso ao ensino técnico público, na modalidade a distância. O programa é resultado de uma parceria entre o Ministério da Educação, por meio das Secretarias de Educação a Distância (SEED) e de Educação Profissional e Tecnológica (SETEC), as universidades e escolas técnicas estaduais e federais.

A educação a distância no nosso país, de dimensões continentais e grande diversidade regional e cultural, longe de distanciar, aproxima as pessoas ao garantir acesso à educação de qualidade, e promover o fortalecimento da formação de jovens moradores de regiões distantes, geograficamente ou economicamente, dos grandes centros.

O e-Tec Brasil leva os cursos técnicos a locais distantes das instituições de ensino e para a periferia das grandes cidades, incentivando os jovens a concluir o ensino médio. Os cursos são ofertados pelas instituições públicas de ensino e o atendimento ao estudante é realizado em escolas-polo integrantes das redes públicas municipais e estaduais.

O Ministério da Educação, as instituições públicas de ensino técnico, seus servidores técnicos e professores acreditam que uma educação profissional qualificada – integradora do ensino médio e educação técnica, – é capaz de promover o cidadão com capacidades para produzir, mas também com autonomia diante das diferentes dimensões da realidade: cultural, social, familiar, esportiva, política e ética.

Nós acreditamos em você!

Desejamos sucesso na sua formação profissional!

Ministério da Educação  
Janeiro de 2010

Nosso contato  
[etecbrasil@mec.gov.br](mailto:etecbrasil@mec.gov.br)



# Indicação de ícones

Os ícones são elementos gráficos utilizados para ampliar as formas de linguagem e facilitar a organização e a leitura hipertextual.



**Atenção:** indica pontos de maior relevância no texto.



**Saiba mais:** oferece novas informações que enriquecem o assunto ou "curiosidades" e notícias recentes relacionadas ao tema estudado.



**Glossário:** indica a definição de um termo, palavra ou expressão utilizada no texto.



**Mídias integradas:** remete o tema para outras fontes: livros, filmes, músicas, *sites*, programas de TV.



**Atividades de aprendizagem:** apresenta atividades em diferentes níveis de aprendizagem para que o estudante possa realizá-las e conferir o seu domínio do tema estudado.



# Sumário

<b>Palavra das professores-autores</b>	9
<b>Apresentação da disciplina</b>	11
<b>Projeto instrucional</b>	13
<b>Aula 1 - Linguagens de Programação</b>	15
1.1 Conceito de linguagem de programação	15
1.2 Visão geral da linguagem Java	17
1.2.1 Edições da Linguagem Java	18
1.3 Ambiente de Desenvolvimento Integrado	19
1.4 Visão geral da IDE NetBeans	21
<b>Aula 2 - A IDE Netbeans e a linguagem de programação Java</b>	25
2.1 IDE NetBeans	25
2.1.1 Organização do Ambiente	27
2.2 Orientação a objetos em Java	29
2.3 A Linguagem de programação Java	31
2.3.1 Estrutura básica de um programa Java	31
2.3.2 Elementos básicos da linguagem java	33
<b>Aula 3 - Utilizando o Netbeans para o desenvolvimento visual de aplicações</b>	45
3.1 Desenvolvendo o primeiro projeto	45
3.2 Desenvolvendo o primeiro Formulário	47
3.3 Primeiro programa: calculadora	48
3.4 Mudando o layout da tela	51
3.5 Executando o aplicativo	52
3.6 Debugando projetos no NetBeans	53
3.6.1 Usando Breakpoint	54
3.6.2 Variáveis Locais	56
3.6.3 Observadores	57
<b>Aula 4 - Componentes visuais do Netbeans</b>	59
4.1 Componentes visuais no NetBeans	59

4.2 Paleta de componentes	61
4.2.1 Componentes da paleta Controle Swing mais utilizados	62
4.2.2 Outros componentes visuais do controles swing	66
4.2.3 Utilizando outras Janelas	71
4.2.4 Utilizando Menus	76
<b>Aula 5 - Desenvolvendo aplicativos com acesso à banco de dados</b>	<b>79</b>
5.1 Entendendo o que é um banco de dados	79
5.2 Trabalhando com Banco de Dados no NetBeans	81
5.3 Desenvolvendo o primeiro programa para manipular dados do banco de dados através do wizard do NetBeans	86
5.3.1 Entendendo o modelo de acesso a dados do NetBeans	90
5.4 Desenvolvendo aplicativo sem wizard com acesso a banco de dados	90
5.4.1 Desenvolvendo o projeto	91
5.4.2 Desenvolvendo a Interface Visual do projeto	91
5.4.3 Criando as classes para acesso ao banco de dados	91
5.4.4 Utilizando classes geradas pelo NetBeans para acessar banco de dados	95
<b>Aula 6 - Desenvolvendo relatórios no Netbeans</b>	<b>109</b>
6.1 Desenvolvendo relatórios no NetBeans	109
6.2 Obtendo e instalando o plugin do iReport para NetBeans	110
6.3 Criando relatórios para utilização em aplicações Desktop	112
6.4 Personalizando Relatório	117
6.4.1 Estrutura de um relatório	118
6.4.2 Paleta de Componentes de relatório	119
6.4.3 Propriedades dos componentes de um relatório	120
6.5 Chamando o Relatório desenvolvido	123
<b>Aula 7 - Distribuição do aplicativo desenvolvido</b>	<b>129</b>
7.1 Arquivo de Instalação	129
7.2 Gerando um .jar para projetos do NetBeans	130
<b>Retomando a palavra dos professores-autores</b>	<b>135</b>
<b>Referências</b>	<b>137</b>
<b>Curso técnico em meio ambiente</b>	<b>139</b>
<b>Curriculum dos professores-autores</b>	<b>141</b>

# Palavra dos professores-autores

---

## **Caro estudante!**

Convidamos você para participar da disciplina Linguagem de Programação Comercial.

É muito bom tê-lo(a) em nossa companhia para estudar este módulo por meio da modalidade à distância. No seu processo avaliativo, você resolverá questões dissertativas e objetivas, fará trabalhos e resolverá exercícios que irão compor a sua avaliação.

Além disso, também queremos que você faça as outras atividades propostas neste material para consolidar o conhecimento adquirido durante seus estudos. Assim, dedique tempo para fazer a leitura, as atividades e esclarecer dúvidas. Sempre que considerar necessário, volte ao texto e refaça as atividades. Não se limite a este material, existem indicações de várias outras leituras e estudos, faça pesquisas, converse com professores e colegas, questione, tudo isto para aprimorar seu conhecimento e auxiliar na sua formação profissional.

Aprender a desenvolver programas, utilizar uma linguagem de programação, está diretamente relacionado a fazer atividades práticas, portanto faça e refaça os exercícios até que tudo esteja bem compreendido.

Ao final desta disciplina, você estudante estará apto(a) a desenvolver pequenos aplicativos comerciais para plataforma desktop. Dentre os recursos, o aplicativo desenvolvido poderá contar com uma boa variedade de componentes visuais de interação com o usuário, acesso a uma base de dados para gravar e recuperar informações, emissão de relatórios impressos e em tela. Além de conhecer alguns dos comandos mais utilizados na linguagem de programação Java.

Bom estudo!



# Apresentação da disciplina

Nesta disciplina você poderá aprender conceitos básicos de uma linguagem de programação e desenvolver programas simples utilizando a linguagem Java.

Os conteúdos serão apresentados na seguinte ordem:

**Aula 1:** Linguagens de programação.

**Aula 2:** IDE NetBeans e a linguagem de programação Java.

**Aula 3:** Utilizar o NetBeans para o desenvolvimento visual de aplicações.

**Aula 4:** Componentes visuais disponíveis no NetBeans.

**Aula 5:** Desenvolver aplicativos com acesso à banco de dados.

**Aula 6:** Desenvolver relatórios no NetBeans.

**Aula 7:** Distribuição do aplicativo comercial desenvolvido.

A aula 1 apresenta os conceitos básicos de uma linguagem de programação para que você saiba o que é um programa (software) de computador.

A aula 2 apresenta a linguagem de programação Java, a IDE de desenvolvimento NetBeans e conceitos de orientação a objetos. São esses os três elementos básicos que norteiam esse curso. Os conceitos de orientação a objetos são a essência da programação em Java e você utilizará o ambiente NetBeans para fazer programas nessa linguagem.

A aula 3 apresenta o desenvolvimento visual de aplicações desktop utilizando a IDE NetBeans. Essa aula apresenta um passo a passo de como utilizar esse ambiente de desenvolvimento.

A aula 4 apresenta os principais componentes visuais utilizados para o desenvolvimento de aplicativos comerciais com a linguagem Java.

A aula 5 apresenta o acesso a um banco de dados utilizando componentes da IDE NetBeans. Com o conteúdo dessa aula, você pode fazer programas que

mantenham informações armazenadas em disco, recuperarem essas informações e trabalharem com elas. Por exemplo, fazer um cadastro de clientes para um sistema de controle de uma loja.

A aula 6 apresenta o desenvolvimento de relatórios para aplicações desktop. As informações armazenadas em banco de dados podem ser listadas sob a forma de relatórios, podendo estes ser impressos ou não.

A aula 7 apresenta o processo para distribuir e instalar o aplicativo desenvolvido nos computadores clientes.

No decorrer deste caderno serão apresentadas várias imagens do ambiente de desenvolvimento. Todas elas foram capturadas utilizando o sistema operacional Ubuntu (uma versão do Linux) e a IDE de desenvolvimento Java chamada Netbeans, na sua versão 6.5.1. Versões superiores podem ser utilizadas, uma vez que a IDE mantém a compatibilidade com versões mais antigas.

Portanto, esperamos que, ao final deste caderno, você esteja habilitado a entender a estrutura básica de uma linguagem de programação comercial e a desenvolver programas simples na linguagem Java.

# Projeto instrucional

**Disciplina:** Linguagem de Programação Comercial (carga horária: 60 horas).

**Ementa:** Implementar programas de computador desde o levantamento das necessidades do usuário, utilizando como ferramenta de desenvolvimento a linguagem de programação Visual; Desenvolver aplicações em computador utilizando um ambiente de programação visual; Empregar um linguagem/ambiente de programação de alto nível para transformar algoritmos em programas executáveis pelo computador, explorando o uso intensivo de bibliotecas.

AULA	OBJETIVOS	MATERIAIS	CARGA HORÁRIA
1. Linguagens de programação.	Apresentar os conceitos básicos de uma linguagem de programação para que você saiba o que é um programa (software) de computador.	Caderno de Disciplina Software: Netbeans Full	4 horas
2. DE NetBeans e a linguagem de programação Java.	Apresentar a linguagem de programação Java, a IDE de desenvolvimento NetBeans e conceitos de orientação a objetos.	Caderno de Disciplina Vídeo Aula Software: Netbeans Full	8 horas
3. Utilizar o NetBeans para o desenvolvimento visual de aplicações.	Apresentar o desenvolvimento visual de aplicações Desktop utilizando a IDE NetBeans.	Caderno de Disciplina Vídeo Aula Software: Netbeans Full	8 horas
4. Componentes visuais disponíveis no NetBeans	Apresentar os principais componentes visuais utilizados para o desenvolvimento de aplicativos comerciais com a linguagem Java.	Caderno de Disciplina Vídeo Aula Software: Netbeans Full	10 horas
5. Desenvolver aplicativos com acesso à banco de dados.	Apresentar o acesso a um banco de dados utilizando componentes da IDE NetBeans.	Caderno de Disciplina Vídeo Aula Software: Netbeans Full	15 horas
6. Desenvolver relatórios no NetBeans.	Apresentar o desenvolvimento de relatórios para aplicações desktop.	Caderno de Disciplina Vídeo Aula Software: Netbeans Full Plugin iReport	10 horas
7. Distribuição do aplicativo comercial desenvolvido.	Apresentar o processo para distribuir e instalar o aplicativo desenvolvido nos computadores clientes.	Software: Netbeans Full	5 horas



# Aula 1 - Linguagens de programação

## Objetivos

- Conceituar linguagem de programação.
- Apresentar conceitos relacionados às linguagens de programação visual, comercial, para web e desktop.
- Familiarizar o estudante com os conceitos básicos da linguagem de programação Java e o ambiente de desenvolvimento NetBeans.

**A aula é composta dos seguintes tópicos:**

- 1.1.** Conceito de linguagem de programação
- 1.2.** Visão geral da linguagem Java
- 1.2.1.** Edições da linguagem Java
- 1.3.** Ambiente de desenvolvimento integrado
- 1.4.** Visão geral da IDE Netbeans

## 1.1 Conceito de linguagem de programação

Para Mitchell (2003), as linguagens de programação são o meio de expressão na arte de programar computadores. Os programas de computador, também denominados sistemas ou software, fazem com que a máquina denominada computador tenha alguma utilidade. São exemplos de computador: servidor de rede (para gerenciar redes de computadores), desktop (computador de mesa), notebook (computador portátil), e telefone celular que permite jogar e executar aplicativos como agenda ou despertador.

Considerando a diversidade de tipos de computadores e sistemas (sejam operacionais ou aplicativos), é natural que existam muitos tipos de linguagens para programá-los. Há linguagens mais específicas, como as utilizadas para desenvolver sistemas para ambiente desktop, web ou para dispositivos portáteis como celulares. Existem também linguagens dependentes de sistema operacional ou plataforma. E ainda existem linguagens, como Java, que é

independente de plataforma (pode-se utilizá-la para desenvolver programas em várias plataformas).

Como existem várias linguagens de programação, podendo estas ser bastante diferentes e com finalidades distintas, todas compartilham o objetivo básico de permitir que um computador realize ações definidas por um ser humano. Porém, realizar ou executar essas ações depende: do problema (o objetivo que o programa visa alcançar) ter uma solução definida, ou seja, conhecida e que seja possível implementar essa solução com os recursos computacionais (incluindo linguagem de programação) existentes.



**Atenção:** Não é possível fazer um sistema computacional único para resolver qualquer problema existente. E em outras vezes os programas podem não ser tão eficientes como se esperaria, como exemplo cita-se sistemas para diagnóstico de doenças e previsão de tempo. Esses sistemas podem não ser tão eficientes como se esperaria porque não se tem uma solução completa para eles. Normalmente são muitas as variáveis envolvidas e nem sempre é conhecida a ação, interação e interdependência dessas variáveis, ou mesmo todas as variáveis que influenciam no problema.

Essas ações são instruções ou comandos para o computador. Para que o computador possa realizar uma instrução é necessário que ela seja escrita de acordo com uma sintaxe (a forma, por exemplo, para somar é utilizado o símbolo +) e a semântica (o significado, por exemplo, o símbolo + significa adicionar dois valores obtendo-se um terceiro que é resultado da soma dos dois primeiros) de uma linguagem de programação.

A Figura 1 apresenta o fluxo necessário para o desenvolvimento de um aplicativo, onde o programador desenvolve um algoritmo, codifica este utilizando uma linguagem de programação, transformando o algoritmo em um programa que será executado em um computador.

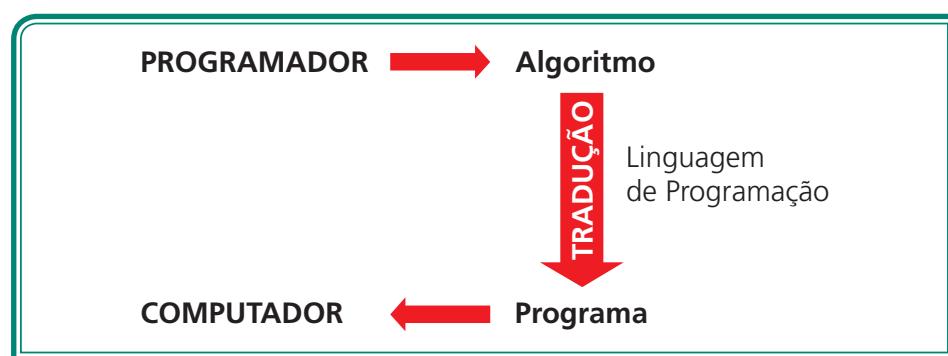


Figura 1: Fluxo para o desenvolvimento de um programa para computador

Os objetivos de uma linguagem estão relacionados ao seu tipo e aplicabilidade. Existem linguagens mais adequadas para implementar aplicações científicas, como, por exemplo, simulação de um terremoto e o cálculo do crescimento de uma população de bactérias de acordo com determinadas variáveis. Outras linguagens se destinam ao desenvolvimento de programas para aplicações comerciais. Outras, ainda, para sistemas de auxílio à tomada de decisões, para controle de acesso a ambientes, controle financeiro de uma empresa, automação de um supermercado, de uma loja, de um banco, de uma indústria, etc. Há também linguagens específicas para entretenimento, como jogos, animações, etc.

Dentre as linguagens estão as destinadas ao desenvolvimento de programas comerciais, sendo elas utilizadas no desenvolvimento de aplicações para automação de processos, rotinas e procedimentos comerciais. A ênfase dos programas feitos nessas linguagens está na interface gráfica, visando facilitar o seu uso, e no acesso, manipulação e armazenamento de dados.

**Atenção:** A facilidade de uso de um aplicativo comercial se refere aos recursos gráficos que facilitam a comunicação com o usuário, incluindo o uso de mouse e teclas de atalho. A usabilidade também está relacionada ao uso de recursos (como comandos de voz) que facilitam a acessibilidade para pessoas portadoras de necessidades especiais.



Uma linguagem de programação que possui recursos e componentes gráficos para desenvolver um programa de computador é comumente denominada linguagem de programação visual. Isso porque a composição do programa está centrada na estruturação e na definição de uma interface a partir de componentes visuais. A programação (definição dos códigos de acordo com a sintaxe e semântica da linguagem) é definida em função desses componentes.

A linguagem de programação Java é um exemplo de linguagem que permite o desenvolvimento visual de aplicativo comerciais. Em conjunto com ambientes integrados de desenvolvimento de aplicativos, é possível desenvolver aplicativos comerciais de maneira rápida e simples.

**A-Z**

Um ambiente integrado de desenvolvimento, ou simplesmente IDE (do inglês Integrated Development Environment) é um programa de computador que reúne características e ferramentas de apoio ao desenvolvimento de software com o objetivo de agilizar e facilitar a produção.

## 1.2 Visão geral da linguagem Java

Java é uma linguagem de programação de alto nível que faz uso da orientação a objetos. Na linguagem Java, a maior parte dos elementos de um programa são objetos. Uma exceção são os tipos primitivos, como o int e o double, muito semelhante aos tipos primitivos de linguagens como C e C++. Os códigos

gos Java são organizados em classes, que podem estabelecer relacionamentos entre si, como herança.

Quando um programa Java é compilado, um código intermediário é gerado, esse chamado de bytecode. Este bytecode é um código binário (baixo nível) e é interpretado pelas máquinas virtuais Java (JVMs). Essas máquinas virtuais também são responsáveis por carregar de forma segura todas as classes do programa, verificar se os bytecodes estão de acordo com a especificação JVM e se eles não violam a integridade e a segurança do sistema. Existem máquinas virtuais para diferentes sistemas operacionais, como Windows, Linux, etc.

### 1.2.1 Edições da Linguagem Java

A linguagem Java, também denominada plataforma ou tecnologia Java, pode ser classificada em três edições de acordo com a abrangência de suas funcionalidades e os propósitos de desenvolvimento. Essas especificações são: Java SE, Java EE e Java ME.

a) Java Standard Edition (Java SE) é a tecnologia Java que tem como foco o desenvolvimento de sistemas para desktop, notebooks e dispositivos com capacidade de processamento e memória consideráveis. Várias APIs (Application Programming Interface) acompanham esta versão e outras podem ser facilmente incorporadas aos ambiente. É com essas APIs que as aplicações são implementadas e executadas. Como exemplos de API estão a API de Gráfico (JFreeChart) e a API para relatórios (iReport).

Java SE contém todo o ambiente necessário para a criação e a execução de aplicações Java, incluindo a máquina virtual Java (JVM), o compilador Java, as APIs do Java e outras ferramentas utilitárias. O Java SE possibilita o desenvolvimento de aplicações com interface gráfica, acesso a bancos de dados, acesso à rede, dentre outros. O Java SE possui duas divisões, JDK e JRE:

a.1) Java Development Kit (JDK) ou Java Standard Development Kit (JSDK) - contém ferramentas de desenvolvimento, o ambiente de execução (JRE), a API Java SE (compilada e código-fonte), programas de exemplo, bibliotecas adicionais e documentação (que é obtida separadamente).

a.2) Java Runtime Edition (JRE) - é necessária para executar programas Java (bytecodes compilados) em outras máquinas, diferentes da utilizada para o desenvolvimento.

- b) Java EE (Java Enterprise Edition) - voltada para o desenvolvimento de aplicações corporativas complexas ou de porte maior. É no Java EE que são desenvolvidos aplicativos para ambiente internet (programas que rodam no servidor, como JSP e Servlet).
- c) Java ME (Java Micro Edition) – utilizado para o desenvolvimento de aplicações para dispositivos computacionais móveis de pequeno porte, tais como celulares, PDAs (Personal Digital Assistants) e set-top boxes (TV Digital). O Java ME possui máquinas virtuais adaptadas para dispositivos com recursos limitados, como velocidade de processamento e tamanho da tela pequena.

A Figura 2 apresenta as plataformas de desenvolvimento Java, dividido em Java Micro Edition, Java Standard Edition e Java Enterprise Edition.

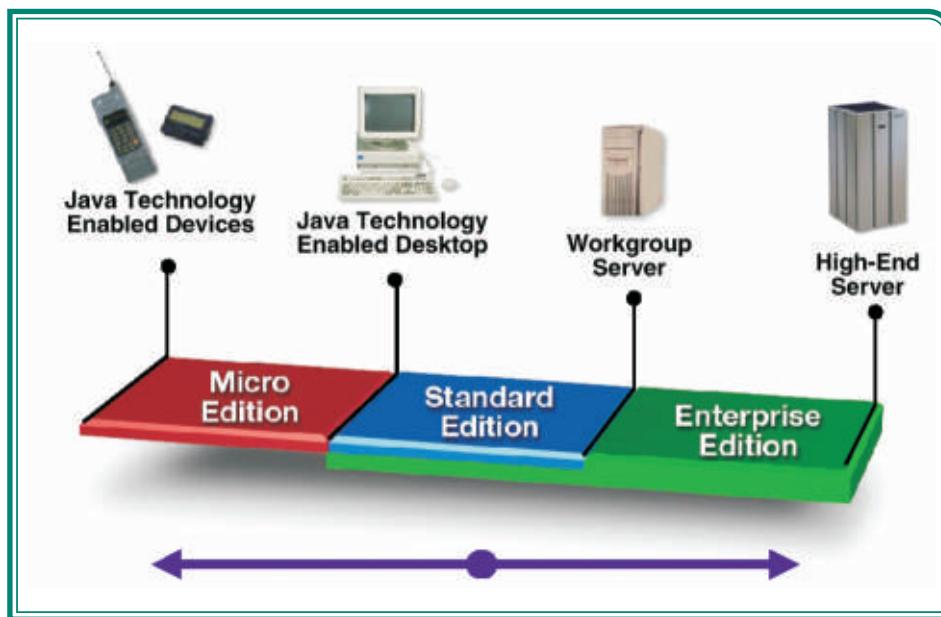
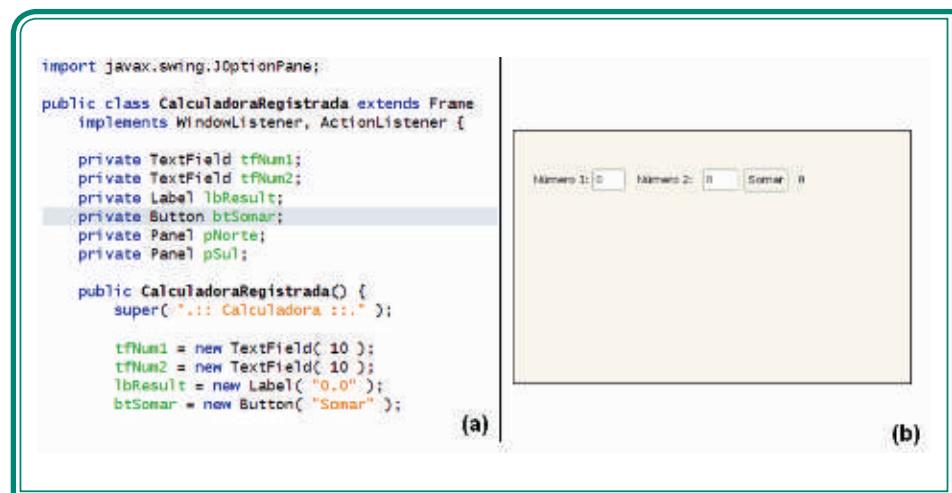


Figura 2: Plataformas de desenvolvimento Java (Site da Sun, 2010)

## 1.3 Ambiente de Desenvolvimento Integrado

A linguagem de programação Java, através de ambientes visual de desenvolvimento, também é utilizada para o desenvolvimento de aplicações comerciais. Para desenvolver programas com interface gráfica de maneira rápida é necessário utilizar um Ambiente de Desenvolvimento Integrado. A linguagem Java juntamente com o ambiente de desenvolvimento integrado NetBeans é denominada ambiente visual para desenvolvimento comercial.

A Figura 3 apresenta um comparativo entre o desenvolvimento de programas através de códigos fontes e utilizando recursos de IDE, o qual permite trabalhar diretamente com componentes prontos, clicando e arrastando eles para a tela do aplicativo.



**Figura 3: Comparativo entre o desenvolvimento através de códigos (a) e IDEs de desenvolvimento visual (b)**

As ferramentas mais comuns encontradas nas IDEs são:

- a)** Editor de código – permite escrever e editar o código-fonte do programa escrito na(s) linguagem(ns) suportada(s) pela IDE. O código fonte é o programa escrito de acordo com a sintaxe e a semântica de determinada linguagem. Esse código precisa passar por um processo de transformação para que ele possa ser executado pelo computador.
- b)** Compilador – verifica erros no código-fonte do programa, editado em uma linguagem específica e o transforma em códigos binários (baixo nível). Esses códigos binários contém as instruções que são efetivamente executadas pelo compilador
- c)** Linkeditor – liga as várias partes de código-fonte, compiladas em linguagem de máquina, em um programa executável.
- d)** Depurador (debugger) - auxilia no processo de encontrar e corrigir erros no código-fonte do programa.
- e)** Modelagem (modelling) - criação do modelo que auxilia a desenvolver um software. Esses modelos podem estar relacionados à interface do sistema,

à forma de navegação, aos diagramas que representam a composição interna do sistema, como classes, dentre outros.

**f)** Geração de código – o IDE pode gerar todo ou parte do código fonte do sistema automaticamente a partir de modelos.

**g)** Distribuição - auxilia no processo de criação do instalador do software, ou outra forma de distribuição do mesmo.

**h)** Testes Automatizados - realiza testes em software de forma automatizada de acordo com planos de testes especificados.

Pode-se citar como exemplos de IDE:

**a)** NetBeans - Gera código Java, dentre outras linguagens, considerada hoje uma das IDE's mais didáticas para o desenvolvimento de aplicativos comerciais. Esta é independente de plataforma (Windows, Linux, MacOS, etc).

**b)** Eclipse – Utilizado no desenvolvimento e gerenciamento de código Java (por meio de plugins, o Eclipse pode suportar diversas outras linguagens como Python e C/C++). Muito utilizado por desenvolvedores experientes, principalmente no desenvolvimento de aplicações WEB.

**c)** BlueJ - Gera código Java. Bastante utilizado no meio acadêmico para o ensino de lógica de programação e estrutura de dados.

**d)** Visual Studio .NET - Gera código para Framework .NET, suportando linguagens como Visual Basic .NET, C#, C++, e J#. Esta é um ambiente pago da Microsoft.

**e)** DEV-C++ e Code::Blocks - Geram código C e C++.

A Figura 4 apresenta algumas IDE de desenvolvimento Java, cada uma com suas características, suas especialidades, algumas são livres, outras pagas.

## 1.4 Visão geral da IDE NetBeans

A IDE NetBeans é um ambiente de desenvolvimento utilizado para produzir códigos fontes. Com ele é possível desenvolver aplicativos que fazem uso da linguagem de programação Java.



**Figura 4: Logotipo de algumas IDE utilizadas para o desenvolvimento Java**

A IDE NetBeans é multiplataforma (pode ser executado em vários sistemas operacionais), sendo utilizado para auxiliar o programador a escrever, compilar, depurar (debug) e instalar aplicações. Outra característica é possuir uma estrutura com componentes reutilizáveis, que visa simplificar o desenvolvimento e aumentar a produtividade.

Sua distribuição é realizada sob as condições da SPL (Sun Public License), uma variação da MPL (Mozilla Public License, resumidamente, através deste modelo de distribuição o Netbeans pode ser baixado e utilizado gratuitamente).

Alguns dos seus principais recursos são:

- a)** Editor de código fonte integrado, com recursos para aplicações Web (Servlets e JSP, JSTL, EJBs) e aplicações visuais com Swing. Swing é uma API Java para elaborar (desenhar) interfaces gráficas para aplicações desktop.
- b)** Visualizador de classes integrado ao de interfaces, que gera automaticamente o código dos componentes de forma organizada.
- c)** Suporte a Database (banco de dados), Data view e Connection wizard que são os módulos incopordados na IDE.
- d)** Geração de Javadoc, que é a documentação do programa, em arquivos HTML a partir dos comentários inseridos no código. NeBeans possui recursos que facilitam a inclusão de comentários no código.

## Resumindo

Esta aula apresentou os conceitos básicos de linguagem de programação, bem como uma visão geral da linguagem Java e da IDE NetBeans.

## Atividades de Aprendizagem



- 1)** O que é uma linguagem de programação? E uma IDE?
- 2)** Diferencie uma linguagem de programação compilada de uma linguagem de programação interpretada.
- 3)** Qualquer programa de computador pode ser implementado em qualquer linguagem de programação? Por quê?



# Aula 2 - A IDE Netbeans e a linguagem de programação Java

## Objetivos

- Apresentar a IDE NetBeans e a forma básica de utilizá-la.
- Exemplificar o uso da IDE NetBeans por meio de exemplos de programas em Java.
- Apresentar conceitos de orientação a objetos.
- Apresentar comandos básicos da linguagem Java.
- Definir a estrutura de uma classe na linguagem Java.

**A aula é composta dos seguintes tópicos:**

- 2.1.** IDE NetBeans
  - 2.1.1.** Organização do Ambiente
- 2.2.** Orientação a objetos
- 2.3.** Linguagem de programação Java
  - 2.3.1.** Estrutura básica de um programa Java
  - 2.3.1.** Elementos básicos da linguagem Java
    - 2.3.1.1.** Variáveis e tipos de dados
    - 2.3.1.2.** Comentários
    - 2.3.1.3.** Entrada e Saída de dados
    - 2.3.1.4.** Estruturas de controle
      - 2.3.1.4.1.** Estrutura de decisão
      - 2.3.1.4.2.** Estrutura de repetição

## 2.1 IDE NetBeans

A IDE NetBeans é um ambiente de desenvolvimento visual gratuito mantido pela Sun Microsystem, recentemente comprada pela empresa Oracle, que possibilita desenvolver aplicativos utilizando a linguagem de programação Java de modo visual, através do Clicar-Arrastar (Drag-and-Drop).

Com o NetBeans, é possível desenvolver aplicativos comerciais para as plataformas Desktop, Web e Mobile (celulares), aumentando significativamente a produtividade no desenvolvimento de aplicações, já que a IDE implementa automaticamente o código referente à interface visual, permitindo ao programador dedicar mais tempo à lógica de negócios do aplicativo.

Para ter acesso à ferramenta, deve-se acessar o site do projeto (<http://www.netbeans.org>) e fazer o download da IDE. Existem algumas versões da ferramenta, como apresentada na Figura 5.

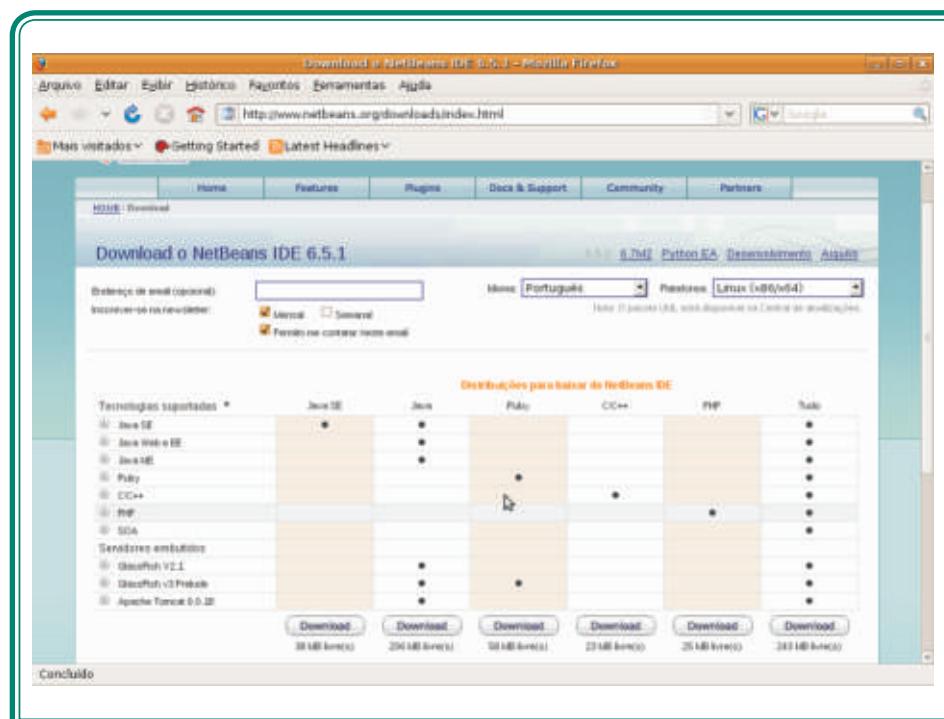


Figura 5: Versões para Download da IDE NetBeans

Para o desenvolvimento comercial desktop, pode ser utilizada a versão Java SE, o qual dá suporte ao clicar-arrastar de componentes visuais, modificar visualmente suas propriedades, possuindo também uma ferramenta de debugação e distribuição do aplicativo.



**Atenção:** Para a instalação do NetBeans, é necessário que o computador já possua o Kit de Desenvolvimento Java Padrão (JSDK-Java Standard Development Kit). Caso não o possua, o mesmo pode ser baixado do link <http://java.sun.com/javase/downloads>. Nesse site é possível baixar isoladamente o Kit de Desenvolvimento Java, ou uma versão do NetBeans já integrado com o JSDK.

O NetBeans está disponível para vários sistemas operacionais, como Linux, Windows e MacOs. Após o download da IDE para o sistema operacional desejado, basta realizar a instalação. Não existem configurações especiais, bastando avançar todas as telas de instalação.

## 2.1.1 Organização do Ambiente

Após a execução da IDE, uma interface visual será apresentada, interface esse apresentada na Figura 6.

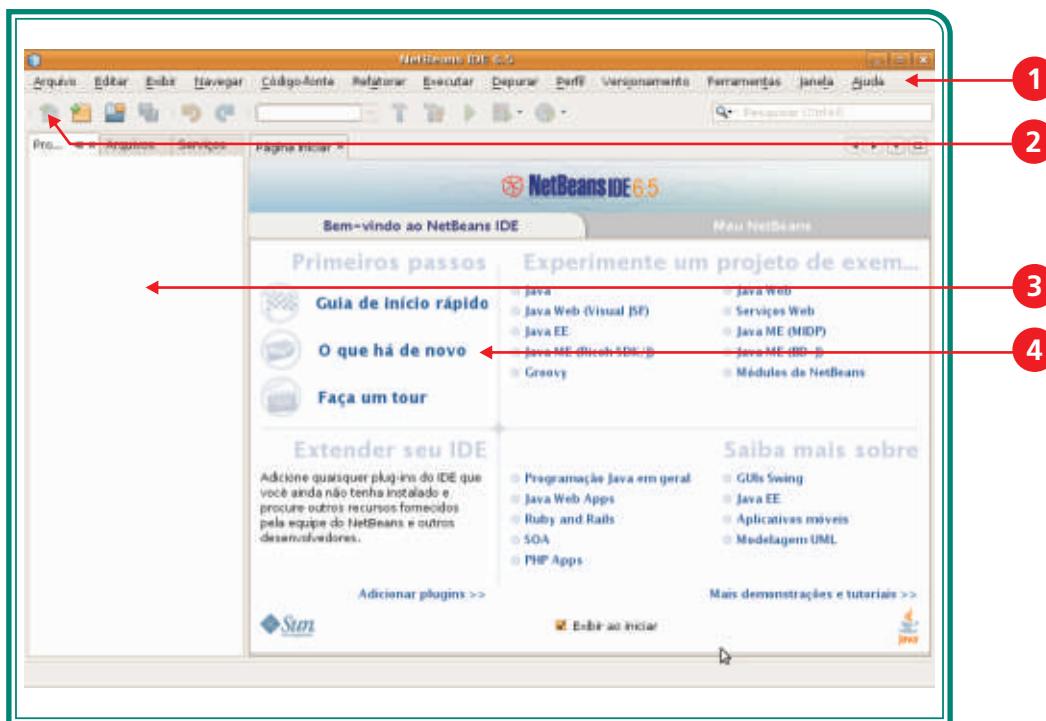


Figura 6: Interface visual da IDE NetBeans

A interface é dividida em barra de menus (1), barra de ferramentas (2), área para organização do projeto (3) e área para o desenvolvimento (4). Essa interface pode ser personalizada pelo usuário.

Para os exemplos apresentados na sequência, cujo objetivo é apresentar a sintaxe da linguagem Java, não será utilizada interface visual. A captura dos comandos e a saída dos dados serão feitos através da console (prompt de comando).

Para iniciar o desenvolvimento de um projeto que faz uso da console, deve-se escolher o menu Arquivo e escolher a opção Novo Projeto... Será apresentada a interface visual conforme Figura 7.

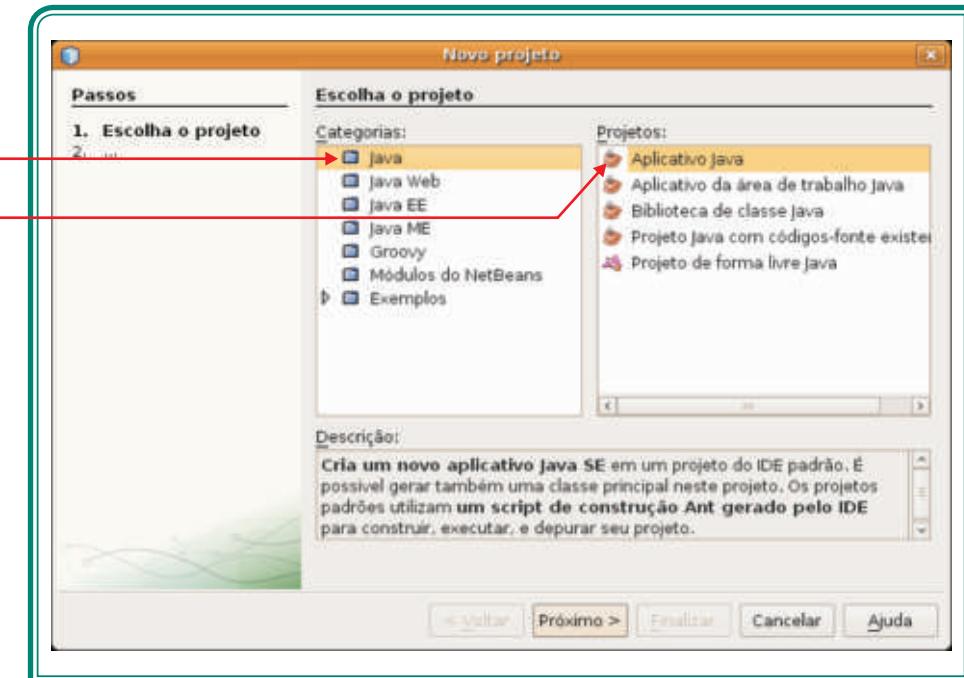


Figura 7: Tela do Wizard para criação de um novo projeto no NetBeans

Na tela apresentada, deve ser escolhida a Categoria Java (5) e o Projeto Aplicativo Java (6). Após clicar em próximo será apresentada uma nova tela - Figura 8.

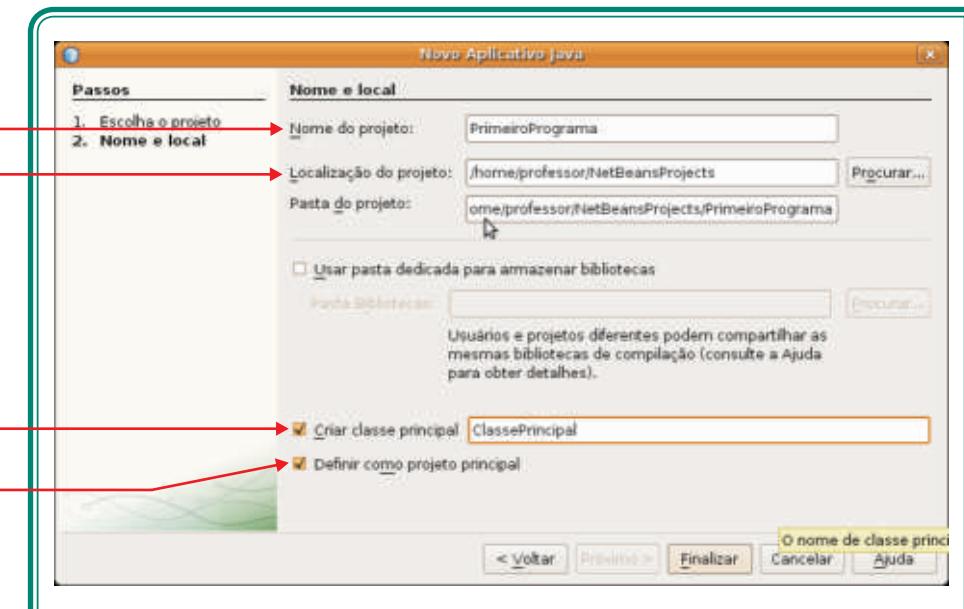


Figura 8: Tela para informar nome do projeto e local onde será armazenado

Nessa tela, deve ser informado o nome do projeto (7), o local onde o mesmo será salvo no computador (8), o nome da classe principal do projeto, caso exista (9) e se o projeto será definido como projeto principal (10).

Sendo definido como projeto principal, ao clicar no botão executar (Figura 9), o conteúdo da classe principal é executado.

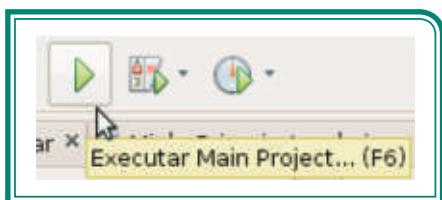


Figura 9: Botão utilizado para executar o projeto

Com as informações apresentadas anteriormente, é possível desenvolver classes para testar os conceitos que serão apresentados na sequência, lembrando que é aconselhável a criação de um projeto para cada programa que se deseja implementar.

Antes de apresentarmos os primeiros comandos da linguagem de programação Java, serão apresentados os conceitos básicos da orientação a objetos, uma vez que a linguagem Java usa este paradigma no desenvolvimento de aplicações.

## 2.2 Orientação a objetos em Java

Nesta seção inicialmente são apresentados os conceitos básicos da orientação a objetos. Isso para que se tenha um entendimento do que é um objeto e de como ele é tratado em uma linguagem de programação orientada a objetos.

O termo orientação a objetos pressupõe organizar um software como uma coleção de objetos.

Em termos gerais, objetos podem representar coisas do mundo real (um botão da tela, uma caixa de texto, um formulário). No desenvolvimento de aplicativos orientados a objetos, todo objeto possui uma identificação única, permitindo individualizar cada objeto de todos os demais.

Além de uma identificação única, um objeto pode possuir atributos (ou características) e métodos (ou ações). Por exemplo, pode-se citar características de um botão seu texto, sua cor, a largura e a altura (medida em píxeis, etc.). Como métodos, destaca-se o procedimento executado quando este botão é pressionado, ou quando o usuário coloca o mouse sobre o mesmo.

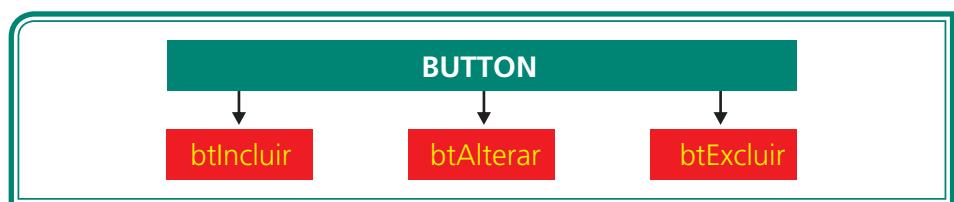
**Atenção:** Um objeto é uma entidade que possui identidade, atributos (características, propriedades) e operações (comportamento, métodos). Na lingua-



gem de programação comercial, pode-se considerar todos os componentes visuais como sendo objetos.

Todos os objetos em um programa são instâncias de classes. Exemplificando, as classes representam os tipos de componentes e os objetos os componentes em si.

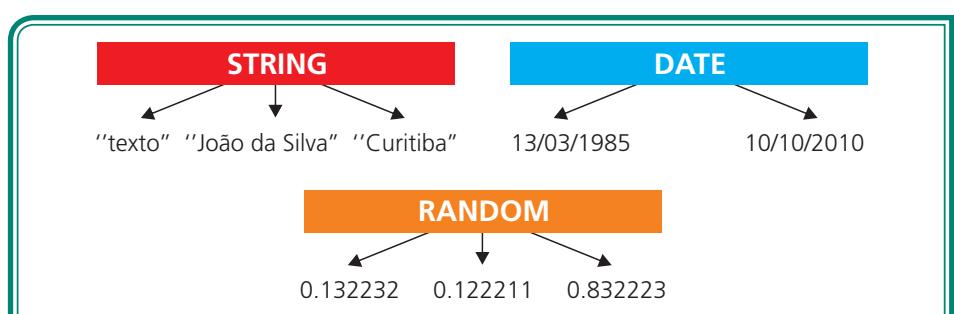
Na Figura 10 é possível visualizar os objetos identificados por btIncluir, btAlterar e btExcluir, estes objetos reais que executam as funções de incluir, alterar e excluir registros de um banco de dados. Estes objetos são instâncias da classe Button, que define as características de um Botão, como formato, estilo da borda, formato do texto apresentado, etc.



**Figura 10:** Objetos “reais” btIncluir, btAlterar e btExcluir, estes instâncias da Classe Button

Uma classe é uma abstração que define como serão os seus objetos. Em outras palavras, um objeto é a instância de uma classe. Assim, todos os objetos instanciados a partir da classe Button possuem a mesma estrutura (características), como por exemplo, os botões serão desenhos retangulares, representados em alto relevo, e ao receber um clique executam uma ação específica.

Porém, nem sempre as classes representam componentes visuais, como caixas de texto, botões, formulários, etc. Em algumas situações, classes podem representar informações, como um texto ou os dados referentes a um banco de dados. A Figura 11 mostra exemplos de outros tipos de classe.



**Figura 11:** Exemplos de Classes e instâncias não visuais, como String (textos), Datas e Número randômicos

## 2.3 A Linguagem de programação Java

A linguagem Java foi projetada para ser pequena, simples e portável a diversas plataformas e sistemas operacionais. Esta portabilidade é obtida pelo existência de uma máquina virtual java.

### 2.3.1 Estrutura básica de um programa Java

Esta seção apresenta a estrutura básica de um programa em linguagem Java, visando indicar o que é necessário para implementar um programa nessa linguagem. Nas subseções seguintes cada um desses elementos é apresentado em detalhes.

Uma linguagem de programação possui um conjunto de palavras reservadas, que são as palavras com significado para o interpretador/compilador, os seus elementos fundamentais e uma sintaxe e semântica próprias. As palavras reservadas ou palavras-chave são identificadas pelo compilador/interpretador e possuem significado determinado pela linguagem. Os itens fundamentais representam, em essência e fundamentalmente, o que a linguagem pode fazer. A sintaxe e a semântica definem como representar as instruções, os comandos da linguagem.

De maneira geral, um programa implementado em qualquer linguagem de programação deve tratar variáveis, obter dados de entrada, apresentar dados de saída, controlar se determinadas instruções do programa serão realizadas e se instruções serão repetidas.

As variáveis são os dados manipulados pelo programa. Esses dados podem ser fornecidos pelo usuário, como o nome e o endereço para um cadastro e o valor do salário e do percentual de impostos para calcular o valor do salário líquido. Uma constante é uma variável que não é alterada durante a execução do programa em todas as vezes que o programa for executado. O valor de PI, que é 3.1415, a quantidade de dias de uma semana são sempre os mesmos, independentemente de quantas vezes um programa é executado. Esses são exemplos de constantes. Uma constante normalmente possui o seu valor atribuído pelo programador durante a implementação do programa.

Os dados de entrada de um programa são os valores necessários para que o mesmo possa executar. É útil que o usuário possa informar valores para um programa. Por exemplo, em um programa de controle de estoque, o usuário

### A-Z

A Máquina Virtual Java, ou JVM do inglês Java Virtual Machine, é um software com a função de executar o código Java desenvolvido pelos programadores. Como existem várias versões da máquina virtual Java, um programa desenvolvido em Windows pode ser executado por uma máquina virtual Java no sistema operacional Linux ou Mac OS, o que garante a portabilidade de um aplicativo Java.

insere as informações dos produtos, cadastrando-os, inclui a quantidade comprada, a quantidade vendida e o programa faz o cálculo do valor em estoque.

Os dados de saída são provenientes das operações realizadas pelos programas. Esses dados são, normalmente, apresentados na tela do computador ou impressos. Contudo, eles também podem ser armazenados em arquivos (discos e outros meios físicos de armazenamento), ser transmitidos para outros programas, emitir comandos para equipamentos, como, por exemplo, para um sensor para abrir uma porta ao reconhecer a impressão digital de uma pessoa.

O controle que determina se instruções de um programa são realizadas ou repetidas é realizado por estruturas, ou comandos, denominadas de estruturas de decisão e de repetição. Uma estrutura de decisão determina se um conjunto de comandos será ou não executado. Por exemplo, se o salário for menor que R\$ 700,00, é isento de descontos. Uma estrutura de repetição permite repetir determinado conjunto de instruções até que uma condição seja satisfeita. Por exemplo, ler o nome e a idade de trinta alunos; ler valores e verificar se cada um desses valores é um número primo e o programa continuar em execução até que o usuário informar zero como valor a ser verificado.

Além dos elementos básicos, uma linguagem possui uma forma, normalmente particular, de organizar esses elementos. Os elementos definem a sintaxe (a forma) da linguagem. A forma de organizar esses elementos constitui a sua semântica.

Como a linguagem Java é orientada a objetos, a sua estrutura é baseada em classes. Classes e os seus objetos possuem atributos (características) e métodos (operações).

Para exemplificar, a Listagem 1 contém um programa Java que escreve uma mensagem na tela.

```
1. public class Mensagem {  
2.     public static void main(String args[ ]) {  
3.         System.out.println("Linguagem Java");  
4.     }  
5. }
```

**Listagem 1: Programa Java para mostrar uma mensagem na tela**

Entendendo o código:

Linha 1 - Todo programa em Java inicia com a palavra reservada class, a qual pode ser pública (palavra public no início da instrução) ou não, seguida pelo nome da classe, nesse caso Mensagem.

Linha 2 e Linha 7 - Um par de chaves (abre e fecha chaves) envolve todo o código de uma classe. Isso ocorre para cada um das classes do programa, seja a principal ou as demais classes. Chaves também são utilizadas para agrupar conjuntos de instruções pertencentes a estruturas de controle ou de decisão. Entre as linhas 2 a 7 está o código da classe Mensagem. Entre as linhas 4 a 6 está o código pertencente ao método main.

Linha 3 - Toda classe Java que é executável deve, obrigatoriamente, possuir o método main. Esse método é invocado quando a classe é executada. A instrução da linha 5 será executada quando o método main for invocado. Essa instrução imprimirá a mensagem “Linguagem Java” na console do computador.

**Atenção:** A linguagem Java é sensível a letras maiúscula/minúscula, por esse motivo deve-se programar com muito cuidado. Por exemplo, a variável nomeDoCliente é diferente da variável nomedocliente e NomeDoCliente.



**Atenção:** Entretanto existe uma regra para minimizar os problemas com os nomes. Em Java todas as palavras iniciam com letras minúsculas. A única exceção são as classes, ou seja, palavras iniciadas em letra maiúscula costumam ser utilizadas apenas por nome das classes (Ex. Mensagem, String, System). A letra maiúscula costuma ser utilizada também quando se possui uma variável/objeto com nome complexo, sendo nessa situação utilizado para facilitar a leitura (ex. nomeDoUsuario, dataDeNascimento).



**Atenção:** A indicação das linhas (o número no início de cada linha de código) é apenas para facilitar a referência nesse material. Esse número e nem o ponto que o segue não são indicados quando da escrita do programa Java no editor.



## 2.3.2 Elementos básicos da linguagem java

As subseções a seguir apresentam os elementos básicos da linguagem de programação Java.

### 2.3.2.1 Variáveis e tipos de dados

Variáveis são endereços (espaços) de memória nas quais são armazenados

dados utilizados pelo programa quando o mesmo está sendo executado. Elas são definidas de acordo com um tipo e armazenam um dado de cada vez. Por exemplo, a variável `idade` armazena um dado do tipo numérico inteiro, a variável `preço` um dado do tipo numérico real.

A seguir um exemplo de declaração de variável na linguagem Java.

```
int idade;  
double preço;
```

Após a declaração de uma variável é possível atribuir um valor para a mesma. Essa atribuição é realizada usando o operador '='.

```
idade = 18;  
preço = 125.98;
```

Exemplo de declaração e atribuição imediata de um valor inteiro para uma variável:

```
int idade = 30;
```

Apesar da String ser uma classe, a mesma tem um comportamento idêntico aos tipos primitivos, por exemplo:

```
String nome = "João da Silva";
```

O código anterior determina que o nome é uma variável do tipo texto (uma sequência de caracteres que em inglês é denominada String). Ela pode armazenar valores que sejam uma conjunto de caracteres (letras, números e símbolos como %, #, \*, &).

### 2.3.2.2 Comentários

Comentários são técnicas para fazer o compilador ignorar um trecho de código. Na linguagem Java, dois tipos de comentários são utilizados com mais freqüência:

- a) /\* e \*/ -** comenta tudo que estiver entre os dois delimitadores é ignorado.
- b) // -** usados para comentar a partir das suas barras até o final da respectiva linha.

### 2.3.2.3 Entrada e Saída de dados

A entrada fornece meios do usuário digitar informações que serão tratadas por um programa Java. A saída apresenta o resultado do processamento dessas informações.

Como exemplo de comando para saída de dados, tem-se o:

```
System.out.println( );
```

Este já utilizado anteriormente em alguns exemplos. Como entrada de dados, destaca-se a classe Scanner, que pertence ao pacote java.util. Este permite obter dados informados pelo usuário por meio do teclado no modo console. Essa classe contém métodos para leitura de vários tipos de dados.

A Listagem 2 contém um exemplo de uso da classe Scanner.

```
1. import java.util.Scanner;
2. public class UsandoScanner {
3.
4.     public static void main(String[] args) {
5.
6.         String nome;
7.         int idade;
8.         Scanner entrada = new Scanner(System.in);
9.         System.out.print("Informe o seu nome. ");
10.        nome = entrada.nextLine();
11.        idade = entrada.nextInt();
12.        System.out.println("Olá " + nome + " você tem " + idade + " anos. ");
13.    }
14. }
```

Listagem 2: Uso da classe Scanner

Nesse exemplo é obtido um valor texto (String – linha 10) e um valor inteiro (int – linha 11) a partir da console.

### 2.3.2.4 Estruturas de controle

As estruturas de controle permitem estabelecer que determinado conjunto de instruções seja ou não executado ou que um conjunto de instruções seja repetido determinado número de vezes.

Nas estruturas de controle, são definidas através de testes lógicos se uma instrução será ou não executada ou a quantidade de vezes que esta será executada.

As estruturas de controle podem ser divididas em estruturas de decisão e estruturas de repetição. Essas estruturas também são denominadas comandos.

#### 2.3.2.4.1 Estrutura de decisão

Uma estrutura de decisão também é conhecida como estrutura condicional, isto porque a execução de um determinado conjunto de instruções está condicionada ao resultado de um teste lógico.

#### Estrutura if

O comando if tem duas variações, estas apresentadas na Figura 12. A primeira variação trata apenas a condição de verdadeiro (a), já a segunda variação trata ambas as condições: verdadeiro e falso.

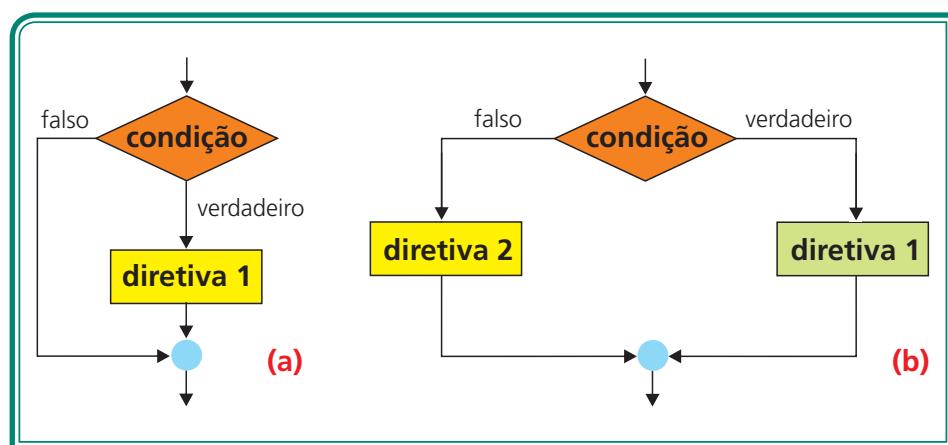


Figura 12: Fluxo das variações do comando if. Em (a) é tratada apenas a condição de verdadeiro, em (b) são tratadas as condições de verdadeiro e falso (Bertol, 2010).

Na Listagem 3 tem-se um exemplo prático da estrutura if utilizando somente o tratamento da condição verdadeira.

```
1. if (x < y) {  
2.     System.out.println("x é menor do que y");  
3. }
```

Listagem 3: Utilização da estrutura if simples

Neste exemplo, se o conteúdo da variável x for maior que o conteúdo da variável y, então a instrução System.out.println(" x é menor do que y"); será executada.

Já no exemplo da Listagem 4, verifica se o conteúdo de uma variável x é maior que o conteúdo de uma variável y e informar a condição.

```
1. if (x <= y)  {  
2.     System.out.println("x é menor ou igual a y");  
3. } else {  
4.     System.out.println("y é maior que x");  
5. }
```

Listagem 4: Utilização da estrutura if else

Neste exemplo, a instrução `System.out.println("x é menor do que y")` será executada se o resultado do teste lógico for verdadeiro; e `System.out.println("y é maior que x")` será realizada se o resultado do teste lógico for falso.

Um bloco é definido por {} e contém um conjunto de instruções. Quando um bloco é criado um escopo local é definido e permite a definição de variáveis locais. As variáveis definidas dentro de um bloco só podem ser vistas internamente a este, e são terminadas ou extintas no final da sua execução.

### Estrutura switch case

O switch case é **outra estrutura** de decisão e possui funcionalidade parecida com o comando if, a diferença é que o if pode tratar apenas duas situações. Para tratar mais situações deve-se utilizar vários if's. Já no switch case pode-se tratar várias situações com um único comando, conforme apresentado na Figura 13.

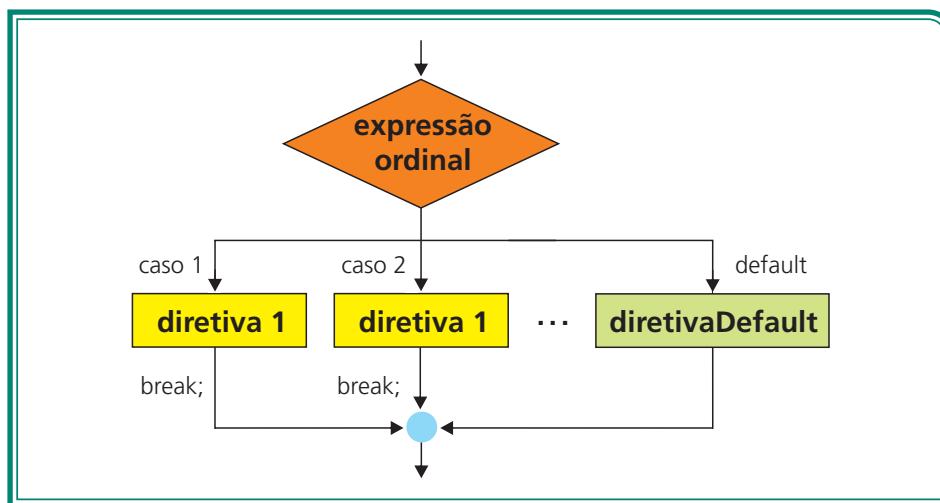


Figura 13: Fluxo do comando switch..case (Bertol, 2010).

A Listagem 5 apresenta um exemplo na linguagem de programação Java da estrutura switch case.

```
01. switch (opcao) {  
02.     case 'a':  
03.         case 'A': System.out.println("Alteração");  
04.             break;  
05.     case 'i':  
06.         case 'I': System.out.println("Inclusão");  
07.             break;  
08.     case 'e':  
09.         case 'E': System.out.println("Exclusão");  
10.             break;  
11.     default: System.out.println("Opção inválida ! ");  
12. }
```

**Listagem 5: Utilização da estrutura switch case**

Nesse exemplo, a variável opcao está valorizada com um caractere (char) digitado pelo usuário, então na linha 01 esse caractere é testado, se o mesmo for letra 'a' (minúscula ou maiúscula – linhas 02 e 03) é impresso na console a mensagem Alteração (linha 03) e a instrução é finalizada com o break (linha 04). Os testes continuam para a letra i ou e (linhas 05 a 07 e 08 a 10 respectivamente). Caso o caractere digitado for diferente desses, será executada a opção default (linha 11).

#### **2.3.2.4.2 Estrutura de repetição**

Uma estrutura de repetição é utilizada para repetir um determinado conjunto de instruções:

- um determinado número de vezes;
- até que determinada condição seja atendida, ou;
- enquanto determinada condição está sendo atendida.

A linguagem Java possui três estruturas de repetição: for (para), while (enquanto) e do...while (faça enquanto). De uma maneira geral, esses três tipos de estrutura de repetição podem ser utilizados indistintamente, porém há casos em que uma delas se aplica de maneira mais adequada que as outras.

#### **Estrutura for**

A estrutura do comando for pode ser traduzida por: para ... até ... faça. Essa

estrutura é apresentada no fluxo da Figura 14.

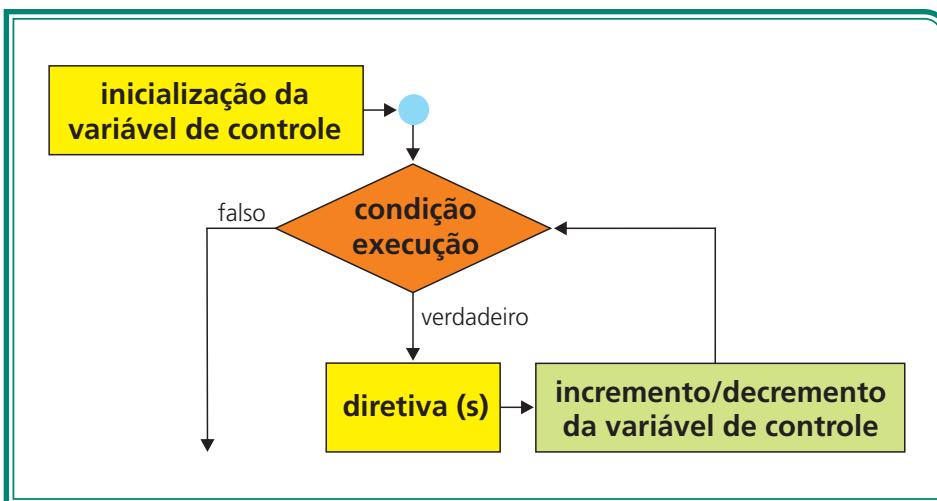


Figura 14: Fluxo da estrutura de repetição for (Bertol, 2010).

Na sequência, um exemplo de instruções na linguagem Java utilizando a estrutura for para imprimir a tabuada de um número (Listagem 6).

```
1.   for (int cont=0; cont<=10; cont++) {  
2.  
3.       resposta = cont * tabuadaNumero;  
4.       System.out.println( cont + " x " + tabuadaNumero + " = " +  
5.                           resposta);  
6.   }
```

Listagem 6: Exemplo da estrutura for

O conjunto de instruções das linhas 3 e 4 será repetido 10 vezes. A variável cont na linha 1 controla essas repetições. Essa variável inicia com 0 e a cada vez que o conjunto de instruções é executado essa variável é incrementada em 1, isso é feito em cont++. A variável tabuadaNúmero é recuperada do usuário antes do looping, sendo a tabuada desse número impressa na lógica.

A condição cont <= 10 é testada toda vez que o conjunto de instruções é realizado. Como a variável cont é incrementada, após a décima repetição essa variável terá o valor 11 e ao iniciar uma nova execução a condição é verificada. Ela resultará falso porque cont estará valendo 11, e assim as instruções do for não serão mais executadas, seguindo a lógica do programa.

## Estrutura while

A estrutura de repetição while pode ser traduzida como enquanto. Ela tem o fluxo conforme Figura 15.

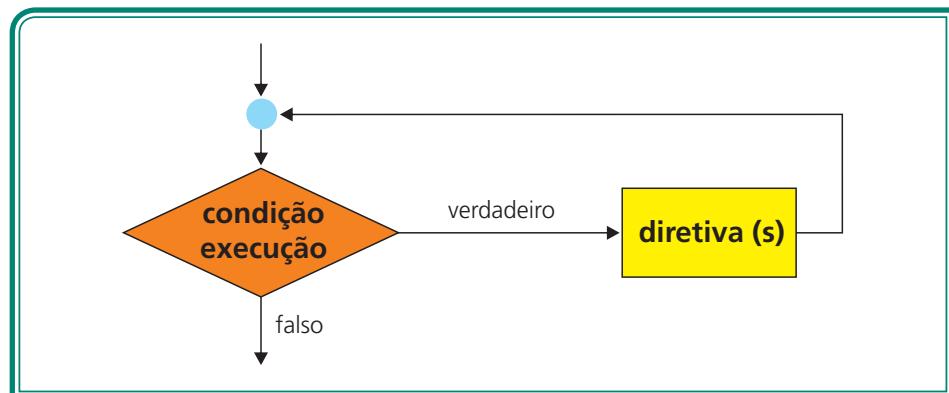


Figura 15: Fluxo da estrutura de repetição while (Bertol, 2010).

Exemplo de instruções na linguagem Java utilizando a estrutura while para imprimir os números pares entre 0 e 10 (Listagem 7).

```
1.   while (numero <= 10) {  
2.     if (numero % 2 == 0) {  
3.       System.out.println(numero);  
4.     }  
5.     numero = numero + 1;  
6.   }
```

Listagem 7: Exemplo da estrutura while

Nesse exemplo, o conjunto de instruções do while (linhas 2 a 7) é executado enquanto a condição de teste (`numero <=10`) é verdadeira. Isso ocorre enquanto a variável `numero` não chega ao valor 10. Cada vez que esse conjunto de instruções é executado a variável `numero` é acrescentada de 1, isso ocorre na linha 7 (`numero = numero + 1`). Esse processo se repete até chegar no valor 11, quando o teste `numero <= 10` tem resultado falso e a execução desse trecho do programa é finalizada.

## Estrutura do while

A estrutura de repetição do while repete um determinado conjunto de

instruções enquanto uma condição testada pelo while for verdadeira. A principal diferença entre o while e o do while é que o teste condicional no caso do while é feita antes de se executar o conjunto de instruções que definem o bloco. Assim, se a condição testada no while tiver resultado falso, o conjunto de instruções não será executado. Com o do while, o conjunto de instruções é executado pelo menos uma vez, pois o teste lógico é realizado no fim da estrutura, conforme Figura 16.

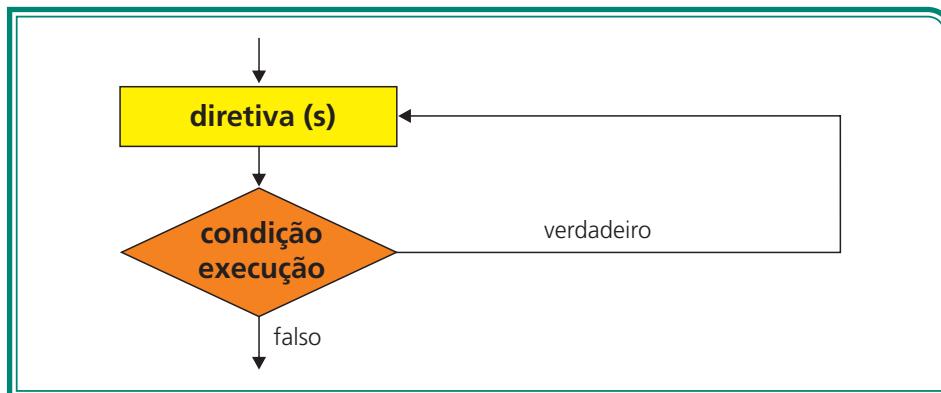


Figura 16: Fluxo da estrutura de repetição do..while (Bertol, 2010).

Exemplo de instruções na linguagem Java utilizando a estrutura do..while para imprime uma mensagem 10 vezes na tela (Listagem 8).

```
1.   do {  
2.     System.out.println("Contador: " + contador);  
3.     contador = contador + 1;  
4.   } while (contador <= 10);
```

Listagem 8: Exemplo da estrutura while

Esse conjunto de instruções de programa imprime na tela “Contador: 1” “Contador: 2” até “Contador: 10”. Leva-se em consideração que o contador é inicializado com 1 antes do looping.

Nesse exemplo após ser mostrada a mensagem na tela com o conteúdo da variável (linha 03), o contador é incrementado de 1, isso é feito na instrução “contador = contador + 1” da linha 04. Depois dessa instrução a condição do while “contador <= 10” é avaliada. Se o resultado for verdadeiro, isto é, se o valor do contador for menor ou igual a 10, as instruções das linhas 3 e 4 são executadas novamente, caso contrário a execução do programa continua a partir do looping.



## Resumindo

Mídias Integradas: No portal do curso é possível ter acesso a uma apostila detalhada, em formato digital, sobre os comandos da linguagem de programação Java.

Esta aula apresentou a IDE NetBeans para o desenvolvimento Java, sua instalação e o processo para desenvolver algoritmos. Também foram apresentados os conceitos da orientação a objetos e a estrutura básica da linguagem Java. Os itens fundamentais da linguagem Java são:

- a)** Variáveis que armazenam valores utilizados e manipulados pelos programas. As variáveis possuem um tipo que determina que tipo de informação pode-se armazenar.
- b)** As estrutura de decisão, que permitem executar ou não determinado conjunto de instruções. if é a estrutura de decisão básica que pode ser complementada com else; switch case é outra estrutura de decisão mais complexa.
- c)** As estruturas de repetição que permitem executar repetidamente. while, do while e for são as três estruturas de repetição da linguagem Java.

As estruturas de decisão e de repetição devem possuir uma condição de execução que é obtida em decorrência de um teste lógico, para determinar que instruções serão realizadas, a continuidade ou a finalização da repetição.



## Atividades de Aprendizagem

- 1)** A palavra reservada class é utilizada para definir uma classe na linguagem Java. Por que é necessário utilizar palavras reservadas, ou seja, para que elas servem? Indique mais três palavras reservadas da linguagem Java e explique o seu uso. converse com alguém que conhece alguma outra linguagem de programação, verifique se outras linguagens também possuem palavras reservadas e se há semelhanças com a linguagem Java.
- 2)** Escreva um programa com a linguagem Java para imprimir todos os números entre 0 e 100, utilizando para isso uma estrutura de repetição.
- 3)** Escreva um programa para imprimir a soma dos números entre 10 e 20. Aqui você também precisará de uma estrutura de repetição para “passar” pelos números entre 10 e 20.

**4)** Escreva um programa para imprimir os múltiplos de 3, entre 1 e 50. Lembre-se que um número é múltiplo de 3 se a sua divisão por 3 resultar 0, ou seja, se o resto da divisão desse número por três é zero ou divisão exata.

**5)** Imprimir o fatorial de um número.

O fatorial de um número n é  $n * n-1 * n-2 \dots$  até  $n = 1$ .

O fatorial de 0 ( $0!$ ) é = 1

O fatorial de 1 ( $1!$ ) é = 1

O fatorial de 3 ( $3!$ ) é  $3 * 2 * 1 = 6$

O fatorial de 4 ( $4!$ ) é  $4 * 3 * 2 * 1 = 24$

Observe que as multiplicações ( $4 * 3 * 2 * 1$ ) são realizadas por meio de uma estrutura (comando) de repetição.

**6)** Escreva um programa que leia um número e apresente a metade desse número se o mesmo é par e o seu dobro se ele é ímpar.

Dica: testar (if ) se o número é par ou ímpar. Uma forma de definir se é número é par é verificar o resto da divisão desse número por dois. Se o resto é zero o número é par.



# Aula 3 - Utilizando o Netbeans para o desenvolvimento visual de aplicações

## Objetivos

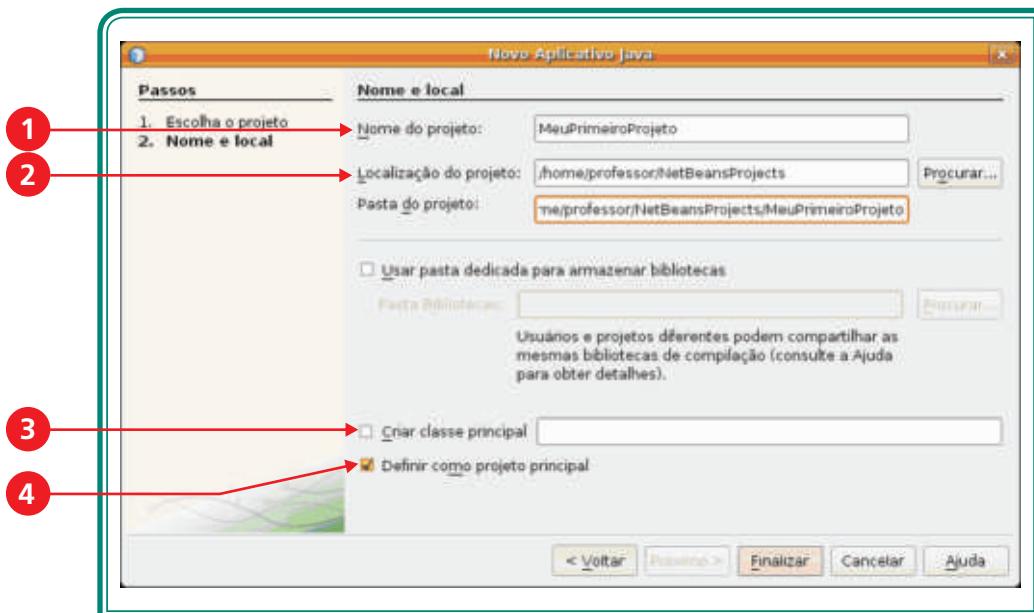
- Iniciar o desenvolvimento de um projeto visual;
- Realizar o tratamento do evento de clique em um botão;
- Abordar técnicas de debugação.

**A aula é composta pelos seguintes tópicos:**

- 3.1** Desenvolvendo o primeiro projeto
- 3.2** Desenvolvendo o primeiro formulário
- 3.3** Primeiro programa: calculadora
- 3.4** Mudando o layout da tela
- 3.5** Executando o aplicativo
- 3.6** Debugando projetos no NetBeans
  - 3.6.1** Usando Breakpoint
  - 3.6.2** Variáveis Locais
  - 3.6.3** Observadores

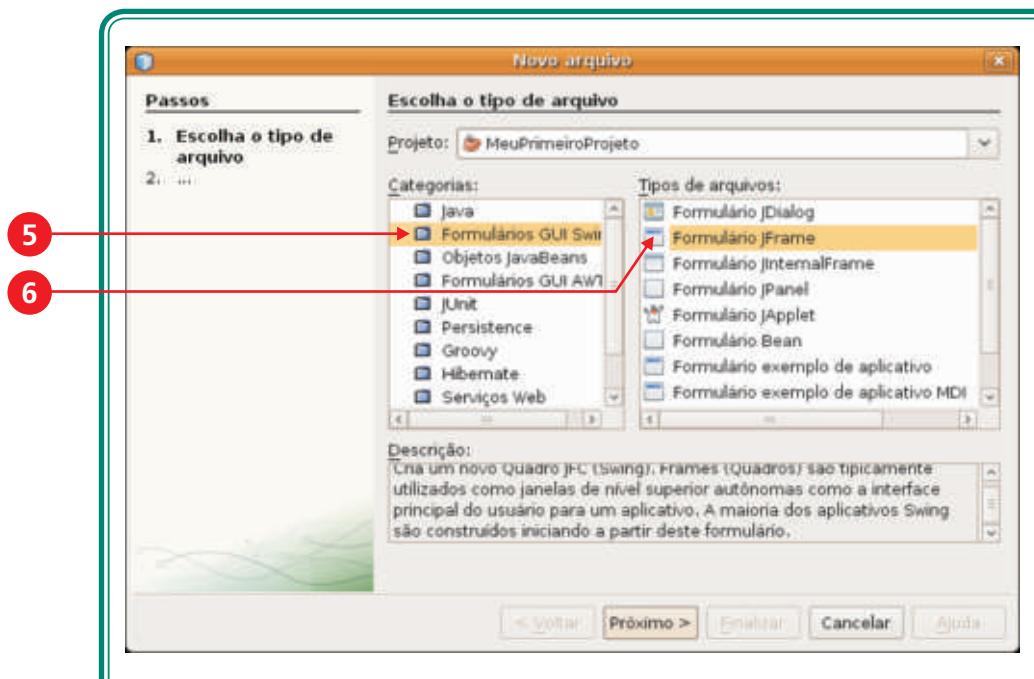
## **3.1 Desenvolvendo o primeiro projeto**

Para iniciar o desenvolvimento de um projeto que faz uso da interface visual, deve-se escolher a opção Arquivo-Novo Projeto... Na tela apresentada, deve ser escolhida a categoria Java e o projeto Aplicativo Java, conforme apresentado anteriormente na seção 2.1. Na sequência é apresentada a tela da Figura 17.



**Figura 17: Início do desenvolvimento de um novo projeto**

Nessa tela, deve ser informado o nome do projeto(1), o caminho onde o mesmo será salvo (2) e desliga-se a opção Criar classe principal (3), já que será desenvolvido um aplicativo *desktop* de forma visual. A opção Definir como projeto principal deve permanecer marcada (4), fazendo com que o projeto desenvolvido seja o projeto principal da IDE. Para finalizar, pressiona-se o botão Finalizar.



**Figura 18: Tela de escolha da classe desejada**

## 3.2 Desenvolvendo o primeiro Formulário

Após o desenvolvimento do projeto, deve-se criar a interface gráfica da aplicação, para isso seleciona-se o menu Arquivo – Novo Arquivo... Na tela seguinte escolhe-se o tipo de tela. Para o exemplo, seleciona-se Formulário GUI Swing (5) – Formulário JFrame (6) – Figura 18.

Na tela seguinte é informado o nome da classe que armazenará as informações da interface gráfica (7), bem como o pacote onde o mesmo será armazenado (8). Após finaliza-se a criação da interface gráfica clicando no botão Finalizar (9) – Figura 19.

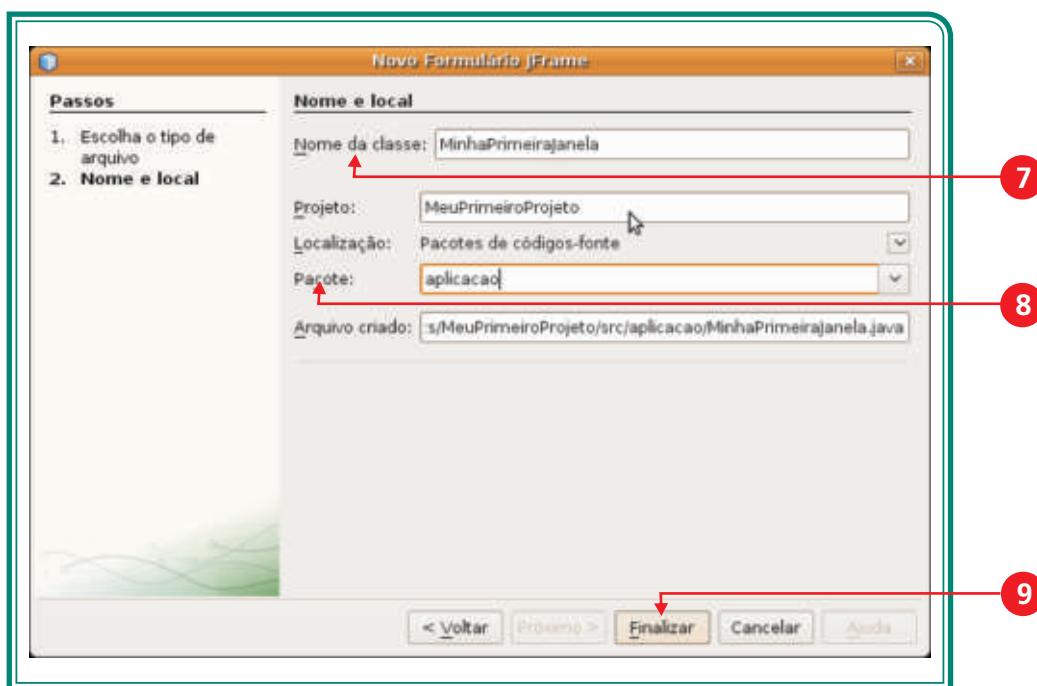


Figura 19: Tela de atribuição de nome e pacote para a interface gráfica

Ao selecionar o arquivo MinhaPrimeiraJanela.java, que se encontra no pacote aplicacao, a interface de desenvolvimento é apresentada conforme Figura 20.

Essa área é dividida em:

- (10) Tela do Aplicativo: Representação gráfica dos componentes que serão apresentados na MinhaPrimeiraJanela.java.
- (11) Paleta de Componentes Visuais: Os componentes exibidos podem ser clicados-arrastados para a tela do aplicativo. Dentre os componentes destacam-se os botões, os campos de textos e os rótulos.
- (12) Paleta de Propriedades: Ao selecionar um componente visual, é possível

modificar algumas características visuais dele (por exemplo sua cor, sua fonte) e não visuais (por exemplo nome do componente).

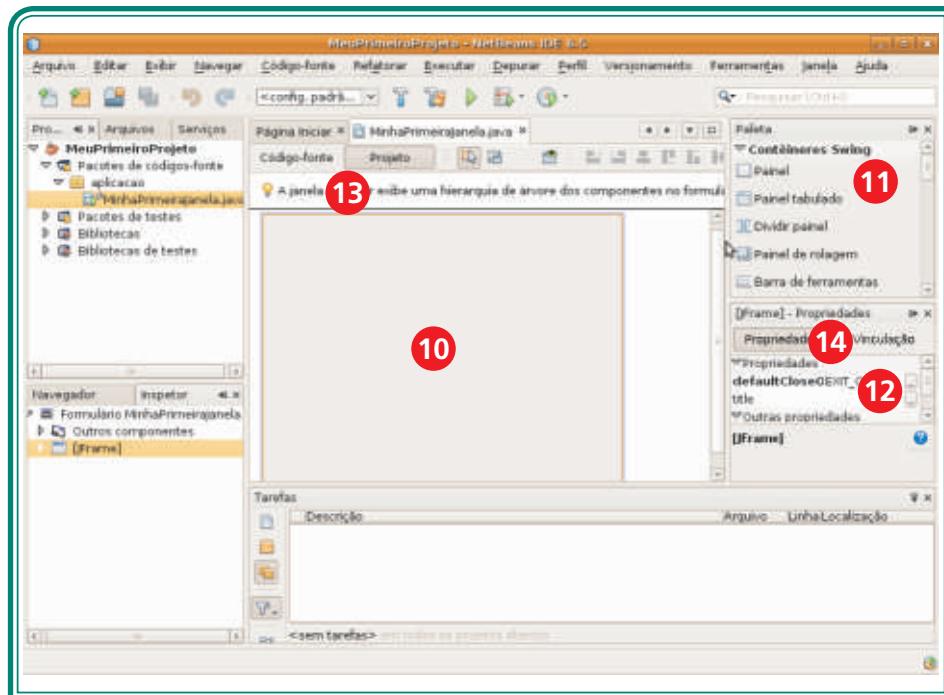


Figura 20: Área de desenvolvimento de aplicativos visuais

- (13) Alternância entre o modo Código-Fonte/Projeto: É possível através desses botões visualizar o código fonte da classe e voltar para o modo de desenvolvimento visual. Embora o NetBeans possua um poderoso ambiente de desenvolvimento visual, muitas vezes é necessário adicionar ou modificar o código fonte da classe.
- (14) Alternância entre Propriedades: É possível modificar a forma de ver as propriedades de um componente. Por exemplo, na guia Vinculação é possível ver a qual informação o componente está vinculado. Na guia Eventos é possível adicionar um código que será executado quando o usuário realizar um evento no componente visual (por exemplo, um clique).

### 3.3 Primeiro programa: calculadora

Para conhecer melhor o ambiente de desenvolvimento visual do NetBeans, será desenvolvido um programa didático: uma calculadora, que é composto por dois campos de textos onde serão digitados os dois números que serão somados, três componentes de rótulo, um para exibir a resposta da somatória e os outros dois para identificar os campos de textos, e um botão que será pressionado pelo usuário para realizar a soma. Os componentes visuais se

encontram na paleta Controle Swing e devem ser clicados/arrastados para a tela do projeto.

A interface visual do aplicativo é apresentada na Figura 21.

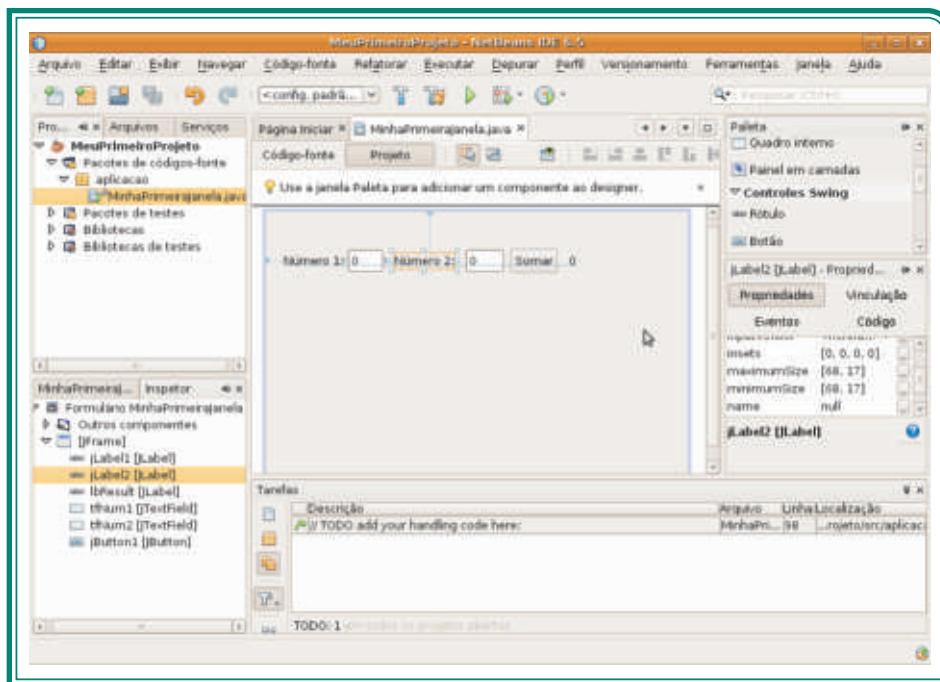


Figura 21: Interface gráfica do primeiro aplicativo - calculadora

Para os objetos, foram alteradas as seguintes propriedades:

jTextField1	jTextField2	jLabel1	jLabel2	jLabel3	jButton1
text=0	text=0	text: Número 1	text: Número 2	text: 0	Text: Somar
name=tfNum1	name=tfNum2			name: lbResult	name: btSomar

A propriedade *text* representa o texto inicial do componente que será exibido quando o aplicativo é executado. A propriedade *name* renomeia o componente visual e sempre que se fizer necessário recuperar as informações de um componente visual, deve se referenciar ao nome dado.

O nome de um componente também pode ser alterado clicando com o botão direito sobre o componente e selecionando a opção Alterar o nome da variável.

Para o aplicativo realizar a soma dos valores, deve-se selecionar o componente responsável pela soma (botão Somar). Na sequência clica-se no botão Eventos dentro da paleta de propriedades, escolhendo a opção actionPerformed (esse

evento representa o evento de clique sobre o botão). Após clica-se no botão de reticências (...) - Figura 22.

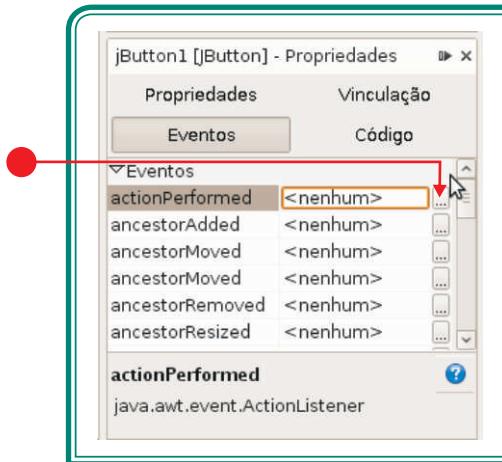


Figura 22: Área para associar um método ao clique de um botão

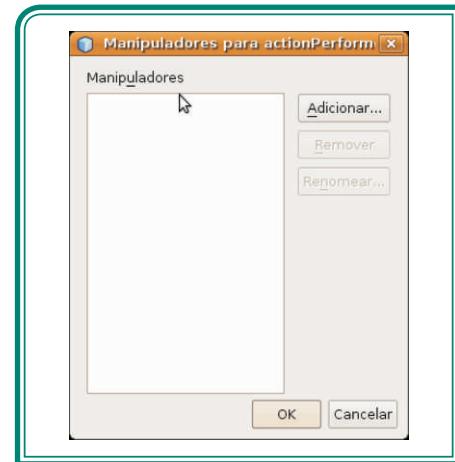


Figura 23: Área para associar um método ao clique de um botão

Na tela apresentada, deve-se informar qual método será executado ao clicar no botão. Para definir um método, clica-se em Adicionar... será exibida uma tela para informar o nome do método – Figura 23. Após informar o nome do método (no exemplo será somar), deve-se confirmar clicando nos botões OK das janelas abertas.

Assim será alternada para o modo código fonte, onde deve-se inserir a lógica para somar. O método somar é apresentado na Listagem 9.

```
1. private void somar(java.awt.event.ActionEvent evt) {  
2.     double num1 = Double.parseDouble( tfNum1.getText() );  
3.     double num2 = Double.parseDouble( tfNum1.getText() );  
4.  
5.     double result = num1 + num2;  
6.  
7.     lbResult.setText( String.valueOf( result ) );  
8. }
```

Listagem 9: Método somar: Recebe as informações de duas caixas de texto, somando e apresentando o resultado na tela

A Listagem 9, como a maioria dos programas/rotinas computacionais, pode ser dividida em três momentos: entrada, processamento e saída.

Na entrada (linhas 02 e 03), é recuperado o texto digitado nas caixas de texto

(tfNum1 e tfNum2) através do método `getString()`. Como esse método retorna um texto, o mesmo necessita ser convertido para ser armazenado na variável `num1`, que é do tipo `double`, por esse motivo a utilização do método `Double.parseDouble()`.

No processamento, as variáveis criadas anteriormente serão somadas, para isso foi definida uma variável `double` com o nome de `result` (linha 05), a qual recebe a soma das variáveis `num1 + num2`.

A saída apresenta o resultado do processamento para o usuário, por isso o conteúdo da variável `result` é exibido através do componente `lbResult`. O método `setText()` atribuí um valor para o componente visual, entretanto, o método espera uma `String` por parâmetro e a variável `result` é do tipo `double`, assim necessita-se fazer a conversão através do comando `String.valueOf()` – linha 07.

### 3.4 Mudando o layout da tela

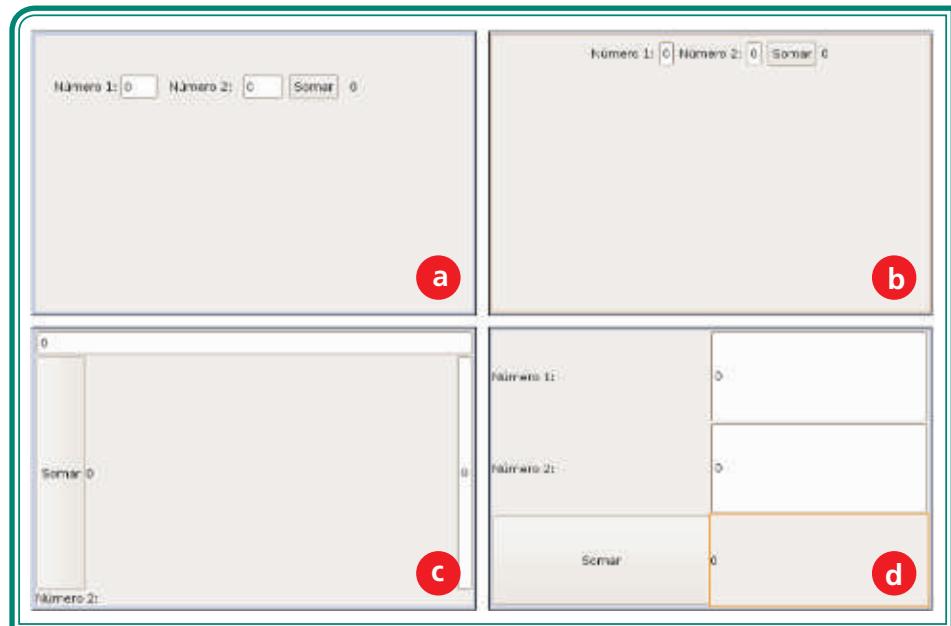
A linguagem de programação Java possui algumas classes para o gerenciamento de layout. Essas classes garantem a portabilidade do sistema desenvolvido em Java para outras plataformas e sistemas operacionais.

Ao se iniciar um projeto no NetBeans, o layout default é Desenho Livre. Esse layout utiliza posições relativas para os componentes. Utilizando esse layout, um componente possui vínculo com outros componentes da tela, dessa maneira a tela se reorganiza automaticamente ao adicionar um novo componente.

Para se mudar o layout basta clicar com o botão direito sobre a área do formulário ou no objeto formulário da janela Inspetor, selecionando na sequência a opção Definir Layout.

Outros layouts também merecem destaque em Java, entre eles o layout absoluto, que adiciona os componentes levando em considerações posições x e y, o layout de fluxo que permite adicionar um componente após o outro, o layout de borda, onde os componentes são adicionados respeitando cinco posições: Norte, Sul, Leste, Oeste e Centro da janela, e layout de grid, onde a tela é dividida em uma grade formada por linhas e colunas, sendo os componentes adicionados às intercessões (células).

A Figura 24 apresenta os layouts citados anteriormente.



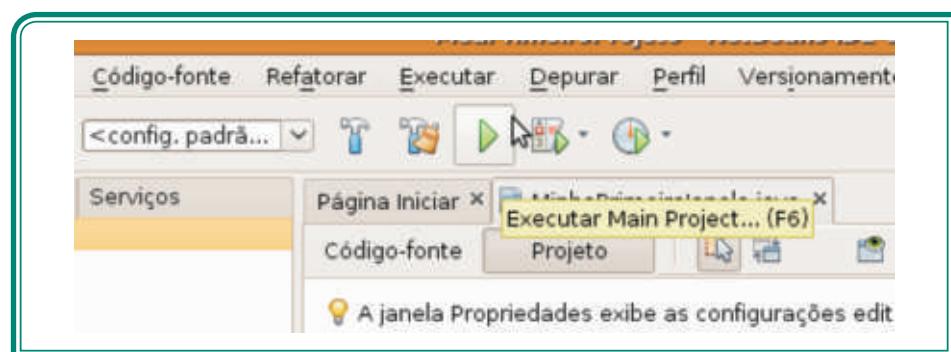
**Figura 24:** (a) layout absoluto, (b) layout de fluxo, (c) layout de borda e (d) layout de grid (três linhas, duas colunas).

### 3.5 Executando o aplicativo

Existem duas maneiras de executar um aplicativo desenvolvido no NetBeans. Se o mesmo não está definido como projeto principal, pode-se executar clicando com o botão direito sobre o projeto e selecionando a opção Executar...

Para definir um projeto como principal, deve-se clicar com o botão direito sobre o projeto e selecionar a opção Definir como Projeto Principal, ou ainda, ao iniciar o desenvolvimento de um novo projeto, deixar a opção “Definir como projeto principal” ligada. Essa opção é apresentada na tela da Figura 17.

Após o projeto está configurado como principal, basta clicar no botão Executar Main Project da barra de tarefas (Figura 25).



**Figura 25:** Barra de ferramentas com a opção de executar o projeto

Após a execução, o aplicativo é apresentado conforme a Figura 26.

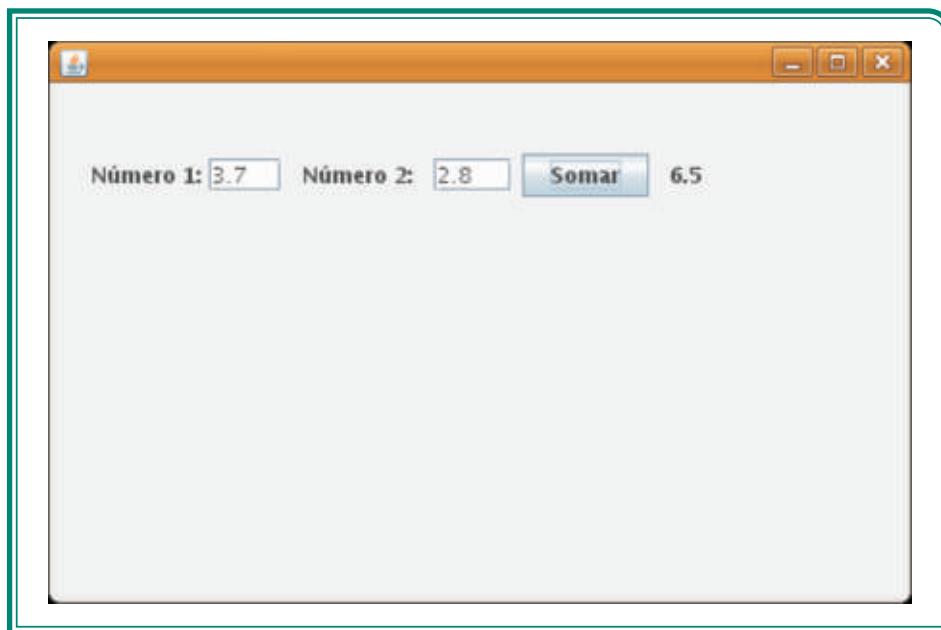


Figura 26: Tela do primeiro aplicativo (calculadora) em execução

### 3.6 Debugando projetos no NetBeans

Processo para debugação é muito útil no desenvolvimento de software. Muitas vezes cometem-se erros de programação que são de difícil percepção quando se analisa o código fonte desenvolvido isoladamente.

Na linguagem de programação Java, podemos citar como um exemplo de erro de programação a comparação do conteúdo de String utilizando o operador (==), conforme apresentado na Listagem 10.

```
1.  String name1 = "John";
2.  String name2 = "John";
3.
4.  if ( name1 == name2 ) {
5.      System.out.println( "A comparação é verdadeira" );
6. }
```

Listagem 10: Trecho de código Java que apresenta um erro de programação

No exemplo acima, o programa pode não executar a linha 05, uma vez que o operador == faz uma comparação de um objeto com outro e não do seu conteúdo. Como se tratam de objetos diferentes, os mesmos ocupam posi-

ções diferentes na memória, e com isso o teste lógico da linha 04 retornará falso.

Para fazer a comparação do conteúdo de duas Strings, deve-se utilizar o método equals, conforme apresentado na Listagem 11.

```
1.     if ( name1.equals(name2) ) {  
2.         System.out.println( "A comparação é verdadeira" );  
3.     }
```

**Listagem 11:** Correção para o código apresentado anteriormente

Entretanto, embora o erro acima seja um erro já conhecido de muitos programadores Java, o mesmo é cometido frequentemente, ou pela falta de conhecimento do programador ou ainda pela falta de atenção na codificação, sendo de qualquer maneira um erro de difícil percepção analisando apenas o código fonte.

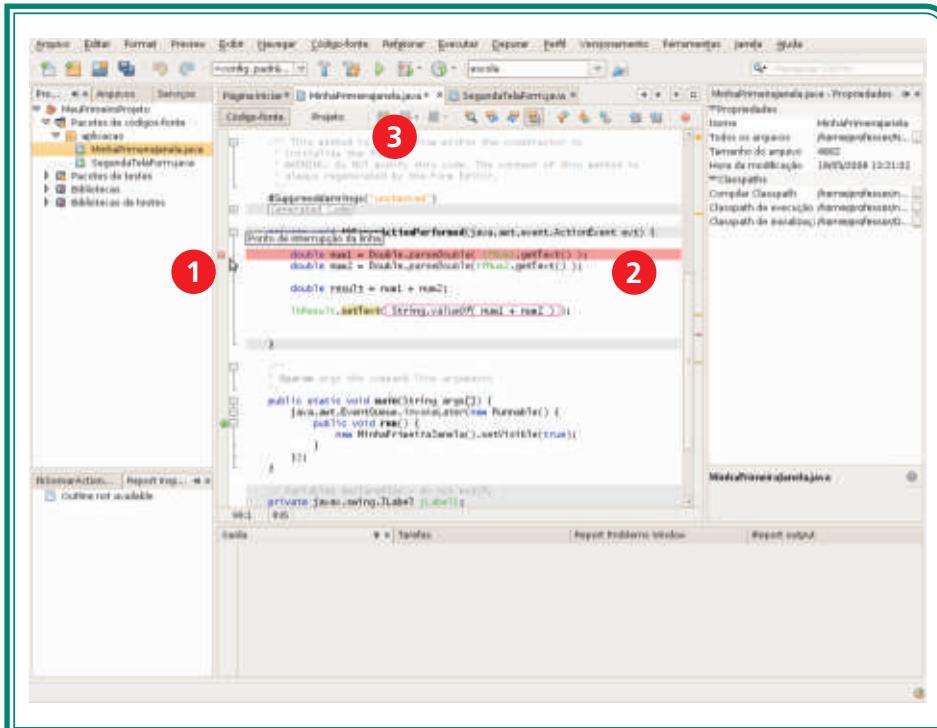
Uma solução para encontrar esses tipos de erros é utilizar recursos de debugação, destacando-se: Breakpoints, análise das variáveis locais e utilização de observadores.

### 3.6.1 Usando Breakpoint

Breakpoint tem como tradução literal “ponto de parada”. Utilizando esse recurso é possível executar o código passo a passo quando o ponto de parada for atingido. Dessa maneira consegue-se visualizar como o programa está se comportando.

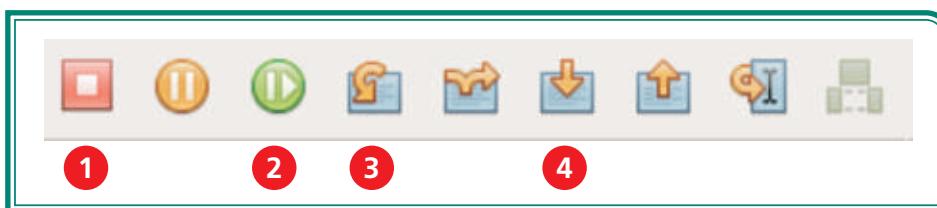
Para inserir um breakpoint, basta escolher a linha de parada no código e clicar sobre a barra cinza no lado esquerdo do editor de código, conforme apresentado pelo ponto (1) da Figura 27. Com isso será apresentada uma linha vermelha, indicando o ponto de parada (2), após definir um ou mais pontos de paradas, o projeto deve ser executado em modo de debugação, clicando no botão Debug Main Project (3) ou clicando com o botão direito sobre o projeto e escolhendo a opção Debug.

 **Atenção:** O Break-point deve ser inserido em uma linha de código executável, ou seja, não pode ser adicionado em uma linha de comentário ou uma linha de declaração de variável ou objeto.



**Figura 27:** Tela apresentando um breakpoint

Após a execução, o aplicativo é apresentado normalmente como se não estivesse sendo debugado. A única diferença está no fato de ser apresentada uma barra de tarefas nova na IDE NetBeans (Figura 28).



**Figura 28:** Barra de ferramentas para debug

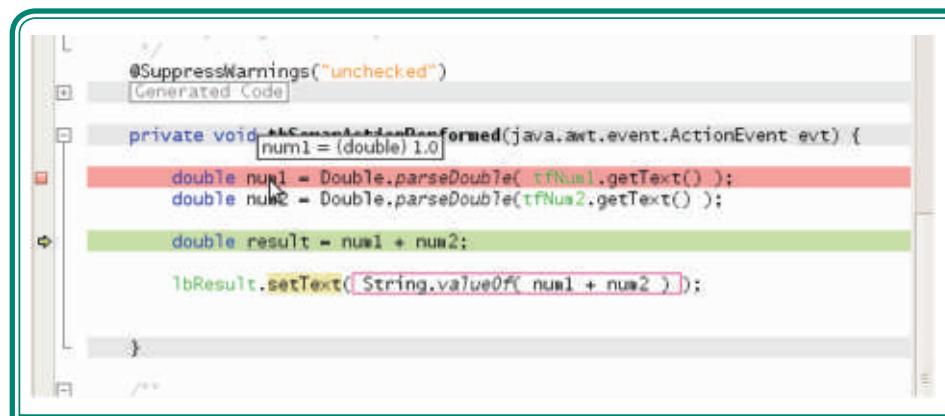
Nessa barra, os principais botões têm as funções de:

- (1) Finish Debug Session: Finaliza a execução passo a passo, fechando o aplicativo.
- (2) Execute: Executa o restardo do código até encontrar um novo breakpoint. Não encontrando breakpoints, o processamento continua normalmente.
- (3) Step Over: Comando utilizado para a execução passo a passo. Com essa opção o interior dos métodos não é debugado. A tecla de atalho para esse comando é o F8.

- (4) Step Into: Comando utilizado para a execução passo a passo. Com essa opção o interior dos métodos são debugados, ou seja, é executado passo a passo cada método chamado pelo programa.

No modo de Debug, o programa é executado normalmente até o processamento alcançar a linha do breakpoint. A partir desse ponto o processamento pára e aguarda a intervenção do programador. O mesmo pode avançar linha por linha de código com a tecla F8/F7 ou ainda clicar nos botões Step Over/Step Into. A linha atual de execução é representada pela cor verde.

No processo de debugação, é possível verificar o valor que uma variável assumiu levando o mouse sobre ela, conforme apresentada na Figura 29.



```

@SuppressWarnings("unchecked")
Generated Code

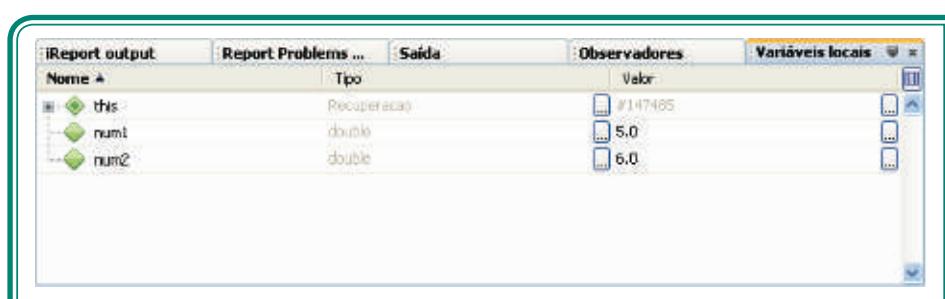
private void tfNum1ActionPerformed(java.awt.event.ActionEvent evt) {
    num1 = (double) 1.0;
    double num1 = Double.parseDouble(tfNum1.getText());
    double num2 = Double.parseDouble(tfNum2.getText());
    double result = num1 + num2;
    tbResult.setText(String.valueOf(num1 + num2));
}

```

Figura 29: Verificar o valor de uma variável levando o mouse sobre ela no modo debug

### 3.6.2 Variáveis Locais

Outra maneira de visualizar o valor das variáveis é analisando a janela Local Variable. Caso a mesma não seja apresentada no editor do NetBeans, deve-se acessar o menu Janela → Depurando → Variáveis Locais. A janela de inspeção das variáveis é apresentada conforme Figura 30.



Nome	Tipo	Observadores	Variáveis locais
this	Recuperado		
num1	double		5.0
num2	double		6.0

Figura 30: Janela das Variáveis Locais

Nessa janela é possível verificar as variáveis do escopo (nesse exemplo as variáveis declaradas no método somar) e também as variáveis de classe, estando essa dentro do objeto this. A janela Variáveis Locais possui as colunas: nome (nome do objeto), tipo (tipo do objeto) e valor (valor atual do objeto).

Com a janela Variáveis Locais, além de visualizar o conteúdo das variáveis, também é possível modificá-las, clicando nas reticências (...) ao lado do campo valor.

### 3.6.3 Observadores

Na janela Watches, que pode ser obtida através do menu Janela → Depurado → Observadores, é possível visualizar o conteúdo de uma variável ou de uma expressão ao longo do processamento.

Para isso, no modo debug deve-se clicar com o botão direito sobre a janela Observadores, escolhendo a opção Novo Observador... - Figura 31.

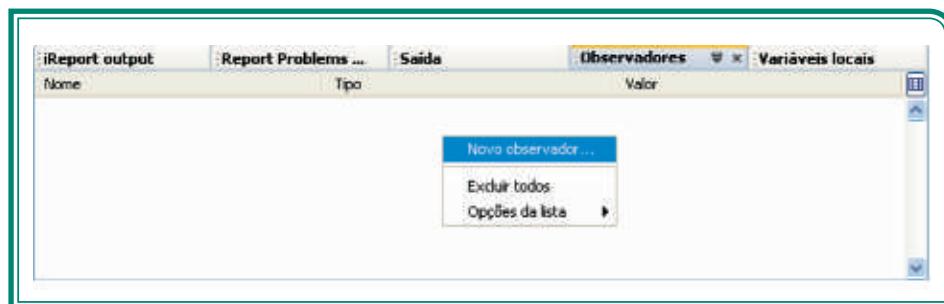


Figura 31: Janela para inclusão de Observadores

Na tela seguinte deve-se informar o nome da variável que se deseja observar ou uma expressão. Como exemplo foram adicionados dois Observadores, um para visualizar o conteúdo da variável num1 e outro para recuperar o conteúdo da constante Math.PI. O resultado pode ser visualizado na Figura 32.

Nome	Tipo	Valor
num1	double	5.0
Math.PI	double	3.141592653589793

Figura 32: Conteúdo de dois Observadores adicionados no programa

## Resumindo

Esta aula apresentou o desenvolvimento visual de aplicativos Java utilizando a IDE NetBeans.

Como exemplo foi desenvolvido um programa de calculadora, o qual possuía os três tipos de componentes mais utilizados na programação visual: campo de texto, rótulo e botão.

Também foi apresentado nessa aula o tratamento de eventos (clicar do botão), e algumas técnicas de debugação, como breakpoint, variáveis locais e observadores.



## Atividades de Aprendizagem

- 1)** Implementar as operações de subtração, multiplicação e divisão no programa desenvolvido anteriormente.
- 2)** Desenvolver um programa visual para verificar se um número é primo. Um número é primo se ele é divisível (divisão exata ou que não tem resto) somente por um e ele mesmo. O programa deve possuir um componente Campo de Texto para digitação do número, um componente rótulo para indicar se o número é primo e um botão de verificar, o qual executará a lógica do programa.
- 3)** Desenvolver um programa que receba via interface visual (componente campo de texto) um número, e apresente seu fatorial em um componente rótulo (deve ser utilizado um botão para realizar o processamento).
- 4)** Escreva um programa que receba um número em um campo de texto e apresente a metade desse número se o mesmo é par e o seu dobro se ele é ímpar em um componente rótulo. Utilizar um componente botão para realizar o processamento.

# Aula 4 - Componentes visuais no NetBeans

## Objetivos

- Apresentar os principais componentes visuais utilizados em aplicações comerciais;
- Mostrar alguns códigos que fazem uso de recursos dos componentes visuais.

**A aula é composta pelos seguintes tópicos:**

**4.1** Componentes visuais no NetBeans

**4.2** Paleta de componentes

**4.2.1** Componentes da paleta Controle Swing mais utilizados

**4.2.2** Outros componentes visuais do controle swing

**4.2.3** Utilizando outras janelas

**4.2.4** Utilizando menus

## 4.1 Componentes visuais no NetBeans

O NetBeans 6.5 é um ambiente completo para desenvolvimento de aplicativos comerciais. Como foi visto no capítulo anterior, é possível desenvolver com NetBeans aplicações desktop de forma rápida, apenas clicando e arrastando componentes.

Quando se inicia um projeto no NetBeans, existem algumas opções de interfaces visuais. Para visualizá-las acesse o menu Arquivo – Novo Arquivo...

Na caixa de diálogo apresentada – Figura 26, existem duas categorias para o desenvolvimento de interfaces visuais utilizando o NetBeans, são elas:

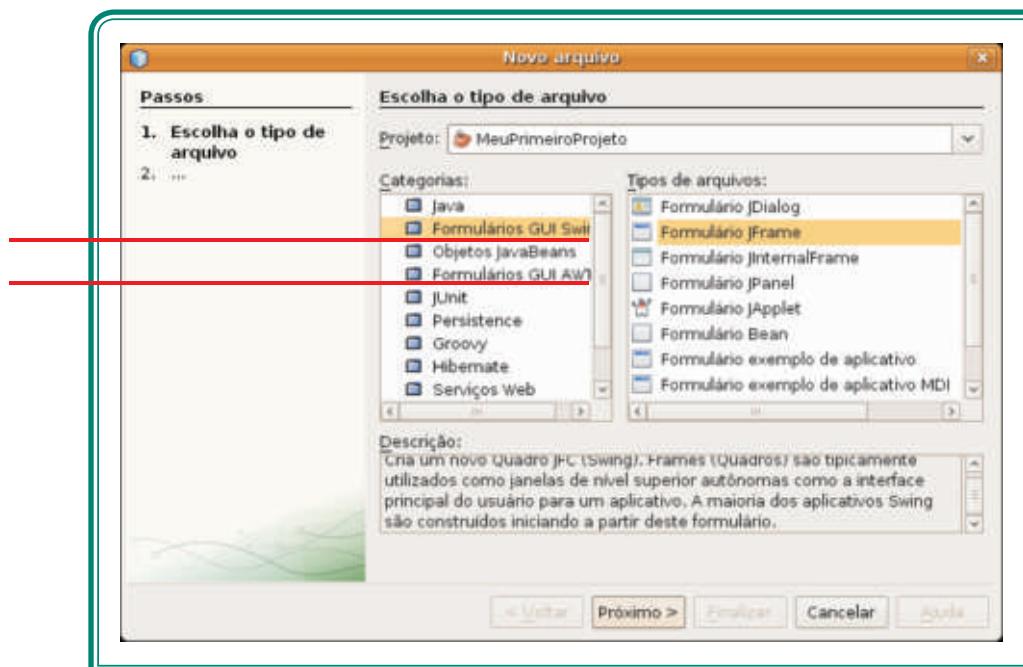


Figura 33: Tela com os tipos de aplicações visuais disponíveis no Netbeans

- a)** Formulário GUI AWT: Permite desenvolver formulário desktop utilizando a biblioteca de componentes AWT. Essa foi a primeira biblioteca de componentes Java para do desenvolvimento desktop, possuindo apenas alguns componentes básicos.
- b)** Formulário GUI Swing: Permite desenvolver formulário desktop utilizando a biblioteca de componentes swing. Essa biblioteca é uma evolução da biblioteca AWT, sendo disponibilizados nela novos componentes e novos recursos.

Para o desenvolvimento serão utilizadas apenas as opções da categoria *swing*, sendo que essa biblioteca possui uma quantidade maior de componentes se comparado com a AWT. Ao selecionar a categoria Formulário GUI Swing, são apresentadas algumas opções de tipos de arquivos (lado direito). Destacam-se:

- a)** Formulário JDialog: Permite desenvolver interfaces visuais simples, como caixas de diálogo de confirmação (ex: “Deseja Fechar o aplicativo?”) e de entrada de dados (ex: “Digite sua idade na caixa de texto abaixo.”).
- b)** Formulário JFrame: São os formulários utilizados pelas aplicações desktop. Dentro de um JFrame é possível adicionar componentes

visuais, como botões, campos de textos, rótulos, etc. É a interface visual mais utilizada em aplicações desktop.

- c) Formulário JApplet: Semelhante ao JFrame, a diferença está no fato do JApplet ser utilizado em aplicações web (os applet são aplicativos executados dentro dos browsers web).
- d) Formulários de Exemplos (de aplicativo, de aplicativo MDI, etc.): os formulários de exemplos permitem desenvolver aplicativos desktop com a utilização de um wizard. Dessa maneira, são apresentadas telas solicitando informações para o desenvolvimento das interfaces visuais, sendo que ao final a tela pronta é adicionada ao projeto. A maior vantagem de sua utilização é a velocidade de desenvolvimento, a desvantagem é a falta de flexibilidade, já que muitas vezes é difícil modificar o conteúdo gerado pelo wizard.

## 4.2 Paleta de componentes

No desenvolvimento de qualquer um dos tipos de interfaces apresentadas anteriormente (na maioria das vezes é utilizado o JFrame), é apresentada uma paleta de componentes visuais subdividida em categorias, esta apresentada na Figura 34.

As categorias são:

- a) Conteineres Swing: São componentes que possuem a função de receber outros componentes visuais. Alguns exemplos são: Barra de ferramenta, Painel tabulado e Quadro Interno.
- b) Controles Swing: Nessa paleta são apresentados os componentes visuais que podem ser adicionados no interior de um conteiner ou formulário. Alguns exemplos são: Botões, Campos de Texto, Rótulos, etc.

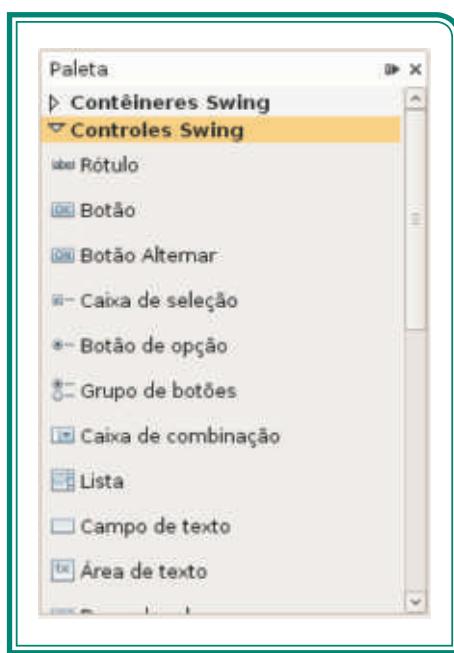


Figura 34: Paleta de componentes visuais no Netbeans

- c) Menus Swing: Nessa paleta estão disponíveis os componentes para adicionar menus a um formulário. Como exemplo, podemos citar os menus existentes na parte superior de alguns formulários e os menus popups.
- d) Janelas Swing: Nessa paleta são apresentados os componentes de nova janela. Dentre eles, destacam-se as telas para escolha de cores, de arquivos e os formulários (Frames).
- e) AWT: Possui os componentes visuais da biblioteca AWT. Muitos dos componentes são parecidos com os da paleta Controles Swing, como botões e campos de textos, entretanto os componentes da paleta AWT são mais simples.
- f) Bean: Essa paleta possui um único componente Escolher Bean, utilizado para instanciar um objeto a partir de uma classe.
- g) Java Persistence: Possui componentes que provêm acesso a base de dados.

## 4.2.1 Componentes da paleta Controle Swing mais utilizados

A seguir serão apresentados os componentes visuais mais utilizadas em uma aplicação *desktop*. Será apresentado o componente, sua utilização, como o mesmo pode ser adicionado à tela, as principais características e um exemplo de código que utiliza o componente.

### 4.2.1.1 Campo de Texto

Este é um dos principais componentes da programação *desktop*, possui a função de recuperar um texto digitado pelo usuário no sistema. Esse componente é apresentado na Figura 35.



Figura 35: Componente Campo de Texto

Para utilização, basta clicar e arrastar o componente Campo de Texto da paleta Controles Swing para o formulário ou conteiner onde se deseja utilizar o componente.

Dentre as propriedades do componente, destacam-se:

- a)** Background: essa propriedade permite modificar a cor do componente.
- b)** Editable: permite habilitar ou não a edição do componente visual.
- c)** Font: É possível através dessa propriedade escolher o tipo da letra (tamanho, estilo, etc) utilizado pelo componente.

Na Listagem 12 é possível verificar um código que fazem uso do conteúdo de um campo de texto cujo o nome é tfTexto.

```
1. String texto = tfTexto.getText();
2. 3. if ( texto.equalsIgnoreCase( "vermelho" ) ) {
3. tfTexto.setBackground( Color.RED );
4. }else if (texto.equalsIgnoreCase( "azul" ) ) {
5. tfTexto.setBackground( Color.BLUE );
6. }
7. }
```

**Listagem 12:** Código responsável por recuperar o texto digitado pelo usuário e dependendo do valor digitado faz a alteração da cor do componente tfTexto

Na linha 01 é recuperado o texto digitado no Campo de Texto cujo nome é tfTexto, sendo seu conteúdo recuperado pelo método getText().

Após recuperar a *String*, a mesma é comparada com a palavra “vermelho” – linha 03. Para comparação foi utilizado o método equalsIgnoreCase() que faz a comparação de textos sem levar em consideração letras maiúsculas ou minúsculas.

Na sequência, caso o texto digitado for diferente de vermelho é realizada uma nova comparação, agora comparando a *String* texto com a palavra “azul” – linha 05. Se o resultado dessa comparação for verdadeiro, é alterado a cor do componente tfTexto através do método setBackground(), o qual espera por parâmetro uma cor.

**Atenção:** O código anterior pode ser adicionado no evento de clique de um botão, ou em algum dos eventos disponíveis na categoria eventos da paleta de propriedades.



#### 4.2.1.2 Rótulo

Um componente rótulo é utilizado para exibir uma informação textual para o usuário. O conteúdo desse componente pode ser alterado pelo programa, porém não pelo usuário.

Muitas vezes esses componentes são utilizados como um texto estático para informar que tipo de informação um componente de entrada espera, por exemplo um campo de texto. Na Figura 36 tem-se um exemplo de utilização de rótulo.

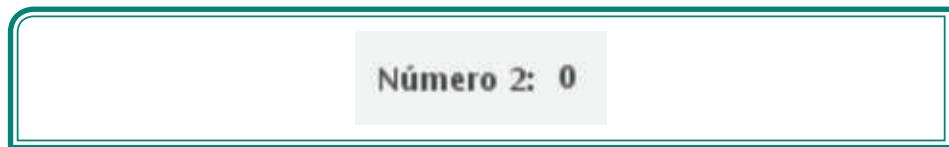


Figura 36: Componente do rótulo

Para adicionar um componente Rótulo no formulário, deve-se clicar e arrastar o componente da paleta Controles Swing até um formulário ou conteiner.

Dentre as propriedades do componente, destacam-se:

- a) Text: É o texto inicial do componente, a qualquer momento essa propriedade pode ser alterada pelo programa.
- b) Border: Essa propriedade permite modificar o tipo da borda do componente.
- c) Foreground: Permite modificar a cor da letra do componente Rótulo.

Na Listagem 13 é apresentada um exemplo de utilização do componente rótulo. Nesse exemplo é recuperado o conteúdo de um rótulo, é aplicada uma lógica para contar o número de vogais e o resultado é apresentado ao usuário em um novo rótulo.

```
1.  String nomeCompleto = lbNomeCompleto.getText();
2.  int contaVogal = 0;
3.  for( int i = 0; i < nome.length(); i++ ) {
4.      switch( nome.charAt(i) ) {
5.          case 'a':
6.          case 'e':
7.          case 'i':
8.          case 'o':
9.          case 'u':
10.             contaVogal++;
11.             break;
12.         }
13.     }
14.   }
15.   lbResult.setText( "Vogais: " + String.valueOf(contaVogal) );
```

Listagem 13: Código responsável por recuperar o conteúdo de um rótulo, contar as vogais e apresentar essa informação para o usuário

Na linha 01 é recuperado o texto existente no rótulo `lbNomeCompleto`. O conteúdo é recuperado pelo método `getText()`.

Na linha seguinte (linha 3) é declarada uma variável inteira que somará a quantidade de vogais existente na `String nomeCompleto`. Na sequência (linha 04) é realizado um *looping* com o comando `for`, onde a variável `i` varia de 0 até a quantidade de caracteres existente na `String nomeCompleto`.

A cada iteração do *looping* é comparado o caractere do índice `i` (caractere recuperado pelo método `charAt()`) com as vogais minúsculas (da linha 06 a 10), caso o caractere for uma vogal entrará no condicional e executará o comando da linha 11, onde é acrescido um na quantidade de vogais.

Ao final do *looping*, a variável `contaVogal` é concatenada com a `String "Vogais:"` e adicionado no rótulo `lbResult` através do método `setLabel()`.

#### 4.2.1.3 Botão

O componente botão é um componente muito utilizado no desenvolvimento de interfaces com o usuário. Na maioria das vezes é através de um botão que o usuário confirma uma ação e executa algum comando dentro do aplicativo.

Esse componente possui um texto que indica ao usuário sua função. Na Figura37 tem-se um exemplo de utilização de botão.



**Figura 37: Componente botão**

Assim como os componentes anteriores, para adicionar um botão na janela deve-se clicar e arrastar o componente da paleta Controles Swing até um formulário ou conteiner.

Dentre as propriedades do componente botão, destacam-se:

- a)** icon: É possível escolher uma representação gráfica para o botão. Para escolher um ícone, é necessário que a imagem esteja armazenada na pasta do projeto.
- b)** text: Um texto informativo para o botão, é através dele que o usuário saberá a função do botão.

- c) enabled: Permite habilitar ou desabilitar o botão.

Apesar de, na maioria das vezes, serem utilizados para executar códigos, os botões também podem ser referenciados dentro do código fonte. Na Listagem 14 tem-se um exemplo de captura e mudança das informações do componente Botão.

```
1.     if ( tfTexto.getText().equals( "" ) ){  
2.         btConfirmar.setEnabled( false );  
3.     }else {  
4.         btConfirmar.setEnabled( true );  
5.     }
```

**Listagem 14:** Código responsável por verificar o conteúdo de um campo de texto chamado `tfTexto`, se a mesma estiver vazia o botão `btConfirmar` fica desabilitado, caso contrário, o botão é habilitado.

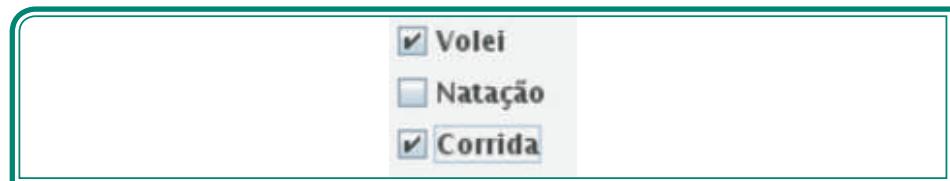
Na primeira linha é realizado um condicional para saber se o componente `tfText` está vazio (para isso é recuperado o conteúdo do `tfText` utilizando o método `getText()` e comparado com `""` através do método `equals()`). Caso positivo, o botão é desabilitado através do método `setEnabled()`, o qual recebe por parâmetro `false`, caso contrário o botão é habilitado.

## 4.2.2 Outros componentes visuais do controles swing

Na sequência são apresentados sucintamente outros componentes visuais da categoria Controles Swing.

### 4.2.2.1 Caixa de Seleção

Esse componente fornece ao aplicativo um mecanismo simples de entrada, onde deve-se ativar ou desativar opções – Figura 38.



**Figura 38:** Tela com três componentes de caixa de seleção

A caixa de seleção é formada por uma caixa de verificação, que pode iniciar ativada ou não, dependendo do valor da propriedade `selected`, e um rótulo que pode ser modificado na propriedade `text`.

Para saber via código fonte se uma caixa de seleção foi selecionada, deve-se utilizar o método `isSelected()`, o qual pode retornar `true` ou `false`.

A Listagem 15 verifica se uma caixa de seleção está selecionada ou não, caso positivo é apresentada uma mensagem na console.

```
1. 01. if ( cbVolei.isSelected() ) {  
2.   02.   System.out.println( "Volei selecionado" );  
3. 03. }
```

Listagem 15: Código para verificar se uma caixa de seleção está marcada

#### 4.2.2.2 Caixa de Combinação

A caixa de combinação fornece um tipo de entrada simples, onde o usuário pode selecionar um dos itens de uma lista. O componente caixa de combinação é apresentado na Figura 39.



Figura 39: Componente caixa de combinação

Ao clicar em uma Caixa de Combinação, é exibida uma lista de itens que foram adicionados a ela. Observa-se que os itens adicionados são objetos `String` e são atribuídos na propriedade `model` do componente.

O elemento inicial pode ser alterado a partir da propriedade `selectedIndex`, sendo que o código zero representa o primeiro elemento da caixa.

No código fonte, para recuperar um elemento da caixa de combinação é utilizado o método `getSelectedItem()`, porém esse método retorna uma informação do tipo `Object` e por esse motivo deve ser feito a conversão (`Cast`).

O exemplo da Listagem 16 recupera o conteúdo de uma caixa de combinação e adiciona em uma variável `String`.

```
1.     String nome = (String) combo.getSelectedItem();
```

Listagem 16: Código para recuperar o conteúdo de uma caixa de combinação

#### 4.2.2.3 Área de Texto

A área de texto é um componente de entrada que permite a digitação de várias linhas. Algumas das propriedades desse componente são iguais ao do componente campo de texto, entretanto algumas características são exclusivas desse componente, como a exibição de barras de rolagens horizontal e vertical quando a quantidade de texto for maior do que o espaço disponível do componente.

Uma imagem do componente área de texto é apresentada na Figura 40.

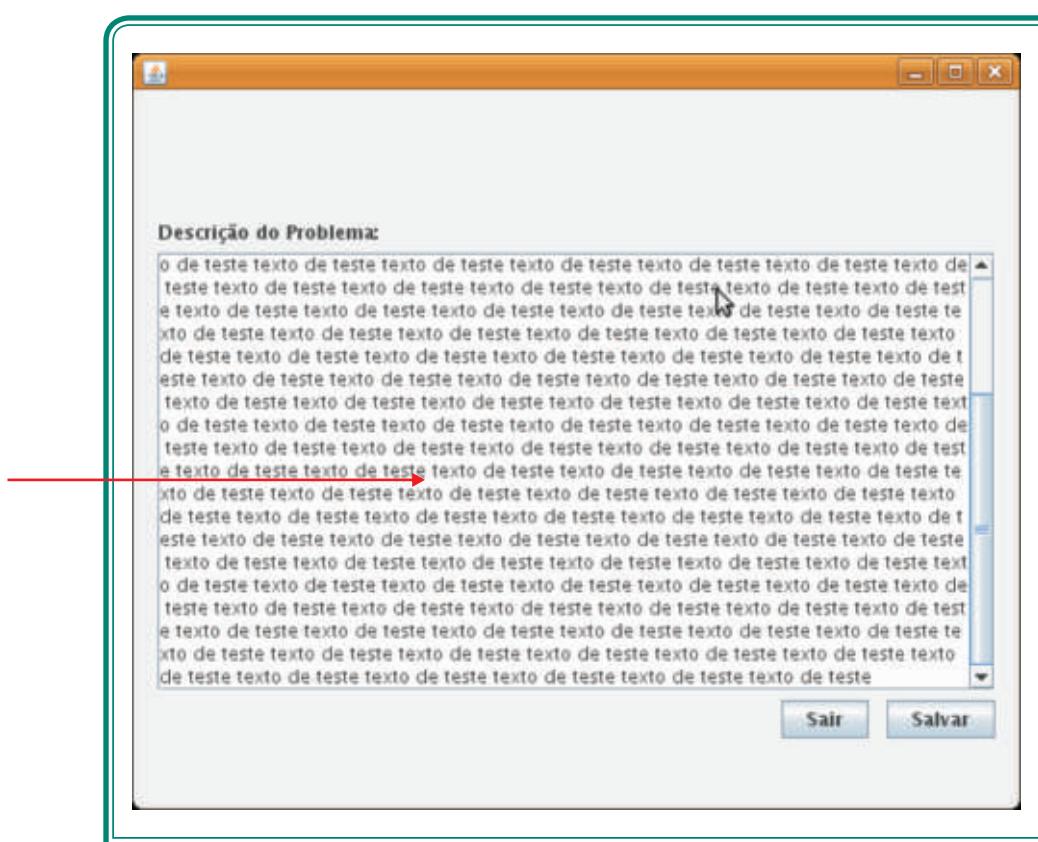


Figura 40: Componente área de texto

Dentre as propriedades específicas desse componente estão a propriedade `text`, utilizada para informar o texto inicial do componente. A propriedade `autoscrolls` que permite ligar/desligar as barras de rolagens e a `lineWrap` que informa se haverá quebra de linha ou não.

Para adicionar uma nova linha no componente área de texto é utilizado o comando `append()`, entretanto o mesmo não faz quebra de linha, sendo necessário adicionar um “`\n`” para realizar a quebra, conforme apresentado na Listagem 17.

```
1. memo.append( resposta.getText() + "\n" );
```

#### Listagem 17: Código para adicionar um texto no componente área de texto

Outro método importante da área de texto é o `getText()`, que permite recuperar o texto do componente.

#### 4.2.2.4 Lista

Com o objeto lista pode-se apresentar uma lista de linhas de texto. Esse objeto pode ser percorrido e suporta os modos de seleção única e múltipla. A Figura 41 apresenta um exemplo de Lista.

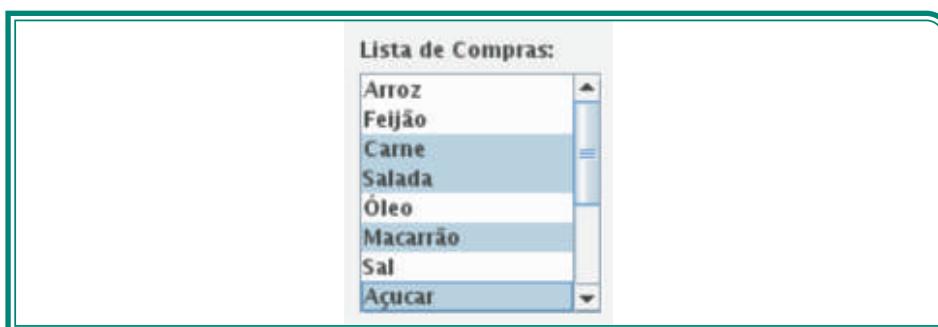


Figura 41: Componente lista

Para informar os elementos iniciais de uma Lista é modificada a propriedade `model` da mesma. Outra propriedade interessante é a `selectionMode`, o qual permite modificar entre os modos de seleção `Multiple` (selecionar mais de um item com a tecla `CTRL`) ou `Single` (apenas uma linha de cada vez).

Quando se escolhe o modo `Single`, a linha selecionada pode ser obtida através do comando `getSelectedValue()`, o qual retorna um `Object`. Na Listagem 18 um exemplo de código para recuperar um item da Lista.

```
1. String conteudo = (String) list.getSelectedValue();
```

#### Listagem 18: Código para recuperar um conteúdo da lista

Ao se trabalhar com linhas múltiplas, deve-se utilizar o método `getSe-`

lectedValues() para recuperar os itens selecionados. Como o método retorna um *array* de *Object*, os valores devem ser percorridos individualmente. Na Listagem 19 um exemplo de sua utilização:

```
1. Object elementos[] = lsAlimento.getSelectedValues();
2.
3. String textoCompleto = "";
4.
5. for( int i = 0; i < elementos.length; i++ ) {
6.     textoCompleto += (String) elementos[i] + " ";
7. }
```

Listagem 19: Código para recuperar os itens selecionados de uma lista múltipla

No código acima, a linha 1 recupera um *array* com os elementos selecionados da lista, na linha 3 tem-se a declaração de um objeto String que irá receber o conteúdo de todos os elementos selecionados. Na linha 05 é realizado um *looping* para percorrer todos os elementos da Lista, cada elemento recuperado é adicionada a String textoCompleto.

#### 4.2.2.5 Tabela

O componente tabela é muito útil na programação *desktop*, principalmente quando se trabalha com dados vindos de um banco de dados – Figura 42.

Nome	Endereço	Cidade	Estado
João da Silva	Rua América, 223	Pato Branco	PR
Maria de Souza	Rua das Flores	Cuiabá	MT
Pedro Correia	Av. Brasil, s/n	Palmas	

Figura 42: Componente tabela

Esse componente é formado por células, estas originadas pelas intercessões entre as linhas e colunas da tabela. Para atribuir um conteúdo inicial na tabela, deve-se modificar sua propriedade model. Nessa propriedade é possível modificar os títulos de colunas (Guia Configurações da Tabela) e o conteúdo da tabela (Valores Padrão).

Para recuperar o conteúdo de uma posição específica, é utilizado o comando da Listagem 20.

```
1. String conteudo = (String) tbCidades.getModel().getValueAt( 0, 0 );
```

**Listagem 20:** Código para recuperar o conteúdo de uma posição específica da tabela

Esse comando retorna o conteúdo da primeira célula, localizado no canto superior esquerdo. O conteúdo retornado pelo comando é um Object, por esse motivo a necessidade da conversão para String.

### 4.2.3 Utilizando outras Janelas

#### 4.2.3.1 JFrame

No desenvolvimento de aplicativos comerciais é comum a utilização de várias telas para apresentar diferentes informações para o usuário. Para adicionar uma nova tela (JFrame) o aconselhável é desenvolver uma nova classe, sendo esta tratada como um novo arquivo.

Para adicionar uma nova tela, deve-se clicar com o botão direito sobre o projeto e escolher a opção Novo... ou ainda pode-se acessar o menu Arquivo e escolher a opção Novo Arquivo. O novo formulário deve ser do tipo Formulário JFrame, encontrado na categoria Formulários GUI Swing.

O processo de criação de um JForm é idêntico ao apresentado na seção 3.2. Para esse exemplo, vamos dar o nome da classe de SegundaTelaForm.

O desenvolvimento de uma nova tela é idêntico ao já apresentado até o momento, sendo que o novo JFrame pode receber componentes visuais.

Para apresentar um novo formulário, deve-se chamar seu método construtor e posteriormente torná-lo visível, conforme comandos da Listagem 21.

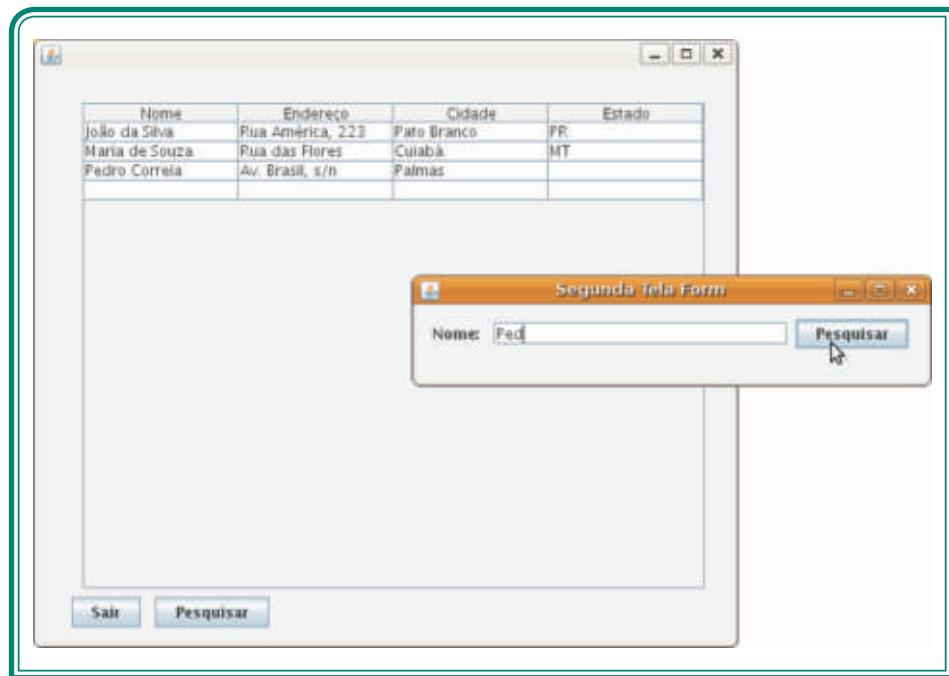
```
1. SegundaTelaForm novaJanela = new SegundaTelaForm();  
2. novaJanela.setVisible( true );
```

**Listagem 21:** Código para exibir uma nova janela (jframe)

O código anterior tem a função de instanciar um objeto (novaJanela) do tipo SegundaTelaForm, dessa forma a interface visual desenvolvida para a

SegundaTelaForm é apresentada na tela do computador. Não existe limite para abertura de janelas, assim podem ser aberta várias janelas de uma mesma classe ao mesmo tempo.

A Figura 43 apresenta a utilização de duas Janelas em um mesmo aplicativo.



**Figura 43: Utilização de duas janelas (JFrame) no mesmo aplicativo**

Dentre as propriedades de um JFrame, destacam-se:

- defaultCloseOperation: Informa qual será a forma de fechar um formulário; essa propriedade comumente é valorizada com DISPOSE (fecha a tela e a aplicação continua executando) e EXIT\_ON\_CLOSE (fecha todas as telas abertas da aplicação).
- resizable: Verdadeiro permite redimensionar a tela através das bordas.

#### **4.2.3.2 Componente Selecionador de Arquivos**

Essa é a implementação de uma interface para seleção de arquivos. Ele possui uma janela autônoma com acessórios que permite ao usuário percorrer o sistema de arquivos para selecionar um arquivo específico. Esse componente se encontra na categoria Janelas Swing.

Para adicionar esse componente à tela, deve-se clicar e arrastar o componente da paleta até a área de fora do formulário, área essa apresentada na Figura 44.

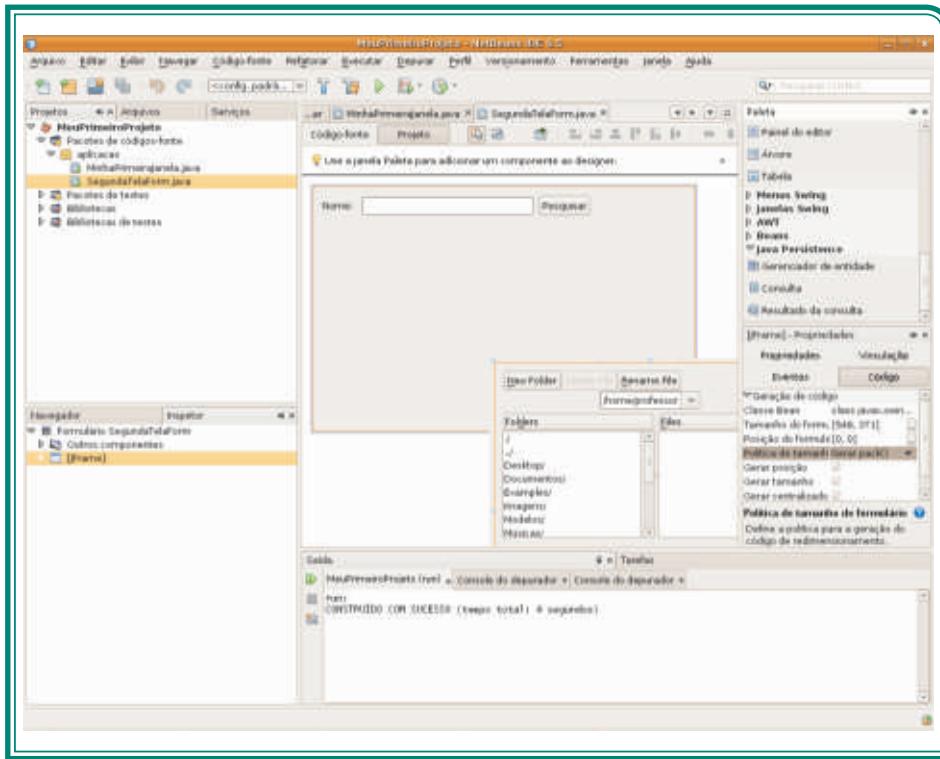


Figura 44: Adicionando o componente selecionador de arquivos na aplicação

**Atenção:** O componente Seletor de Arquivos não deve ser adicionado sobre um JFrame. Se isso ocorrer o componente fará parte do formulário. Para ele ser exibido separadamente, ele deve ser adicionado fora do formulário e sua seleção deve ser feita via paleta de Inspetor – Figura 45.

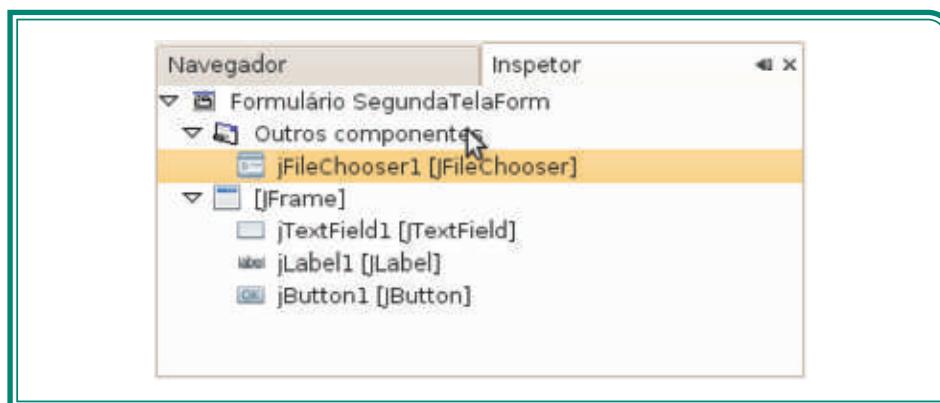


Figura 45: Adicionando o componente selecionador de arquivos na aplicação

Após adicionar o componente no projeto, como o mesmo não foi inserido dentro do formulário, a única maneira de selecioná-lo é através da janela Inspetor. Nessa janela é possível visualizar todos os componentes adicionados no projeto. Na Figura 35 é possível ver a tela Inspetor na qual o com-

ponente jFileChooser1 (Nome Default atribuído ao Seletor de Arquivos) está selecionado.

Sobre o componente, pode-se clicar com o botão direito e escolher a opção “Alterar o Nome da Variável” para atribuir um novo nome que será utilizado nas referências do código fonte.

Com o componente selecionado, podem-se mudar algumas propriedades, como por exemplo:

- a) CurrentDirectory: O diretório no qual será aberto o componente Seletor de Arquivo
- b) DialogTitle: Título da Janela
- c) DialogType: Tipo da janela, podendo ser OPEN\_DIALOG (caixa de diálogo para abrir um arquivo), SAVE\_DIALOG (caixa de diálogo para salvar um arquivo) ou CUSTOM\_DIALOG (caixa de diálogo personalizada).

Levando em consideração que foi adicionado um Seletor de Arquivo no projeto e este foi renomeado para fcAbrirArquivo, o mesmo pode ser chamado através de um botão, sendo este código apresentado na Listagem 22:

```
1. int ret = fcAbrirArquivo.showOpenDialog( this );
2. if (ret == JFileChooser.APPROVE_OPTION ) {
3.     String nome =
4.         fcAbrirArquivo.getSelectedFile().getAbsolutePath();
5.     System.out.println( nome );
6. }
```

**Listagem 22: Código para chamar e recuperar o conteúdo escolhido em um seletor de arquivos**

No código anterior, na linha 01 é apresentado o Seletor de Arquivo para o usuário, o mesmo vai ser apresentado sobre o JFrame atual (passado por parâmetro através do this). Esse comando retorna um int com o código do botão pressionado pelo usuário no Seletor de Arquivo. Esse código é validado na linha 02, verificando se o usuário escolheu o botão de confirmação (Abrir no tipo Load ou Salvar no tipo Save), se isso ocorreu é recuperado o caminho completo do arquivo (linha 3) sendo este armazenado na variável nome, que posteriormente (linha 4) será apresentado na console.

#### 4.2.3.3 Classe JOptionPane

A classe que implementa caixas de diálogos em Java chama-se JOptionPane. Ela oferece métodos para criar diversos tipos de caixas de diálogo. Para utilizar essa classe, deve-se fazer o import da mesma, através do comando apresentado na Listagem 23.

```
1. import javax.swing.JOptionPane;
```

#### Listagem 23: Comando para importar uma classe

O comando acima deve ser realizado no inicio da classe, junto com os outros imports.

A classe JOptionPane oferece métodos para criar diversos tipos de caixas de diálogo. Por exemplo, para exibir uma caixa de texto simples para informar algo ao usuário, usa-se o método showMessageDialog(), conforme Listagem 24.

```
1. JOptionPane.showMessageDialog( this,  
2. "Essa é uma caixa de diálogo Simples" );
```

#### Listagem 24: Comando para importar uma classe

Esse comando espera dois parâmetros, o primeiro deles é o formulário no qual a caixa de dialogo será apresentada, caso não exista um formulário, pode-se passar null para esse parâmetro, o segundo parâmetro é a String que será exibida na caixa de diálogo.

A Figura 46 apresenta uma imagem da caixa de diálogo que será apresentada.

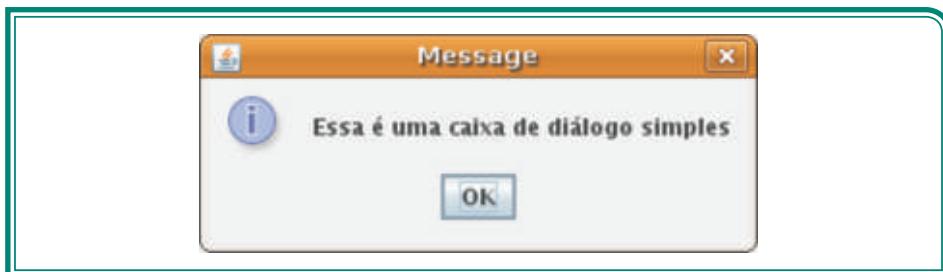


Figura 46: Adicionando o componente selecionador de arquivos na aplicação

Existe outro método no JOptionPane utilizado para exibir uma caixa de entrada: showInputDialog(). Esse método retorna sempre a String que foi digitada pelo usuário. O código para sua utilização é apresentado na Listagem 25.

```
1.     String nome = JOptionPane.showInputDialog("Digite o seu nome");
```

Listagem 25: Código para exibir uma caixa de diálogo de entrada

Nessa situação a informação digitada pelo usuário é armazenada na variável String. Se o usuário cancelar a digitação fechando a janela, essa variável fica valorizada com null. Na Figura 47 se tem um exemplo do caixa de entrada.

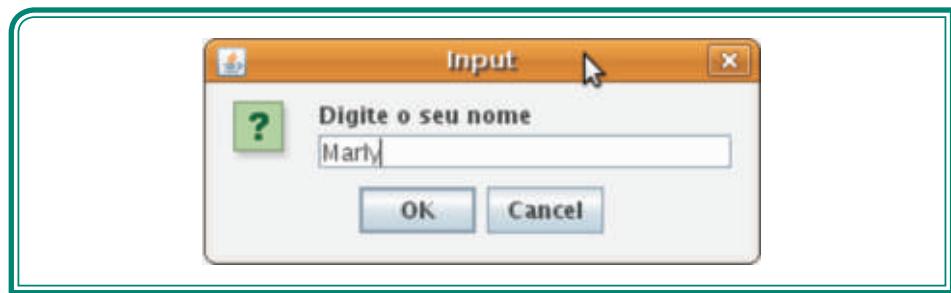


Figura 47: Caixa de diálogo do JOptionPane.showInputDialog()

#### 4.2.4 Utilizando Menus

Menus são componentes visuais de interação que costumam ficar na parte superior de um formulário. É através dos menus que diferentes comandos podem ser agrupados, sendo essa uma das alternativas para telas complexas, onde o usuário tem acesso a vários comandos.

Normalmente os menus são organizados em árvore, onde uma barra de menu pode ser composta por um ou mais menus, e cada um desses pode ser formado por um ou mais menu itens.

No NetBeans, os menus estão disponíveis na categoria Menus Swing da paleta de componentes. Para iniciar o desenvolvimento de um menu, o primeiro passo é adicionar ao formulário o componente Barra de Menu. Ele é inserido na parte superior do formulário.

Inicialmente junto com a Barra de Menu são inseridos dois Menus: File e Edit. Esses menus podem ser excluídos, clicando com o botão direito sobre eles e escolhendo a opção Excluir ou renomeados, selecionando o menu e alterando sua propriedade Text.

Se for necessário, novos menus poderão ser adicionados à tela, para isso o componente Menu deve ser selecionado na barra de tarefas e arrastado sobre o componente Barra de Menu previamente inserido na tela.

Para adicionar um item no Menu, deve-se selecionar o componente Item de Menu e arrastá-lo dentro de um Menu previamente inserido na tela. Não existe limite de Item de Menu dentro de um Menu.

Após a inserção dos Itens de Menu, pode-se modificar o nome apresentado através da propriedade Label.

Na Figura 48 tem-se o exemplo de uma aplicação em que a mesma possui uma barra de menu, composto por 3 menus (Arquivo, Opções e Ajuda) e sendo que a opção Arquivo possui quatro Item de Menu (Novo, Abrir, Salvar e Fechar).



Figura 48: Formulário com menu

Para atribuir ação a um Item de Menu, ele deve ser selecionado e na paleta de propriedades deve ser escolhida a opção Eventos. Dentro dessa tela deve-se clicar nas reticências (...) do actionPerformed para então ser adicionado um novo método, que será atribuído ao clique do menu. Esse procedimento é semelhante ao tratamento de eventos de botões, apresentados anteriormente na seção 3.3.

## Resumindo

Esta aula apresentou os componentes visuais mais utilizados em uma aplicação desktop. Foi apresentado exemplos de componentes de formulários, janelas, menus e caixa de diálogos, sendo apresentada algumas propriedades e exemplos de código.

## Atividades de Aprendizagem:

- 1) Desenvolver uma interface visual para um aplicativo de cadastro de alunos. Essa interface deve ser composta por campos de texto para digitar a identi-



ficação do aluno, o nome, o endereço e o coeficiente de rendimento do mesmo. Para a manipulação dos dados, devem ser inseridos os botões para incluir, alterar, excluir e pesquisar registro.

- 2)** Desenvolver o layout de um editor de texto, o qual é formado por uma área de texto ocupando toda a tela do formulário, e dois menus, o primeiro com o rótulo de Arquivo, possuindo os itens Novo, Abrir, Salvar e Sair, e o segundo com o rótulo Ajuda, possuindo o item Ajuda sobre o sistema. Ao clicar no item Ajuda sobre o sistema, deve ser apresentado um novo formulário, como os dados pessoais do programador em componentes rótulos.
- 3)** Desenvolver uma interface visual com apenas um botão, que ao ser pressionado apresenta um caixa de entrada para o usuário (`JOptionPane.showInputDialog`) solicitando um número, ao confirmar a entrada apresentar uma caixa de saída (`JOptionPane.showMessageDialog`) com o valor do fatorial do número informado na caixa de entrada.

# Aula 5 - Desenvolvendo aplicativos com acesso à banco de dados

## Objetivos

- Apresentar os conceitos básicos de banco de dados;
- Utilização e manutenção do sistema gerenciador de banco de dados Apache Derby;
- Desenvolvimento de aplicativos utilizando wizard do NetBeans;
- Comandos para pesquisa e manipulação de dados no banco.

**A aula é composta pelos seguintes tópicos:**

- 5.1** Entendendo o que é um banco de dados
- 5.2** Trabalhando com banco de dados no NetBeans
- 5.3** Desenvolvendo o primeiro programa para manipular dados do banco de dados através do wizard do NetBeans
  - 5.3.1** Entendendo o modelo de acesso a dados do NetBeans
  - 5.4** Desenvolvendo um aplicativo sem wizard com acesso a banco de dados
    - 5.4.1** Desenvolvendo o projeto
    - 5.4.2** Desenvolvendo a interface visual do projeto
    - 5.4.3** Criando as classes para acesso ao banco de dados
    - 5.4.4** Utilizando as classes geradas pelo NetBeans para acessar o banco de dados

## 5.1 Entendendo o que é um banco de dados

É muito difícil pensar em uma aplicação desktop sem pensar em banco de dados. Pode-se entender por banco de dados um sistema que reúna e mantenha organizada uma série de informações relacionadas a um determinado assunto em uma determinada ordem.

A lista telefônica de uma região pode ser considerada um exemplo de banco de dados. Nela os dados estão agrupados por cidade, podendo estas ser consideradas as tabelas do banco de dados. Os dados referentes a uma pessoa estão na mesma linha da lista telefônica, sendo esta linha um registro da tabela. Já os tipos ou categorias da informação (nome, endereço, telefone,etc.) sobre uma pessoa estão separadas em colunas, que são conhecidas como campos.

No exemplo da Figura 49 observam-se duas tabelas referentes aos alunos de uma escola e as disciplinas ofertadas para elas. Na tabela alunos tem-se três campos: nome, endereço e telefone, sendo armazenado nessa tabela quatro registros. Ao repositório onde se encontram as duas tabelas dá-se o nome de Banco de dados.

Aluno		Disciplina
Nome	Endereço	Telefone
Carlos Pereira	Rua Xavantes, 100	3220-1332
João da Silva	Rua das Flores, 66	3225-1580
Maria Alves	Rua Sete de Setembro	3220-7665
Pedro da Costa	Av. Marechal Deodoro, 15	3220-2332

**Figura 49: Exemplo de Banco de dados**

Para armazenar e gerenciar as informações de um banco de dados, de uma forma que permita às pessoas examiná-las de diversas maneiras, utiliza-se um Sistema Gerenciador de Banco de Dados (SGBD). Hoje existem vários SGBDs, dentre elas alguns gratuitos, como o apache Derby (<http://db.apache.org/derby>), mySQL (<http://dev.mysql.com>) e PostgreSql (<http://www.postgresql.org/>). Entre as opções pagas destacam-se o SGBD Oracle (<http://www.oracle.com>), o SQL Server (<http://www.microsoft.com/SQL>) e DB2 (<http://www.ibm.com/db2>).

Para manter um padrão no que diz respeito à sintaxe para consulta e manipulação dos dados no banco, foi desenvolvida uma linguagem de banco chamada Structured Query Language (SQL). Os comandos SQL podem ser usados para trabalharem interativamente com um banco de dados ou podem ser embutidos em uma linguagem de programação para interfacear um banco de dados. Essa linguagem dá suporte a vários comandos, como pesquisa, inclusão, alteração e exclusão de registros.

## 5.2 Trabalhando com Banco de Dados no NetBeans

O NetBeans 6.5 possui um conjunto de ferramentas que permite a utilização e a manutenção de banco de dados, podendo ser facilmente reconfigurado para qualquer banco de dados que dê suporte à Java via JDBC.

**Atenção:** Java DataBase Connectivity, ou JDBC, é um conjunto de classes e interfaces (API) escritas em Java que permite a execução de comandos em diferentes bancos de dados utilizando a linguagem de programação Java. O JDBC foi desenvolvido para manter a compatibilidade de um programa com diferentes bancos de dados.



Junto com o NetBeans é instalado um SGBD livre chamado Apache Derby, também conhecido como JavaDB, que será utilizado no decorrer deste documento.

Para ter acesso às ferramentas de gerenciamento do banco de dados existente no NetBeans, deve-se acessar a janela Serviços (menu Janela – Serviços) e escolher a categoria Banco de Dados, como apresentado na Figura 50.

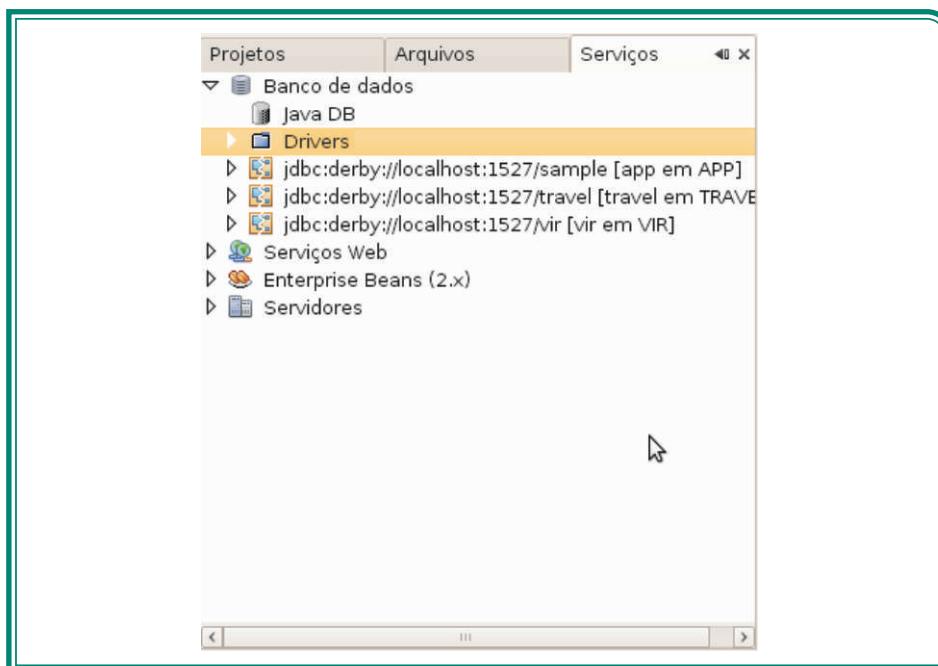


Figura 50: Janela com as ferramentas para manutenção do banco de dados

Nessa tela visualizam-se os bancos de dados já vinculados ao NetBeans, como é o caso do sample, travel e vir apresentados na figura. Todos esses bancos estão armazenados no Apache Derby. Também é possível gerenciar os

Drivers JDBC para se ter acesso à diferentes bancos de dados (opção Drivers) e gerenciar o banco de dados Apache Derby, através da opção Java DB.

Ao ser iniciado, o NetBeans mantém o SGDB Derby em estado de espera, para executá-lo (só é possível executar comandos no Apache Derby se o mesmo estiver em estado de execução) clica-se com o botão direito sobre o Java DB e escolhe a opção Inicializar Serviço.

Clicando com o botão direito sobre o Java DB, tem-se acesso também há algumas outras opções, dentre elas a de criar um novo banco de dados.

Para os exemplos utilizados na sequência, será utilizado um banco de dados chamado escola, o qual possui uma tabela de alunos, formado pelos campos código (campo numérico inteiro), nome (campo texto), endereço (campo texto) e coeficiente (campo numérico real).

Para criar o banco de dados, clica-se com o botão direito sobre o item Java DB e escolhe-se a opção Criar Banco de Dados...

Na tela apresentada (Figura 51), informa-se o nome do banco de dados, um usuário e uma senha para se ter acesso aos dados armazenados.



Figura 51: Tela para criação de um banco de dados

Assim é estabelecido um vínculo entre o NetBeans e o banco de dado criado. Para se conectar com o novo banco clica-se com o botão direito sobre o novo vínculo e escolhe-se a opção Conectar, conforme Figura 52.

Serão apresentadas as opções de tabelas, visões e procedimentos existentes para o banco de dados. Como o banco acabou de ser criado, as opções citadas anteriormente aparecem em branco.

Para criar uma nova tabela, clica-se com o botão direito sobre a opção Tabelas e escolhe-se a opção Criar Tabela – Figura 53.

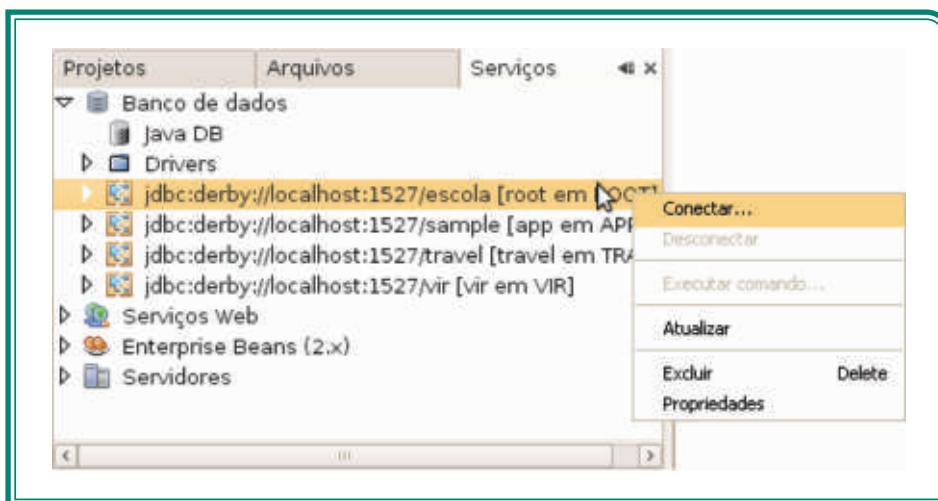


Figura 52: Conectando com o banco de dados

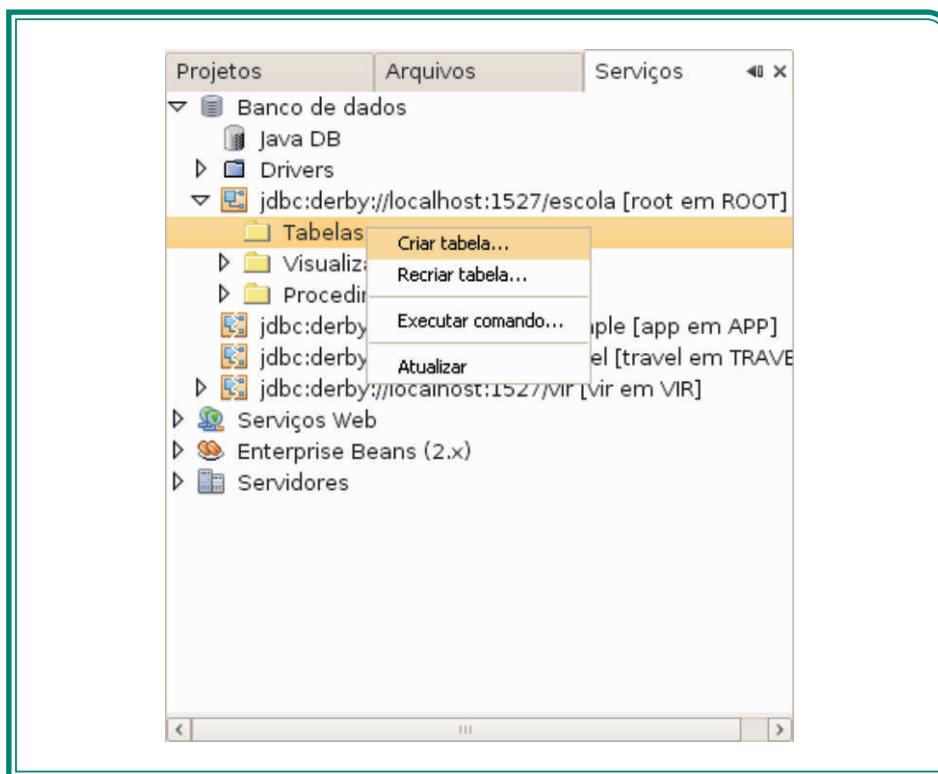


Figura 53: Criando uma nova tabela para o banco de dados escola

Na janela seguinte é definido o nome da tabela e os campos que a formarão, conforme Figura 54. Para a inclusão de um novo campo, deve-se clicar no botão Adicionar Coluna.

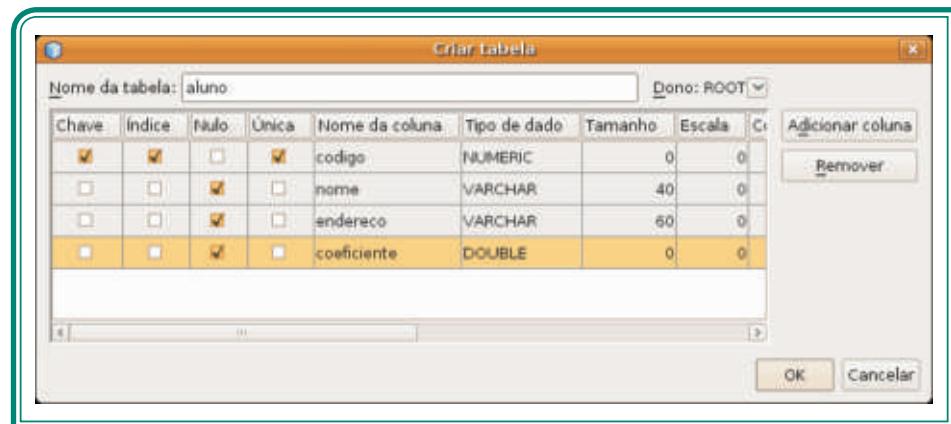


Figura 54: Tela para criar uma nova tabela

A tabela é composta por quatro campos, sendo o campo de chave primária (opções chave e índice ligadas) código, o qual armazenará informações numéricas. O campo chave primária é o que individualiza o registro, ou seja, esse campo não poderá ter informações repetidas em toda a tabela.

O campo nome é do tipo texto (varchar) que armazenará até 40 caracteres, o campo endereço do tipo texto (varchar) que armazena até 60 caracteres, e por fim o campo coeficiente do tipo real (double). Após preencher as informações, deve-se clicar no botão OK, sendo a tabela criada.

Após criar a tabela é possível manipular seus registros, clicando com o botão direito sobre a tabela e escolhendo a opção Visualizar Dados – Figura 55.

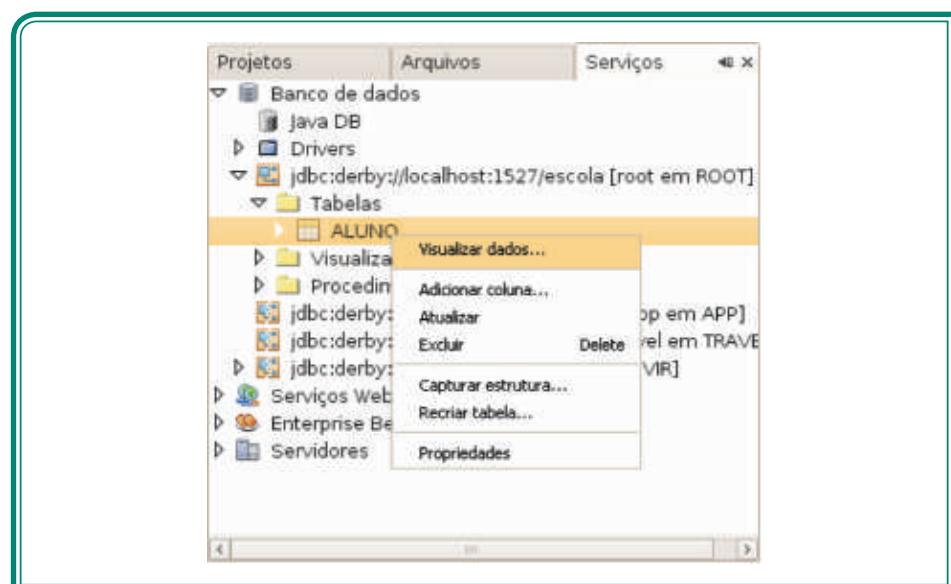


Figura 55: Manipulando os registros da tabela alunos

Dessa maneira, é apresentada uma tela para manipulação das informações. Essa tela permite a execução de comandos SQL, conforme apresentado em (1) na barra, visualizar os registros resultantes desses comandos em (2) ou adicionar informações manualmente na tabela através do botão em (3). Ao clicar no botão Inserir Registro, uma nova tela é apresentada – Figura 56.

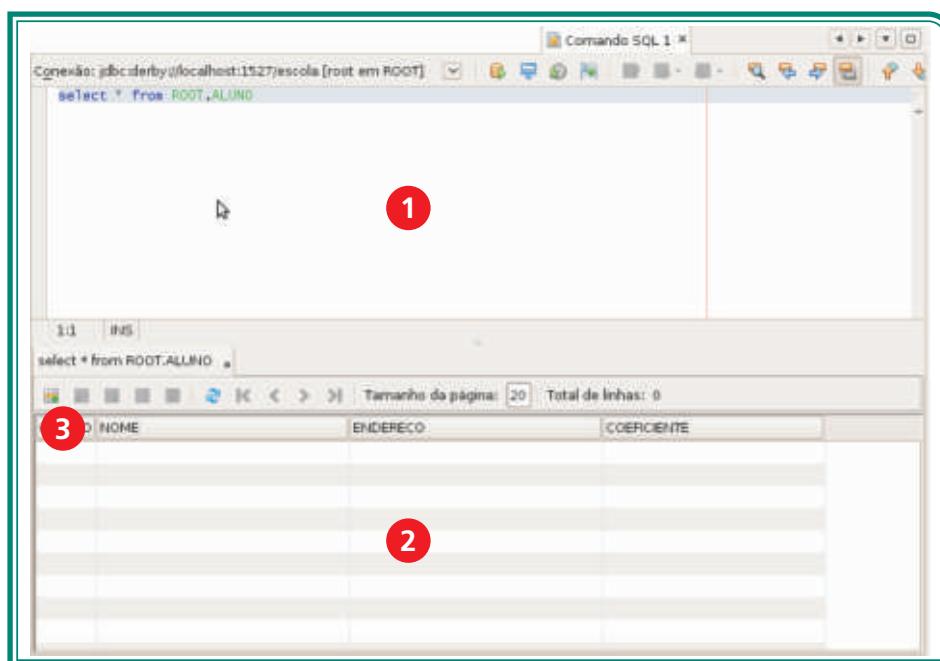


Figura 56: Tela para gerenciamento de registros

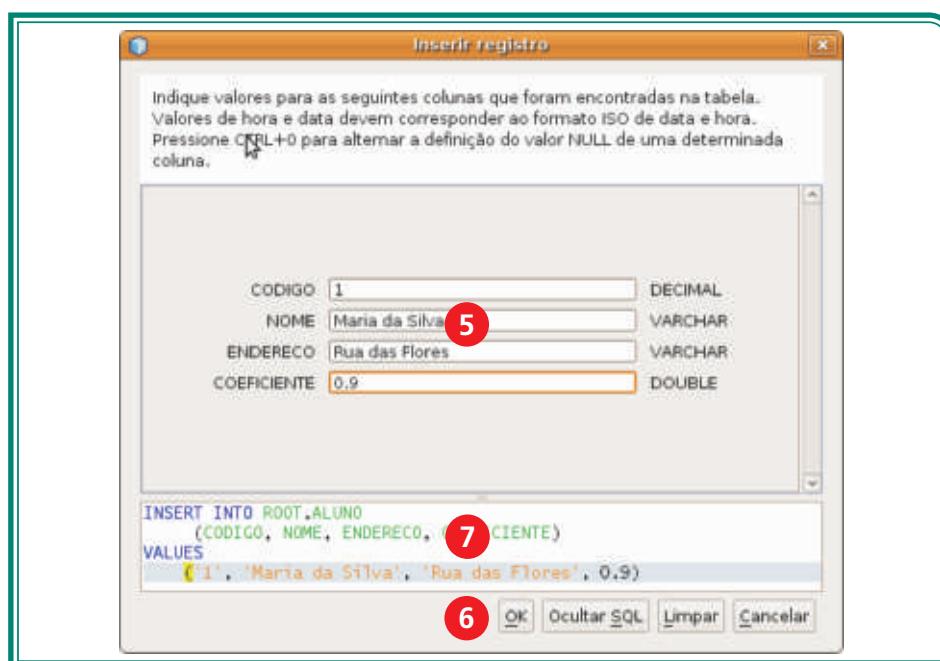


Figura 57: Tela de inclusão de registros

Para incluir um registro preenchem-se os campos da tela (5) e clica-se no botão OK (6). Se o botão Mostrar SQL for pressionado, é exibida a sintaxe do comando SQL para incluir o registro na tabela, conforme (7).

### 5.3 Desenvolvendo o primeiro programa para manipular dados do banco de dados através do wizard do NetBeans

Para iniciar o desenvolvimento de um primeiro aplicativo que faça uso do banco de dados criado anteriormente, deve-se escolher o menu Arquivo – Novo Projeto.

Na tela apresentada, deve-se escolher a categoria Java – Aplicativo da área de trabalho Java – Figura 58.

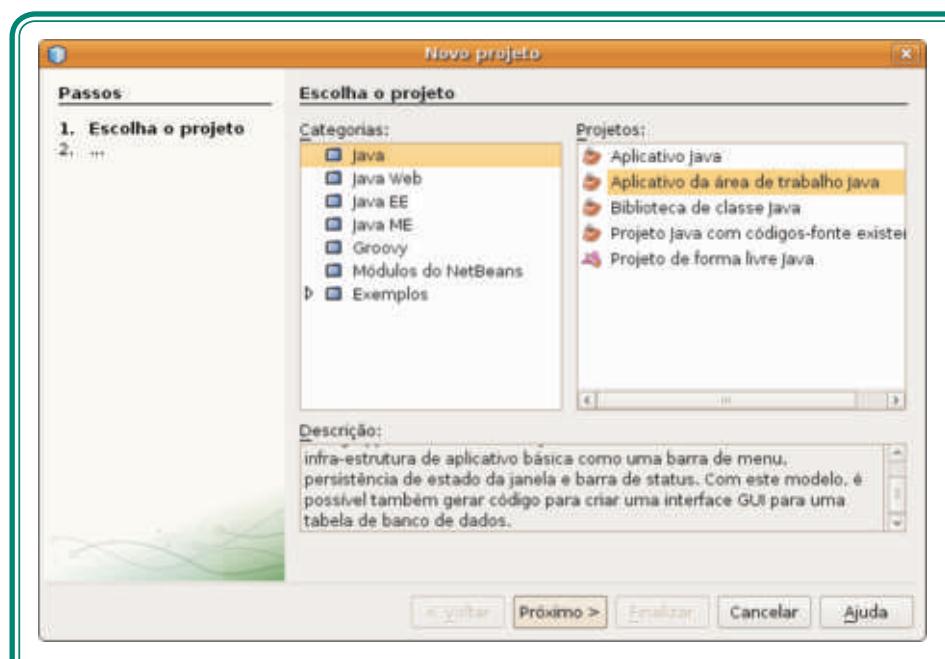


Figura 58: Desenvolvimento de um novo projeto com banco de dados

Na tela seguinte – Figura 59 – são digitadas algumas informações, como o nome do projeto, o local onde o mesmo será salvo e o nome da classe principal do projeto. Na opção Escolher shell de aplicativo escolhe-se a opção Aplicativo do Banco de Dados, o qual permite o desenvolvimento de uma aplicação com banco de dados através de um wizard. Após preencher as informações, deve-se clicar em Próximo.

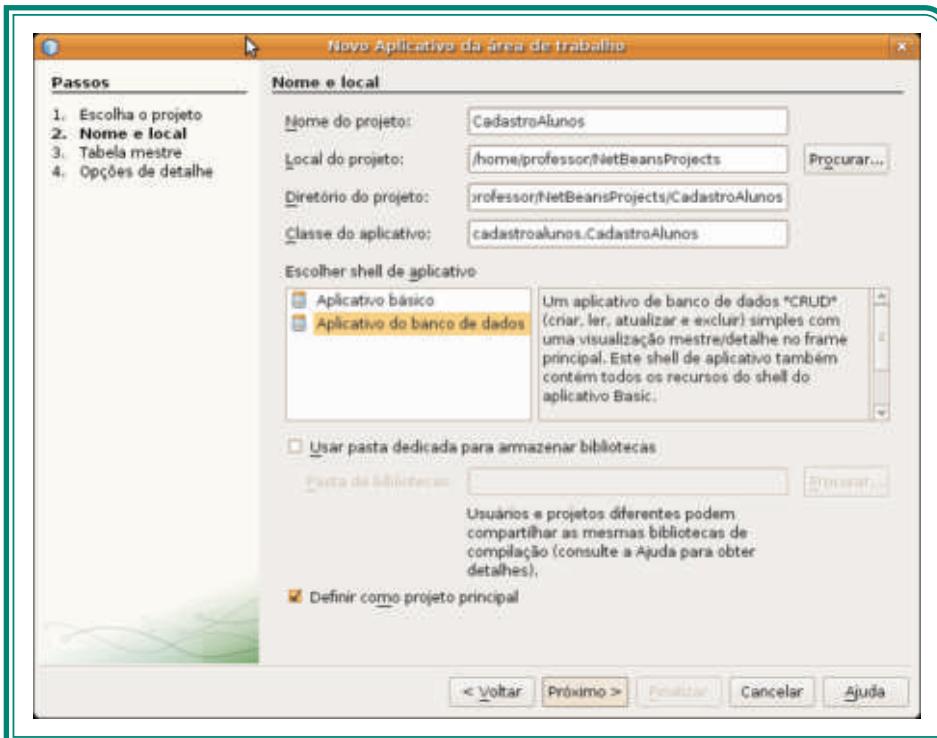


Figura 59: Desenvolvimento da classe principal

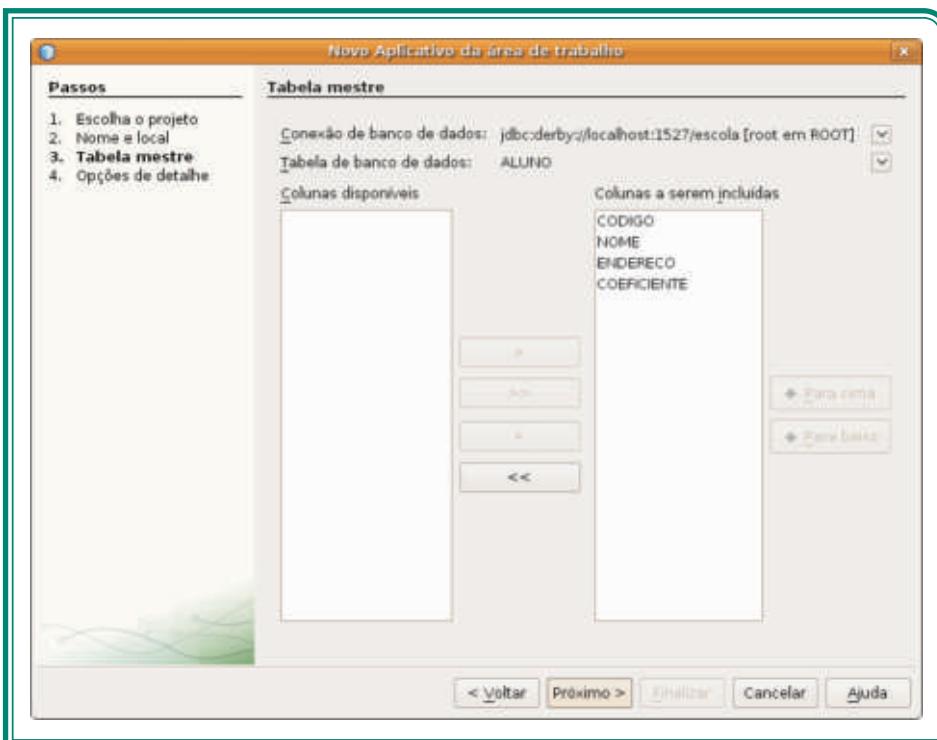


Figura 60: Seleção dos campos que serão utilizados no programa

Na tela da Figura 60, é apresentada a opção para a escolha do banco de dados

(Conexão de banco de dados) e da tabela (Tabela de banco de dados) que será utilizada pelo aplicativo. Após escolher a tabela, é apresentada uma lista de campos. Os campos desejados para o aplicativo devem permanecer em Colunas a serem incluídas, caso contrário deve-se movê-las para Colunas Disponíveis.

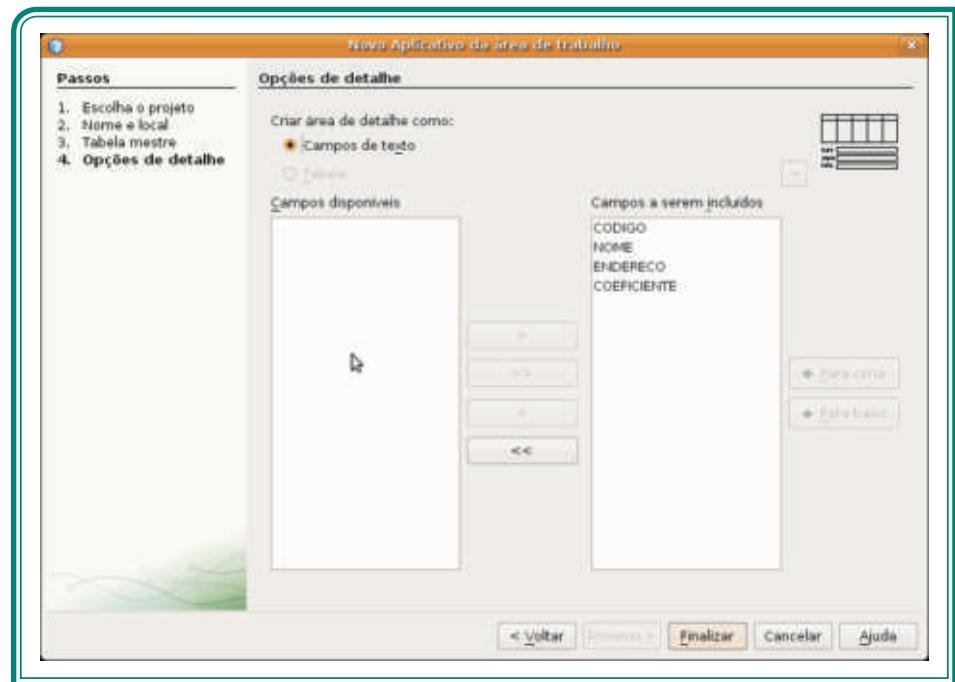


Figura 61: Seleção dos campos que serão utilizados na interface visual

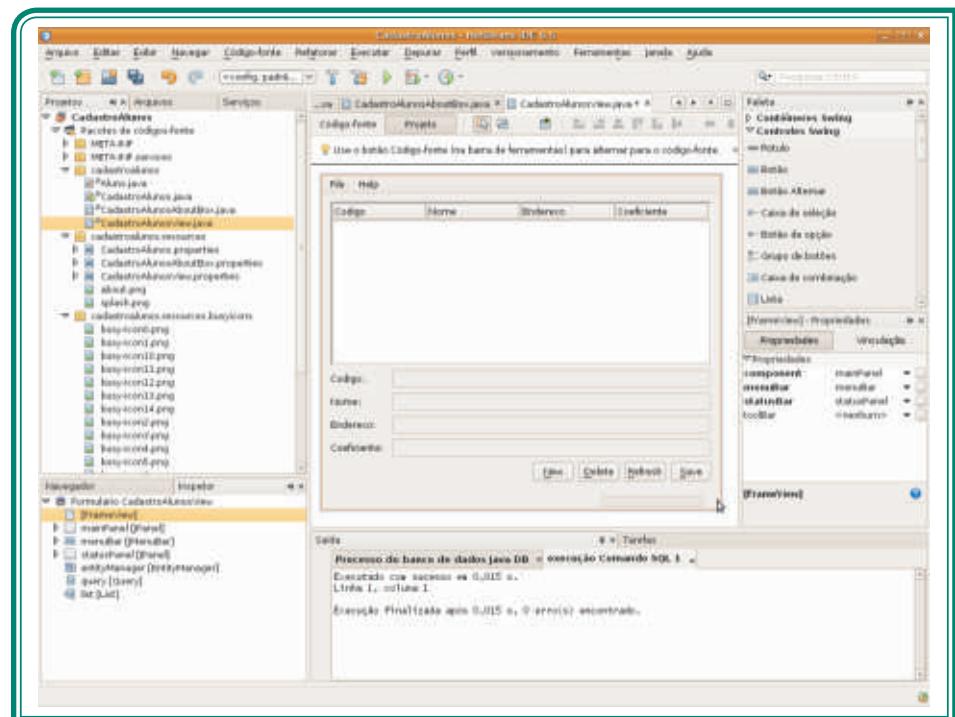


Figura 62: Tela para o desenvolvimento do aplicativo

Na última tela do wizard – Figura 61 – refere-se à interface visual do aplicativo, onde é possível retirar alguns campos da tela e optar pelos componentes Campos de texto/Tabela. Para finalizar o desenvolvimento clica-se no botão de Finalizar.

No pacote META-INF se encontra o arquivo xml responsável pela integração do banco de dados com a aplicação desktop. Apesar de ser utilizado um banco de dados relacional (apache Derby), a utilização do arquivo persistence.xml permite trabalhar com o banco de dados como se o mesmo fosse orientado a objetos. No código fonte do projeto toda a manipulação dos registros acontecem de forma orientada a objetos, através de um camada de persistência chamada JPA (Java Persistence API, ou API de Persistência Java).

O pacote META-INF.service possui um único arquivo, o qual armazena o nome da aplicação.

No pacote cadastroalunos encontram-se as classes do programa. Aluno é uma classe de persistência, possuindo em seu interior os atributos referentes a tabela aluno que encontra-se no banco de dados escola. A classe CadastroAlunos é responsável por iniciar o aplicativo. A interface visual do aplicativo encontram-se nas classes CadastroAlunosView, que apresenta a interface de (1) na Figura 63 e a classe CadastroAlunosAboutBox possui uma tela de Sobre (2). Para executar o aplicativo, o botão Executar Main Project deve ser pressionado.

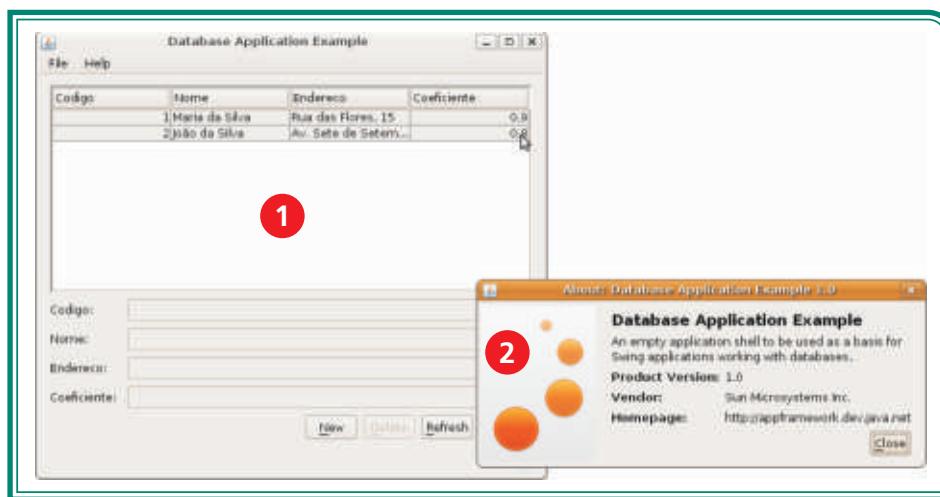


Figura 63: Tela do aplicativo executando

Já os pacotes aplicacao.resources a aplicacao.resources.busyicons possuem respectivamente os arquivos de configurações do aplicativo e as imagens utilizadas por este.

### **5.3.1 Entendendo o modelo de acesso a dados do NetBeans**

Para acessar um banco de dados relacional, o NetBeans utiliza uma camada intermediária que abstrai os códigos SQL, dessa maneira o programador pode fazer a manipulação dos registros, como a inclusão, exclusão e alteração sem conhecer a sintaxe SQL.

A camada intermediária de persistência utilizada no aplicativo anterior é o JPA, que abstrai um banco de dados relacional para um modelo orientado a objetos. Assim, para armazenar um registro no banco é necessário criar um objeto referente a esse registro, valorizar as propriedades do objeto e só então fazer a persistência.

Dentre as principais classes utilizadas pelo JPA no programa, destacam-se:

- a)** Alunos – Essa classe representa um registro na tabela. É instanciando um objeto dessa classe com o registro que se deseja incluir, excluir ou alterar.
- b)** EntityManagerFactory – Essa classe é responsável pela conexão com o banco de dados. Também é responsabilidade dessa classe carregar todas as estruturas de tabelas e fazer os mapeamentos dos dados do banco de dados para serem utilizados como objetos.
- c)** EntityManager – Essa classe é responsável pela execução dos comandos no banco de dados. Normalmente é criado um EntityManager para cada comando que se deseja executar no banco. Para a inclusão e alteração de registros é utilizado o método persist(), e para a exclusão o método remove(). Se for necessário fazer uma pesquisa no banco, o método createQuery() é utilizado. O mesmo espera um comando SQL e pode retornar um ou mais registros.

## **5.4 Desenvolvendo aplicativo sem wizard com acesso a banco de dados**

Embora o NetBeans possua um Wizard que permite o desenvolvimento rápido de aplicativos com acesso ao banco de dados, muitas vezes torna-se necessário tratar as informações de forma personalizada, como por exemplo, navegar pelos registros do banco capturando informações específicas e realizar pesquisas personalizadas.

Para abordar esses recursos específicos, será desenvolvido um aplicativo de controle de alunos, no qual o usuário terá acesso a botões de navegação (prime-

íro registro, registro anterior, próximo registro e último registro), além das operações de inclusão, alteração, exclusão e pesquisa através do campo código.

### **5.4.1 Desenvolvendo o projeto**

Para iniciar o desenvolvimento desse aplicativo, escolha a opção Novo Projeto no menu Arquivo. Na janela apresentada deve-se escolher a categoria Java e o projeto Aplicativo Java. Na tela seguinte deve ser informado o nome do projeto (GerenciamentoAluno) e o local onde o mesmo será armazenado. A opção “Criar Classe Principal” deve ser desligada, uma vez que a classe principal será JFrame. Clica-se no botão de Finalizar.

### **5.4.2 Desenvolvendo a Interface Visual do projeto**

Depois de criado o projeto, deve-se desenvolver a interface visual do aplicativo. Para isso clica-se com o botão direito sobre o nome do projeto (GerenciamentoAluno), escolhendo a opção Novo – Formulário JFrame. Na tela apresentada, deve-se informar o nome da classe (TelaPrincipal) e clicar no botão Finalizar. Dessa maneira será iniciado o desenvolvimento da tela para gerenciamento de registro.

A TelaPrincipal será formada por quatro componentes Rótulo e quatro Campos de Texto, um para cada campo da tabela aluno, e nove botões conforme apresentado na Figura 64.

Para facilitar o desenvolvimento, foi alterado o nome dos componentes Campos de Texto e Botões (clicando com o botão direito sobre o respectivo componente e escolhendo a opção “Alterar o nome da variável”) para:

- Campos de Texto: tfCod, tfNome, tfEndereco e tfCoeficiente
- Botões: btPrimeiro, btAnterior, btProximo, btUltimo, btPesquisar, btNovo, btIncluir, btAlterar e btExcluir.

### **5.4.3 Criando as classes para acesso ao banco de dados**

Para linkar o banco de dados com o aplicativo desenvolvido deve-se criar uma classe de persistência. Para isso clica-se com o botão direito sobre o projeto (GerenciamentoAluno). Na tela apresentada escolhe-se a categoria Persistente, o qual possui as classes para trabalhar com banco de dados, e em Tipos de Arquivos escolhe-se Classes de Entidade do banco de dados – Figura 65.

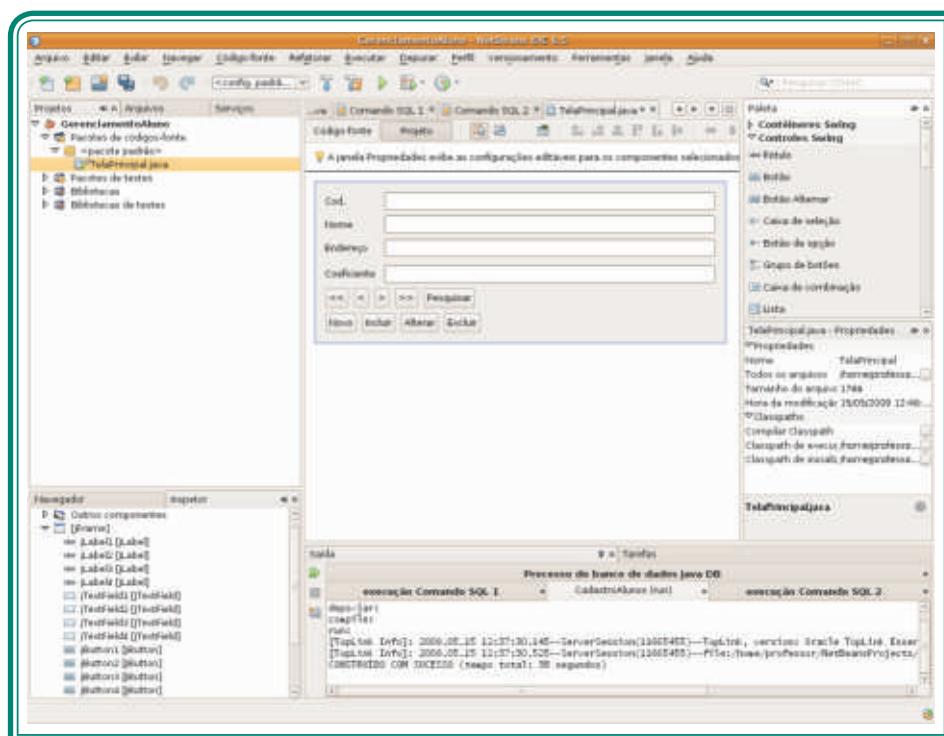


Figura 64: Tela do projeto para gerenciamento de alunos

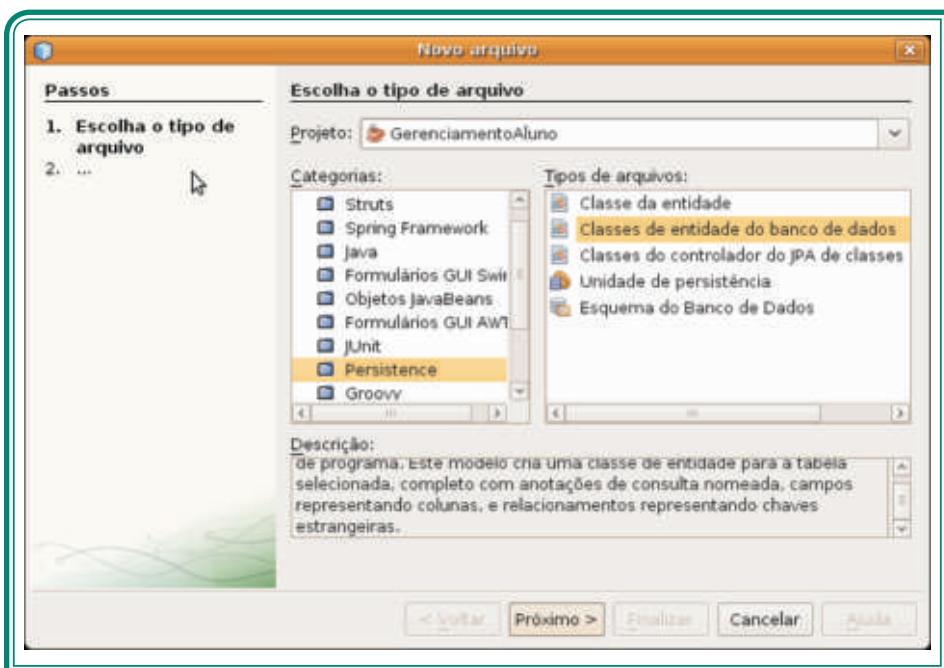


Figura 65: Janela para criar as classes de persistência

Na tela seguinte é feito o vínculo da classe gerada com a tabela no banco de dados, para isso na opção Conexão de banco de dados deve-se escolher a conexão referente ao banco de dados escola, esta definida anteriormente

(Seção 5.2), assim é exibida na lista de tabelas disponíveis. A tabela ALUNO deve ser adicionada em Tabelas selecionadas, conforme Figura 66 .

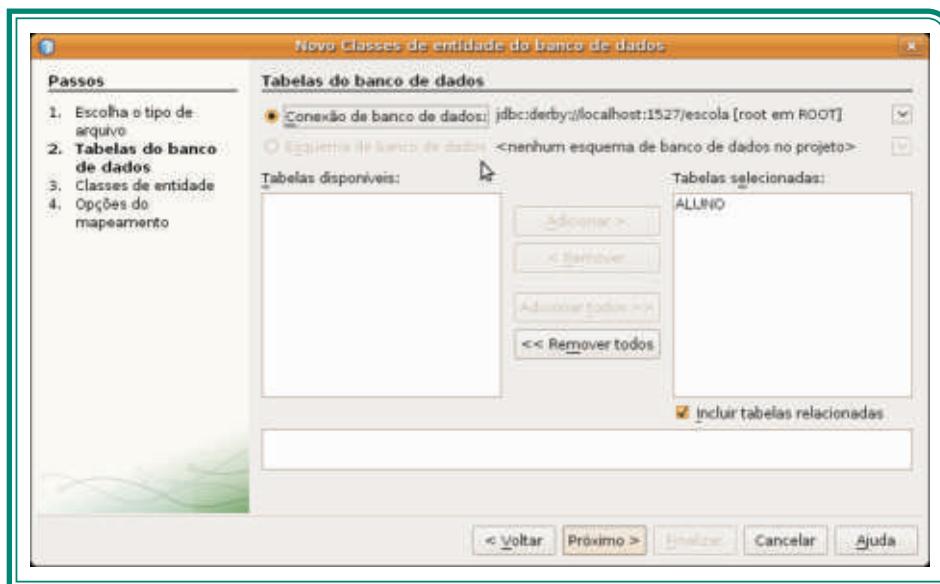


Figura 66: Vinculação do projeto com o banco de dados

Na tela seguinte são apresentadas as classes de entidades associadas às tabelas (para o exemplo tem-se a classe da tabela Aluno), que será armazenado em um pacote chamado bd, entretanto deve-se criar uma unidade de persistência, sendo responsabilidade desta as operações de banco de dados, como inclusão, exclusão, alteração e pesquisa no banco. Essa unidade de persistência pode ser criada a partir do botão Criar Unidade de Persistência – Figura 67.

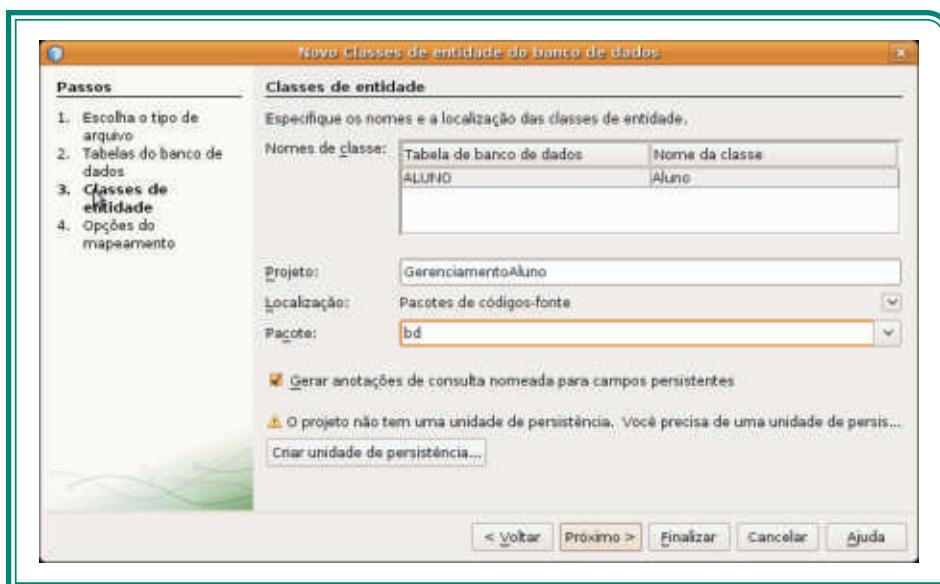


Figura 67: Informações sobre as entidades do programa

Ao clicar no botão criar unidade de persistência, a tela da Figura 68 é exibida, solicitando um nome para a classe da Unidade de Persistência, que costuma ter o nome do projeto e o sufixo PU (Persistence Unit). Como biblioteca para persistência será utilizada a TopLink, por possuir um bom desempenho com o NetBeans. Ao final clica-se em Criar e conclui-se a criação das classes para persistência clicando no botão Finalizar.

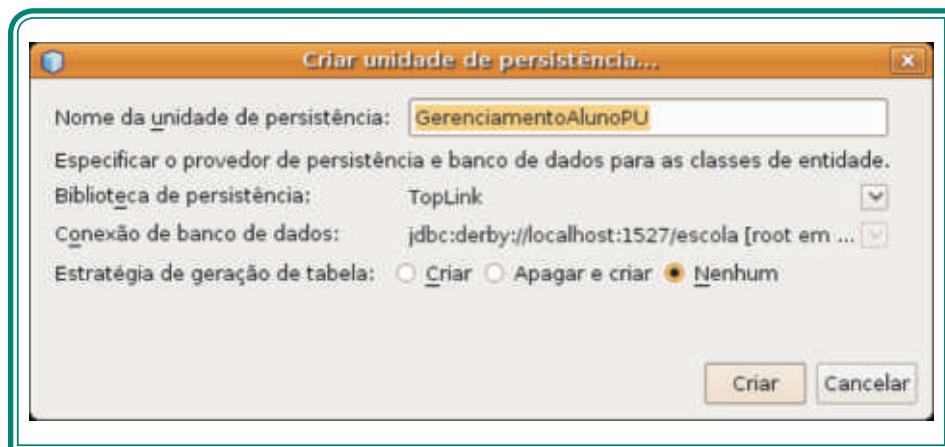


Figura 68: Tela para criar a unidade de persistência

Agora deve-se adicionar ao projeto a biblioteca para acesso ao banco de dados Apache Derby, clicando-se com o botão direito do mouse sobre a categoria biblioteca e escolhendo a opção Adicionar Biblioteca – Figura 69.

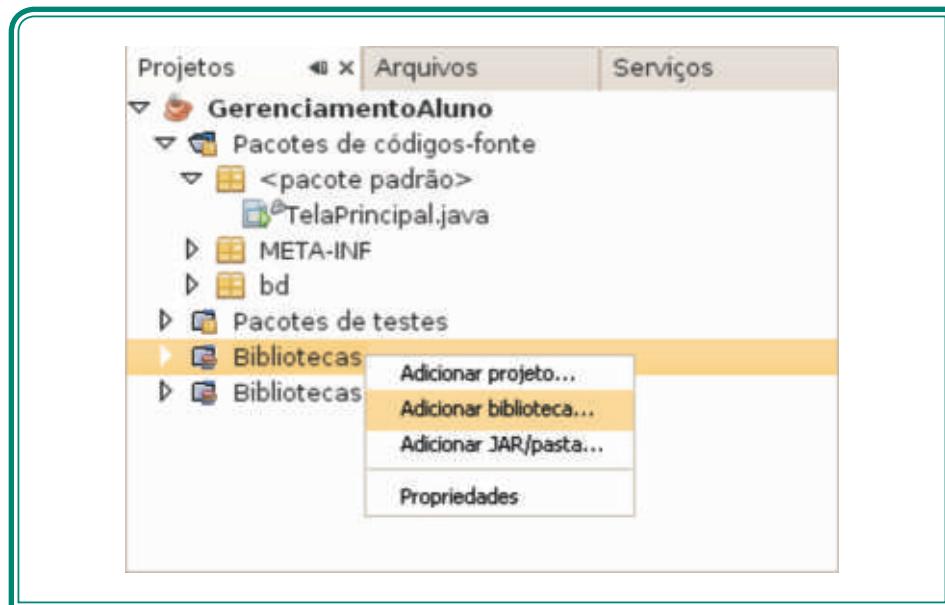


Figura 69: Método para adicionar uma biblioteca ao projeto

Na tela apresentada - Figura 70, deve-se escolher o opção Driver do Java DB. Para confirmar a inclusão utiliza-se o botão Adicionar Biblioteca.

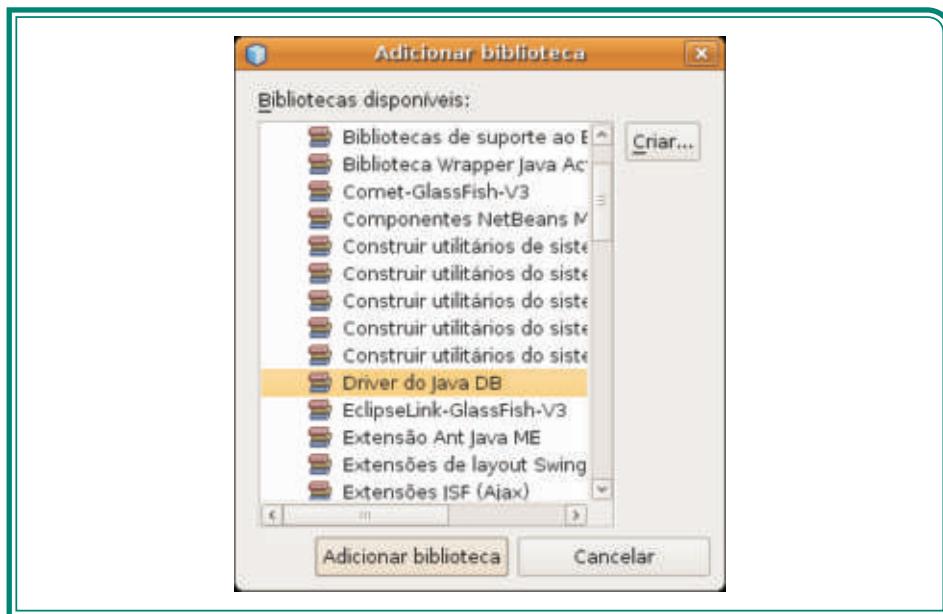


Figura 70: Incluir ao projeto a biblioteca para utilização do Apache Derby (Java DB)

#### 5.4.4 Utilizando classes geradas pelo NetBeans para acessar banco de dados

O objetivo desse programa é apresentar uma interface visual para manipular os dados da tabela Aluno, essa armazenada no banco de dados escola.

Ao ser executado, o projeto GerenciamentoAluno apresentará os dados referentes ao primeiro registro da tabela. A medida que se deseja navegar pelos registros, os botões <<, <, >, >> serão utilizados, e possuem as funções respectivas de posicionar no primeiro registro, registro anterior, próximo e último registro da tabela.

O usuário também poderá localizar um registro no banco através do botão pesquisar, o qual apresenta uma caixa de entrada (JOptionPane.showInputDialog) solicitando um código. O registro localizado será apresentado em uma caixa de saída (JOptionPane.showMessageDialog).

Além dos botões de navegação, o usuário pode modificar o conteúdo da tabela Aluno. Para adicionar um novo registro se utiliza o botão Incluir, o qual fará a inserção de um novo registro com os dados digitados nos campos de texto.

Para alterar e excluir o registro apresentado na tela utiliza-se respectivamente os botões Alterar e Excluir. Já o botão novo tem a função de limpar os campos de texto para a digitação dos dados de um novo registro.

Para iniciar a codificação do programa, deve ser incluído no inicio do código fonte da classe TelaPrincipal.java a importação das classes que serão utilizadas ao longo do projeto, conforme Listagem 26.

```
1. import bd.Aluno;
2. import java.util.Vector;
3. import javax.persistence.Persistence;
4. import javax.persistence.EntityManagerFactory;
5. import javax.persistence.EntityManager;
```

**Listagem 26: Código referente a importação das classes utilizadas no projeto**

A linha 01 importa a classe de entidade Aluno, o qual representa um registro dentro da tabela. A classe Vector (linha 02) é utilizada para armazenar uma lista com todos os registros da tabela, permitindo a recuperação dos registros para apresentar na tela (através dos botões <<, <, >, >>). A classe Persistence (linha 03) tem a função de realizar a conexão com o banco de dados, essa conexão é armazenada no objeto EntityManagerFactory (linha 04). A partir da EntityManagerFactory é possível instanciar uma EntityManager (linha 05), sendo essa classe responsável pelos comandos de manipulação da tabela, como inclusão, alteração e exclusão de registros, além de ser possível localizar registros com a instância dessa classe.

Seguindo, deve ser incluído o código para realizar a conexão com o banco e recuperar o primeiro registros da tabela, que é apresentando na tela inicial do projeto. Esse código é apresentado na Listagem 27.

```
1. public class TelaPrincipal extends javax.swing.JFrame {
2.
3.     private Vector listaRegistros;
4.     private int id = 0;
5.
6.
7.     /** Creates new form TelaPrincipal */
8.     public TelaPrincipal() {
9.         initComponents();
10.        iniciarConexaoBanco();
```

```

11. }
12.
13. private void iniciarConexaoBanco() {
14.     EntityManagerFactory emf =
15.         Persistence.createEntityManagerFactory("GerenciamentoAlunoPU");
16.
17.     EntityManager em = emf.createEntityManager();
18.
19.     listaRegistros = (Vector) em.createQuery ( "select a from Aluno a")
20.         .getResultSet();
21.
22.     posicionarPrimeiroRegistro();
23.
24.
25. }
26.
27. private void posicionarPrimeiroRegistro() {
28.
29.     if ( ! listaRegistros.isEmpty() ) {
30.         id = 0;
31.
32.         Aluno aluno = (Aluno) listaRegistros.get( id );
33.         tfCod.setText( String.valueOf( alunos.getCodigo() ) );
34.         tfNome.setText( alunos.getNome() );
35.         tfEndereco.setText( alunos.getEndereco() );
36.         tfCoeficiente.setText( String.valueOf( alunos.getCoeficiente() ) );
37.     }
38.
39. }

```

**Listagem 27:** Código utilizado no início da classe, sendo necessário para a conexão com o banco e a recuperação/apresentação do primeiro registro

Após a declaração da TelaPrincipal, é declarado o objeto listaRegistros do tipo Vector (linha 2), o qual armazenará todos os registros existentes na tabela Aluno, e uma variável inteira id (linha 4), que armazena o índice do registro atual apresentado na tela. O índice varia de zero (primeiro registro) até a quantidade de registro menos um (último registro). Por exemplo, existindo dez registros na tabela, o índice do primeiro registro é zero e do último registro é o número nove.

Dentro do método construtor da classe TelaPrincipal, após o método initComponents() é executado o método iniciarConexaoBanco() – linha 10. Esse método terá a função de executar a conexão com o banco de dados, recuperar os registros da tabela Aluno e apresentar o primeiro registro na tela, esse método é codificado na linha 13.

A primeira linha do método (linha 14) tem a função de instanciar um objeto do tipo EntityManagerFactory. Para a instanciação, é necessário passar o nome da Unidade de persistência desenvolvida (o nome da unidade de persistência é apresentado na Figura 68). Na linha seguinte (linha 17), é instanciado um objeto do tipo EntityManager, o qual terá a função de manipular os dados do banco.

Na linha 19 é executada uma pesquisa para retornar todos os registros existentes na tabela Aluno, para isso é executado um comando SQL de pesquisa (“select a from Aluno a”), esse comando irá retornar uma lista de registros através do método getResultList(). Essa lista com todos os registros da tabela será armazenado no objeto listaRegistro, declarada anteriormente na linha 3.

Uma vez recuperado o registro do banco de dados, é executado um método para apresentar o primeiro registro na tela – `posicionarPrimeiroRegistro` – linha 22.

Esse método inicialmente verifica se o objeto listaRegistro não está vazio – validação da linha 29. Não estando vazio, é possível posicionar no primeiro registro da tabela, assim é armazenado na variável id o índice do primeiro registro (0). Para recuperar o registro é utilizado o método `get()` do objeto listaRegistro. O retorno desse método é armazenado na classe de entidade Aluno.

A instância aluno (iniciado com letra minúscula) possui os campos da tabela aluno, sendo estes exibidos nos componentes campos de texto, respeitando os respectivos formatados (na linha 33 é convertido o valor do campo código de inteiro para String para então ser exibido na tela e na linha 35 o campo coeficiente foi convertido de double para String para exibição).

Além do método para posicionar no primeiro registro, outros métodos de navegação foram codificados (`posicionarRegistroAnterior`, `posicionarProximoRegistro`, `posicionarUltimoRegistro`), conforme Listagem 28.

```
1. private void posicionarRegistroAnterior() {  
2.  
3.     if ( id > 0 ) {  
4.         id--;  
5.         Aluno aluno = (Aluno) listaRegistros.get( id );  
6.         tfCod.setText( String.valueOf( aluno.getCodigo() ) );  
7.         tfNome.setText( aluno.getNome() );  
8.         tfEndereco.setText( aluno.getEndereco() );
```

```

9.     tfCoeficiente.setText( String.valueOf( aluno.getCoeficiente() ) );
10.    }
11.
12. }
13.
14. private void posicionarProximoRegistro() {
15.
16.     if ( id < listaRegistros.size() -1 ) {
17.         id++;
18.         Aluno aluno = (Aluno) listaRegistros.get( id );
19.         tfCod.setText( String.valueOf( aluno.getCodigo() ) );
20.         tfNome.setText( aluno.getNome() );
21.         tfEndereco.setText( aluno.getEndereco() );
22.         tfCoeficiente.setText( String.valueOf( aluno.getCoeficiente() ) );
23.     }
24.
25. }
26.
27. private void posicionarUltimoRegistro() {
28.
29.     if ( ! listaRegistros.isEmpty() ) {
30.         id = listaRegistros.size() - 1;
31.         Aluno aluno = (Aluno) listaRegistros.get( id );
32.         tfCod.setText( String.valueOf( aluno.getCodigo() ) );
33.         tfNome.setText( aluno.getNome() );
34.         tfEndereco.setText( aluno.getEndereco() );
35.         tfCoeficiente.setText( String.valueOf( aluno.getCoeficiente() ) );
36.     }
37.
38. }

```

#### Listagem 28: Métodos utilizados para a navegação do registro

Para posicionar em um registro anterior é necessário verificar se o sistema não está exibindo o primeiro registro. Por esse motivo é realizada a verificação da linha 3. Caso o índice do registro exibido seja maior do que zero, esse índice é decrementado de um sendo recuperado o registro referente ao novo índice na listaRegistros. O resultado da pesquisa é apresentado na tela.

O método posicionarProximoRegistro também consiste o conteúdo da variável id, verificando se o registro exibido na tela é o último da tabela. Caso isso não ocorra é incrementado um no valor da variável id (linha 17), e recuperado o registro correspondente a esse novo índice no objeto listaRegistros, sendo apresentado os campos desse registro nos respectivos campos de texto.

O método ultimoRegistro valida o objeto listaRegistros para saber se o mesmo não está vazio. Existindo registros no objeto, é atualizado o valor da variável id (linha 30), sendo recuperado o registro com o id e apresentado na tela.

Para vincular a execução dos métodos citados anteriormente com o clique nos respectivos botões, deve-se clicar duplo sobre cada botão de navegação, adicionando ao clique do botão a chamada ao método correspondente, como é exibido na Listagem 29.

```
1. private void btPrimeiroActionPerformed(java.awt.event.ActionEvent evt) {  
2.     posicionarPrimeiroRegistro();  
3. }  
4.  
5. private void btAnteriorActionPerformed(java.awt.event.ActionEvent evt) {  
6.     posicionarRegistroAnterior();  
7. }  
8.  
9. private void btProximoActionPerformed(java.awt.event.ActionEvent evt) {  
10.    posicionarProximoRegistro();  
11. }  
12.  
13. private void btUltimoActionPerformed(java.awt.event.ActionEvent evt) {  
14.    posicionarUltimoRegistro();  
15. }
```

#### Listagem 29: Chamada dos métodos de navegação

O primeiro método (linha 1) é apresentado ao realizar dois cliques sobre o botão primeiro, por esse motivo foi executado o método posicionarPrimeiroRegistro(). Os demais métodos são referentes aos dois cliques no botão anterior (linhas 5 a 7), botão próximo (linhas 9 a 11) e botão ultimo (linhas 13 a 15), executando respectivamente os métodos posicionarRegistroAnterior() – linha 6, posicionarProximoRegistro() – linha 10 e posicionarUltimoRegistro() – linha 14.

O botão Novo tem a função de limpar o conteúdo dos componentes campos de texto. Seu código é apresentado na Listagem 30.

```
1. private void btNovoActionPerformed(java.awt.event.ActionEvent evt) {  
2.     tfCod.setText( "");  
3.     tfNome.setText( "");  
4.     tfEndereco.setText( "");  
5.     tfCoeficiente.setText( "");  
6. }
```

#### Listagem 30: Método para limpar os campos de texto, executado pelo botão novo

O código referente ao botão novo tem a função de valorizar o texto apresentado em todos os Campos de Texto para cadeia vazia (" "). Dessa maneira as caixas de textos ficarão vazias após o clique do botão.

Para a inclusão de registro no banco, o método da Listagem 31 deve ser executado.

```
1. private void incluirRegistro() {  
2.     Aluno alunos = new Aluno();  
3.     aluno.setCodigo( Integer.parseInt( tfCod.getText() ) );  
4.     aluno.setNome( tfNome.getText() );  
5.     aluno.setEndereco( tfEndereco.getText() );  
6.     aluno.setCoeficiente( Double.parseDouble( tfCoeficiente.getText() ) );  
7.  
8.     EntityManagerFactory emf =  
9.         Persistence.createEntityManagerFactory("GerenciamentoAlunoPU");  
10.    EntityManager em = emf.createEntityManager();  
11.  
12.    em.getTransaction().begin();  
13.    try {  
14.        em.persist(aluno);  
15.  
16.        em.getTransaction().commit();  
17.  
18.        iniciarConexaoBanco();  
19.  
20.        javax.swing.JOptionPane.showMessageDialog( this,  
21.            "Inclusão realizada com sucesso" );  
22.  
23.    }catch (Exception e) {  
24.        javax.swing.JOptionPane.showMessageDialog( this,  
25.            "Erro na Inclusão: " + e.getMessage() );  
26.        em.getTransaction().rollback();  
27.    }finally {  
28.        em.close();  
29.    }  
30.  
31. }
```

#### Listagem 31: Inclusão de registros na tabela aluno

O método inicia com a instanciação do objeto de entidade aluno (linha 2), o qual é valorizado com o conteúdo digitado pelo usuário nos campos de textos (linhas 3 a 6).

Após a valorização, são instanciados os objetos para manipulação do banco de dados (linhas 8 a 10). Para fazer a inclusão de um registro é iniciada uma nova transação (linha 12) sendo realizada na sequência a persistência do registro (linha 14). Ocorrendo tudo bem, a transação é encerrada (linha 16), é apresentado na tela o primeiro registro (linha 18) e também uma mensagem de sucesso (linha 20), sendo a conexão finalizada (linha 28); caso contrário é exibida uma mensagem de erro para o usuário (linha 24) e as modificações no banco são desfeitas (linha 26).



**Atenção:** É aconselhável fazer as instanciações dos objetos de acessos à base de dados a cada manipulação do banco; isso garante a integridade dos dados refazendo uma nova conexão, caso ocorra algum problema no banco de dados.

Já o método para alterar dados de um registro é apresentado na Listagem 32.

```
1. private void alterarRegistro() {
2.     Aluno aluno = (Aluno) listaRegistros.get( id );
3.
4.     aluno.setCodigo( Integer.parseInt( tfCod.getText() ) );
5.     aluno.setNome( tfNome.getText() );
6.     aluno.setEndereco( tfEndereco.getText() );
7.     aluno.setCoeficiente( Double.parseDouble( tfCoeficiente.getText() ) );
8.
9.     EntityManagerFactory emf =
10.        Persistence.createEntityManagerFactory("GerenciamentoAlunoPU");
11.     EntityManager em = emf.createEntityManager();
12.     em.getTransaction().begin();
13.
14.     try {
15.         Aluno alunoAux = (Alunos) em.merge( aluno );
16.         em.persist(alunoAux);
17.
18.         em.getTransaction().commit();
19.
20.         iniciarConexaoBanco();
21.
22.         javax.swing.JOptionPane.showMessageDialog( this,
23.             "Alteração realizada com sucesso" );
24.
25.     }catch (Exception e) {
26.         javax.swing.JOptionPane.showMessageDialog( this,
27.             "Erro na Alteração: " + e.getMessage() );
```

```
28.     em.getTransaction().rollback();
29. }finally {
30.     em.close();
31. }
32.
33. }
```

#### Listagem 32: Alteração de registros na tabela aluno

A estrutura do método alterarRegistro é semelhante ao método incluirRegistro. Existem basicamente duas diferenças, a primeira está no fato do método alterarRegistro recuperar o registro da listaRegistro com o id atual, sendo este registro armazenado no objeto alunos. Na sequência o objeto aluno é valorizado com o conteúdo digitado nos componentes campos de texto, são instanciadas as classes de conexão e então existe a segunda diferença com relação ao incluir registro, pois para se alterar um registro é necessário instanciar um objeto Aluno auxiliar, esse vai receber o conteúdo do método merge() – linha 15, sendo que na sequência o objeto alterado será persistido – linha 16. Não ocorrendo erros será apresentada uma mensagem de sucesso.

A Listagem 33 apresenta o método responsável pela exclusão de um registro.

```
1. private void excluirRegistro() {
2.     Aluno aluno = (Aluno) listaRegistros.get( id );
3.
4.     EntityManagerFactory emf =
5.         Persistence.createEntityManagerFactory("GerenciamentoAlunoPU");
6.
7.     EntityManager em = emf.createEntityManager();
8.     em.getTransaction().begin();
9.
10.    try {
11.
12.        Aluno alunoAux = (Aluno) em.merge( aluno );
13.        em.remove(alunoAux);
14.
15.        em.getTransaction().commit();
16.
17.        iniciarConexaoBanco();
18.
19.        javax.swing.JOptionPane.showMessageDialog( this,
20.            "Exclusão realizada com sucesso" );
21.    }catch (Exception e) {
```

```

22.     javax.swing.JOptionPane.showMessageDialog( this,
23.             "Erro na Exclusao: " + e.getMessage() );
24.             em.getTransaction().rollback();
25.         }finally {
26.             em.close();
27.         }
28.
29.     }

```

#### Listagem 33: Exclusão de registros na tabela aluno

Para a exclusão de um registro, ele é recuperado a partir do seu id (linha 2), sendo instanciados os objetos de conexão com o banco (linhas 4 a 8), e na sequência, assim como para alterar um registro, para excluir é necessário fazer o marge das informações (linha 12) para só então realizar a exclusão do registro (linha 13). Não ocorrendo erros, uma mensagem de sucesso é apresentada ao usuário.

Para finalizar o aplicativo, é apresentada na Listagem 34 o código para a rotina de pesquisar.

```

1. private void pesquisarRegistro() {
2.
3.     String nome = javax.swing.JOptionPane.showInputDialog(this,
4.             "Digite um nome para pesquisa" );
5.
6.     EntityManagerFactory emf =
7.             Persistence.createEntityManagerFactory("GerenciamentoAlunoPU");
8.
9.     EntityManager em = emf.createEntityManager();
10.
11.
12.    javax.persistence.Query query = em.createQuery(
13.            "select a from Aluno a where a.nome = :nome" );
14.    query.setParameter("nome", nome );
15.
16.    try {
17.        Aluno aluno = (Aluno) query.getSingleResult();
18.
19.
20.        javax.swing.JOptionPane.showMessageDialog( this, "Resultado: \n\n" +
21.                "Cod: " + aluno.getCodigo() + "\n" +
22.                "Nome: " + aluno.getNome() + "\n" +
23.                "Endereço: " + aluno.getEndereco() + "\n" +

```

```
24.     "Coeficiente: " + aluno.getCoeficiente() );
25.
26. }catch( javax.persistence.NoResultException e ) {
27.     javax.swing.JOptionPane.showMessageDialog( this,
28.         "Registro não encontrado" );
29. }
30. }
```

#### Listagem 34: Método para pesquisar um registro no banco a partir do seu nome

Na linha 3 é apresentada ao usuário uma caixa de texto, a qual solicita um nome para pesquisa. Após são instanciados os objetos de conexão com o banco (linhas 6 e 9) e um objeto query (linha 12), o qual possui uma sintaxe SQL para pesquisar registro que possuam um determinado nome. Na sequência é valorizado o parâmetro nome com o nome digitado pelo usuário (linha 14).

Após, é executado o comando getSingleResult (linha 17), que retorna um registro, sendo este armazenado em uma instância de Aluno. Para finalizar, é exibida uma mensagem para o usuário com as informações do registro recuperado pela consulta. Caso não exista registro para o argumento informado pelo usuário, é apresentada a mensagem da linha 27.

Agora deve-se associar a chamada dos métodos aos cliques dos botões. Para isso deve-se clicar duas vezes sobre os botões de incluir, alterar, excluir e pesquisar, associando os códigos, como apresentado na Listagem 35.

```
1. private void btIncluirActionPerformed(java.awt.event.ActionEvent evt) {
2.     incluirRegistro();
3. }
4.
5. private void btAlterarActionPerformed(java.awt.event.ActionEvent evt) {
6.     alterarRegistro();
7. }
8.
9. private void btExcluirActionPerformed(java.awt.event.ActionEvent evt) {
10.    excluirRegistro();
11. }
12.
13. private void btPesquisarActionPerformed(java.awt.event.ActionEvent evt) {
14.     pesquisarRegistro();
15. }
```

#### Listagem 35: Chamada dos métodos através dos botões

Ao se executar o aplicativo pela primeira vez, pode ser solicitado o nome da classe principal, ou seja, a classe que implementa o método public static void main(), nessa situação deve-se escolher a classe TelaPrincipal e pressionar o botão Ok, conforme Figura 71.

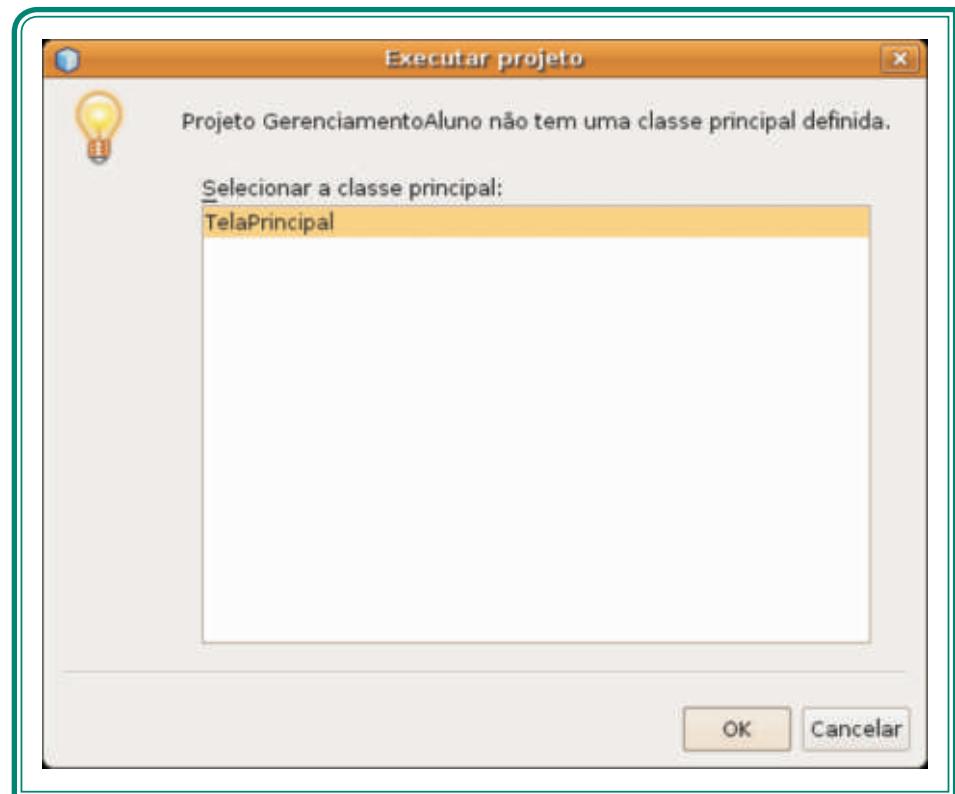


Figura 71: Escolha da classe principal

Após o programa é executado, tendo o usuário acesso às operações de navegação, pesquisa e manutenção do banco de dados, como apresentado na Figura 72.

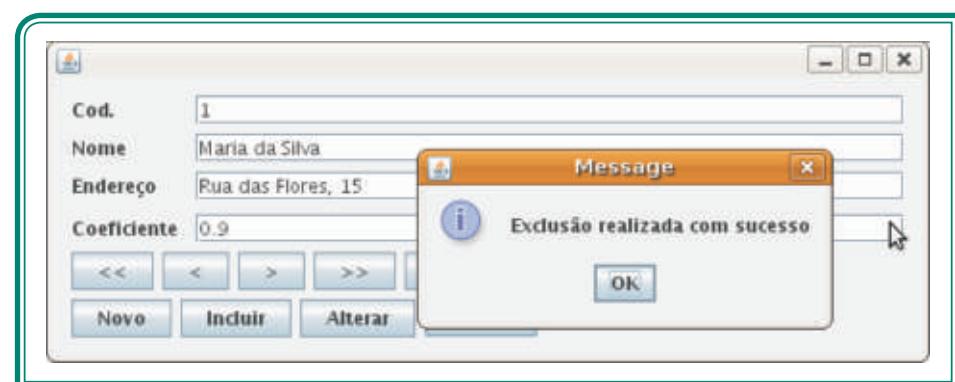


Figura 72: Programa GerenciamentoAluno sendo executado

## Resumindo

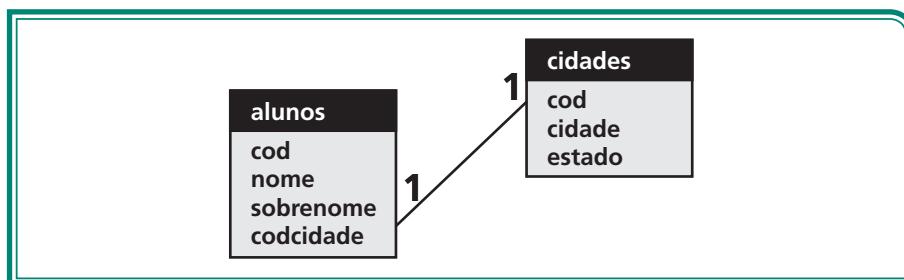
Esta aula apresentou um modelo para acesso a banco de dados utilizando a IDE NetBeans. Inicialmente utilizou-se as ferramentas para gerenciamento do SGBD, o qual acessou informações do Apache Derby.

Na sequência foi desenvolvido um cadastro de aluno utilizando o wizard do NetBeans, e para finalizar o capítulo foi apresentada a sintaxe dos principais comandos referentes ao banco de dados: comandos de navegação, incluir, alterar, excluir e pesquisar registros.

## Atividades de Aprendizagem:



- 1)** Desenvolver um cadastro de cliente, criando um banco de dados com o nome de cadastro no Apache Derby e duas tabelas, conforme abaixo:



- 2)** Desenvolver um aplicativo Desktop utilizando para cadastrar informações na tabela alunos.
- 3)** Desenvolver um aplicativo sem wizard para gerenciar as informações da tabela alunos. O aplicativo deve possuir os botões Novo, Salvar, Excluir e Pesquisar.



# Aula 6 - Desenvolvendo relatórios no Netbeans

## Objetivos

- Apresentar o plugin do iReport para o desenvolvimento de relatórios no NetBeans;
- Desenvolver e personalizar relatórios;
- Apresentar na tela e imprimir os relatórios gerados.

A aula é composta pelos seguintes tópicos:

- 6.1** Desenvolvendo relatórios no NetBeans
- 6.2** Obtendo e instalando o plugin do iReport para o NetBeans
- 6.3** Criando relatórios para utilização em aplicações desktop
- 6.4** Personalizando Relatórios
  - 6.4.1** Estrutura de um relatório
  - 6.4.2** Paleta de componentes de relatório
  - 6.4.3** Propriedades dos componentes de um relatório
- 6.5** Chamando o relatório desenvolvido

## 6.1 Desenvolvendo relatórios no NetBeans

Uma das características das aplicações comerciais é a impressão de relatórios. Esses relatórios podem conter dados simples, como a listagem dos registros existentes em uma tabela de banco de dados, ou um conjunto de dados complexos, envolvendo algoritmos de mineração de dados onde são utilizadas informações referentes a várias tabelas.

De qualquer maneira é muito importante o uso de relatórios em aplicações comerciais, pois são através deles que o usuário poderá imprimir e apresentar/enviar os dados para outras pessoas. Para o desenvolvimento de relatórios, um ponto importante é a escolha de uma boa ferramenta para elaboração dos mesmos. Para a IDE NetBeans uma das melhores opções é o iReport.

O iReport é um programa Open Source capaz de criar visualmente os mais complexos relatórios para aplicações Java. Essa ferramenta possui uma interface gráfica intuitiva, onde o desenvolvedor pode criar praticamente qualquer tipo de relatório de forma rápida e simples. Internamente, o iReport cria um arquivo no formato XML para armazenar a estrutura do relatório, entretanto o usuário não precisa manipular estas informações.

Atualmente existem duas maneiras de trabalhar com o iReport: através de um plugin que pode ser instalado no NetBeans ou através de uma aplicação standalone. Este documento apresenta a utilização do plugin do NetBeans, o qual facilita o processo de desenvolvimento de relatórios e a integração com programas desenvolvidos nessa IDE.

## 6.2 Obtendo e instalando o plugin do iReport para NetBeans

Para obter o plugin do iReport, é necessário acessar a página de plugins para a IDE NetBeans: <http://plugins.netbeans.org>. Essa página dispõe de um sistema de busca onde pode ser informado o nome do plugin ou o recurso desejado. Nessa situação deve ser informado iReport na tela de busca - Figura 73.

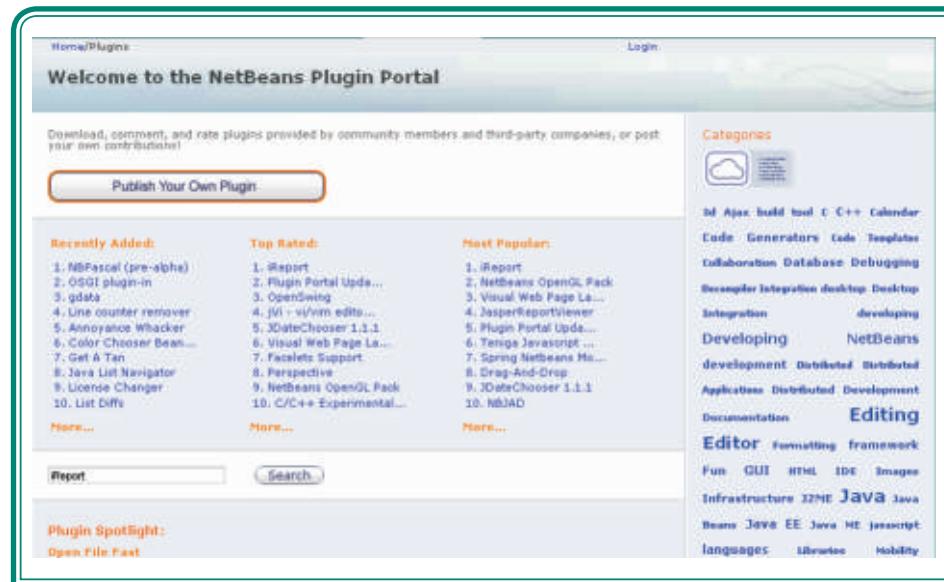


Figura 73: Página para pesquisa de plugins para a IDE NetBeans

Na tela de resultado é apresentado o plugin do iReport, sendo necessário baixar o arquivo através do botão Download – Figura 74.

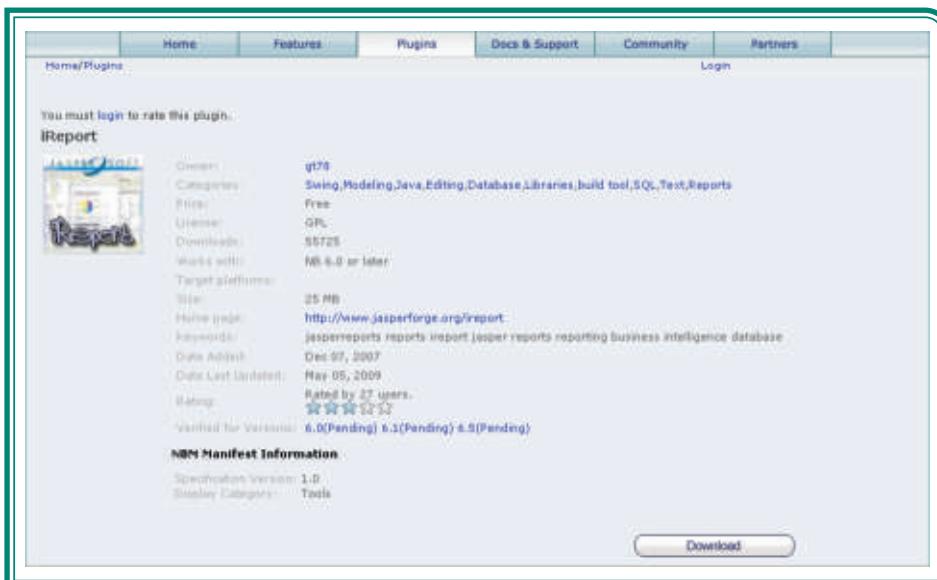


Figura 74: Download do plugin iReport para o NetBeans

Os plugins para o NetBeans comumente possuem a extensão .nbm. Após o download, o arquivo deve ser descompactado (originalmente ele possui a extensão zip) e deve-se instalá-lo no NetBeans, através do menu Ferramentas – Plugins, escolhendo-se a categoria Baixados e clicando no botão Adicionar Plug-ins...

Na tela de seleção deve-se escolher o arquivo iReport-nb-3.5.1.nbm, o qual se encontrava no arquivo compactado (zip) e confirmar no botão Open. Na sequência seleciona-se o botão de Instalar – Figura 75.

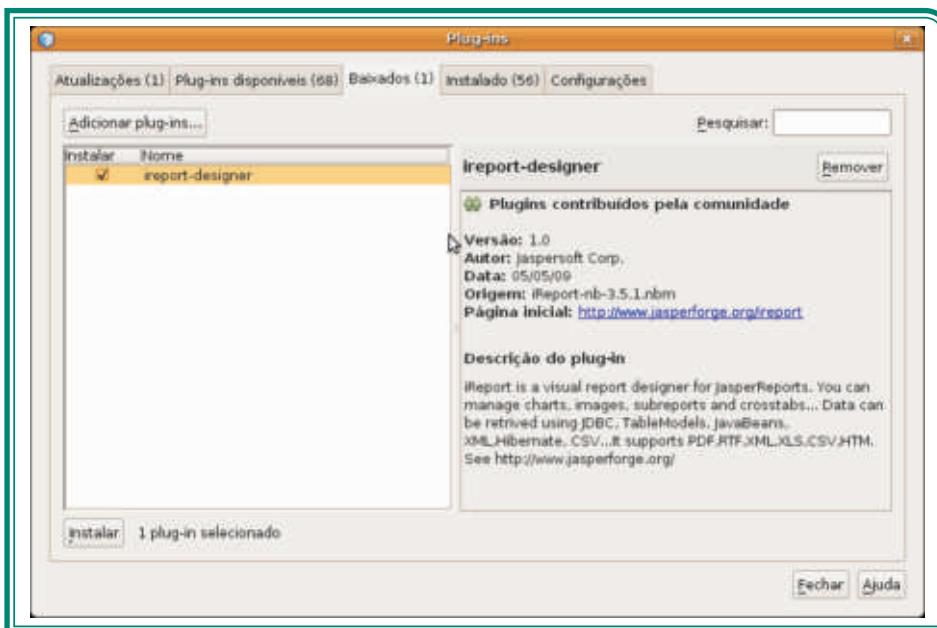


Figura 75: Tela para inclusão de novos plugins na IDE NetBeans

Ao clicar no botão Instalar, a caixa de diálogo NetBeans IDE Instaler surgirá. Clica-se em Próximo para prosseguir a instalação. Na etapa seguinte deve-se aceitar os termos de licença clicando na opção Eu aceito os termos no contrato de todas as licenças e confirma-se clicando em Instalar.

A caixa de diálogo Validar surgirá, dizendo que o plugin a ser instalado não foi assinado pela NetBeans. Apesar da mensagem, o plugin foi desenvolvido e vem sendo testado a um longo tempo, logo não existe problema em seguir a instalação clicando no botão Continuar.

Ao terminar a instalação (botão Finalizar da última tela), será redirecionado para a caixa de diálogo Plug-ins. Para retornar ao ambiente de desenvolvimento, clica-se no botão Fechar.

### 6.3 Criando relatórios para utilização em aplicações Desktop

Para apresentar o desenvolvimento de relatórios em aplicações desktop, será utilizado como base o aplicativo desenvolvido na seção 5.4 – GerenciamentoAluno, sendo desenvolvido um relatório simples com os dados de todos os alunos existentes na tabela.

Antes de criar um novo relatório, é necessário criar um pacote para armazenar os arquivos referentes aos relatórios. Clica-se com o botão direito do mouse sobre o projeto GereciamentoAlunos, escolhendo a opção Novo – Pacote Java. Para o pacote dá-se o nome de relatorios – Figura 76. Para concluir clica-se no botão Finalizar.

Depois de definido o pacote, deve-se clicar com o botão direito sobre este, escolhendo a opção Novo – Outros.... Na tela apresentada deve-se escolher a categoria Reports e o tipo de arquivo ReportWizard - Figura 77.

Na tela seguinte será solicitado o nome do relatório, bem como a pasta onde será salvo – Figura 78.

Na sequência são solicitadas as informações referentes à conexão com o banco de dados. Uma nova conexão pode ser estabelecida através do botão New. Para o exemplo será utilizada a mesma conexão do aplicativo GerenciamentoAlunos.

No wizard de conexão apresentado após o clique no botão New, para Data-source escolhe-se NetBeans Database JDBC connection, na tela seguinte é

solicitado um nome para a conexão (para o exemplo utiliza-se escola) e qual conexão deverá ser utilizada (utiliza-se jdbc:derby://localhost:1527/escola). Para testar a conexão com o banco clica-se no botão Test. Se a configuração foi bem sucedida, é apresentada a mensagem Connection Test Successful – Figura 79.

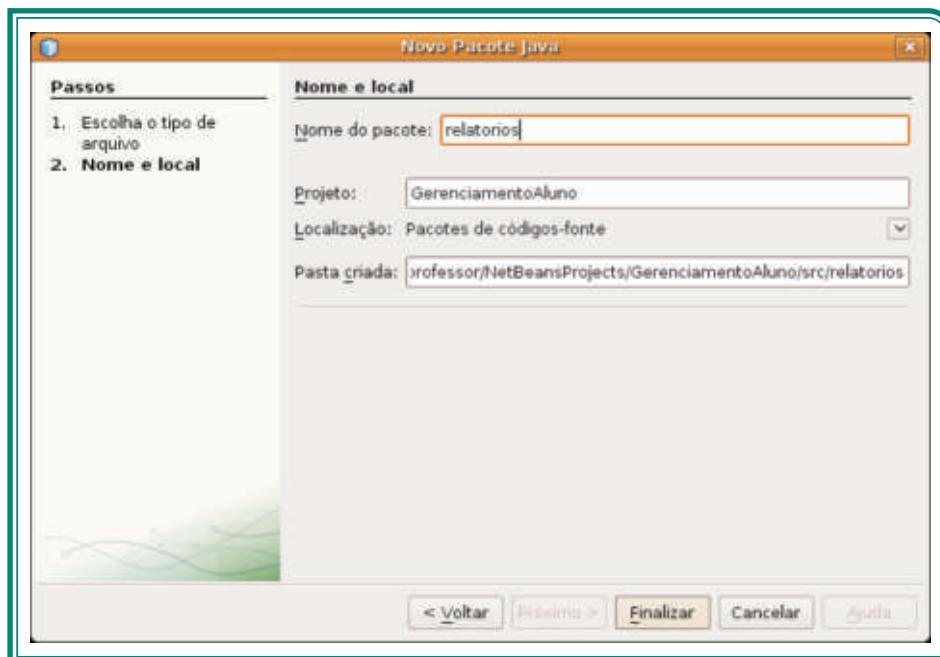


Figura 76: Definição de um pacote para armazenar os códigos referentes ao relatório

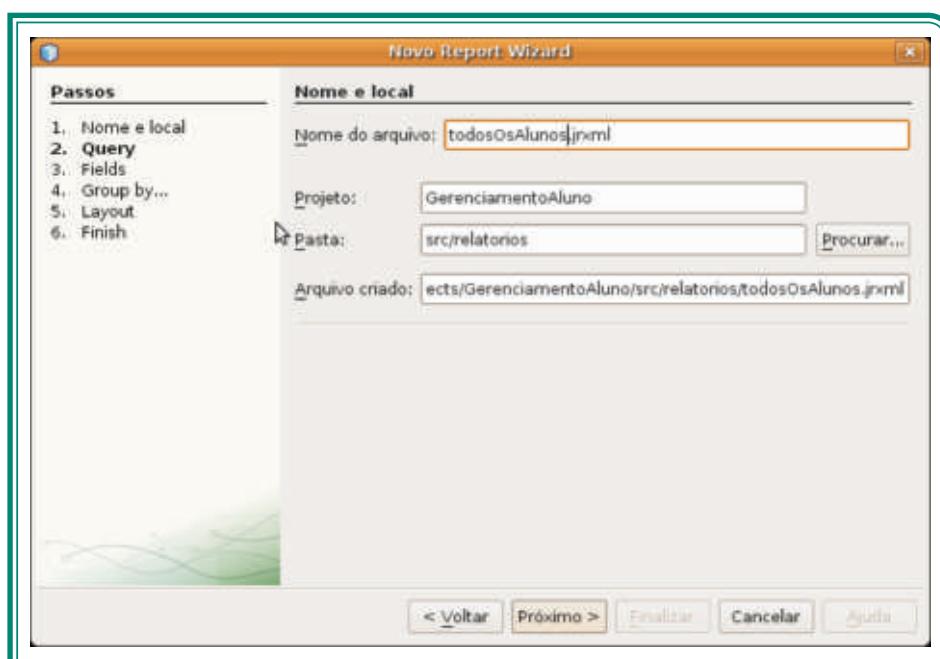


Figura 77: Escolha do wizard para confecção do relatório

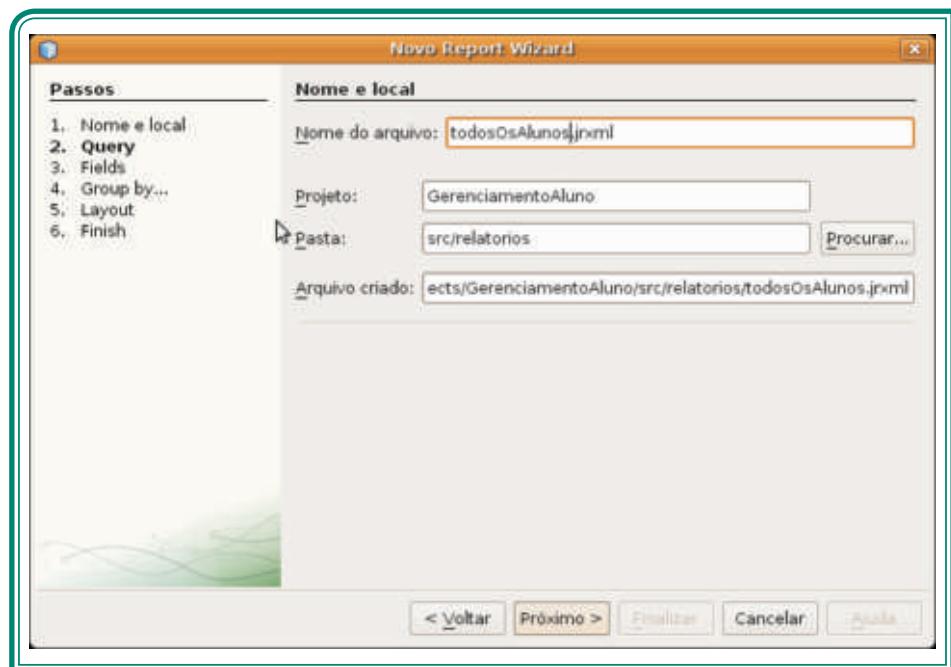


Figura 78: Escolha do wizard para confecção do relatório

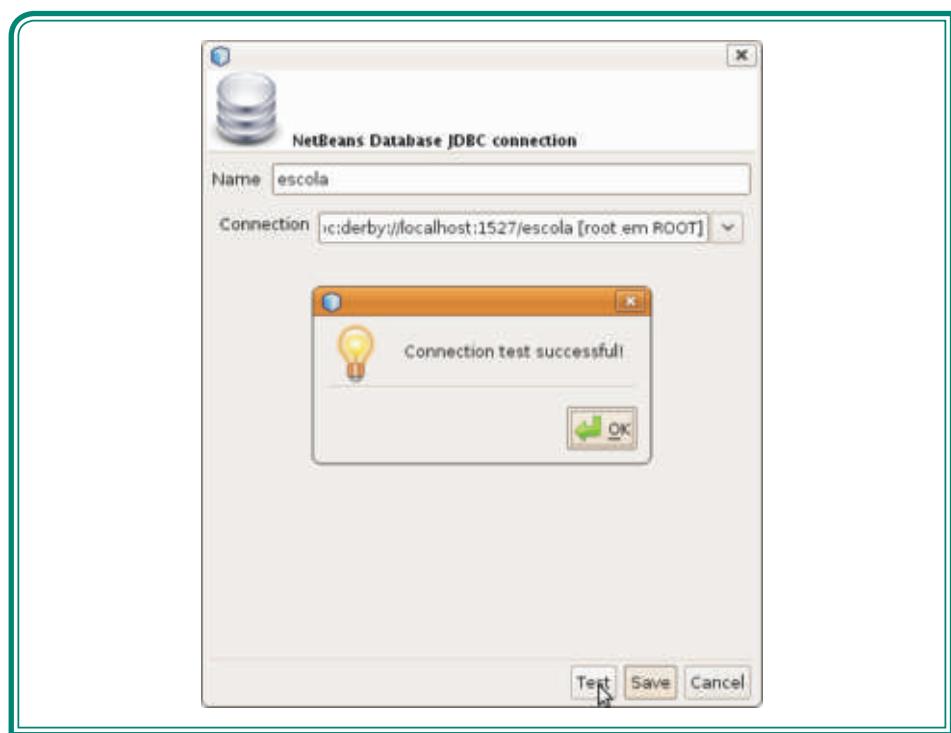


Figura 79: Wizard para realizar a conexão com o banco de dados

Após confirmar a tela de conexão bem sucedida, é necessário salvar as informações (botão Save), sendo apresentada uma tela para realizar a seleção das informações que serão apresentadas no relatório. Para a pesquisa pode

ser utilizada uma sintaxe SQL Simples – Figura 80, ou um assistente para o desenvolvimento de pesquisas SQL – Figura 81.

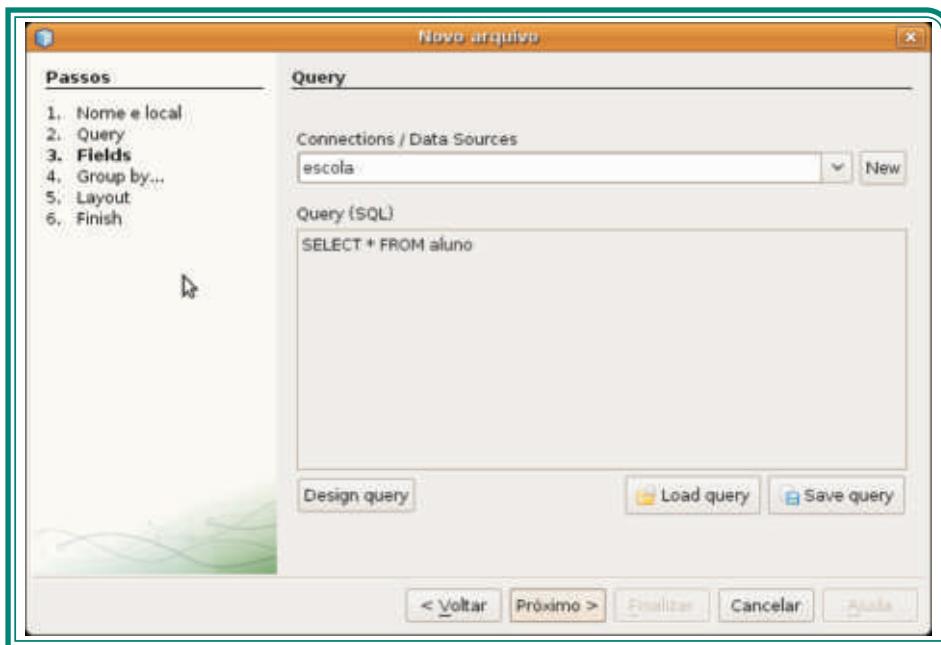


Figura 80: Utilização de um comando SQL simples para pesquisa

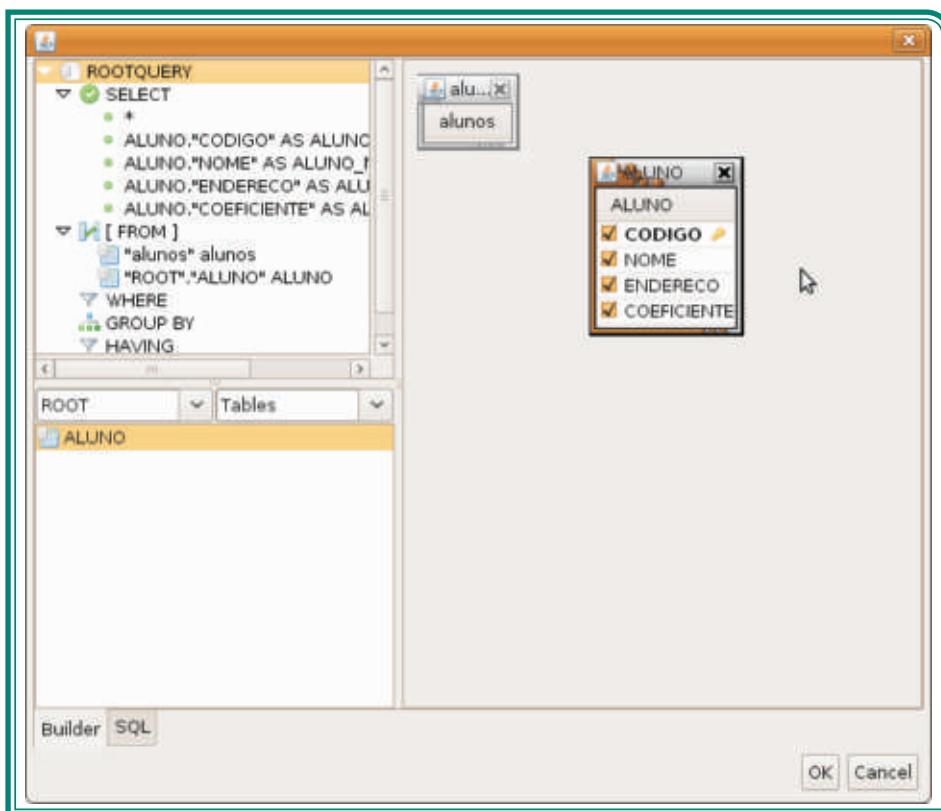


Figura 81: Utilização do wizard para desenvolvimento de sintaxes SQL

Para o exemplo será optada pela primeira opção, sendo utilizado um comando SQL para retornar todos os registros da tabela alunos - Figura 80.

Na próxima tela será apresentada a opção de seleção dos campos para o relatório - Figura 82.

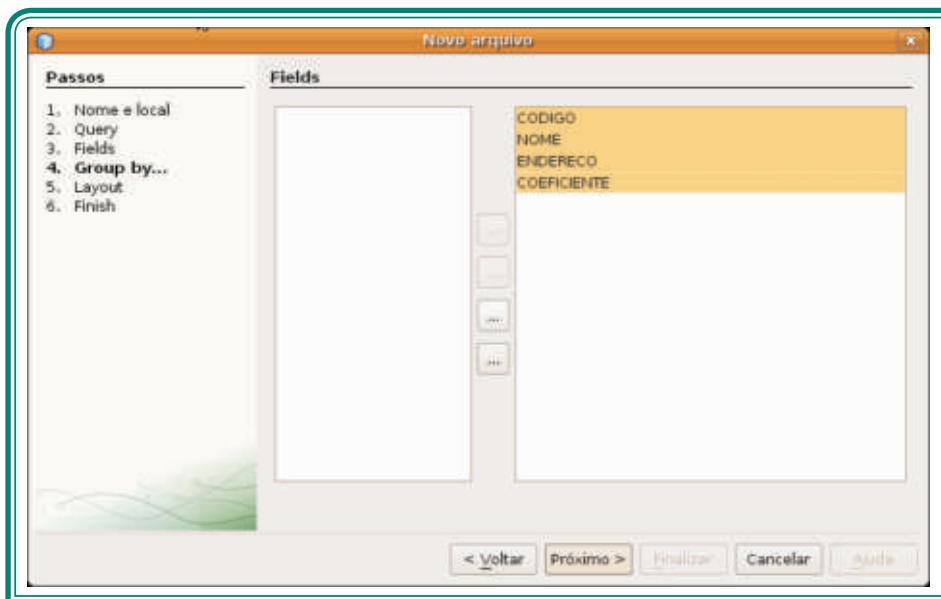


Figura 82: Utilização do wizard para desenvolvimento de sintaxes SQL

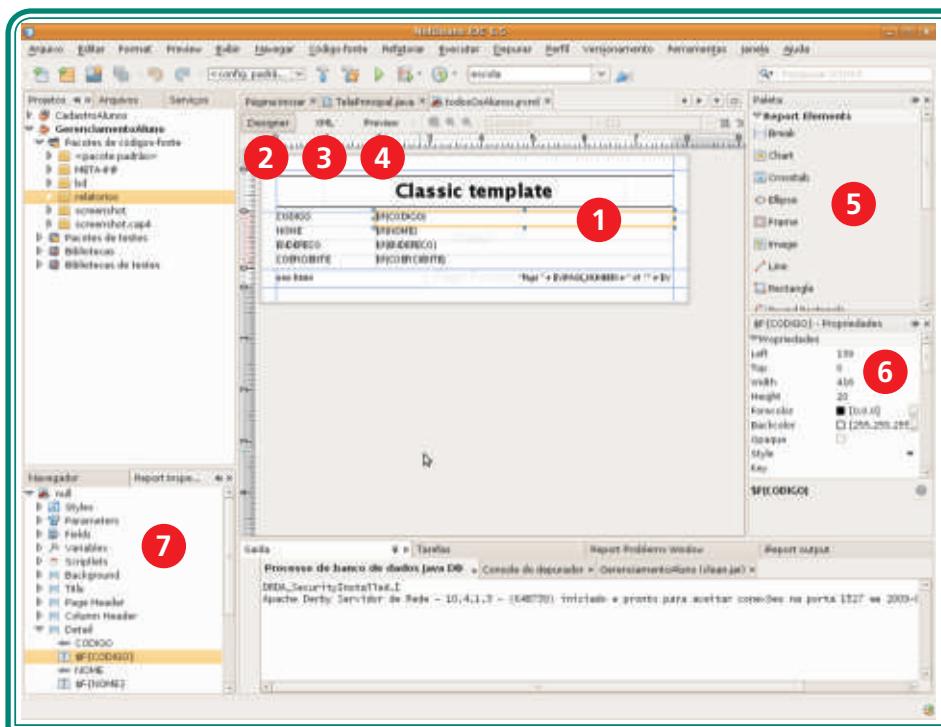


Figura 83: Tela para o desenvolvimento de relatório

Na sequência é possível personalizar o agrupamento de dados (ex. apresentar a quantidade de alunos que possuem o mesmo coeficiente). Para o exemplo proposto não haverá agrupamento (nenhum campo deve ser selecionado). Na última tela solicita-se um layout para o relatório, o qual pode ser Coluna (um campo abaixo do outro) ou Tabular (um registro abaixo do outro). Será escolhido o layout Coluna, sendo concluído o assistente no botão Finalizar.

O relatório então é apresentado na tela, conforme Figura 83.

## 6.4 Personalizando Relatório

Conforme apresentado na Figura 83, o relatório pode ser personalizado de forma visual em (1). É possível mudar a forma de visualização do relatório, podendo este ser Designer (2), XML (3), o qual apresenta o código XML referente ao relatório ou ter uma pré-visualização do relatório em (4).

**Atenção:** É necessário permanecer por alguns segundos no modo de visualização Preview, sendo que nesse modo é gerado o arquivo com a extensão .jasper, arquivo este utilizado para apresentar o relatório. Em alguns casos essa geração pode ser lenta, para forçar a geração, após escolher o modo Preview deve-se clicar com o botão direito sobre o projeto GerenciamentoAluno e escolher a opção Limpar e Construir. Assim o arquivo .jasper é gerado – Figura 84.

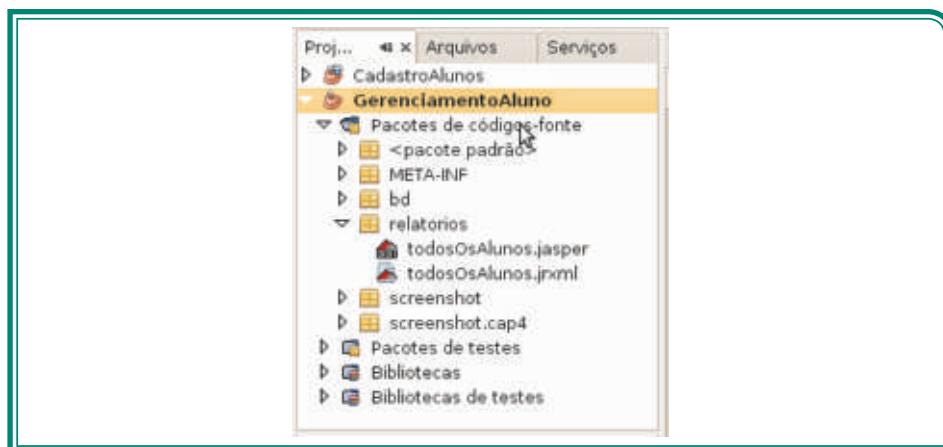


Figura 84: Geração do arquivo .jasper

Para personalizar o relatório, é possível clicar e arrastar componentes da paleta Report Elements (5) até a tela (1) em modo Designer. Para modificar a propriedade de um componente utiliza-se a janela de propriedades (6). Outros elementos também podem ser adicionados através da janela Report Inspector (7), como funções matemáticas, estilos, planos de fundos, etc.

### 6.4.1 Estrutura de um relatório

Um relatório é formado por uma ou mais bandas. As bandas são faixas do relatório onde se pode adicionar informações. A princípio existem sete tipos de bandas, como pode ser visualizado na Figura 85.

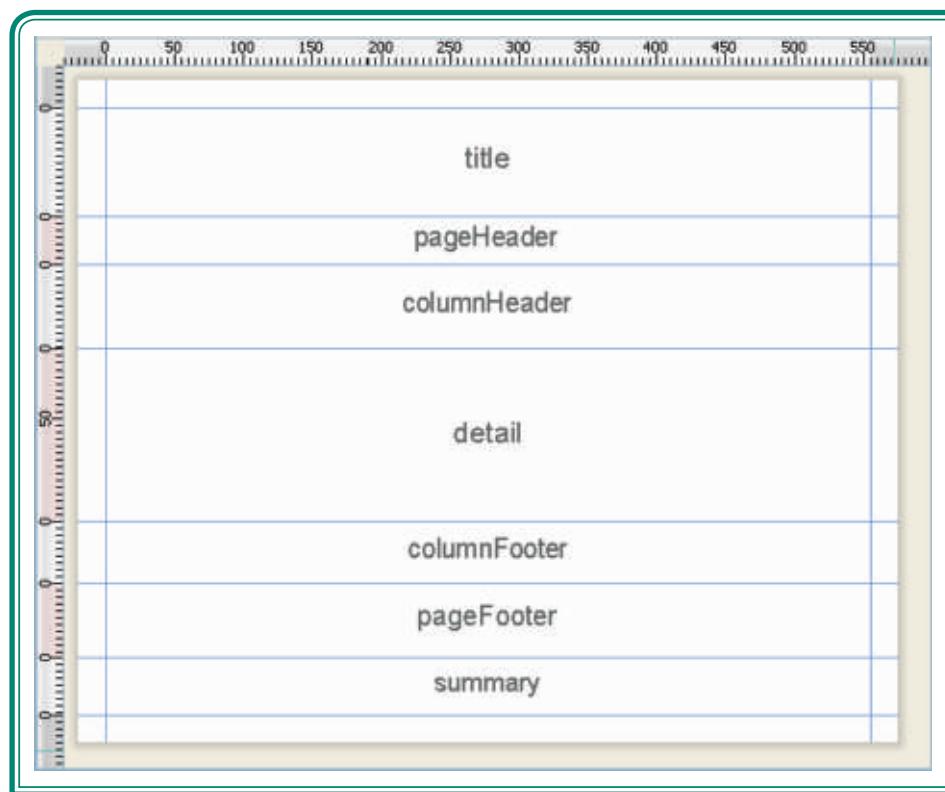


Figura 85: Tipos de bandas para o relatório

Para adicionar uma nova banda, basta clica-las e arrastá-las da janela Report Inspector para a tela do relatório. As bandas têm as respectivas características:

- a) Title: Essa banda corresponde ao título do formulário. A informação adicionada nessa banda (frequentemente são adicionados textos estáticos) é apresentada apenas na primeira página do relatório.
- b) PageHeader: Cabeçalho da página. Essa banda costuma conter o nome do relatório e é apresentada ao topo de todas as páginas do relatório.
- c) ColumnHeader: Cabeçalho de colunas. Essa banda é utilizada para adicionar o nome das colunas, quando se utiliza um relatório do tipo Tabular (um registro abaixo do outro). A informação dessa banda costuma repetir no início de cada página.

- d) Detail:** Essa banda apresenta as informações referentes aos registros. Cada registro da base de dados é apresentado em um Detail distinto.
- e) ColumnFooter:** Rodapé da coluna. É utilizado para apresentar uma informação referente às colunas da tabela, como por exemplo alguns resultados parciais da coluna (soma dos valores, médias, etc.)
- f) PageFooter:** Rodapé da página. É utilizado para apresentar informações como número da página, telefone para contato, etc. Essa informação se repete ao final de cada página.
- g) Summary:** Apresentado apenas na última página do relatório, é utilizado para informar totais do relatório, como a quantidade de registros apresentados, a soma ou a média de uma determinada coluna, etc.

No relatório gerado pelo wizard (Figura 83), foram utilizadas três bandas: Title – formada pelo texto estático “Classic Template”; Detail contém as informações dos registros, sendo formado por quatro textos estáticos (CÓDIGO, NOME, ENDEREÇO e COEFICIENTE) e quatro campos de texto, os quais serão valorizados com os campos da tabela. E, por fim, a banda PageFooter, a qual possui no lado esquerdo uma função que retorna a data do sistema e ao lado direito duas funções, a primeira que retorna o número da página atual e a segunda que retorna o total de páginas do relatório. A Figura 85 apresenta um preview do relatório, onde é possível visualizar as informações existentes nas bandas.

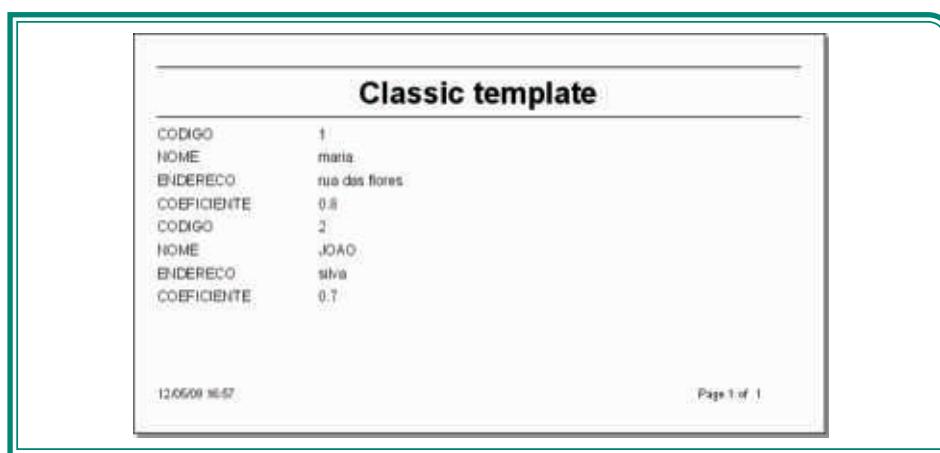
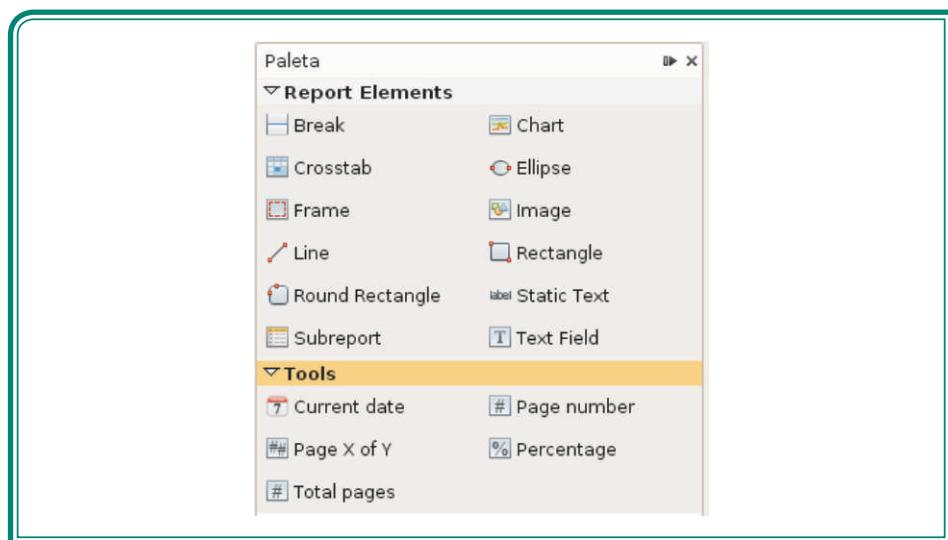


Figura 86: Preview do relatório desenvolvido pelo wizard

## 6.4.2 Paleta de Componentes de relatório

A paleta de componentes de relatório é apresentada na Figura 87.



**Figura 87: Tipos de componentes utilizados em um relatório**

Dentre seus componentes destacam-se:

- a) Chart: Gera um gráfico através de valores existentes na tabela do banco de dados.
- b) Image: Utilizado para exibir imagens no relatório. Estas podem ser imagens dinâmicas (preenchidas por informações existentes no banco de dados) ou estáticas.
- c) Rectangle: Usado para desenhar retângulos ao redor de outro elemento visual, comumente utilizado para criar destaque em informações importantes do relatório.
- d) TextField: Utilizado para criar os campos dinâmicos dos relatórios. É neste elemento que se podem recuperar dados de um determinado campo no banco de dados para exibir para o usuário.
- e) Ellipse: Desenha uma elipse no relatório.
- f) Line: Permite desenhar uma linha. Esse componente é utilizado com frequência para delimitar as bandas.
- g) Static Text: Apresenta um texto estático. Utilizado para criar rótulos ou títulos no relatório.

#### **6.4.3 Propriedades dos componentes de um relatório**

Todo componente visual existente no relatório é formado por propriedades, onde é possível modificar características e seu conteúdo. Para acessar as pro-

priedades de um componente, basta selecioná-lo e acessar a barra de ferramenta de propriedades – Figura 88.



Figura 88: Propriedades do componente visual TextField que corresponde ao campo CODIGO da tabela

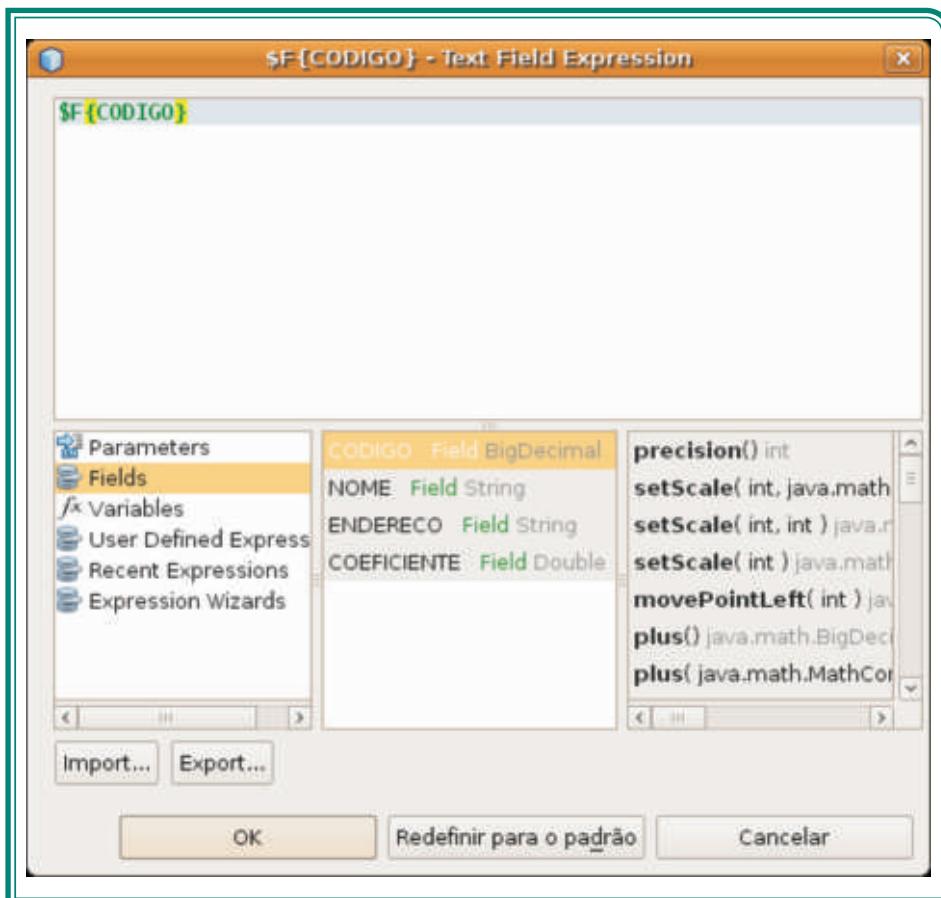


Figura 89: Editor de expressões para um TextField

As propriedades são específicas para cada componente. Por exemplo, um componente Retangle possui a propriedade Pen, no qual pode ser personalizado o formato do contorno, já um componente TextField possui a propriedade TextField Expression, em que é informado o conteúdo que será apresentado por esse componente.

Ao adicionar um novo TextField em uma banda (por exemplo Detail), é necessário acessar o editor de expressões para informar qual será o conteúdo desse campo. O editor de expressões do TextField é apresentado na Figura 89.

No exemplo apresentado, o conteúdo do TextField será o campo NOME do registro correspondente na tabela aluno.

A coluna da esquerda corresponde ao tipo de informação que será apresentada no TextField, essa pode ser:

- a) Parameters:** Parâmetros do relatório, como largura e altura da folha
- b) Field:** Campos da tabela no banco de dados
- c) Variables:** Informações como número de página, número da coluna.
- d) User Defined Expressions:** Expressões definidas pelo usuário. Nesse ponto podem ser utilizados comandos como if.
- e) Recent Expressions:** É apresentada uma lista das últimas expressões utilizadas.
- f) Expression Wizard:** Um assistente para desenvolver expressões.

Na coluna central são apresentados detalhes da coluna selecionada à esquerda. Por exemplo, ao selecionar Fields à esquerda, na coluna central são apresentados os campos existentes no banco de dados.

Na coluna da direita é possível selecionar qual informação da coluna central será apresentado para o usuário no TextField. Por exemplo, é possível exibir apenas a quantidade de caracteres existentes no campo nome, utilizando para isso o método length().

Para selecionar e modificar as características do relatório, também é possível selecionar o item desejado na janela Report Inspector, conforme apresentado na Figura 90.

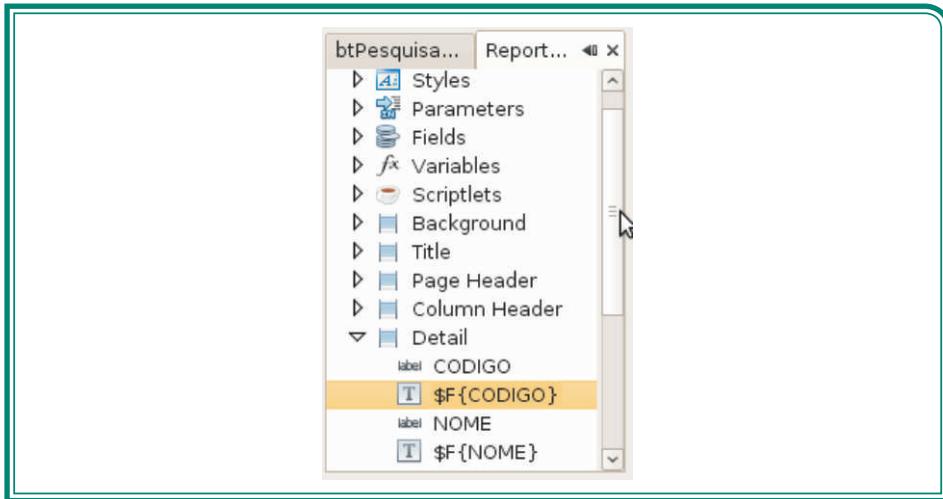


Figura90: Tipos de bandas para o relatório

Na imagem apresentada, é possível visualizar todos os itens das bandas, sendo possível selecionar componentes do relatório nessa janela. Além dos componentes tradicionais de um relatório, estão disponíveis no Report Inspector componentes como Background, onde é possível mudar a cor/formato do fundo do formulário e Styles e definir um novo estilo para o relatório.

## 6.5 Chamando o Relatório desenvolvido

O relatório desenvolvido utiliza algumas bibliotecas adicionais do JasperReport, as quais devem ser adicionadas na categoria Bibliotecas do projeto GerenciamentoAluno.

Para adicionar as bibliotecas adicionais clica-se com o botão direito sobre a categoria Bibliotecas do projeto GerenciamentoAluno e seleciona a opção Adicionar JAR/pasta... na sequência, é selecionada a pasta onde o NetBeans foi instalado (por default é /home/usuário/netbeans-6.5), acessando as subpastas /report/modules/ext. Dentro desta pasta deve-se selecionar os arquivos:

- commons-beanutils-1.7.jar
- commons-collections-3.2.1.jar
- commons-digester-1.7.jar
- commons-logging-1.1.jar
- jasperreports-3.5.1.jar

A Figura 91 apresenta as bibliotecas que devem ser inseridas no projeto. Após adicionar os arquivos .jar do iReport, a estrutura de bibliotecas fica como apresentado na Figura 92.

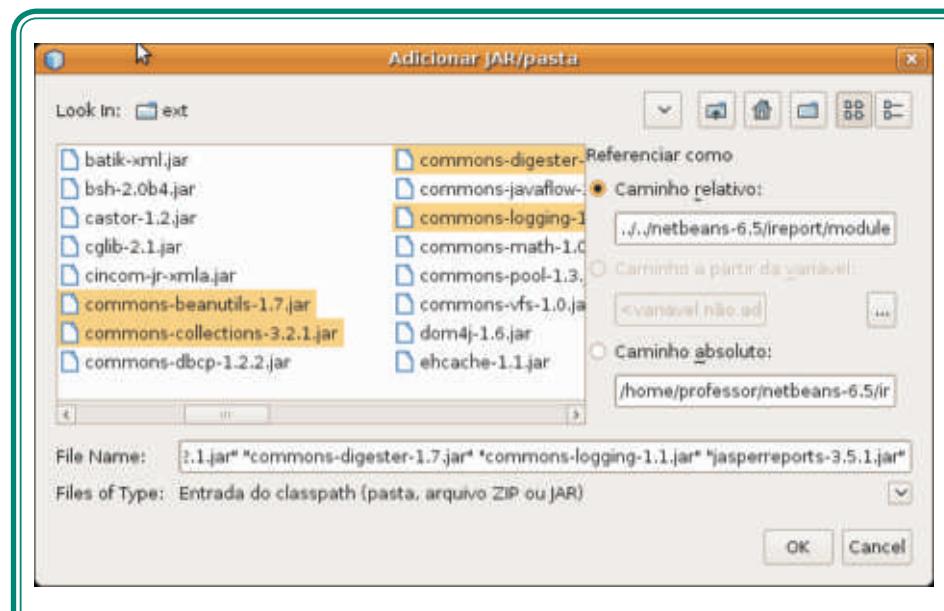


Figura 91: Seleção das bibliotecas utilizadas em tempo de execução para apresentação do relatório

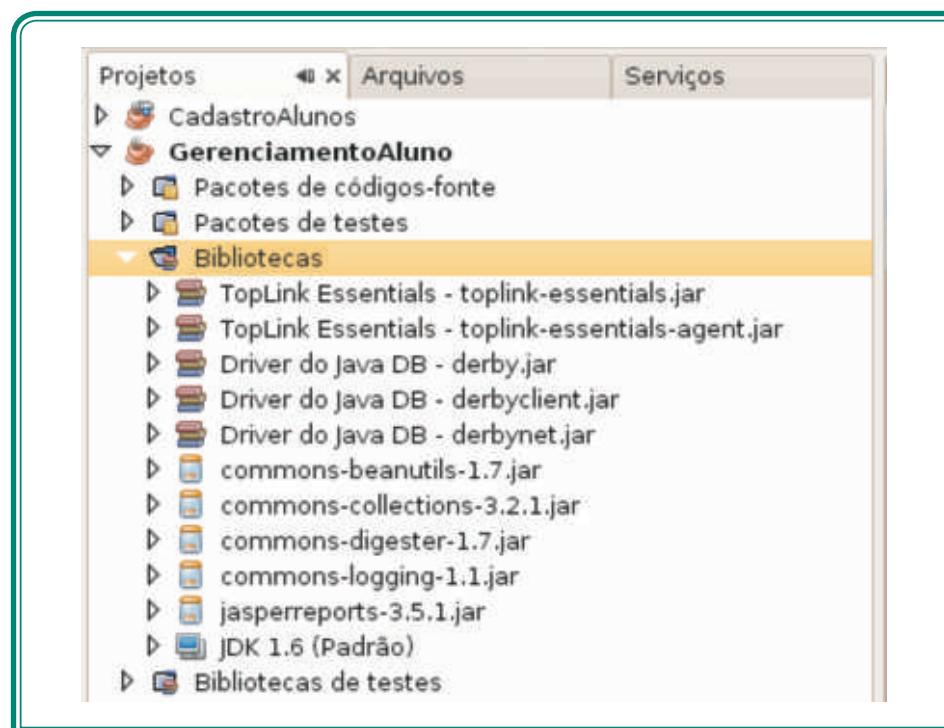


Figura 89: Editor de expressões para um TextField

Para chamar o relatório desenvolvido anteriormente, é necessário adicionar um botão ao projeto GerenciamentoAluno, o qual terá o rótulo “Apresentar Relatório” e o nome de variável btRelatorio – Figura 93.

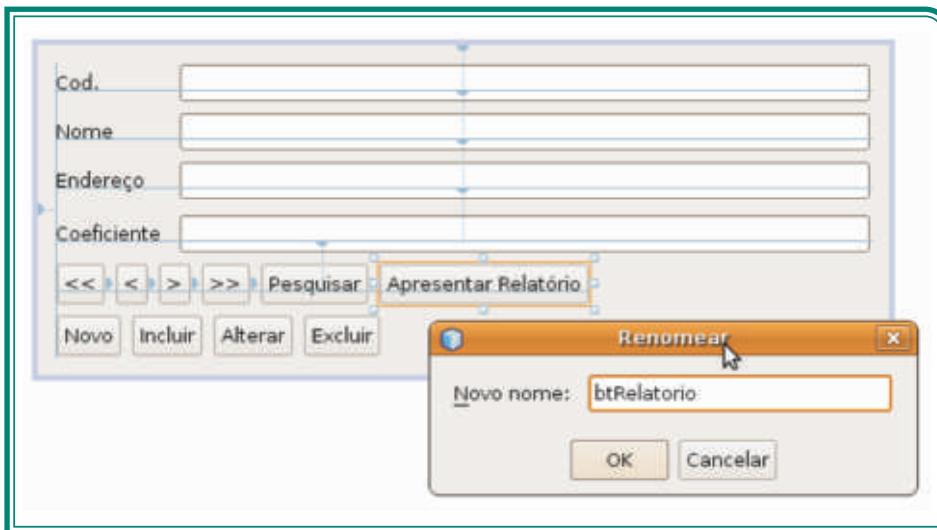


Figura 93: Botão para visualização do relatório

O código para exibir o relatório desenvolvido necessita de algumas classes específicas, essas são apresentadas na Listagem 36. O código para chamar o relatório deve ser associado ao clique do botão btRelatorio, e este é apresentado na Listagem 37.

```
1. import java.io.InputStream;
2. import java.sql.Connection;
3. import java.sql.DriverManager;
4. import java.sql.SQLException;
5. import java.util.HashMap;
6. import net.sf.jasperreports.engine.JRException;
7. import net.sf.jasperreports.engine.JasperFillManager;
8. import net.sf.jasperreports.engine.JasperPrint;
9. import net.sf.jasperreports.view.JasperViewer;
```

#### Listagem 36: Classes utilizadas para apresentar o relatório desenvolvido

No código apresentado, das linhas 1 a 5 são classes do próprio Java, já as classes da linha 6 a 9 são utilizadas pelo iReport, e estão presentes na biblioteca do JasperReport.

```
1. private void btRelatorioActionPerformed(
2.     java.awt.event.ActionEvent evt) {
3.
4.     try {
5.
```

```

6.     String reportName = "relatorios/todosOsAlunos.jasper";
7.
8.     InputStream is = this.getClass().getClassLoader().
9.         getResourceAsStream(reportName);
10.    Class.forName("org.apache.derby.jdbc.ClientDriver");
11.    Connection conn = DriverManager.getConnection(
12.        "jdbc:derby://localhost:1527/escola", "root", "root");
13.    JasperPrint jasperPrint =
14.        JasperFillManager.fillReport(is, new HashMap(), conn );
15.    JasperViewer jv = new JasperViewer(jasperPrint);
16.    jv.setVisible(true);
17.
18. }catch (ClassNotFoundException ex) {
19.     System.out.println( ex.getMessage() );
20. }catch (JRException ex) {
21.     System.out.println( ex.getMessage() );
22. }catch (SQLException ex) {
23.     System.out.println( ex.getMessage() );
24. }
25.
26. }

```

#### Listagem 37: Código responsável pela exibição do relatório

Na linha 06 é declarada uma String contendo o arquivo .jasper do relatório, o qual se encontra no pacote relatórios. Na linha seguinte (linha 08) é recuperado o arquivo .jasper, armazenando-o em uma classe de Stream

Na linha 11 acontece a conexão com o banco de dados, essa conexão será utilizada pelo plugin para impressão dos dados. Na linha 13 acontece a instanciação do objeto que monta o relatório, sendo necessário o arquivo referente ao jasper, um arquivo com as propriedades (nessa situação não existe nenhuma configuração especial para o relatório, por isso o Hash-Map referente às propriedades está vazio) e o objeto com a conexão com o banco.

Para finalizar, a linha 15 instancia um objeto responsável pela visualização do relatório e na linha 16 o relatório é apresentado com o comando setVisible(true). Ocorrendo exceções, as mesmas são tratadas da linha 18 a 24.

Após a codificação do botão btRelatorio, o programa pode ser executado, e ao clicar no botão, uma interface visual para impressão do relatório é apresentado, conforme Figura 94.

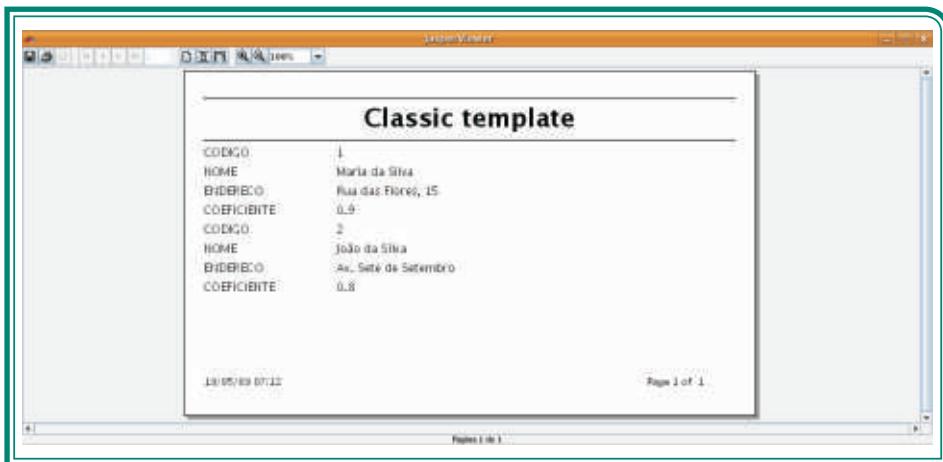


Figura 94: Visualização do relatório

## Resumindo

Esta aula apresentou o desenvolvimento de relatórios comerciais utilizando o plugin do iReport para IDE NetBeans.

Em um primeiro momento foi apresentado o processo de download e instalação de um plugin na IDE NetBeans. Após foi iniciado o desenvolvimento de um relatório utilizando o wizard da ferramenta. Na sequência foi personalizado o relatório, adicionando novos componentes e mudando algumas características visuais, e por fim, foi apresentado os códigos para apresentar o relatório desenvolvido para o usuário através de uma interface visual.

## Atividades de Aprendizagem:



- 1) Modificar o relatório desenvolvido ao longo desse capítulo, para q o mesmo não apresente as informações de forma colunar, e sim de forma tabular (um registro abaixo do outro).
- 2) Adicionar ao relatório desenvolvido a quantidade de registros apresentados em um campo ao final do relatório.
- 3) Apresentar ao final do relatório um campo com a média do coeficiente dos alunos apresentados no relatório.

- 4)** Desenvolver um relatório para a tabela employee, essa existente no banco de dados vir do Apache Derby. Esse relatório deve apresentar todos os campos da tabela, sendo desenvolvido de forma tabular.
- 5)** Criar uma interface visual com um botão, o qual faz a impressão do relatório.

# Aula 7 - Distribuição do aplicativo desenvolvido

## Objetivos

- Apresentar o empacotamento de um aplicativo (arquivo.jar);
- As funcionalidades do Java Runtime Environment;;
- Execução do .jar via console e via ícones;.

**A aula é composta pelos seguintes tópicos:**

**7.1** Arquivo de Instalação

**7.2** Gerando um .jar para projetos do NetBeans

## 7.1 Arquivo de Instalação

Após o desenvolvimento de um aplicativo comercial, o último passo é realizar sua distribuição. Hoje existem várias ferramentas para gerar instaladores padrão wizard, comuns em aplicativos Windows. Dentre eles destacam-se o InstallShield, Launch4J e InnoSetup.

Entretanto, um aplicativo desenvolvido em Java tem a característica de ser multiplataforma, por esse motivo não é interessante gerar instaladores wizard, já que o programa ficará limitado ao sistema operacional Windows.

Dessa maneira, o arquivo executável em Java também não pode ter a extensão .exe, uma vez que essa extensão é utilizada apenas para arquivos executáveis no Windows, já no Linux, por exemplo, o arquivo executável possui como característica uma permissão para execução.

Para tornar o arquivo executável independente do sistema operacional, a plataforma Java fornece um arquivo .jar (Java Archive), podendo este ser executado via console do sistema operacional ou de forma visual mediante um ícone. De qualquer maneira, para executar um arquivo .jar é necessário que

exista no sistema operacional uma JRE (Java Runtime Environment, ou ambiente de execução Java), esse também conhecido como máquina virtual Java, sendo ela quem executa o .jar.

Com isso, um programa desenvolvido em Java para desktop roda em qualquer sistema operacional desktop que possua uma JRE, o que torna o código independente de plataforma.

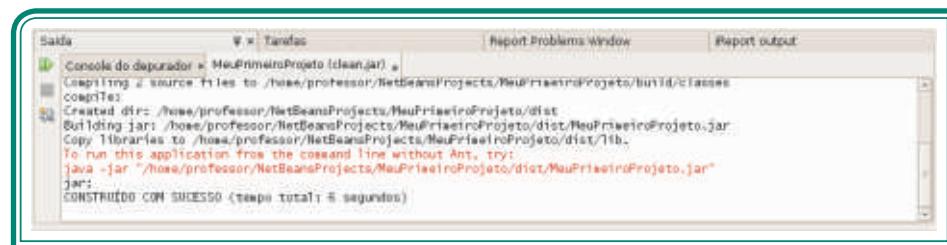
## 7.2 Gerando um .jar para projetos do NetBeans

O arquivo .jar referente ao projeto pode ser gerado pela própria IDE do NetBeans. Para exemplificar o processo, será utilizado um aplicativo simples – MeuPrimeiroProjeto, desenvolvido na seção 3.3 e que implementa uma calculadora simples.

O processo para gerar o instalador de um programa mais complexo, com acesso a banco de dados por exemplo é o mesmo, porém é necessário instalar e configurar o SGBD (ex. apache Derby) na máquina na qual será instalado o programa, sendo que esse processo foge do escopo do presente documento. O site <http://db.apache.org/derby> possui informações sobre o processo de instalação e configuração do apache Derby.

Para gerar o arquivo .jar no NetBeans, basta clicar com o botão direito sobre o projeto desejado (ex. MeuPrimeiroProjeto) e escolher a opção Limpar e Construir, dessa maneira é gerado um .jar para o projeto.

Após gerar o .jar, é apresentada uma mensagem de sucesso na console de saída, bem como o local onde o arquivo .jar foi gerado – Figura 95.



The screenshot shows the NetBeans IDE's 'Console' tab. The output window displays the following text:

```
Saída  Tarjetas  Reporte Problemas  Reporte de salida
Console do depurador < MeuPrimeiroProjeto (clean.jar) >
Compiling 6 source files to /home/professor/NetBeansProjects/MeuPrimeiroProjeto/dist/classes
classes
Created dir: /home/professor/NetBeansProjects/MeuPrimeiroProjeto/dist
Building jar: /home/professor/NetBeansProjects/MeuPrimeiroProjeto/dist/MeuPrimeiroProjeto.jar
Copy libraries to /home/professor/NetBeansProjects/MeuPrimeiroProjeto/dist/lib
To run this application from the command line without Ant, try:
java -jar "/home/professor/NetBeansProjects/MeuPrimeiroProjeto/dist/MeuPrimeiroProjeto.jar"
jar:
CONSTRUIDO COM SUCESSO (tempo total: 6 segundos)
```

Figura95: Console com informações sobre o arquivo .jar

Na console de saída também é apresentada a sintaxe para executar o aplicativo via console:

```
java -jar caminho-do-arquivo/MeuPrimeiroProjeto.jar
```

Como pode ser observado no comando anterior, quem executa o .jar não é o sistema operacional, e sim a JRE; dessa maneira o aplicativo Java só poderá executar em máquinas que já possuem esse recurso.

A diferença básica entre o JDK e o JRE é que se utiliza o primeiro para o desenvolvimento de aplicações Java, o próprio NetBeans possui internamente o JDK, utilizando-o para fazer a compilação e a debugação das classes. Já o JRE é utilizado apenas para a execução de aplicativos, por isso seu tamanho é bastante reduzido.

A JRE está disponível para download na url <http://java.sun.com/javase/downloads>, onde é apresentada a opção Java SE Runtime Environment – Figura 96.



Figura 96: Tela para download do JRE

Na tela seguinte é solicitado a plataforma para o JRE e o idioma, conforme Figura 97.

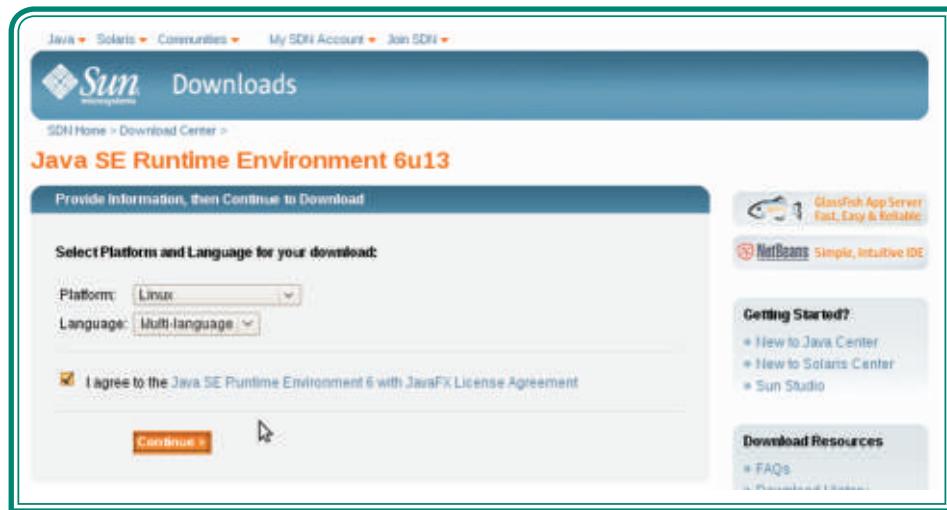


Figura 97: Seleção da plataforma e da linguagem do JRE

Após o download e a instalação do JRE, já é possível executar o aplicativo gerado pelo NetBeans, bastando digitar na console o comando apresentado na Figura 98 e o programa é executado.

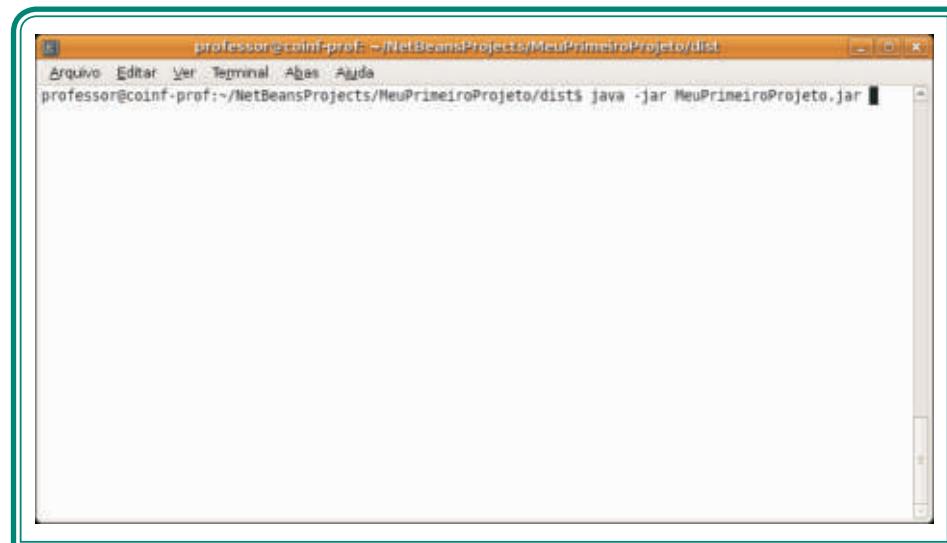


Figura 98: Comando da console para executar o aplicativo

Já no sistema operacional Windows, por exemplo, é possível executar o mesmo aplicativo pela console, digitando o mesmo comando, ou ainda executá-lo clicando com o botão direito do mouse sobre o arquivo e escolhendo a opção Java, conforme Figura 99.



Figura 99: Aplicativo sendo executado no Windows

E dessa maneira, o aplicativo também pode ser executado em qualquer outro sistema operacional desktop que possua a máquina virtual Java. A interface do aplicativo também é mantida, mudando apenas as características específicas de cada sistema operacional, como barra de título, botões de fechar, bordas, etc.

Na Figura 100 tem-se a interface do MeuPrimeiroProjeto executando nos sistemas operacionais Linux e Windows.

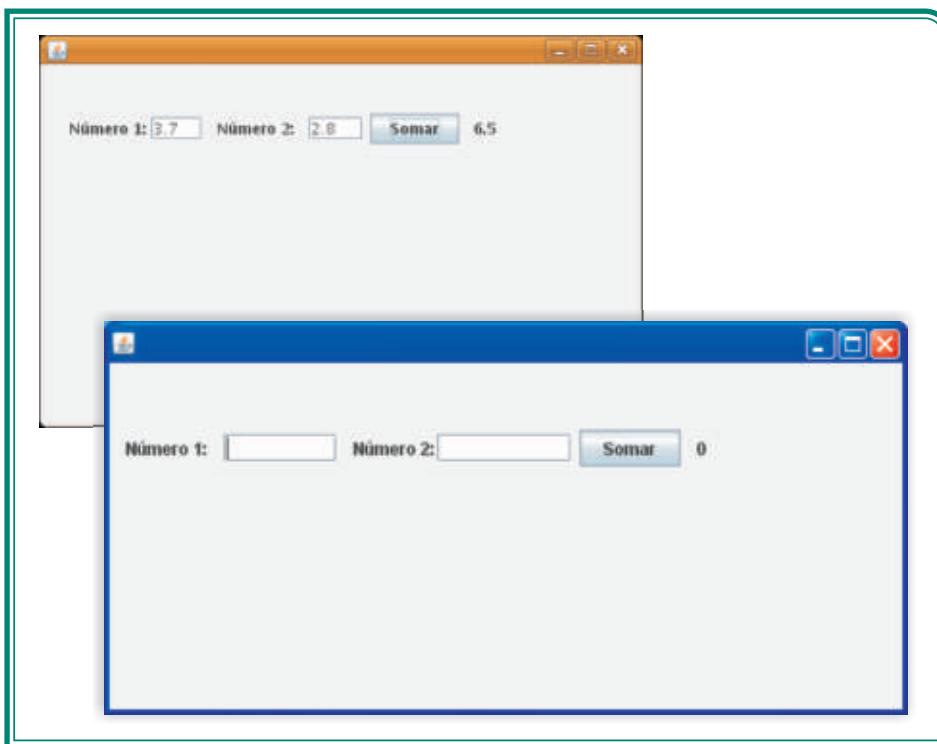


Figura 100: Aplicativo sendo executado Linux (Ubuntu) no Windows XP

## Resumindo

Esta aula apresentou o processo para distribuição de um aplicativo desenvolvido com a linguagem de programação Java, mais especificamente na IDE NetBeans.

Foi apresentada a geração do arquivo .jar, a instalação da JRE, bem como o processo para a execução do aplicativo em diferentes plataformas.



### Atividades de Aprendizagem:

- 1)** Desenvolver o instalador dos aplicativos desenvolvidos ao longo da apostila.
- 2)** Procurar executá-los em diferentes plataformas (Linux, Windows, etc.).

# Retomando a palavra dos professores-autores

Com os conhecimentos adquiridos nessa disciplina – Linguagem de programação comercial – é possível desenvolver aplicativos Desktop para vários sistemas operacionais, como Windows, Linux, MacOS, etc. Aplicativos esses com interface gráfica amigável, banco de dados e relatórios.

Com a constante evolução do mercado da informática, bem como a crescente necessidade de aplicativos desenvolvidos para plataforma livre (ex. Linux), o conhecimento de desenvolvimento visual usando a linguagem Java torna-se um grande diferencial na formação profissional dos novos programados.

Em contrapartida, apesar da grande quantidade de recursos e funcionalidades, a linguagem de programação Java dispõe de uma IDE intuitiva, com muitos recursos – essa IDE foi explorada ao longo do material e chamassem Netbeans. Além disso, a IDE Netbeans é gratuita e possui inúmeros tutoriais que apresenta sua utilização e recursos na Internet.

Agora é aproveitar ao máximo o conhecimento adquirido nessa disciplina e desenvolver vários aplicativos Desktop com Java.



## Referências

BERTOL, O. F. Home Page com Tutoriais e Estudos de caso sobre a tecnologia Java. Disponível em: <http://www.pb.utfpr.edu.br/omero>. Acessado em 24/06/2010.

DEITEL, H. M.; DEITEL, P.J. *JAVA: como programar*. 6a ed. Porto Alegre: Bookman, 2005.

FURGERI, S. *Java 2 – ensino didático*. São Paulo : Érica, 2002.

GONÇALVES, E. Dominando NetBeans. Rio de Janeiro: Ciência Moderna, 2006.

MACENAS, I. NetBeans 6.1. Rio de Janeiro: Alta Book, 2006.

MITCHELL, J. C. *Concepts in Programming Languages*. Reino Unido: Cambridge University Press, 2002.

GONÇALVES, E. *Desenvolvendo Relatórios Profissionais com iReport para NetBeans IDE*. Rio de Janeiro: Ciência Moderna, 2007.'

### INDICAÇÃO DE LINKS

Site oficial do NetBeans: <http://www.netbeans.org>

Site da Sun no Brasil: <http://br.sun.com>

Site com dicas e materiais sobre Java: <http://www.portaljava.com.br>

Site com conceitos básicos de OO: <http://www.dca.fee.unicamp.br/cursos/POOCPP/node3.html>

Apostila com Lógica de programação e OO: <http://www.jack.eti.br/www/arquivos/apostilas/java/logicapoo.pdf>

Site com conceitos básicos de OO: <http://www.linhadecodigo.com.br/Artigo.aspx?id=307>

Apostila introdutório sobre Java com exemplos práticos: <http://www.scribd.com/doc/13198145/Introducao-a-Linguagem-Java-Atraves-de-Exemplos>

Site com dicas e truques sobre linguagem java: <http://www.portaljava.com.br>

Site com material em português sobre java: <http://www.guj.com.br>

Apostila passo a passo do NetBeans: <http://www.scribd.com/doc/260919/Tutorial-NetBeans-By-WallCkguj.com.br>

Apostila com várias dicas da IDE NetBeans: [http://www.oficinadanet.com.br/apostilas/detalhe/52/tutorial\\_netbeans](http://www.oficinadanet.com.br/apostilas/detalhe/52/tutorial_netbeans) <http://java.sun.com>

Site com vídeo aulas sobre NetBeans e banco de dados: <http://www.devmedia.com.br>

Site que apresenta a conexão com um banco de dados passo a passo: [http://blogs.sun.com/ramonlopes/entry/netbeans\\_banco\\_de\\_dados\\_parte](http://blogs.sun.com/ramonlopes/entry/netbeans_banco_de_dados_parte)

Site da ferramenta iReport: <http://jasperforge.org/plugins/mwiki/index.php/Ireport>

Site com tutorial sobre iReport: <http://www.guj.com.br/article.show.logic?id=151>

# Curso técnico em meio ambiente

## Caro estudante:

Os avanços tecnológicos na área de informática e comunicação, associados a modelos pedagógicos apoiados no uso de tecnologia, deram origem à modalidade de ensino chamada Ensino a Distância (EaD). A característica desse modelo é a separação física entre aluno e professor. Para suprir a distância, a interação entre o aluno e o professor é mediada tanto por recursos tecnológicos quanto pelo material impresso. Nessa modalidade de ensino, o material impresso, juntamente com recursos de vídeo, videoconferência e um Ambiente Virtual de Aprendizagem são as bases tecnológicas, às quais você terá acesso durante sua formação.

Todos esses recursos são meios de comunicação entre professor e aluno. Cada recurso possui característica própria e necessita de canal específico de comunicação. Para assistir aos vídeos, participar de videoconferência ou realizar as atividades do Ambiente Virtual de Aprendizagem você precisará ter acesso a computadores e a Internet. Porém, tais recursos tecnológicos nem sempre estão disponíveis em tempo integral, por isso, a importância do material impresso, que permitirá a você ter acesso ao conhecimento independente de possuir a sua disposição as tecnologias de informática e de comunicação.

Aliado às atividades presenciais e às atividades a distância, o material impresso irá também apoia-lo na realização das atividades de estudos, estimulando-o a participar de forma mais ativa no seu processo de ensino-aprendizagem, construindo, progressivamente, conhecimento de maneira interativa. Assim, o professor deixa de ser a única fonte de informação. O distanciamento físico não será impedimento para o processo de cooperação e interação entre você e o professor da sua instituição, responsável pela oferta da disciplina no curso. Esse professor criará oportunidades para que você participe de forma ativa durante seu processo de aprendizagem.

O material foi elaborado visando a formação de Técnicos em Meio Ambiente, segundo os parâmetros do Catálogo Nacional de Cursos Técnicos. O profissional formado deverá ter qualificação para atender à demanda regional em consonância com as tendências tecnológicas. Deverá ser capaz de coletar, armazenar e interpretar informações, dados e documentações ambientais, colaborar na elaboração de laudos, relatórios e estudos ambientais, auxiliar na elaboração, acompanhamento e execução de sistemas de gestão ambiental, atuar na organização de programas de educação ambiental, de conservação e preservação de recursos naturais, de redução, reuso e reciclagem, bem como identificar as intervenções ambientais, analisar suas consequências e operacionalizar a execução de ações para preservação, conservação, otimização, minimização e remediação dos seus efeitos.

Além disso, deve estar ancorado em uma base de relacionamento interpessoal e comunicação oral. Deve também ter pensamento crítico e racional, capacidade para resolver problemas de ordem técnica, capacidade criativa e inovadora, capacidade de gestão e visão estratégica. Essa base o tornará competitivo no mercado de trabalho. Mas, isso somente não é suficiente, você também deve demonstrar: honestidade, responsabilidade, adaptabilidade, capacidade de planejamento, ser ágil e ter capacidade de decisão. Além de ser possuidor de um espírito crítico, uma formação tecnológica generalista e uma cultura geral sólida e consistente.

Foi pensando nessa formação que equipes de professores da rede pública federal de educação elaboraram seu material. Professores que atuam no ensino médio e no ensino superior. Todos profissionais conceituados em suas respectivas áreas de atuação. O objetivo desses profissionais é auxiliar você na sua formação profissional.

Os recursos didáticos pedagógicos e os profissionais envolvidos fazem parte do projeto Escola Técnica Aberta do Brasil, e-TecBrasil. Um projeto que estabelece parceria entre Instituições de Ensino Público Federal, no papel de formadores, e município, ou estado, que disponibilizam os pólos que receberão os cursos oferecidos na modalidade de EaD.

Mas, lembre-se, simplesmente ter acesso aos recursos didáticos e tecnológicos, além de ter a disposição uma equipe especializada de profissionais não é suficiente. É necessário que esse material seja utilizado intensamente, de forma a tornar-se fonte de conhecimento que o auxiliará em todos os momentos de sua formação.

Cientes de que esse também é o seu desejo, a equipe do e-TecBrasil deseja a todos um ótimo processo de aprendizagem.

Atenciosamente,

Equipe de formadores da Escola Técnica Aberta do Brasil  
da Universidade Tecnológica Federal do Paraná.

# Currículo dos professores-autores

**Robison Cris Brito** (robison@utfpr.edu.br) é mestre em Engenharia Elétrica e Informática Industrial pela Universidade Tecnológica Federal do Paraná, onde ministra aulas sobre tecnologia Java e Computação Móvel. Trabalha com a tecnologia Java desde 2003. Atualmente escreve artigos para as revistas Java Magazine e WebMobile Magazine, sendo também membro do Grupo de Estudos e Pesquisa em Tecnologia de Informação e Comunicação (GETIC).



**Beatriz Terezinha Borsoi** (beatriz@utfpr.edu.br) é mestre em Informática pela Universidade Federal do Paraná e doutora em Engenharia Elétrica pela Universidade de São Paulo. Atualmente é professora da Universidade Tecnológica Federal do Paraná, ministrando aulas sobre Fundamentos de Programação e é membro do Grupo de Estudos e Pesquisa em Tecnologias de Informação e Comunicação (GETIC).





