
Ferramentas de Programação Java:

Eclipse

Prof. Ms. Edson Alves de Oliveira Junior
edson@edsonjr.pro.br



Tópicos

- Introdução ao Eclipse:
 - *Eclipse Foundation*, Histórico, Membros, Licença, Projetos, Downloads
- Preparando o Eclipse IDE
 - Configurações iniciais do Eclipse
- Trabalhando com o Eclipse IDE
 - Ambiente, Visões e Perspectivas, Ajuda
- Desenvolvimento e Depuração
 - Edição, Busca e Manutenção, Javadoc
- *Plugins*:
 - Instalação e Gerenciamento
- Aplicações Web

Introdução ao Eclipse: Projeto e IDE

Ferramentas de Programação Java



Eclipse

<http://www.eclipse.org/org>

■ O que é?

- Uma comunidade de projetos *open-source*

■ Qual o seu objetivo?

- Fornecer uma plataforma de desenvolvimento e *frameworks* para a construção de software e a integração de ferramentas

■ O que é a *Eclipse Foundation, Inc.*?

- Empresa sem fins lucrativos
- Visa a criação, evolução, promoção e suporte à plataforma Eclipse
- Estatuto:

http://www.eclipse.org/org/documents/Eclipse%20BYLAWS%202003_11_10%20Final.pdf

Eclipse

<http://www.eclipse.org/org>

- Ferramentas do Eclipse permitem aos desenvolvedores liberdade de escolha de linguagem, plataforma e fabricante
- Possui um *framework* de desenvolvimento de plugins para criar, integrar e utilizar ferramentas, economizando tempo e dinheiro
- Plataforma escrita em Java
- Pode ser utilizada em várias plataforma como Linux, HP-UX, AIX, Solaris, Mac OS e Windows

História do Eclipse

<http://www.eclipse.org/org>

- **Novembro/2001:** formação inicial do projeto
 - Borland, IBM, MERANT, QNX Software Systems, Rational Software, Red Hat, SuSE, TogetherSoft e Webgain
- **Novembro/2003:** mais de 80 empresas já participavam do projeto Eclipse
- **Fevereiro/2004:** eclipse se tornou uma empresa sem fins lucrativos:
 - Lançada a Plataforma Eclipse *open-source*
 - Passou a ser uma empresa dedicada exclusivamente aos interesses de empresas, da academia e de instituições de pesquisa
- **Atualmente** hospeda mais de 50 projetos *open-source* de seus membros

Tipos de Membros

<http://www.eclipse.org/membership>

- ***Strategic Members:*** Intel, Borland, Iona, IBM, etc
 - Empresas que vêem Eclipse como uma plataforma estratégica e investem em seu desenvolvimento
- ***Add-in Provider Members:*** Hitachi, Red Hat, etc
 - Empresas que vêem Eclipse como uma parte importante de seus negócios e produtos.
- ***Associate Members:*** Addison-Wesley, Fraunhofer
 - Empresas sem fins lucrativos, comitês, universidades, institutos de pesquisa que participam do desenvolvimento do Eclipse
- ***Committer Members:***
 - Indivíduos que são um dos núcleos de desenvolvimento do Eclipse e podem fazer mudanças em seu código-fonte

Licença de Uso do Eclipse

- Acesso ao código e utilização é controlada pela *Eclipse Public Licence*:
 - permite re-distribuição de código sem cobrança de taxas
- Quem utiliza a Plataforma Eclipse tem acesso aos logos do Projeto e sua utilização
- A representação e escolha do pessoal do Projeto Eclipse fica a cargo da *Board Representative*

Projetos Eclipse

<http://www.eclipse.org/projects>

- Todo o trabalho desenvolvido pela *Eclipse Foundation* é dividido em 9 projetos:
 1. ***The Eclipse Top-Level Project***: desenvolvimento de aplicações-cliente altamente integradas
 2. ***The Eclipse Tools Project***: desenvolvimento de componentes comuns a todos os projetos
 3. ***The Eclipse Web Tools Platform Project***: plataforma e conjunto de ferramentas para J2EE e Web
 4. ***Test & Performance Tools Platform (TPTP) Project***
 5. ***The Eclipse Data Tools Platform Project***: suporte e data-centers
- Outros 4

Downloads Eclipse

<http://www.eclipse.org/downloads>

- Recursos disponíveis:
 - Eclipse SDK:
 - Projetos
- Todos os recursos estão sob a licença ***Eclipse Foundation Software User Agreement*** ou alguma outra específica:
 - <http://www.eclipse.org/legal/epl/notice.php>
- Os recursos disponíveis em eclipse.org podem ser obtidos por:
 - **Projeto**
 - **Tópico:** C++ plugins, JSP plugins, Swing, etc.
 - **Código-Fonte:** download de código-fonte

Documentação Legal/*Newsgroup*

- Toda a documentação legal pode ser obtida em:
 - <http://www.eclipse.org/legal>
- Exemplos:
 - Licenças da Plataforma e Utilização dos recursos, FAQ
 - Privacidade, Patentes, etc
 - Recursos para a comunidade, contribuições, políticas de membros, etc
- *Newsgroups*:
 - <http://www.eclipse.org/newsgroups/index.php>
 - Canal de comunicação com a comunidade de desenvolvimento do Eclipse

Logotipos

<http://www.eclipse.org/artwork/index.php>

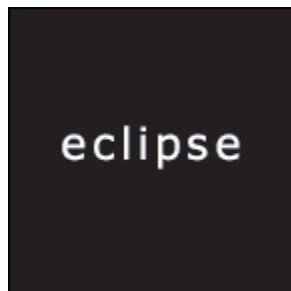
- Devem seguir as orientações de uso apropriado
- Logotipos para os Membros



- Logotipos para a comunidade



eclipse



eclipse



Eclipse.org

The screenshot shows the Eclipse.org website as it appeared in early 2006. The browser window is titled "Eclipse.org home - Mozilla Firefox" and shows the address bar with "http://www.eclipse.org/". The website has a dark blue header with the "eclipse" logo and a search bar. Below the header is a navigation menu with links to Home, Community, Membership, Downloads, Projects, and About Us. A left sidebar contains links to Committers, Newsgroups, Bugs, and Articles. The main content area features a "Welcome" message, a "Getting started" sidebar with links to Downloads, Documentation, and other resources, and two columns of news items. At the bottom, there are sections for "Eclipse News", "Eclipse In The News", and "Eclipse Technical Articles". A "Get Firefox" button is visible in the bottom right corner of the website content.

Eclipse.org home - Mozilla Firefox

Arquivo Editar Exibir Ir Favoritos Ferramentas Ajuda

http://www.eclipse.org/

eclipse

search: [input] [GO]

Contact | Legal

Eclipse.org navigation

Home Community Membership Downloads Projects About Us

Committers

Newsgroups

Bugs

Articles

Welcome

Eclipse is an open source community whose projects are focused on providing an extensible development platform and application frameworks for building software.

[more about eclipse »](#)

eclipseCON™ 2006
8 Days Left To Register
March 20th - 23rd
Santa Clara Convention Center

Eclipse News

- Eclipse Board Election Results posted 09-03-2006
- Eclipse Corner Article published: Teach Your Eclipse to Speak the Local Lingo posted 09-03-2006
- Voting closes today 4 p.m. EST in the Eclipse Foundation Elections posted 24-02-2006
- Eclipse Community Awards voting closes Tuesday, February 28th posted 24-02-2006
- Eclipse Corner Article published: Java Application Profiling using TPTP posted 21-02-2006

Eclipse In The News

- New Developer Tools Coming at EclipseCon by Darryl K. Taft posted 07-03-2006
- Eclipse Awarded IDE of the Year in the 2005 LinuxQuestions.org Members Choice Awards posted 07-03-2006
- Open Source Initiative to Give People More Control Over Their Personal Online Information posted 27-02-2006
- Manage Your Own Identity Online by Clint Boulton posted 27-02-2006
- Sybase launches larger-scale version of RFID stack posted 27-02-2006

Eclipse Technical Articles

- Teach Your Eclipse to Speak the Local Lingo by Kit Lo (IBM)
Translations for the Eclipse Project and several top-level projects are contributed to the Eclipse Foundation in

Getting started

- Downloads
- Documentation
- Eclipse Technical Articles
- Eclipse Workbench
- Eclipse Services
- Newsgroups
- Eclipse Project Wiki
- Community Resources
- PlanetEclipse
- Licensing FAQ

Events Calendar

- Mar 13-17 SD West 2006
- Mar 20-24 AOSD 06
- Mar 20-23 EclipseCon 2006

Get Firefox

Concluído

Preparando o Eclipse IDE: Configurações Iniciais

Ferramentas de Programação Java

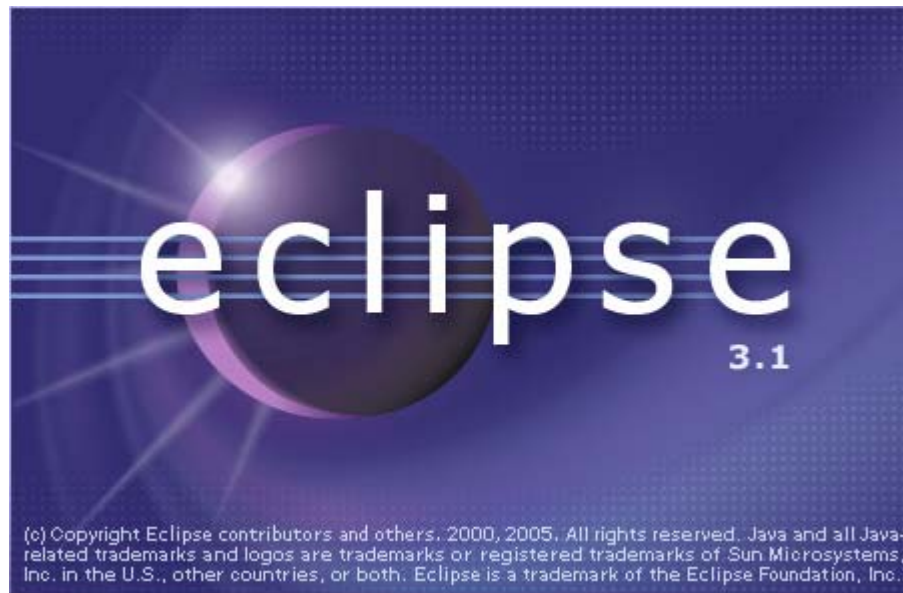


Preparando o Eclipse IDE

- **Etapa 01**: Instalação do Java Runtime Environment (JRE) versão 1.4.2 ou superior
- Instalar o JRE e configurar as seguintes variáveis de ambiente:
 - Adicionar o seguinte conteúdo ao final da variável **PATH**:
“;<dir_JRE>\bin” → Exemplo: “C:\jre1.5.0\bin”
 - Criar a variável **CLASSPATH** com o conteúdo:
“.;<dir_JRE>\lib\rt.jar” → Exemplo:
“C:\jre1.5.0\lib\rt.jar”
 - Criar a variável **JAVA_HOME** com o conteúdo
 - “<dir_JRE>” → Exemplo: “C:\jre1.5.0”

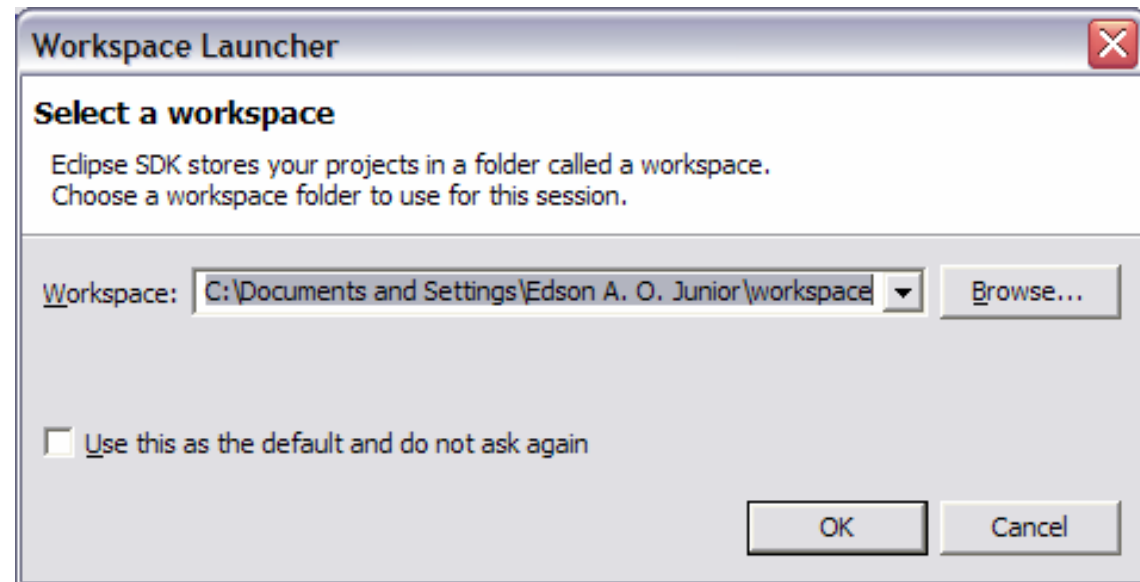
Preparando o Eclipse IDE

- **Etapa 02**: Instalação do Eclipse SDK
- Descompactar o arquivo obtido em qualquer pasta
- Na plataforma Windows, basta executar o arquivo `<eclipse_dir>\eclipse.exe`
- Aparecerá.....



Definindo o *Workspace*

- A primeira tela que aparece ao iniciar o Eclipse é a de seleção de *workspace*
- O *workspace* é o local em que o Eclipse armazena os seus projetos
- É possível modificar futuramente o local do *workspace*: menu *File / Switch Workspace...*

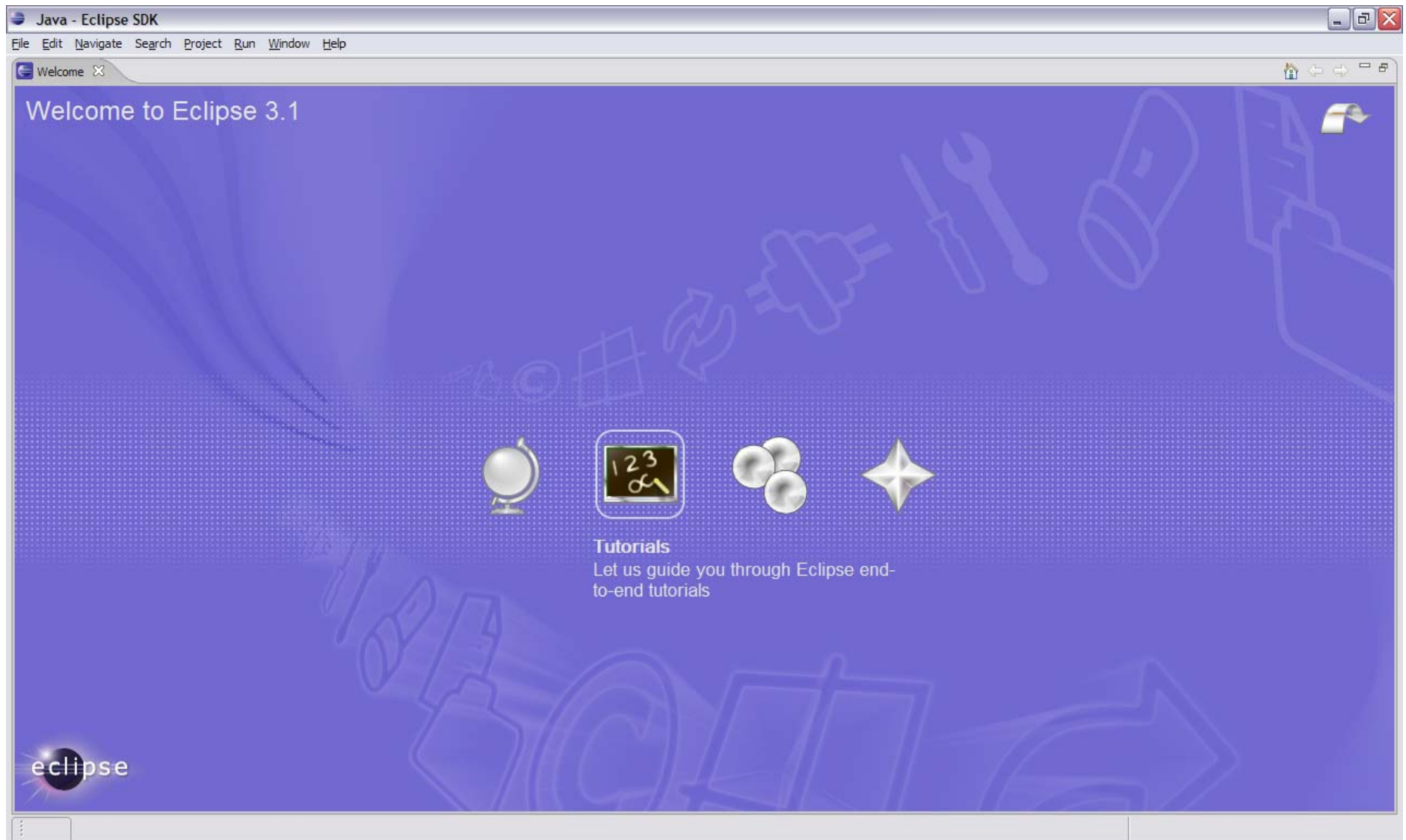


Trabalhando com o Eclipse IDE

Ferramentas de Programação Java



Tela Inicial do Eclipse IDE



Ambiente do Eclipse IDE

The screenshot shows the Eclipse IDE interface with the following components and annotations:

- Package Explorer:** Located on the left, it shows the project structure. Annotation: Package Explorer visão (view) de projetos e suas estruturas (pacotes, classes, etc)
- Code Area:** The central area displaying the source code of `JanelaPrincipal.java`. Annotation: Code Area visão (view) de código-fonte e de arquivos de um projeto
- Perspectives:** A cloud-shaped annotation pointing to the overall layout: Perspectives diferentes formas de organizar as visões do Eclipse IDE
- Outline:** Located on the right, it shows a summary of the elements in the code area. Annotation: Outline visão (view) resumida de um elemento da code area
- Outras visões existentes:** A box at the bottom right lists other available views: Outras visões existentes Ex.: Problems, Javadoc, Declaration, Console, Error Log, outros.

```
package visao;

import java.awt.Dimension;
import javax.swing.JFrame;

public class JanelaPrincipal extends JFrame {

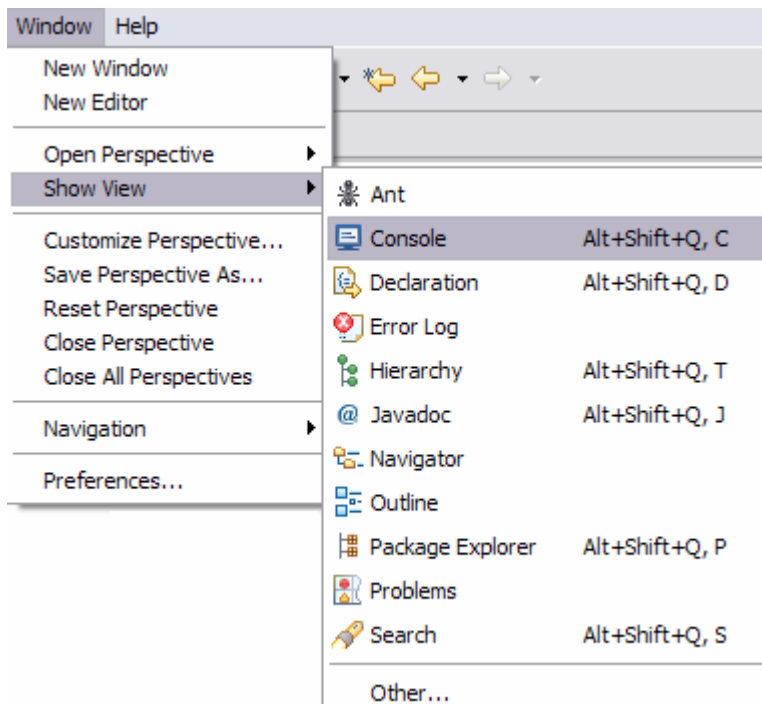
    public JanelaPrincipal() {
        super();
        this.setTitle("Trabalhando com o Eclipse IDE");
        this.setSize(new Dimension(800, 600));
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.setVisible(true);
    }

    public static void main(String[] args) {
        new JanelaPrincipal();
    }
}
```

Visões e Perspectivas

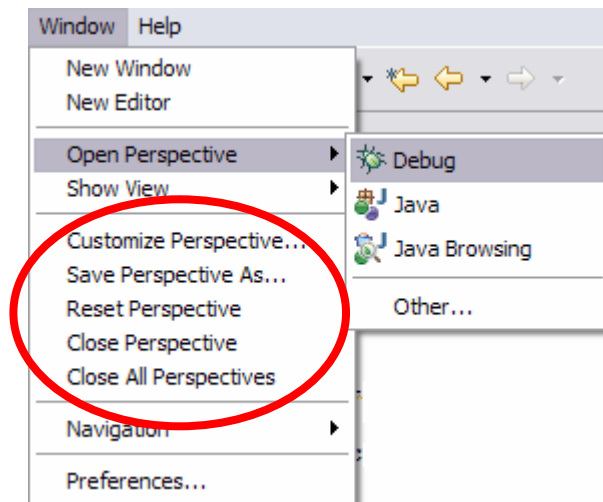
■ Visões:

- ❑ *Window / Show View*



■ Perspectivas:

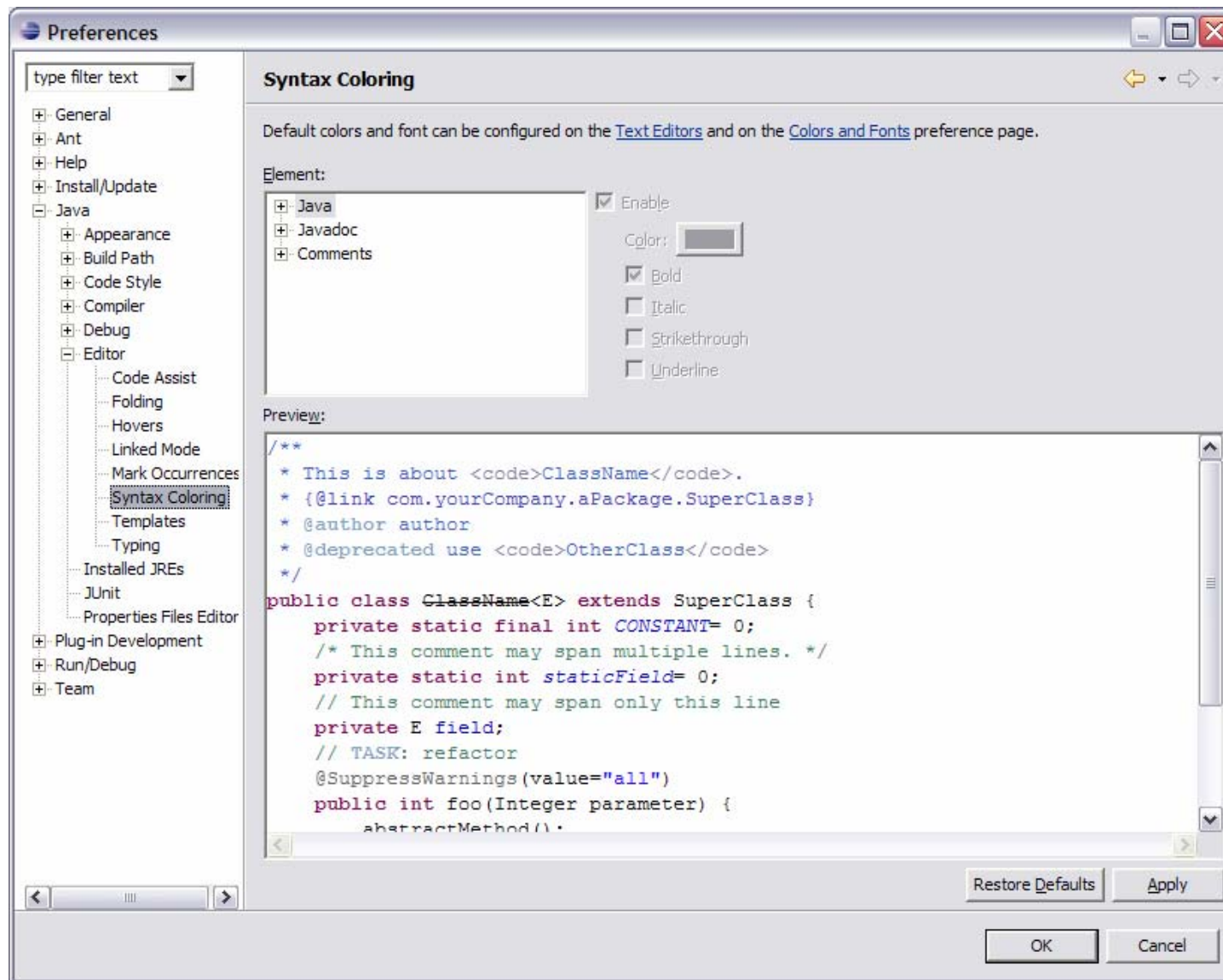
- ❑ *Window / Open Perspective*
- ❑ É possível: personalizar, salvar, reiniciar, fechar e excluir (*Window / Preferences / General / Perspectives*) uma perspectiva



Eclipse IDE: Preferências

- Menu *Window / Preferences...*
- Tipos de Configurações:
 - **Gerais:**
 - Aparência, Associação de Arquivos, Número de Linhas, Teclas de Atalho, Perspectivas, *Workspace*, etc
 - **Instalação / Atualização:**
 - URLs, Agendamento de Atualizações
 - **Java:**
 - Aparência do Código-Fonte, *Classpath*, Bibliotecas, Estilo de Codificação, *Templates* de Código, Versão do Compilador Java, Javadoc, Depuração, *Auto Completion*, JREs, JUnit
 - **Desenvolvimento de *Plug-ins*:**
 - Compiladores, Editores, Bibliotecas para Desenvolvimento
 - **Execução / Depuração de Aplicações:**
 - Ferramentas Externas, Console de Mensagens
 - **CVS:**
 - Conexões com Servidor CVS, Gerenciamento de Versões

Eclipse IDE: Preferências



Ajuda do Eclipse IDE (Menu *Help*)

■ Opção *Welcome*:

- Exibe a tela inicial do Eclipse IDE

■ Opção *Help Contents*:

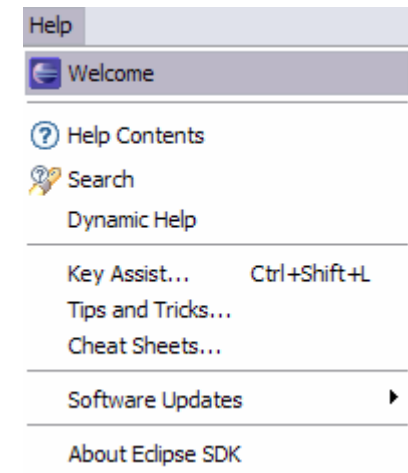
- Abre uma janela em que é possível acessar todo o conteúdo de ajuda e realizar pesquisas por termos
- Guia do Usuário, Desenvolvimento em Java, Ambiente de Desenvolvimento de Plug-ins, tutoriais diversos, etc

■ Opção *Search*:

- Exibe uma visão simplificada do Help Contents em que é possível pesquisar termos rapidamente

■ Opção *Dynamic Help*:

- Help mais elaborado em que os principais tópicos são exibidos para consulta



Ajuda do Eclipse IDE (Menu *Help*)

■ Opção *Key Assist...*:

- ❑ Atalho: Ctrl + Shift + L
- ❑ Exibe todas as teclas de atalho do Eclipse IDE

■ Opção *Tips and Tricks...*:

- ❑ Abre a janela de ajuda com dicas e truques sobre vários assuntos

■ Opção *Cheat Sheets...*:

- ❑ Mini-tutoriais básicos sobre desenvolvimento Java e de plug-ins

■ Opção *Software Updates:*

- ❑ Busca e instalação de atualizações dos recursos já instalados ou de novos recursos disponíveis
- ❑ Gerenciamento de configuração do Eclipse

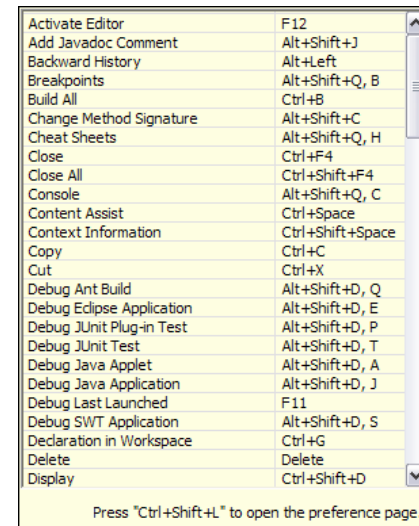
■ Opção *About Eclipse SDK:*

- ❑ Tela com informações sobre a versão do Eclipse e detalhes de *plug-ins*, *features* e configurações

Ajuda do Eclipse IDE (*Screenshots*)



Help Contents



Key Assist...

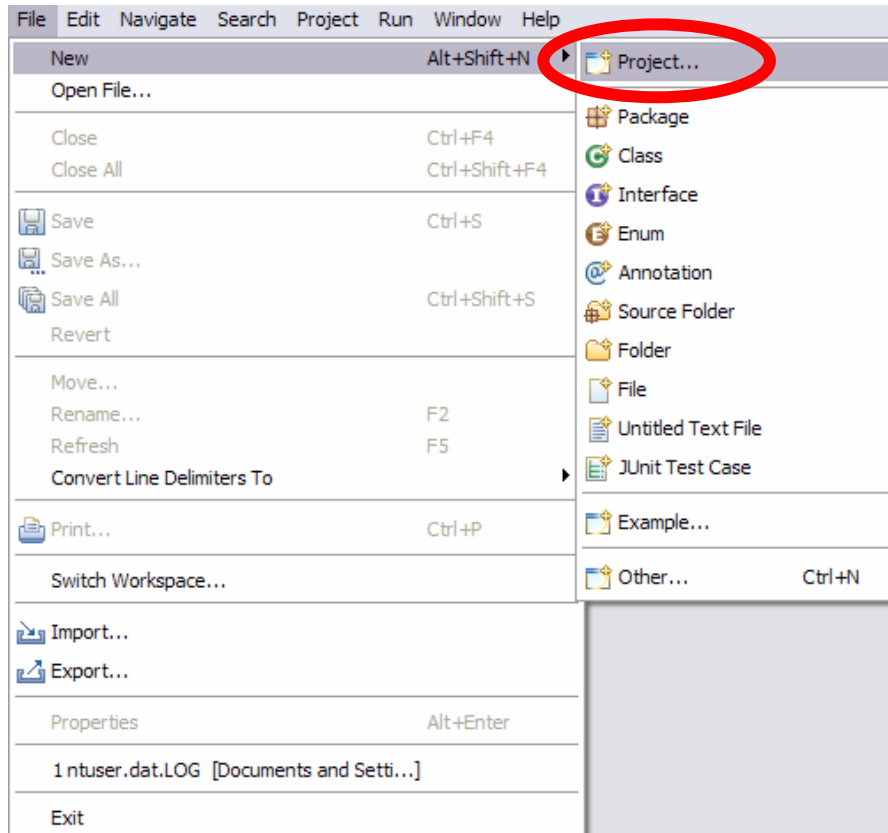


About Eclipse SDK

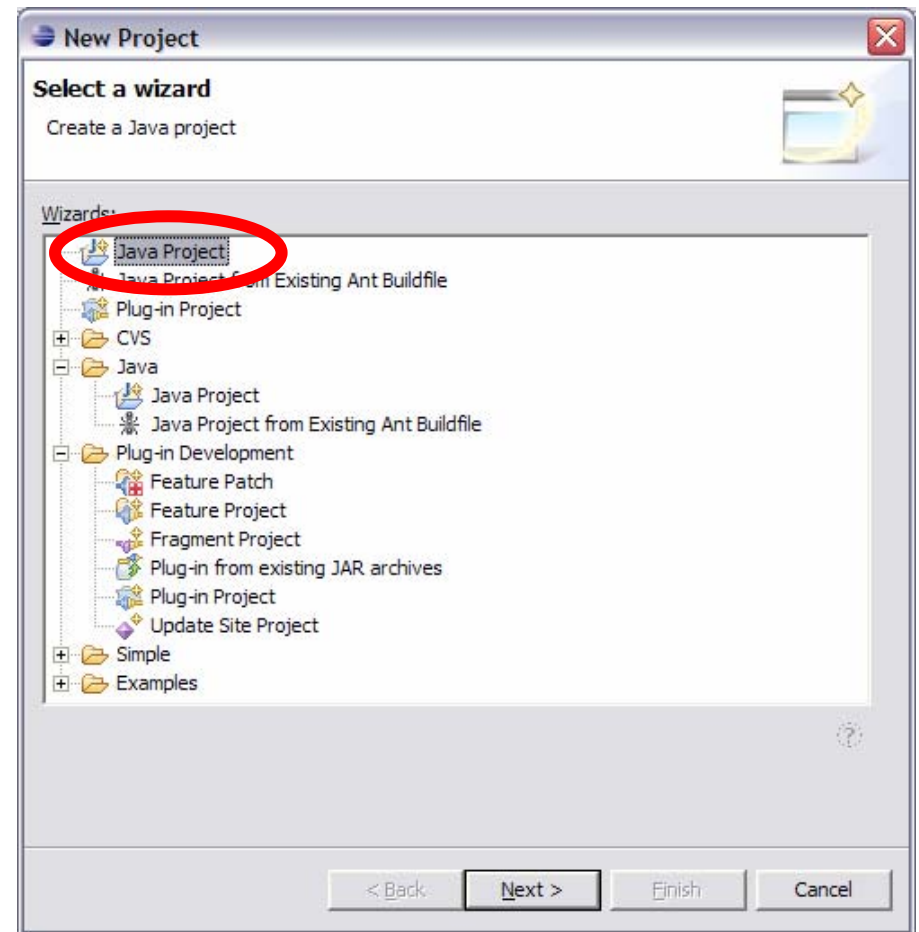
Trabalhando com Projetos

- Os projetos tem por objetivo separar os arquivos e pacotes gerados de acordo com os seus propósitos
- Eclipse IDE permite a criação de vários tipos de projetos, desde Projetos Java passando por Projetos CVS até Projetos de Desenvolvimento de *Plug-ins*
- Criação de Projetos:
 - Menu *File / New / Project*
 - Tecla de Atalho: *Alt + Shift + N*

Trabalhando com Projetos



Criando um Novo Projeto



Definindo o Tipo do Projeto

Definição do Projeto (Etapa 1)

Cria um novo projeto
no *workspace*

Opcionalmente, pode-se
criar um projeto a partir de
código-fonte existente

Define a versão do Java na
qual o projeto será
definido. Ex.: Java 5.0

Define a estrutura de pastas
do projeto, possibilitando
separar o código-fonte (.java)
dos *bytecodes* (.class)

New Java Project

Create a Java project

Create a Java project in the workspace or in an external location.

Project name:

Contents

- ☒ Create new project in workspace
- ☐ Create project from existing source

Directory:

JDK Compliance

- ☐ Use default compiler compliance (Currently 1.4) [Configure default...](#)
- ☒ Use a project specific compliance:

Project layout

- ☐ Use project folder as root for sources and class files
- ☒ Create separate source and output folders [Configure default...](#)

< Back **Next >** Finish Cancel

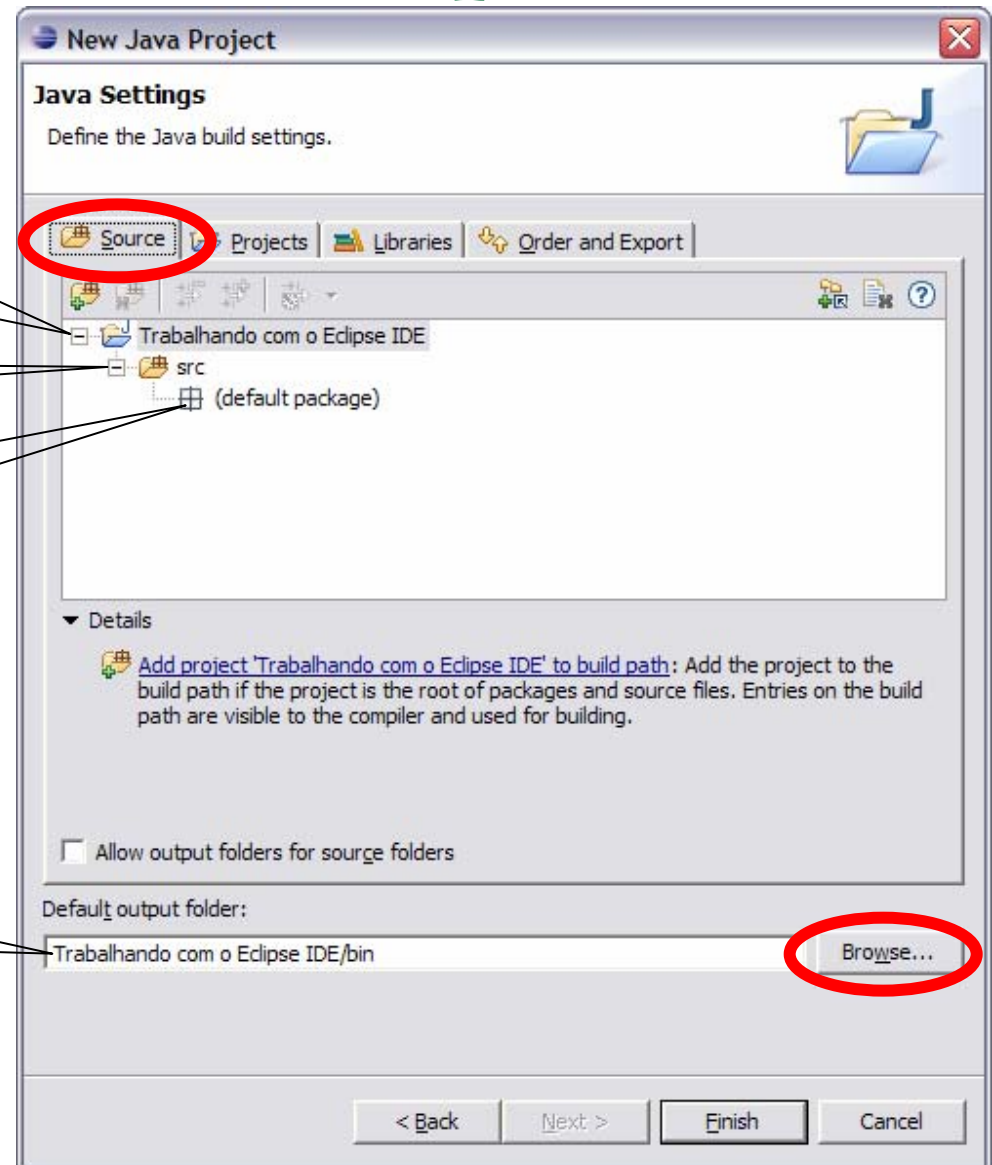
Definição do Projeto (Etapa 2)

Identifica o projeto que será adicionado ao *build path* para compilar e gerar o código-fonte do projeto

Pasta que armazenará o código-fonte do projeto

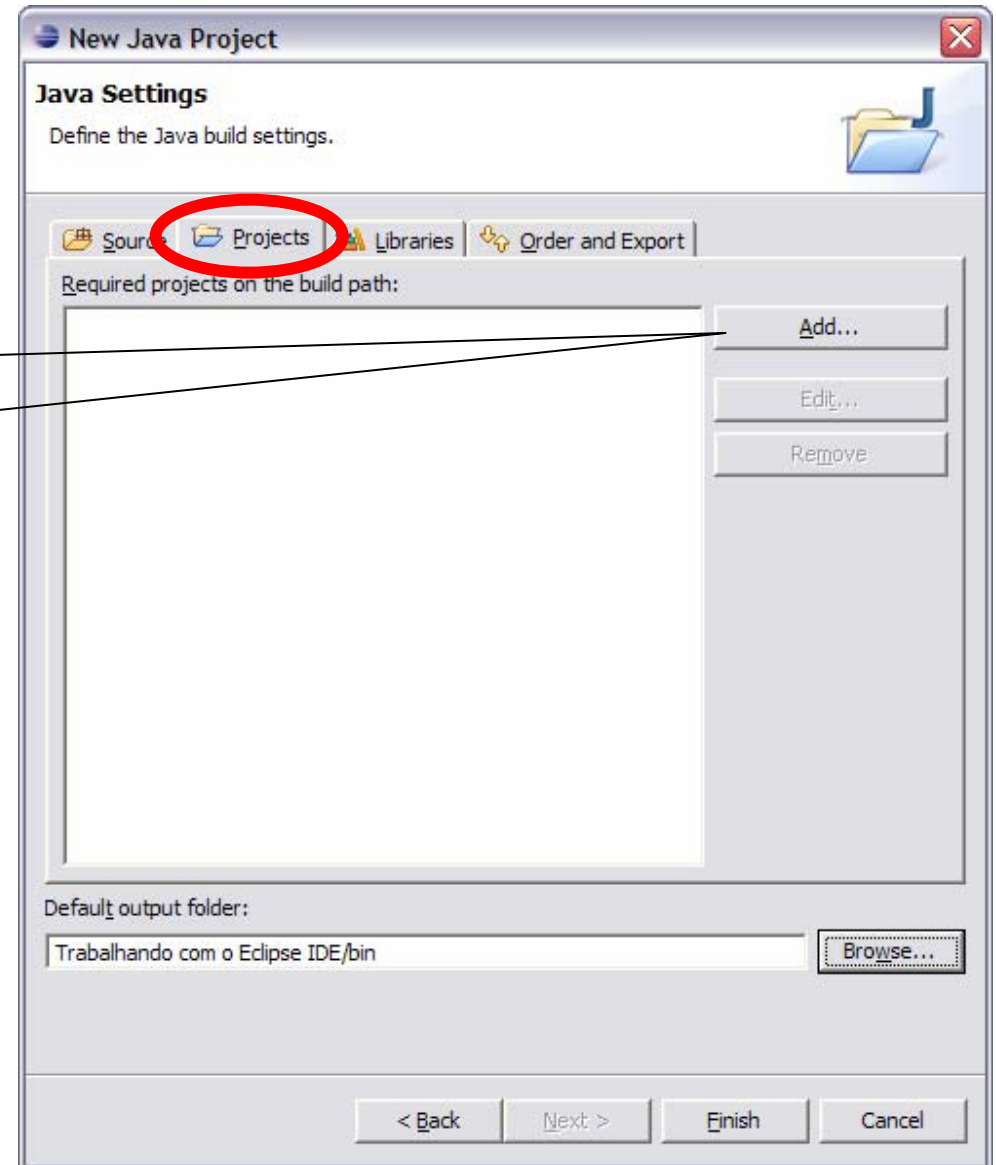
Pacote utilizado caso uma classe não possua a definição *package*

Pasta em que serão gerados os *bytecodes* do projeto



Definição do Projeto (Etapa 2)

Permite definir outros projetos aos quais o novo projeto está vinculado (depende) para o seu desenvolvimento



Definição do Projeto (Etapa 3)

Conjunto de bibliotecas básicas já vinculadas aos projetos Java

Adiciona bibliotecas de JARs

Adiciona pastas com classes compiladas

Edita/Remove entradas do build path

Adiciona arquivos JAR disponíveis de outros projetos

Adiciona arquivos JAR externos ao projeto

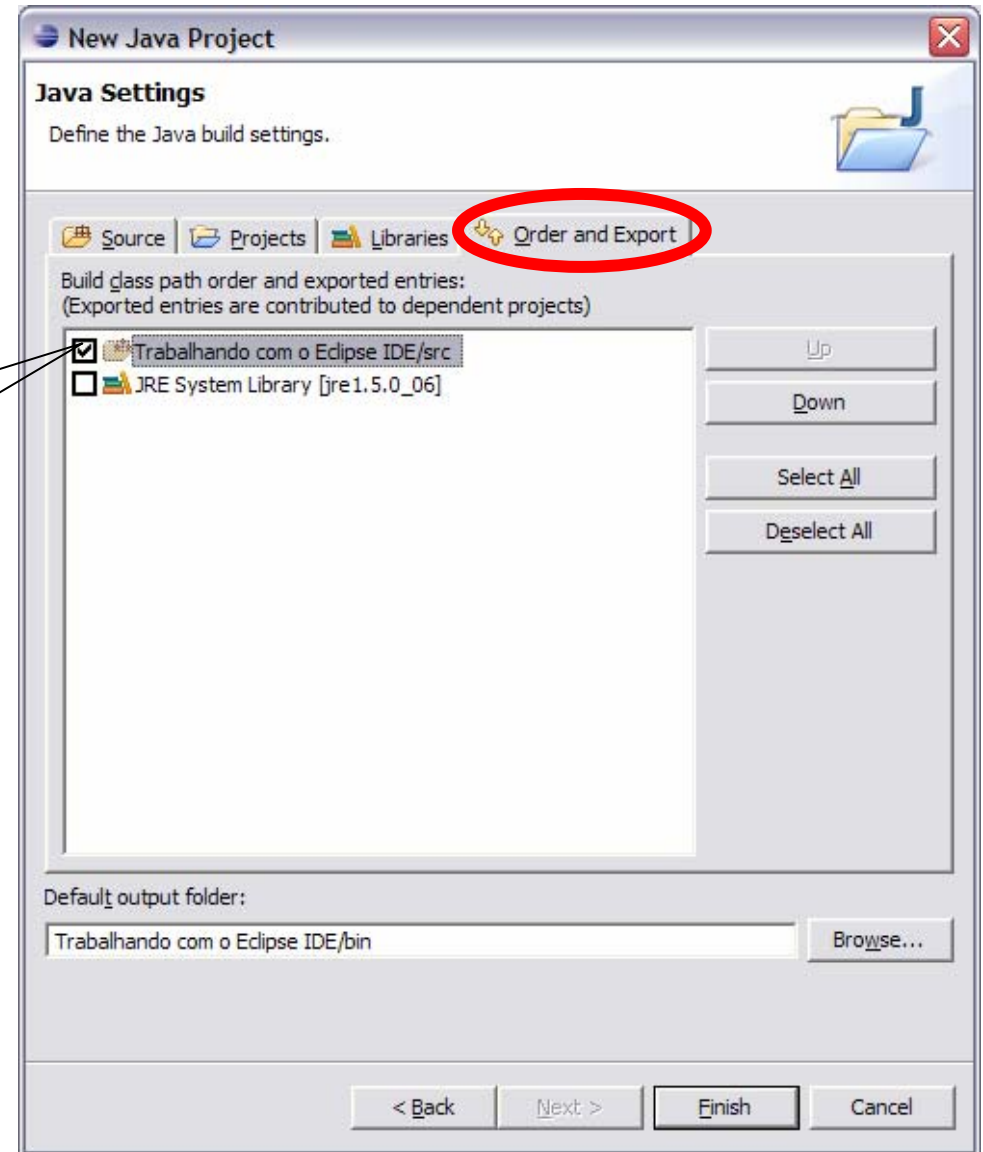
Adiciona variáveis de ambiente

Default output folder:
Trabalhando com o Eclipse IDE/bin

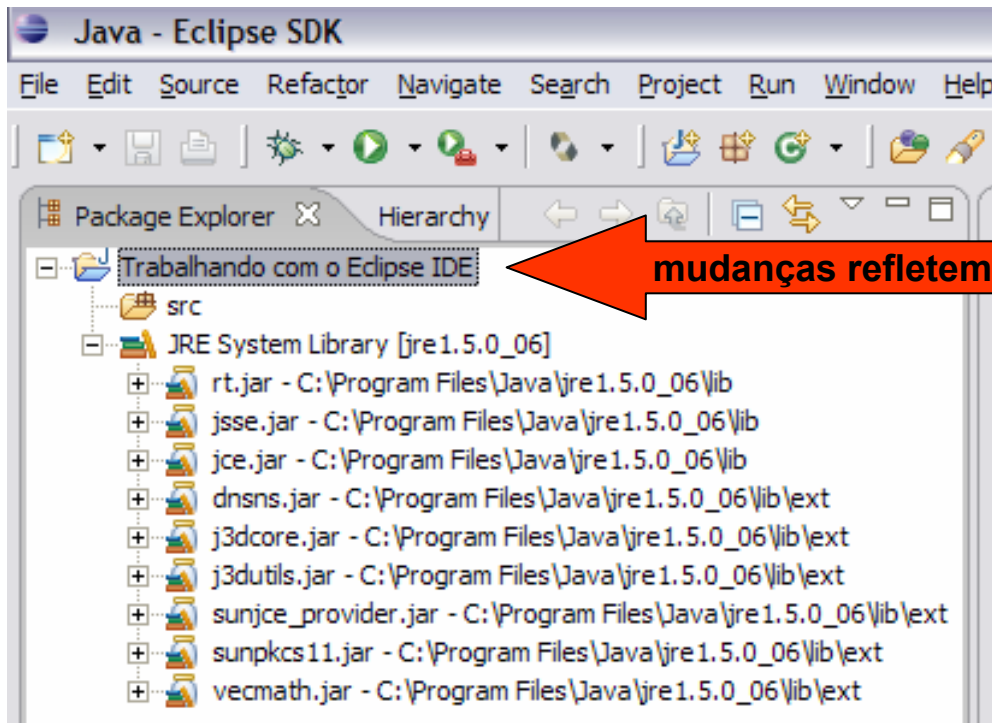
Buttons: Add JARs..., Add External JARs..., Add Variable..., Add Library..., Add Class Folder..., Edit..., Remove, < Back, Next >, Finish, Cancel

Definição do Projeto (Etapa 4)

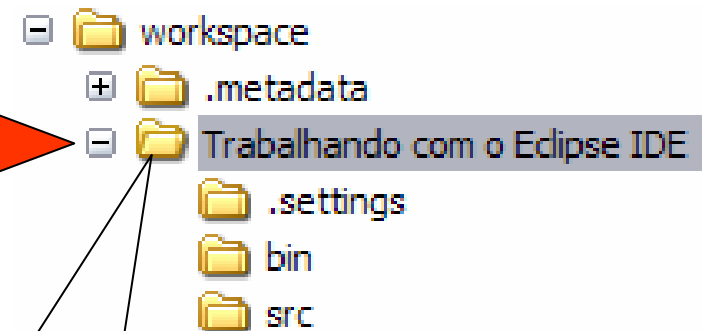
Definição do que será
incluído quando o projeto
for exportado como, por
exemplo, na forma de um
arquivo JAR



Estrutura Inicial do Projeto



No Eclipse



Fisicamente

Contém, inicialmente, os arquivos:

.classpath - informações sobre as pastas ***src*** e ***bin***
.project - informações sobre o projeto e o ***build path***

Criando Pacotes e Classes

- Toda classe ou pacote criado estará localizado na pasta **src** ou uma de suas subpastas
- **Criando Classes:**
 - Menu ***File / New / Class***
 - **Sem pacote definido:** quando não são definidos um pacote para a classe
 - **Em um determinado pacote:** quando um pacote é indicado para uma classe
- **Criando Pacotes:**
 - Menu ***File / New / Package***
 - Subpacotes são indicados com um “.”

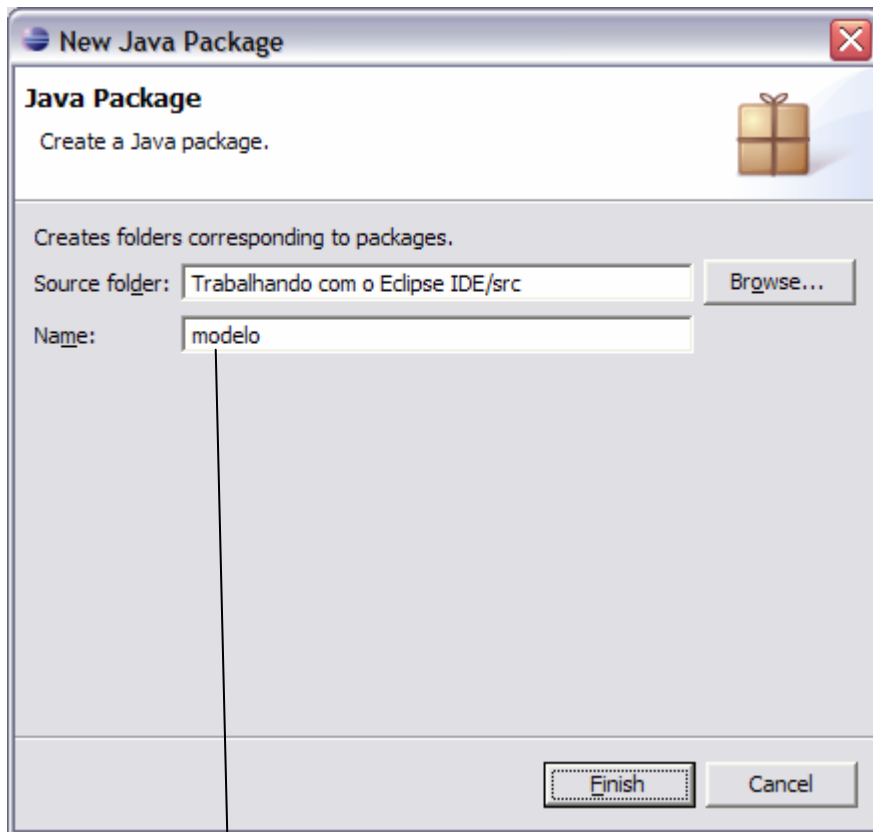
Criando Pacotes



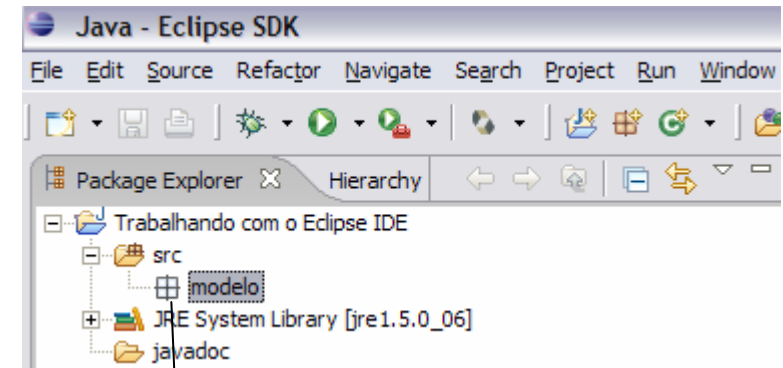
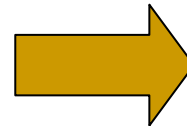
indica pacote vazio



indica pacote com classes



Nome do pacote a ser criado



Novo pacote criado

Criando uma Classe sem Pacote

- Selecione a pasta **src**
- Menu:
 - File / New / Class

Pasta onde será gerado o código-fonte da classe

Se for classe interna, indicar o nome da classe a qual pertence

Nome da classe e modificadores

Superclasse que a classe estende e as interfaces que implementa

Métodos que podem ser inicialmente criados

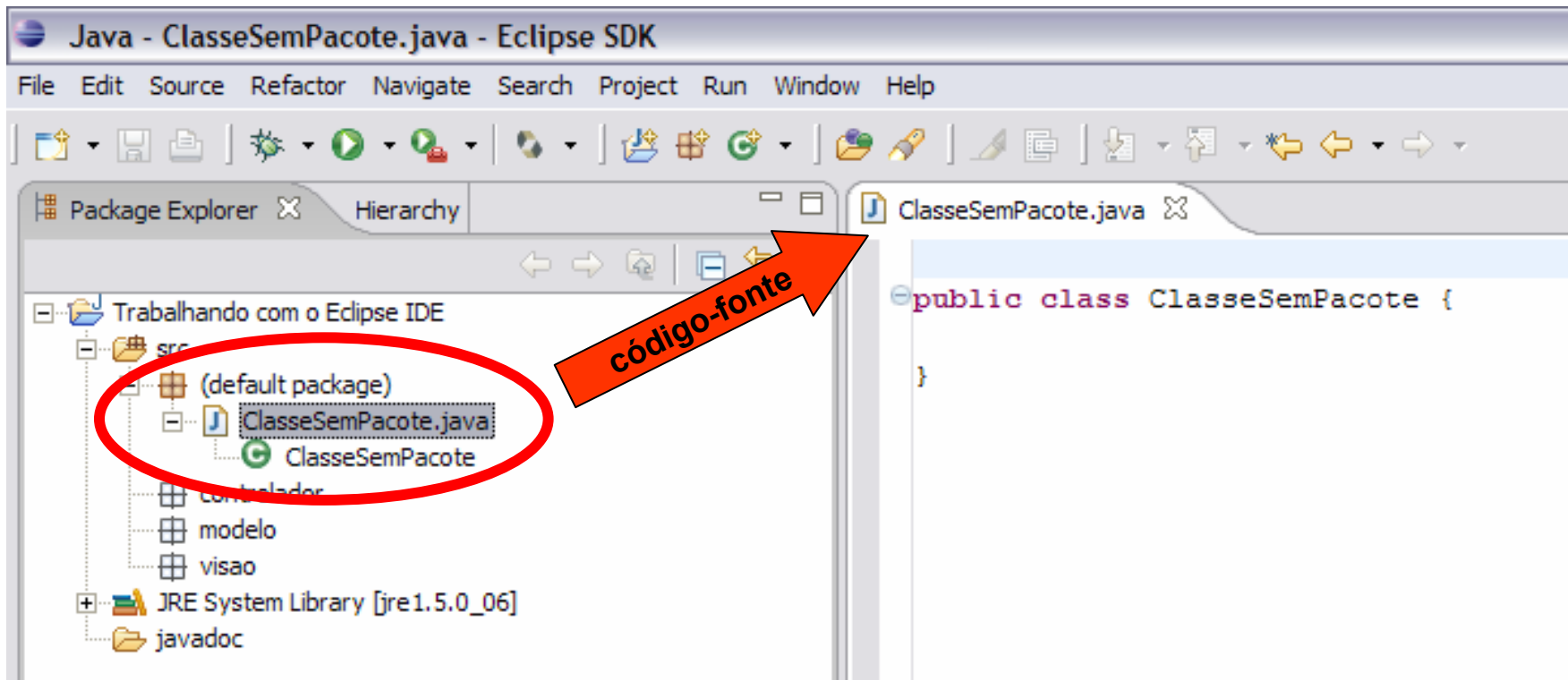
Gera comentários pré-definidos

The screenshot shows the 'New Java Class' dialog box in the Eclipse IDE. The title bar says 'New Java Class'. Inside, there's a section 'Java Class' with a warning icon and text: 'The use of the default package is discouraged.' Below this are several fields and checkboxes:

- Source folder:** 'Trabalhando com o Eclipse IDE/src' with a 'Browse...' button.
- Package:** '(default)' with a 'Browse...' button.
- Enclosing type:** An empty field with a 'Browse...' button.
- Name:** 'ClasseSemPacote'.
- Modifiers:** Radio buttons for 'public' (selected), 'default', 'private', and 'protected'. Checkboxes for 'abstract', 'final', and 'static'.
- Superclass:** 'java.lang.Object' with a 'Browse...' button.
- Interfaces:** An empty list with 'Add...' and 'Remove' buttons.
- Which method stubs would you like to create?** Checkboxes for 'public static void main(String[] args)', 'Constructors from superclass', and 'Inherited abstract methods' (checked).
- Do you want to add comments as configured in the [properties](#) of the current project?** A checkbox for 'Generate comments'.

At the bottom right are 'Finish' and 'Cancel' buttons.

Criando uma Classe sem Pacote



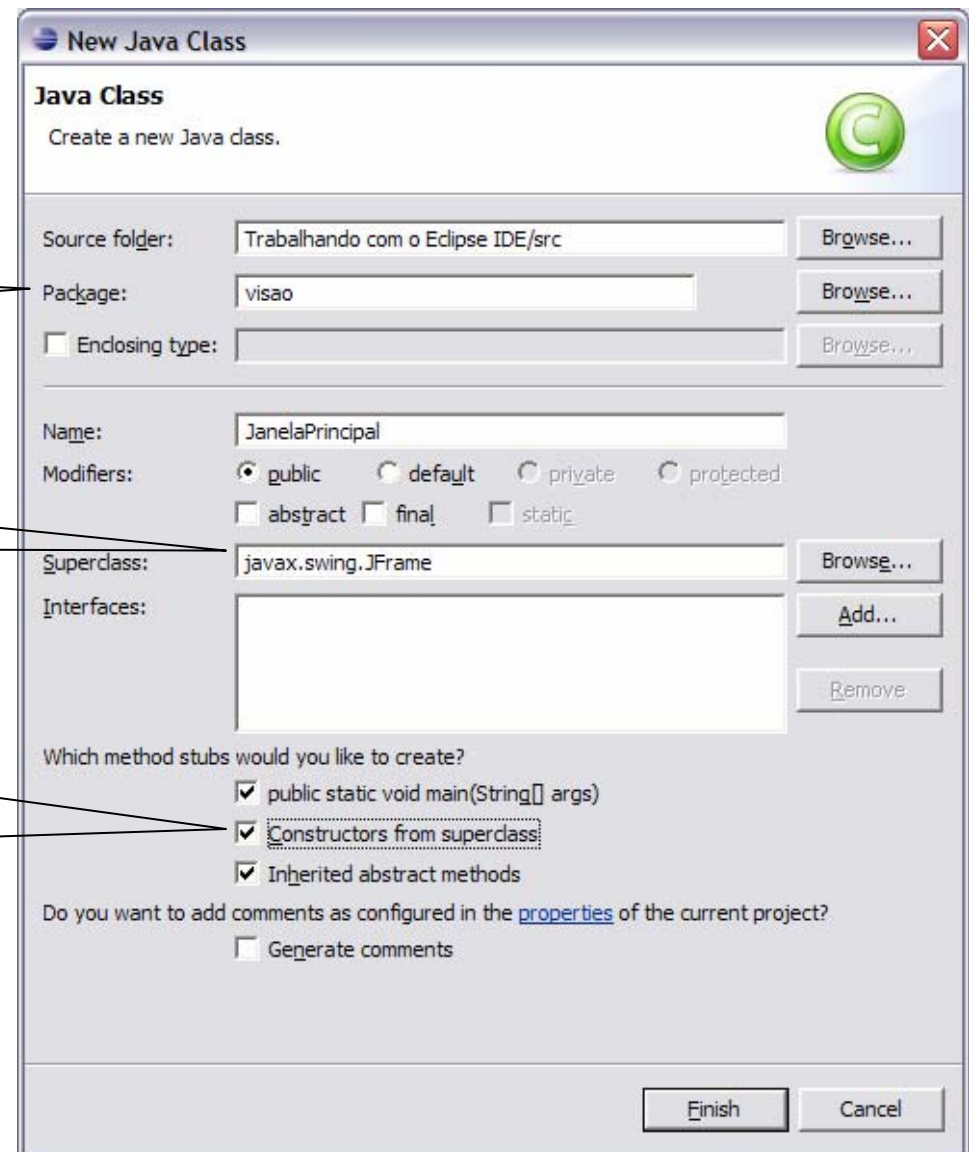
Criando uma Classe em um Pacote

- Selecione o pacote
- Menu:
 - File / New / Class

Pacote da classe:
visao

Classe criada estende a classe
javax.swing.JFrame

Será criado o método *main*, os
construtores da superclasse e os
métodos abstratos herdados da
superclasse



Criando uma Classe em um Pacote

The screenshot shows the Eclipse IDE interface. On the left, the Package Explorer displays a project structure with a package named 'visao'. Inside 'visao', the file 'JanelaPrincipal.java' is highlighted with a red box. Below it, the class 'JanelaPrincipal' and its methods are listed. A red arrow points from the text 'código-fonte' to the 'JanelaPrincipal.java' file. On the right, the editor shows the source code of 'JanelaPrincipal.java'. The code includes imports for 'java.awt.GraphicsConfiguration', 'java.awt.HeadlessException', and 'javax.swing.JFrame'. It defines a public class 'JanelaPrincipal' that extends 'JFrame'. Several constructors are provided, including one with no arguments, one with 'GraphicsConfiguration', one with 'String', and one with both 'String' and 'GraphicsConfiguration'. The 'main' method is highlighted with a red box, and a callout bubble points to it with the text 'Método *main* da aplicação'.

```
package visao;

import java.awt.GraphicsConfiguration;
import java.awt.HeadlessException;

import javax.swing.JFrame;

public class JanelaPrincipal extends JFrame {

    public JanelaPrincipal() throws HeadlessException {
        super();
    }

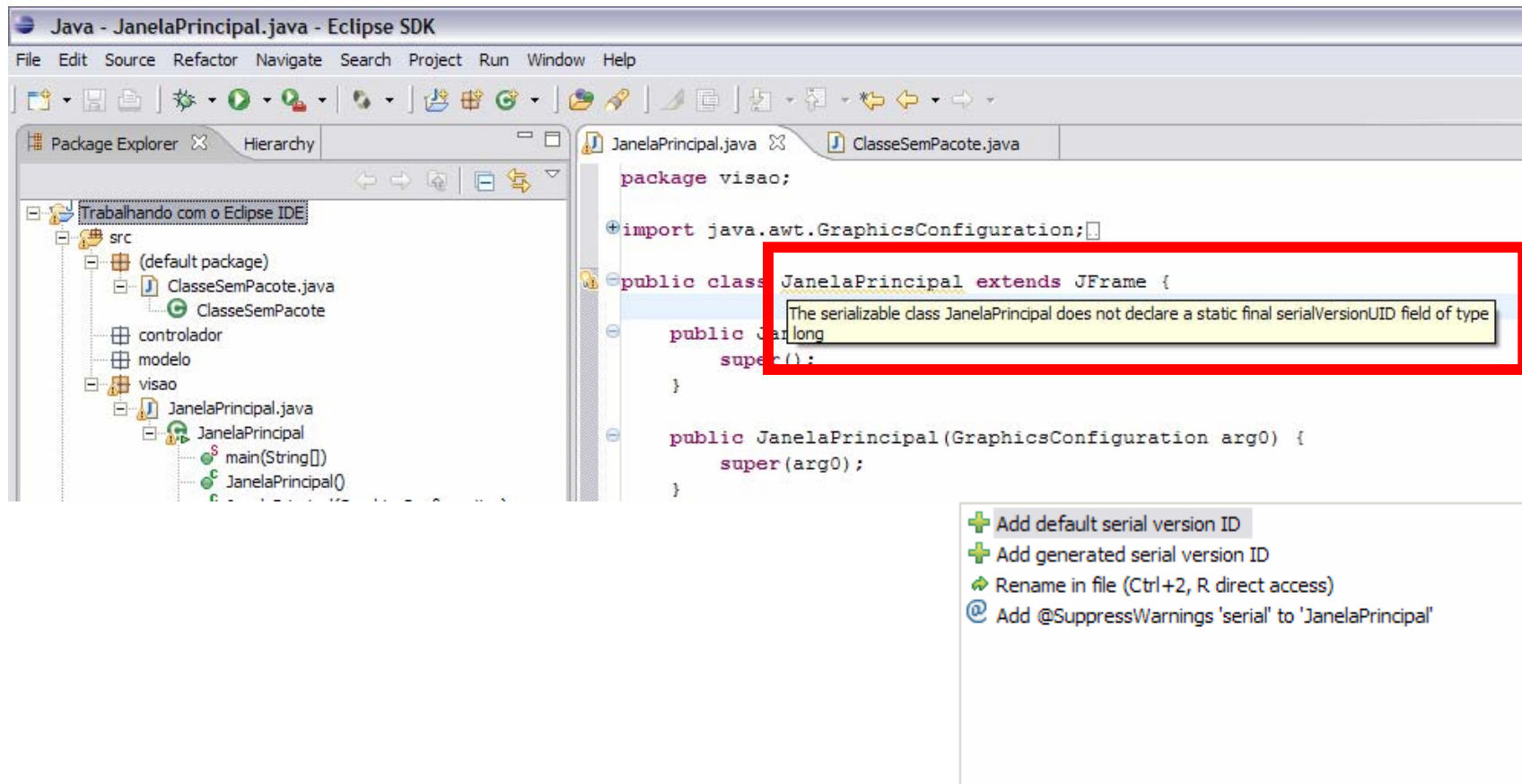
    public JanelaPrincipal(GraphicsConfiguration arg0) {
        super(arg0);
    }

    public JanelaPrincipal(String arg0) throws HeadlessException {
        super(arg0);
    }

    public JanelaPrincipal(String arg0, GraphicsConfiguration arg1) {
        super(arg0, arg1);
    }

    public static void main(String[] args) {
    }
}
```


Serial Version UID



- **serialVersionUID:** utilizado para controle de versão (CVS)
 - *JFrame* implementa a interface *java.io.Serializable*
- **Geração automática:** CTRL + 1
- **Desabilitar:** *Window / Preferences... / Java / Compiler / Errors/Warnings / Potencial programming problems*

Com *serialVersionUID*

Java - JanelaPrincipal.java - Eclipse SDK

File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer Hierarchy

Trabalhando com o Eclipse IDE

- src
 - (default package)
 - ClasseSemPacote.java
 - ClasseSemPacote
 - controlador
 - modelo
 - visao
 - JanelaPrincipal.java
 - JanelaPrincipal
 - serialVersionUID**
 - main(String[])
 - JanelaPrincipal()
 - JanelaPrincipal(GraphicsConfiguration)
 - JanelaPrincipal(String)
 - JanelaPrincipal(String, GraphicsConfiguration)

JRE System Library [jre1.5.0_06]

javadoc

```
package visao;

import java.awt.GraphicsConfiguration;

public class JanelaPrincipal extends JFrame {

    private static final long serialVersionUID = -3829956601547781628L;

    public JanelaPrincipal() throws HeadlessException {
        super();
    }

    public JanelaPrincipal(GraphicsConfiguration arg0) {
        super(arg0);
    }

    public JanelaPrincipal(String arg0) throws HeadlessException {
        super(arg0);
    }

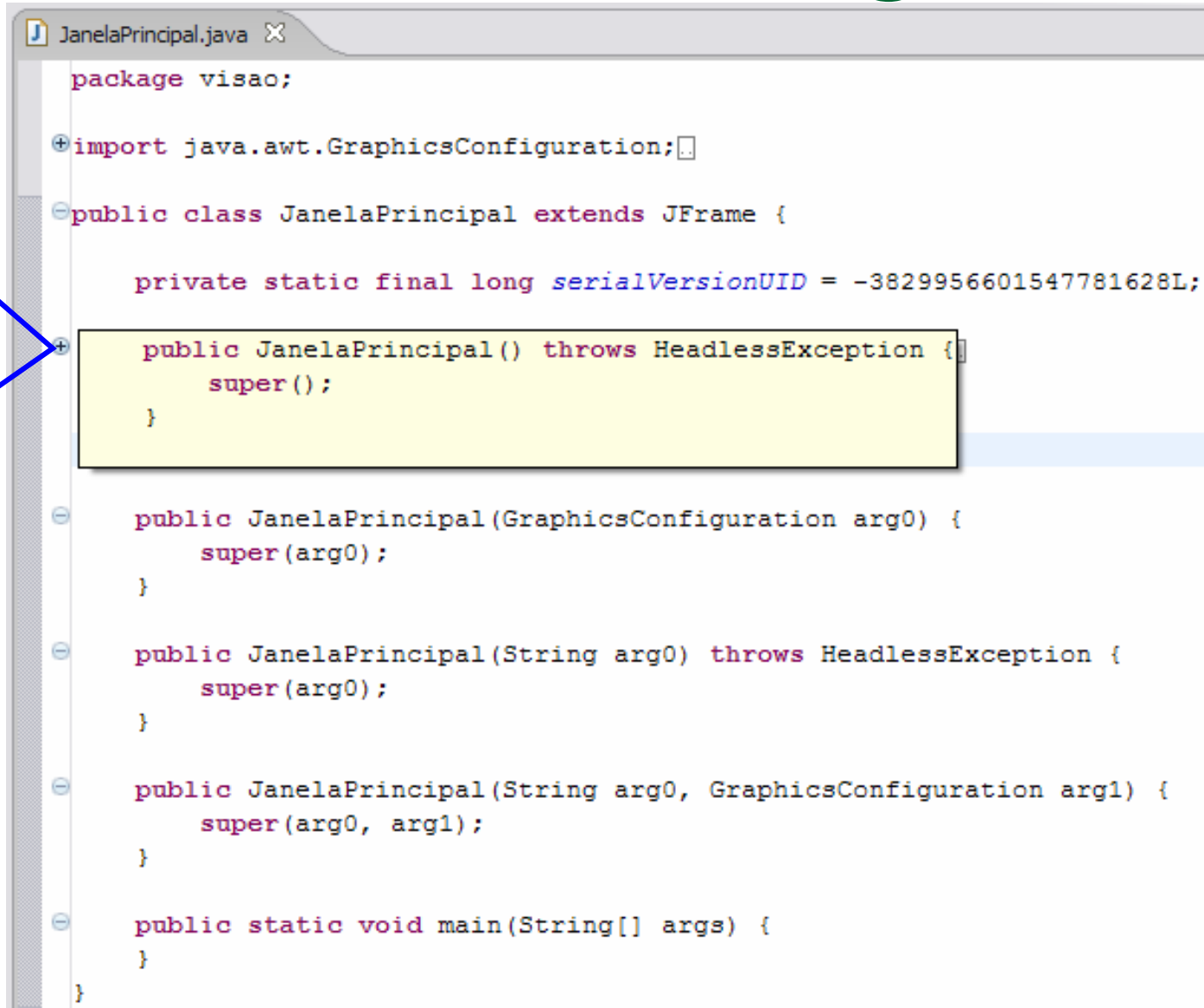
    public JanelaPrincipal(String arg0, GraphicsConfiguration arg1) {
        super(arg0, arg1);
    }

    public static void main(String[] args) {
    }
}
```

Atributos aparecem com a cor azul. Atributos *static* em itálico.

Ocultando/Exibindo Código

O sinal de + significa que o código está escondido.
O código aparece posicionando o mouse no sinal de + ou clicando sobre ele.



```
JanelaPrincipal.java X
package visao;

+import java.awt.GraphicsConfiguration;[]

- public class JanelaPrincipal extends JFrame {

    private static final long serialVersionUID = -3829956601547781628L;

+    public JanelaPrincipal() throws HeadlessException {
        super();
    }

-    public JanelaPrincipal(GraphicsConfiguration arg0) {
        super(arg0);
    }

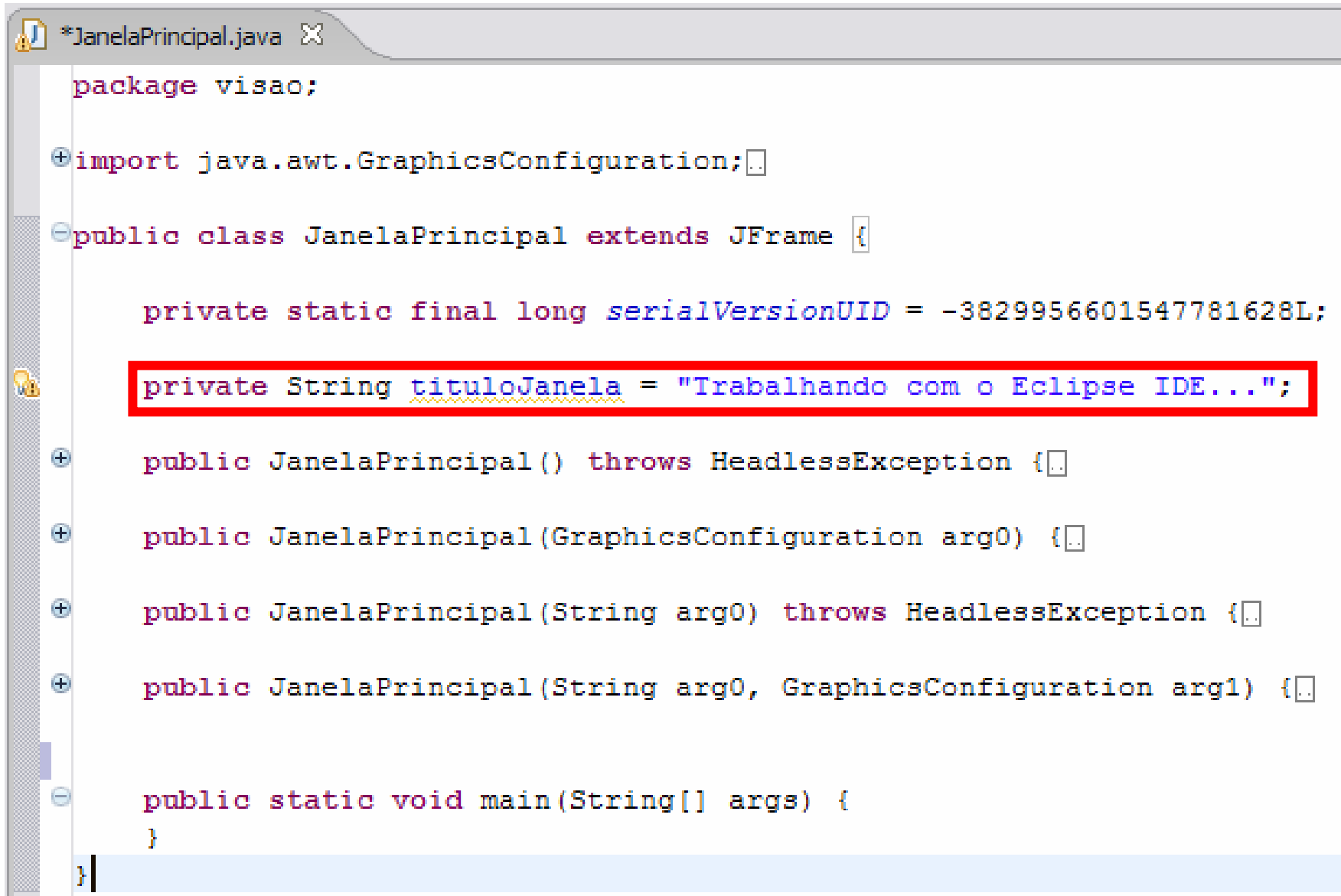
-    public JanelaPrincipal(String arg0) throws HeadlessException {
        super(arg0);
    }

-    public JanelaPrincipal(String arg0, GraphicsConfiguration arg1) {
        super(arg0, arg1);
    }

-    public static void main(String[] args) {
    }

}
```

Adicionando um Atributo à Classe



```
*JanelaPrincipal.java X
package visao;

import java.awt.GraphicsConfiguration;

public class JanelaPrincipal extends JFrame {

    private static final long serialVersionUID = -3829956601547781628L;

    private String tituloJanela = "Trabalhando com o Eclipse IDE...";

    public JanelaPrincipal() throws HeadlessException {

    }

    public JanelaPrincipal(GraphicsConfiguration arg0) {

    }

    public JanelaPrincipal(String arg0) throws HeadlessException {

    }

    public JanelaPrincipal(String arg0, GraphicsConfiguration arg1) {

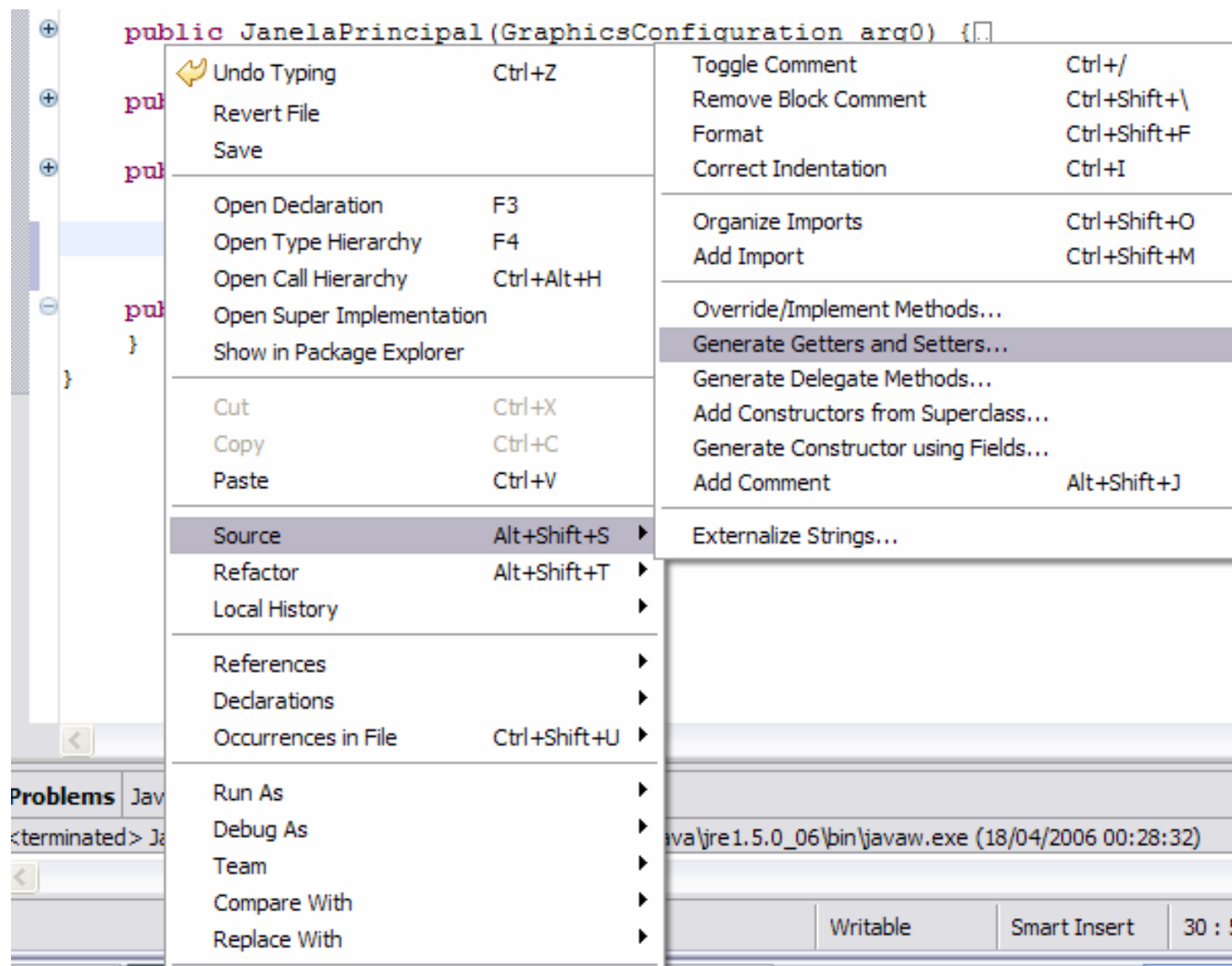
    }

    public static void main(String[] args) {

    }

}
```

Gerando os *Getters/Setters*



Gerando os *Getters/Setters*

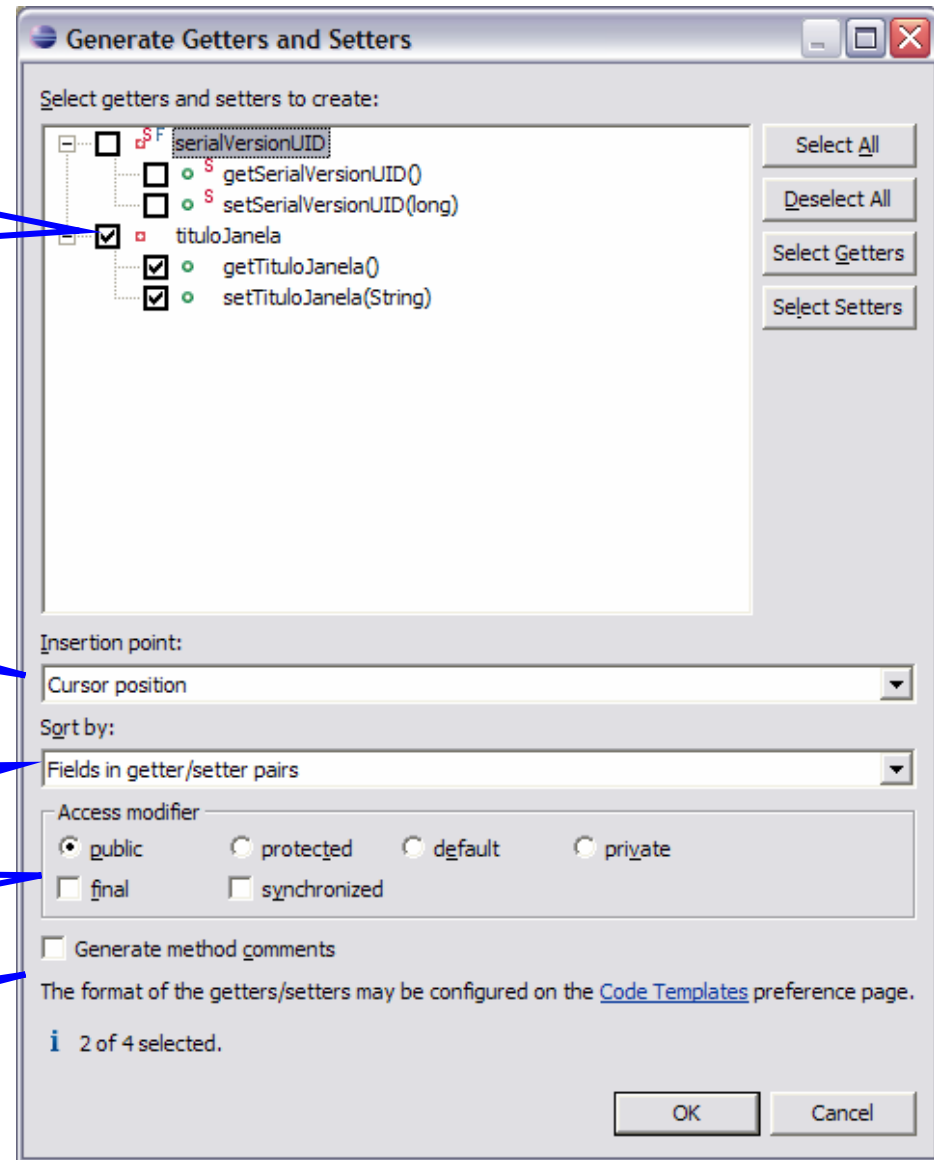
Atributos para os quais
serão gerados os
métodos.

Local onde estarão
posicionados os métodos
gerados.

Ordem dos métodos
getters/setters.

Modificadores dos
métodos.

É possível gerar o
Javadoc dos métodos.



Gerando os *Getters/Setters*

Métodos *getXXX()* retornam o valor de determinado atributo.
Se o atributo for *boolean*, *getXXX()* é substituído por *isXXX()*.

Métodos *setXXX()* definem o valor de determinado atributo.

```
JanelaPrincipal.java X
package visao;

import java.awt.GraphicsConfiguration;

public class JanelaPrincipal extends JFrame {

    private static final long serialVersionUID = -3829956601547781628L;

    private String tituloJanela = "Trabalhando com o Eclipse IDE...";

    public JanelaPrincipal() throws HeadlessException {}

    public JanelaPrincipal(GraphicsConfiguration arg0) {}

    public JanelaPrincipal(String arg0) throws HeadlessException {}

    public JanelaPrincipal(String arg0, GraphicsConfiguration arg1) {}

    public String getTituloJanela() {
        return tituloJanela;
    }

    public void setTituloJanela(String tituloJanela) {
        this.tituloJanela = tituloJanela;
    }

    public static void main(String[] args) {
    }
}
```

Exercícios

- Crie os atributos a seguir (na classe **JanelaPrincipal**):
 - ***larguraJanela***: tipo inteiro, privado, com valor inicial 400;
 - ***alturaJanela***: tipo inteiro, privado, com valor inicial 200;
 - ***janelaRedimensionavel***: tipo booleano, privado, com valor inicial false;
- Para cada atributo acima, crie métodos *getters/setters* usando os recursos do Eclipse IDE

Exercícios

- Depois de gerados os métodos, localize e adicione as seguintes linhas nos métodos a seguir:

```
public void setAlturaJanela(int alturaJanela) {  
    this.alturaJanela = alturaJanela;  
  
    // acrescente a seguinte linha  
    this.setSize(new Dimension(this.getWidth(), alturaJanela));  
}
```

```
public void setLarguraJanela(int larguraJanela) {  
    this.larguraJanela = larguraJanela;  
  
    // acrescente a seguinte linha  
    this.setSize(new Dimension(larguraJanela,  
        this.getHeight()));  
}
```

Exercícios

- No método *main()*, adicione o seguinte código:

```
JanelaPrincipal janela = new JanelaPrincipal();

janela.setSize(new Dimension(janela.getLarguraJanela(),
    janela.getAlturaJanela()));

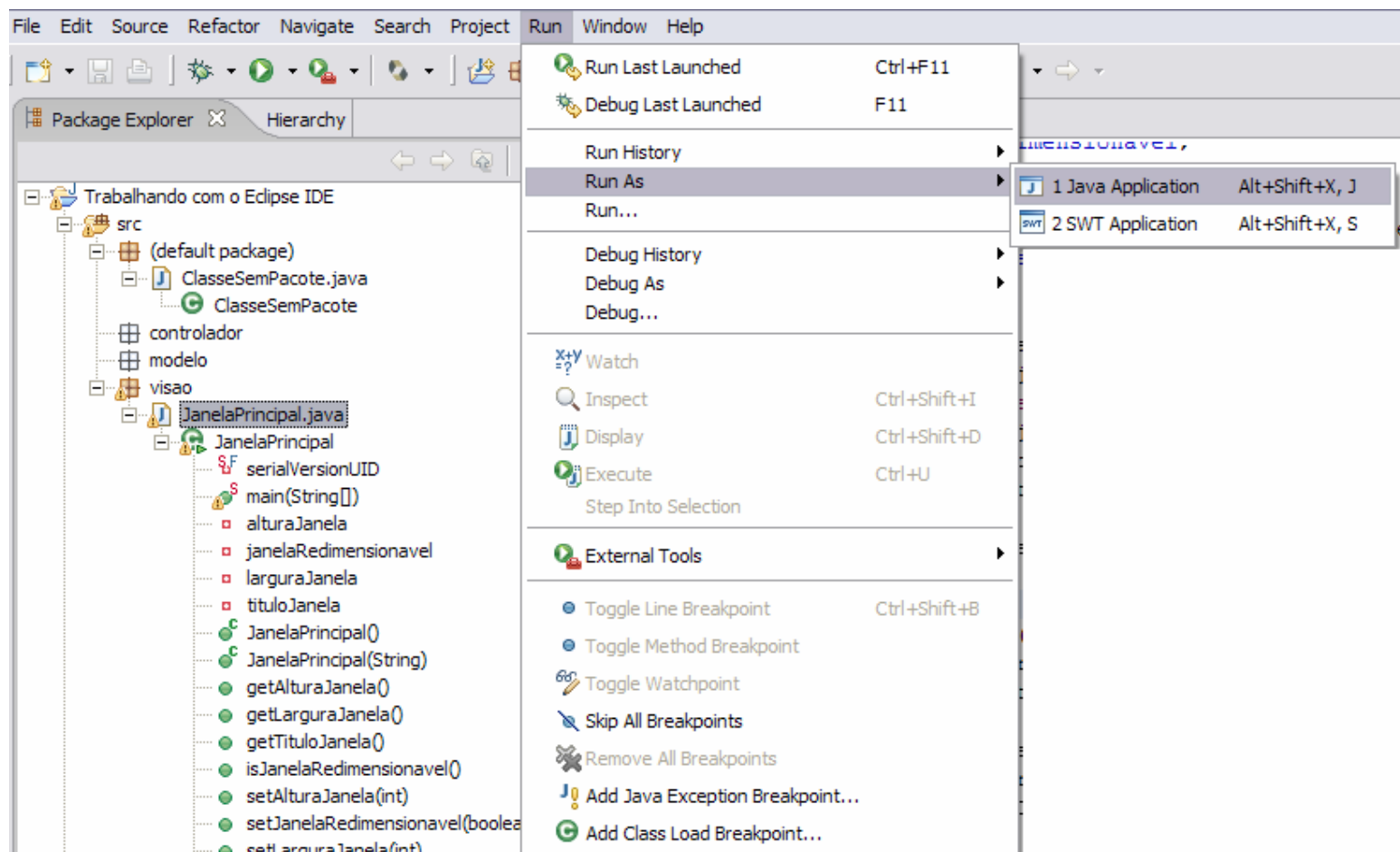
janela.setTitle(janela.getTituloJanela());
janela.setResizable(janela.isJanelaRedimensionavel());
janela.setDefaultCloseOperation(janela.EXIT_ON_CLOSE);

janela.setVisible(true);

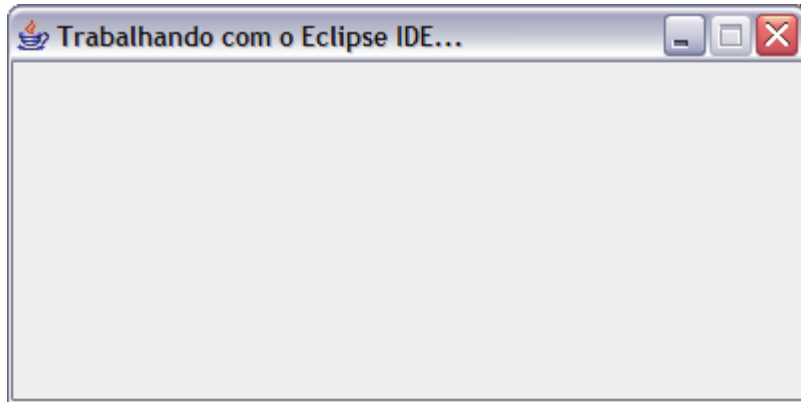
try {
    Thread.sleep(1000 * 5);
    janela.setLarguraJanela(800);
    janela.setAlturaJanela(600);
    System.out.println("Janela redimensionada!!!");
} catch (Exception ex) {
    System.out.println("Erro ao redimensionar a janela...");
    System.exit(0);
}
```

Exercícios

- Executando a aplicação:
 - Com a classe **JanelaPrincipal.java** selecionada, vá no menu *Run / Run As / Java Application*



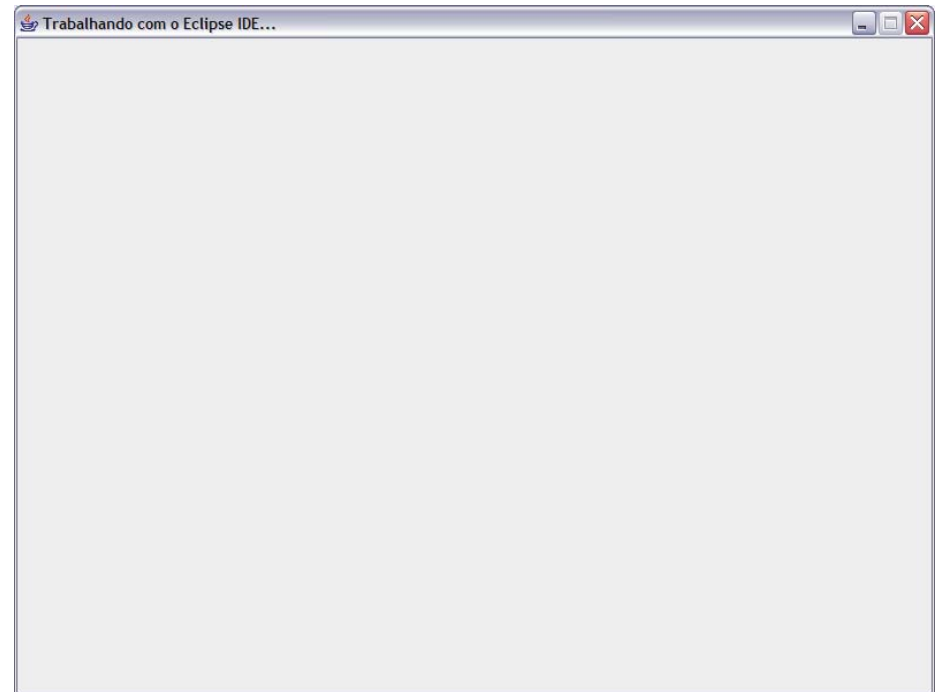
Exercícios



----- 400px -----

----- 200px -----

----- 800px -----



----- 600px -----

Desenvolvimento e Depuração

Ferramentas de Programação Java



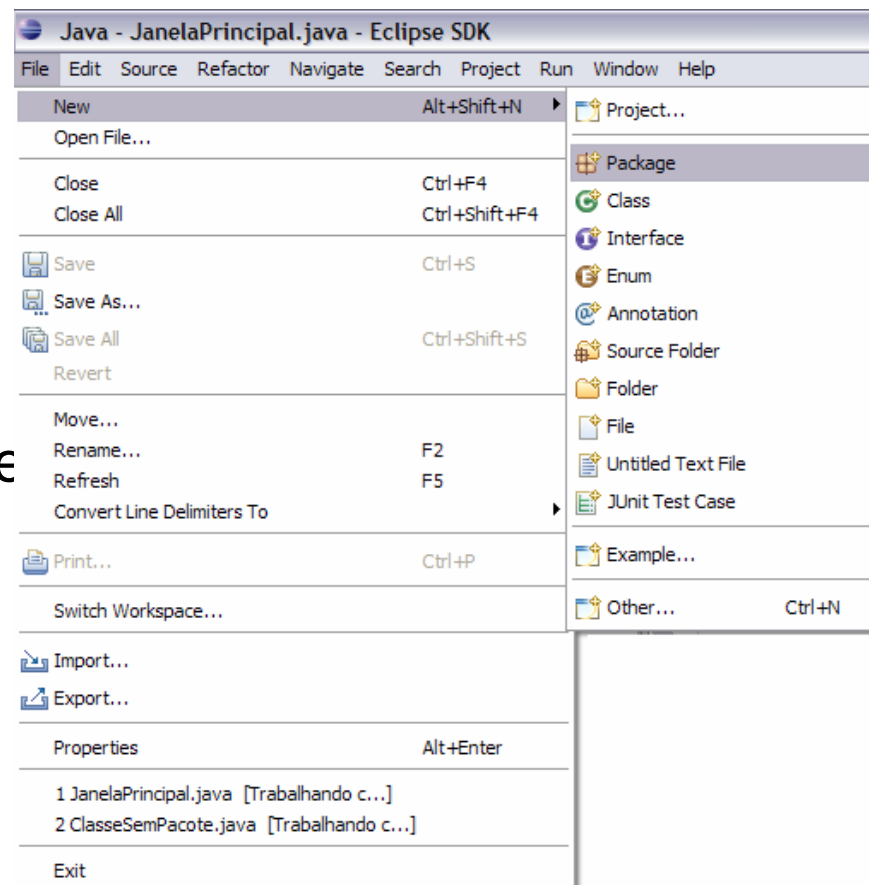
Desenvolvimento e Depuração

- Durante o desenvolvimento usando o Eclipse IDE, o usuário pode:
 - ❑ Incluir, modificar e excluir: arquivos, pastas, classes, pacotes, documentações, etc.
 - ❑ Incluir bibliotecas de terceiros
 - ❑ Modificar a estrutura de organização de uma aplicação
 - ❑ Depurar a aplicação que está sendo desenvolvida em busca de erros
 - ❑ Gerar/exportar o projeto e a sua documentação
 - ❑ Modificar a nomenclatura dos elementos de um projeto sem ter que varrer todos os arquivos em busca das referências de tais elementos

Incluindo Novos Elementos

- O usuário do Eclipse IDE pode incluir, alterar e excluir os seguintes elementos de um projeto Java:

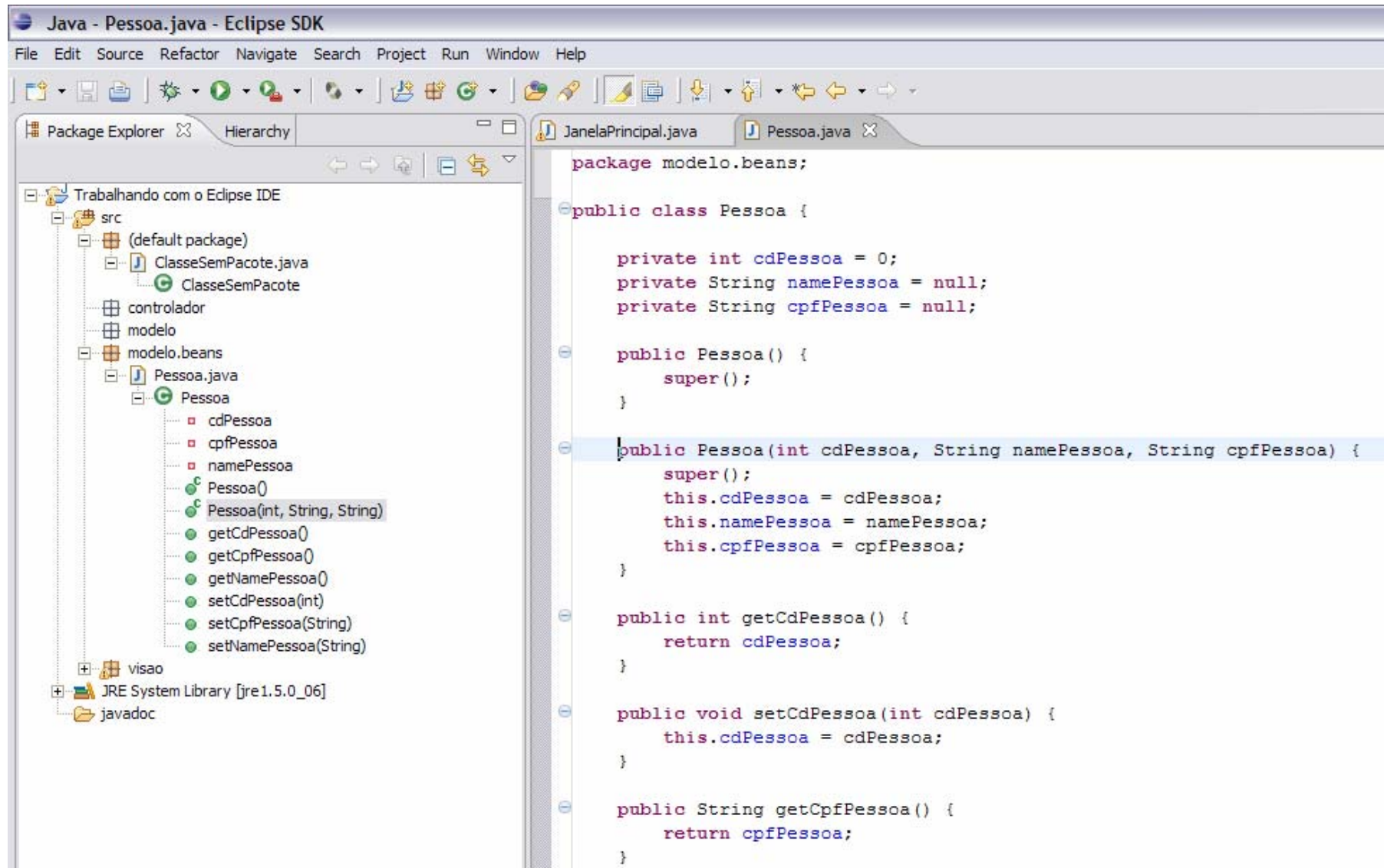
- ❑ Pacotes
- ❑ Classes
- ❑ Interfaces
- ❑ Enum
- ❑ Anotações
- ❑ Pacotes com código-fonte
- ❑ Pastas
- ❑ Arquivos comuns
- ❑ Entre outros



Exercícios

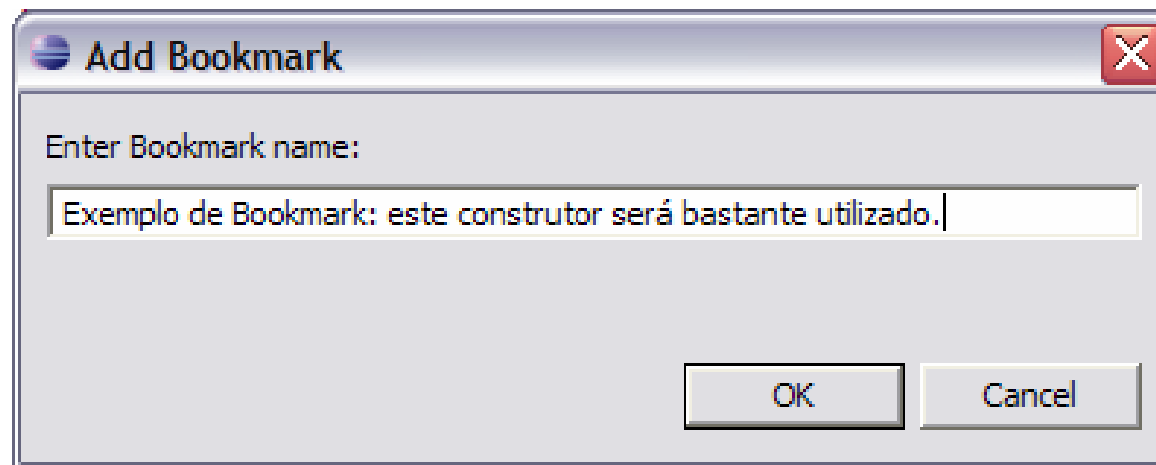
- Inclua o pacote a seguir:
 - ❑ `modelo.beans`
- Em seguida, inclua a classe:
 - ❑ `Pessoa.java` no pacote `modelo.beans`
- Inclua os seguintes elementos na classe ***Pessoa.java***:
 - ❑ ***cdPessoa***: inteiro, privado, valor padrão zero
 - ❑ ***namePessoa***: String, privado, valor padrão null;
 - ❑ ***cpfPessoa***: String, privado, valor padrão null
 - ❑ Construtor que aceita como parâmetros os valores dos atributos definidos acima (fazer automaticamente)
 - ❑ Métodos *getters/setters* para os atributos criados acima

Exercícios



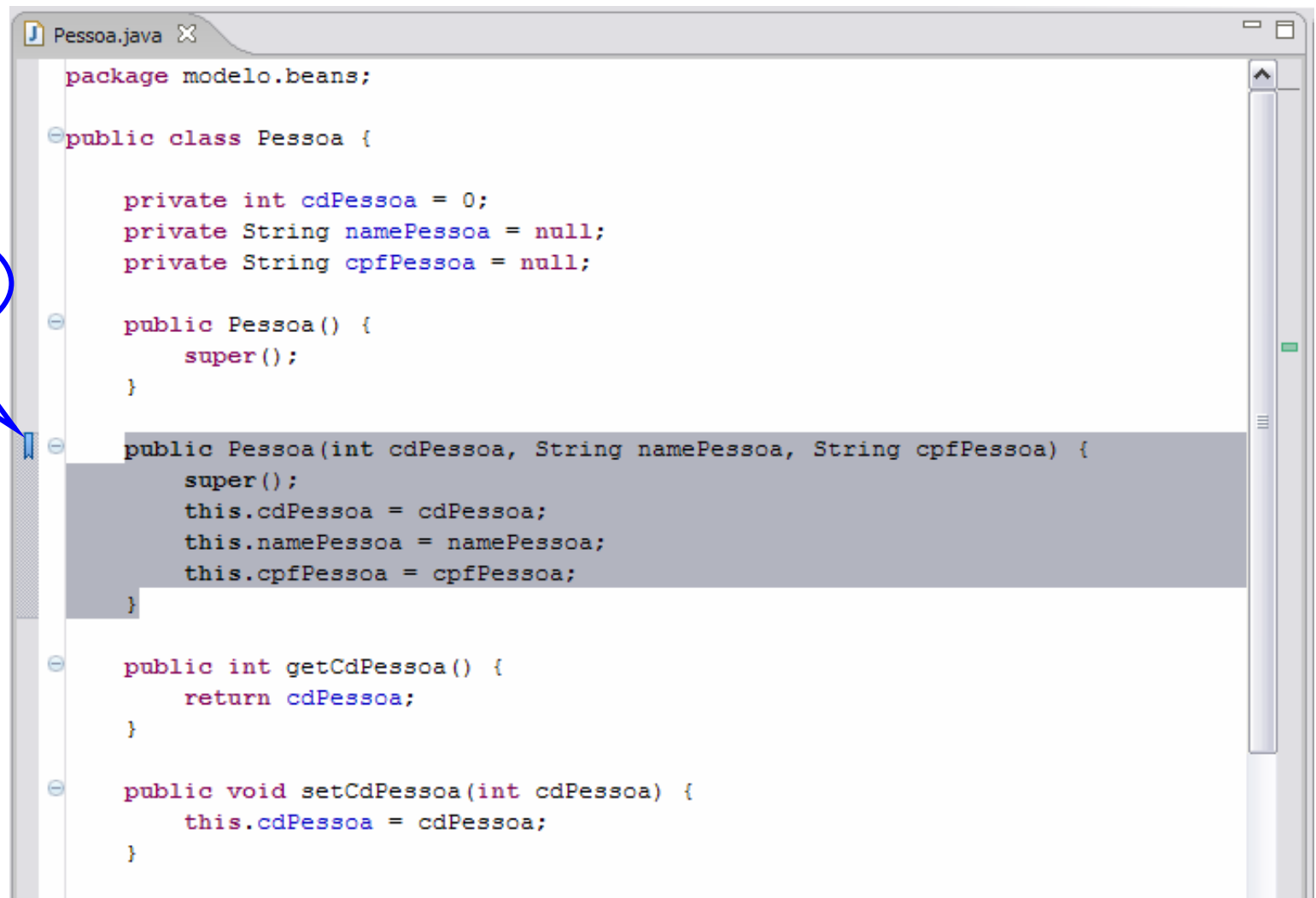
Criando *Bookmarks*

- *Bookmarks* representam marcadores em determinados trechos de código que são bastante visitados
 - Adicionar *bookmarks*:
 - Selecione um trecho de código e vá no menu Edit / Add Bookmark...
 - Aparecerá uma tela pedindo uma descrição para o *bookmark*:



Criando *Bookmarks*

**Bookmark
adicionado**



```
package modelo.beans;

public class Pessoa {

    private int cdPessoa = 0;
    private String namePessoa = null;
    private String cpfPessoa = null;

    public Pessoa() {
        super();
    }

    public Pessoa(int cdPessoa, String namePessoa, String cpfPessoa) {
        super();
        this.cdPessoa = cdPessoa;
        this.namePessoa = namePessoa;
        this.cpfPessoa = cpfPessoa;
    }

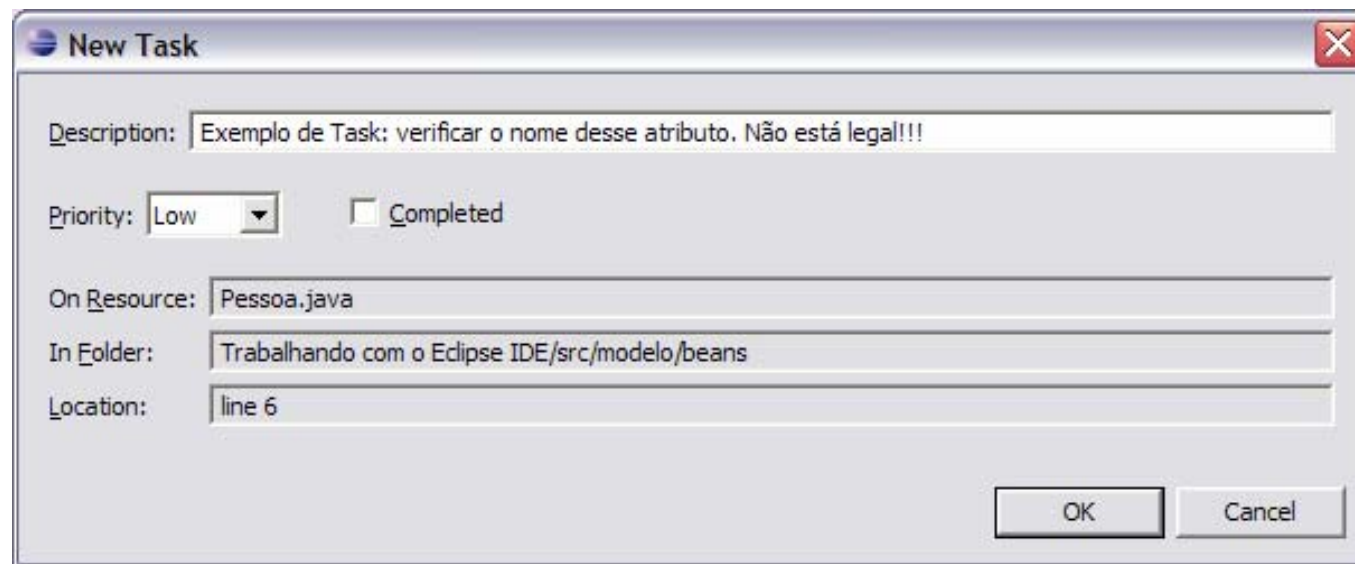
    public int getCdPessoa() {
        return cdPessoa;
    }

    public void setCdPessoa(int cdPessoa) {
        this.cdPessoa = cdPessoa;
    }
}
```

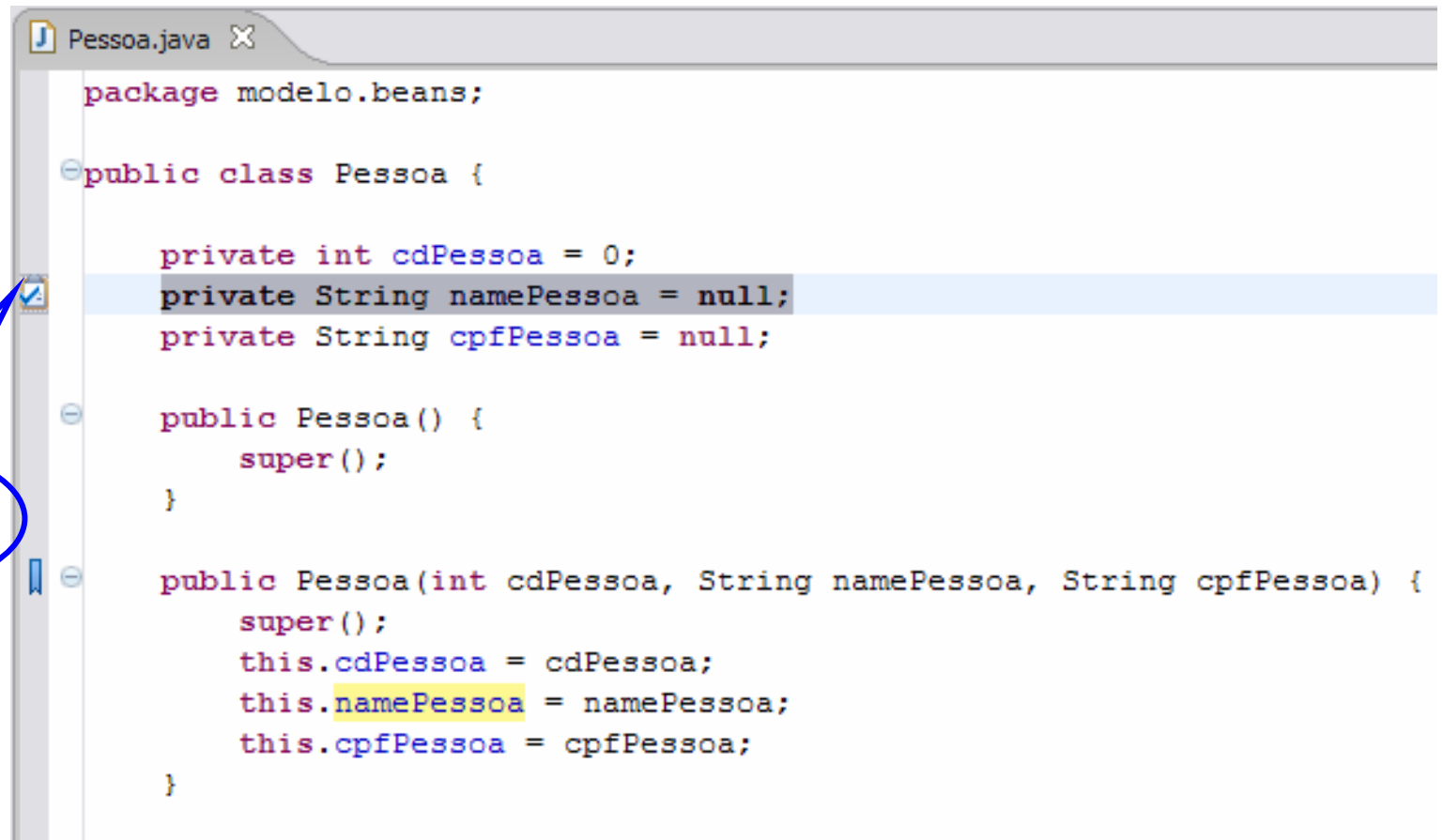
The screenshot shows the Eclipse IDE with a file named 'Pessoa.java' open. The code is a Java class 'Pessoa' in the 'modelo.beans' package. It has private attributes 'cdPessoa' (int), 'namePessoa' (String), and 'cpfPessoa' (String). There are three methods: a no-argument constructor, a constructor with three arguments, and two methods to get and set 'cdPessoa'. A blue bookmark icon is placed in the left margin next to the constructor with three arguments. A blue speech bubble points to this icon with the text 'Bookmark adicionado'.

Criando *Tasks*

- *Tasks* são tarefas que precisam ser realizadas para que um código esteja de acordo com a especificação de uma aplicação e reflita a correta implementação deste
 - ❑ Adicionar *tasks*:
 - ❑ Selecione um trecho de código e vá no menu *Edit / Add Task...*
 - ❑ Aparecerá uma tela pedindo: descrição, prioridade e se a tarefa foi completada ou não



Criando *Tasks*



```
package modelo.beans;

public class Pessoa {

    private int cdPessoa = 0;
    private String namePessoa = null;
    private String cpfPessoa = null;

    public Pessoa() {
        super();
    }

    public Pessoa(int cdPessoa, String namePessoa, String cpfPessoa) {
        super();
        this.cdPessoa = cdPessoa;
        this.namePessoa = namePessoa;
        this.cpfPessoa = cpfPessoa;
    }
}
```

**Task
adicionada**

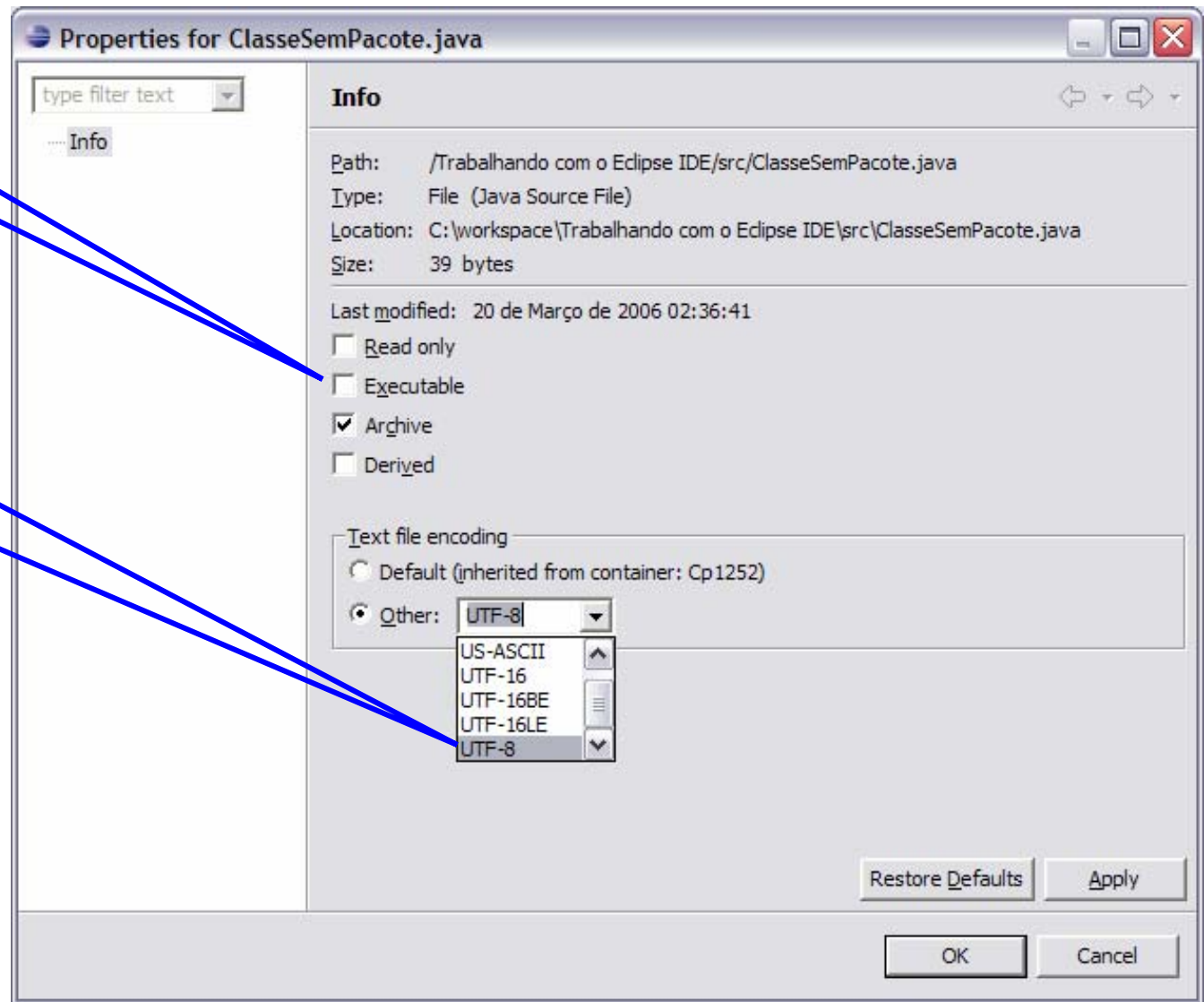
Codificação dos Caracteres

- É possível modificar a codificação dos caracteres de arquivos de um projeto do Eclipse IDE:
 - Cp1252 (padrão), ISO-8859-1, US-ASCII, UTF-16, UTF-16BE, UTF-16LE e UTF-8 (bastante utilizado para caracteres com acentuação)
 - Clique com o botão direito sobre o arquivo desejado e escolha a opção **Properties**
 - Em seguida, no item, **Text file encoding** selecione a codificação desejada, como mostra a tela a seguir

Codificação dos Caracteres

Atributos
do arquivo

Mudando a
codificação
para UTF-8



Trabalhando com o Menu *Source*

- Este menu é importantíssimo do ponto de vista de produtividade do código que está sendo desenvolvido
- **Opções:**
 - ❑ ***Toggle Comment (Ctrl + /)***: comenta/descomenta uma linha inteira
 - ❑ ***Add Block Comment (Ctrl + Shift + /)***: comenta um bloco de comandos inteiro
 - ❑ ***Remove Block Comment (Ctrl + Shift + \)***: remove o comentário de um bloco de comandos inteiro
 - ❑ ***Shift Right e Shift Left***: identa o código para a direita ou para a esquerda
 - ❑ ***Format (Ctrl + Shift + F)***: identa todo o código da classe

Trabalhando com o Menu *Source*

■ Opções:

- ❑ ***Correct Indentation (Ctrl + I)***: corrige a indentação somente dos elementos selecionados
- ❑ ***Sort Members***: ordena os elementos da classe
- ❑ ***Organize Imports e Add Import***: reagrupa os *imports* ou adiciona cláusulas *import* à classe
- ❑ ***Override / Implement Methods***: redefine métodos herdados ou implementa métodos abstratos da superclasse
- ❑ ***Generate Getters and Setters***: gera os métodos de acesso e definição de atributos

Trabalhando com o Menu *Source*

■ Opções:

- ❑ ***Generate Delegate Methods:*** gera os métodos pertinentes aos atributos não-primitivos. Ex. String
- ❑ ***Generate Constructors using Fields:*** gera construtores para a classe utilizando os atributos declarados como parâmetros
- ❑ ***Add Constructors from Superclass:*** gera construtores baseados nos construtores da superclasse
- ❑ ***Add Comment:*** adiciona comentário à um elemento de uma classe baseado no modelo do Javadoc
- ❑ ***Surround with try/catch Block:*** inclui tratamento de exceção em um conjunto de instruções
- ❑ ***Externalize Strings:*** retira todas as Strings de uma classe e as referencia em um arquivo .properties. *Será mostrado na prática*

Exercícios

- Com a classe ***Pessoa.java***, usando o menu **Source**:
 - Adicione um construtor que aceita como parâmetro o nome e o CPF de uma pessoa e associa o valor 0 (zero) ao *cdPessoa*
 - Adicione o seguinte comentário ao atributo *cdPessoa*:
 - “Utilizado como chave primária de Pessoa”
 - Adicione o seguinte comentário ao atributo *cpfPessoa*:
 - “CPF da pessoa formado apenas pelos 11 dígitos”
 - Gerar o método ***toString()*** da superclasse ***Object*** e substituir o código gerado por:
 - ```
return cdPessoa + " - " + nomePessoa + " - " + cpfPessoa;
```

# Exercícios

Comentário para  
o atributo  
*cdPessoa*

Comentário para  
o atributo  
*cpfPessoa*

Construtor com  
os parâmetros:  
nome e cpf

Linha adicionada  
“manualmente”  
ao construtor

```
Pessoa.java X
package modelo.beans;

public class Pessoa {

 /**
 * Utilizado como chave primária de Pessoa
 */
 private int cdPessoa = 0;

 private String namePessoa = null;

 /**
 * CPF da pessoa formado apenas pelos 11 dígitos
 */
 private String cpfPessoa = null;

 public Pessoa(String namePessoa, String cpfPessoa) {
 super();
 this.namePessoa = namePessoa;
 this.cpfPessoa = cpfPessoa;

 this.cdPessoa = 0;
 }
}
```

# Exercícios

- Com a classe ***JanelaPrincipal.java***, usando o menu ***Source***:
  - ❑ Adicione um construtor que aceita como parâmetro o título da janela, a largura e a altura da janela, e se a janela pode ou não ser redimensionável
  - ❑ Adicione o seguinte comentário ao atributo *serialVersionUID*:
    - “Utilizado para controle de versão, por exemplo pelo CVS”
  - ❑ **Por quê você utilizaria o comando *Externalize Strings*? Qual(is) a(s) vantagem(ns) e desvantagem(ns) desse recurso? Tem alguma coisa a ver com manutenção de software?**

# Exercícios

Comentário para o atributo *serialVersionUID*

Construtor com os parâmetros: título, largura, altura e se redimensionável

```
JanelaPrincipal.java X
package visao;

import java.awt.Dimension;

public class JanelaPrincipal extends JFrame {

 /**
 * Utilizado para controle de versão, por exemplo pelo CVS
 */
 private static final long serialVersionUID = -3829956601547781628L;

 private String tituloJanela = "Trabalhando com o Eclipse IDE...";
 private int larguraJanela = 400;
 private int alturaJanela = 200;
 private boolean janelaRedimensionavel = false;

 public JanelaPrincipal(String tituloJanela, int larguraJanela,
 int alturaJanela, boolean janelaRedimensionavel) throws HeadlessException {
 super();
 this.tituloJanela = tituloJanela;
 this.larguraJanela = larguraJanela;
 this.alturaJanela = alturaJanela;
 this.janelaRedimensionavel = janelaRedimensionavel;
 }
}
```

# Trabalhando com o Menu *Refactor*

- Utilizado na manutenção de código e na busca por referência aos elementos que serão “refatorados”
- **Opções:**
  - ***Rename:*** permite renomear um elemento, além de buscar as antigas referências e substituí-las pelas novas. Ex.: atributo `nomePessoa` da classe `Pessoa`
  - ***Move:*** move o elemento para outra classe
  - ***Change Method Signature:*** modifica a assinatura do método selecionado, além de modificar a chamada ao método em outros locais
  - ***Convert Anonymous Class to Nested:*** converte uma classe anônima em uma classe interna
  - ***Move Member Type to New File:*** move uma classe interna para uma nova classe

---

# Trabalhando com o Menu *Refactor*

- ***Push Down***: move um membro de uma classe para uma subclasse
- ***Pull Up***: faz o inverso do Push Down
- ***Extract Interface***: cria uma nova interface com um conjunto de métodos da classe atual
- ***Inline***: busca por ocorrências de elementos uma única vez e que não são utilizados e as remove
- ***Extract Method***: transforma uma instrução em um método
- ***Convert Local Variable to Field***: converte uma variável local em um atributo de instância
- ***Encapsulate Field***: torna um atributo de instância em privado e cria *getter* e *setter* para ele



---

# Exercícios

- Usando as opções do menu ***Refactor***:
  - Modifique o nome do atributo ***namePessoa*** para ***nomePessoa***
    - Todas as referências à ***namePessoa*** foram substituídas por ***nomePessoa***?
    - Quais não foram? Por quê?

# Exercícios

## ■ Na classe ***JanelaPrincipal***:

- ❑ Comente (usando o menu *Source*) todo o código do método *main()*;
- ❑ Inclua o seguinte código no método *main()*:

```
Pessoa pessoa = new Pessoa();
pessoa.setCdPessoa(10);
pessoa.setNomePessoa("Joao da Silva");
pessoa.setCpfPessoa("12345678934");

System.out.println(pessoa);
System.out.println(pessoa.toString());
```
- ❑ Execute a aplicação pelo menu ***Run / Run As / Java Application***
- ❑ As duas últimas linhas imprimiram a mesma informação? Se sim, qual a diferença entre as duas últimas instruções?

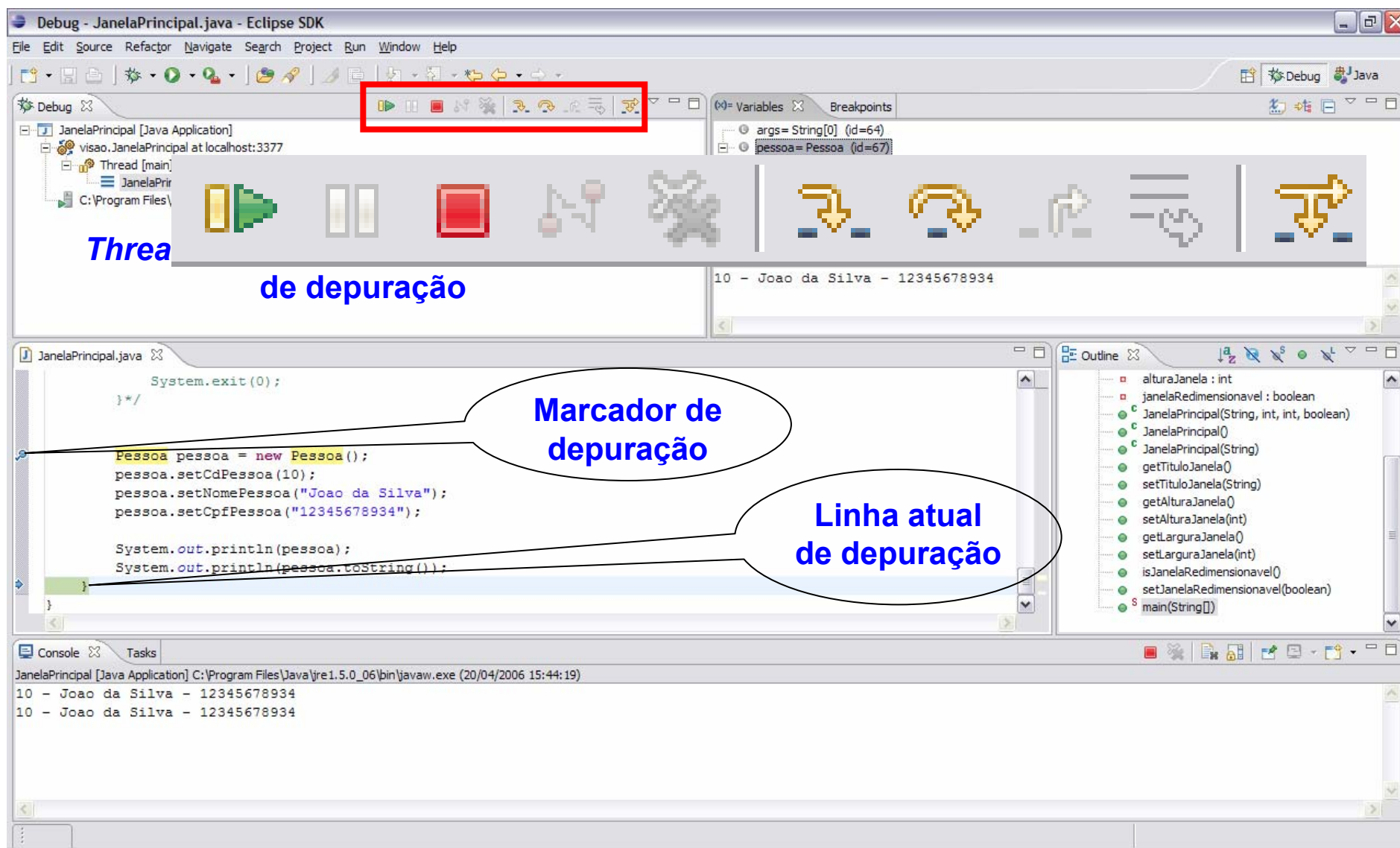
# Depurando uma Aplicação

- Depurar uma aplicação significa demarcar certas regiões de código onde pretende-se inspecioná-lo em busca de erros ou situações que comprometem o correto funcionamento de uma aplicação
- O Eclipse IDE permite depurar as aplicações Java
- Para isso, é necessário:
  - Demarcar os pontos (**breakpoints**) em que a aplicação será inspecionada
  - Percorrer passo a passo as instruções:
    - É possível inspecionar a execução de uma instrução, por exemplo (**Step Into – F5**) ou somente executá-la sem inspecionar o seu código (**Step Over – F6**)

# Depurando uma Aplicação

- Antes de iniciar a depuração é necessário demarcar o(s) ponto(s) em que a aplicação será inspecionada
- A depuração pode ser feita por meio do menu **Run / Debug As / Java Application**
- Note que uma tela surgirá informando que o tipo de execução pretendida (depuração) está associada com a perspectiva “Debug” e se você gostaria de mudar para tal perspectiva
  - Ao escolher sim, veja que a perspectiva do Eclipse IDE agora mudou e as janelas internas foram reorganizadas
- Note, também, que o menu **Run** agora possui somente opções relacionadas à depuração
- Ao final da depuração é só escolher a perspectiva “Java” novamente

# Ambiente de Depuração



# Gerando a Documentação das Classes

- Todas as classes existentes em um projeto devem possuir uma documentação no estilo Javadoc – padrão utilizado para estruturar os elementos de uma classe
- O Javadoc de uma classe é dividido em 3 partes:
  - Relação dos pacotes do projeto
  - Relação das classes pertencentes à um pacote
  - Documentação dos elementos de determinada classe
- O Javadoc é, normalmente, apresentado na forma de hipertexto (páginas HTML)
  - Sua navegação acontece por meio de *hyperlinks*

# Exemplo de Javadoc

Relação de  
pacotes de um  
projeto

The screenshot shows the Javadoc index page for the Java 2 Platform SE 5.0. The left sidebar lists various packages and classes. The main content area displays the documentation for the `String` class, including its inheritance hierarchy, implemented interfaces, and a description of the class. Callouts highlight specific parts of the page:

- Relação de pacotes de um projeto**: Points to the list of packages in the left sidebar.
- Especificação dos elementos de determinada classe**: Points to the `Class String` header in the main content area.
- Relação das classes e interfaces de um pacote**: Points to the `All Implemented Interfaces:` section in the main content area.

---

# Configurando a Geração do Javadoc no Eclipse IDE

- Definindo o local em que será gerado o Javadoc do projeto:
  - Selecione o projeto
  - Menu ***File / New / Folder***
    - Crie a pasta “javadoc”
  - Menu ***Project / Generate Javadoc...***



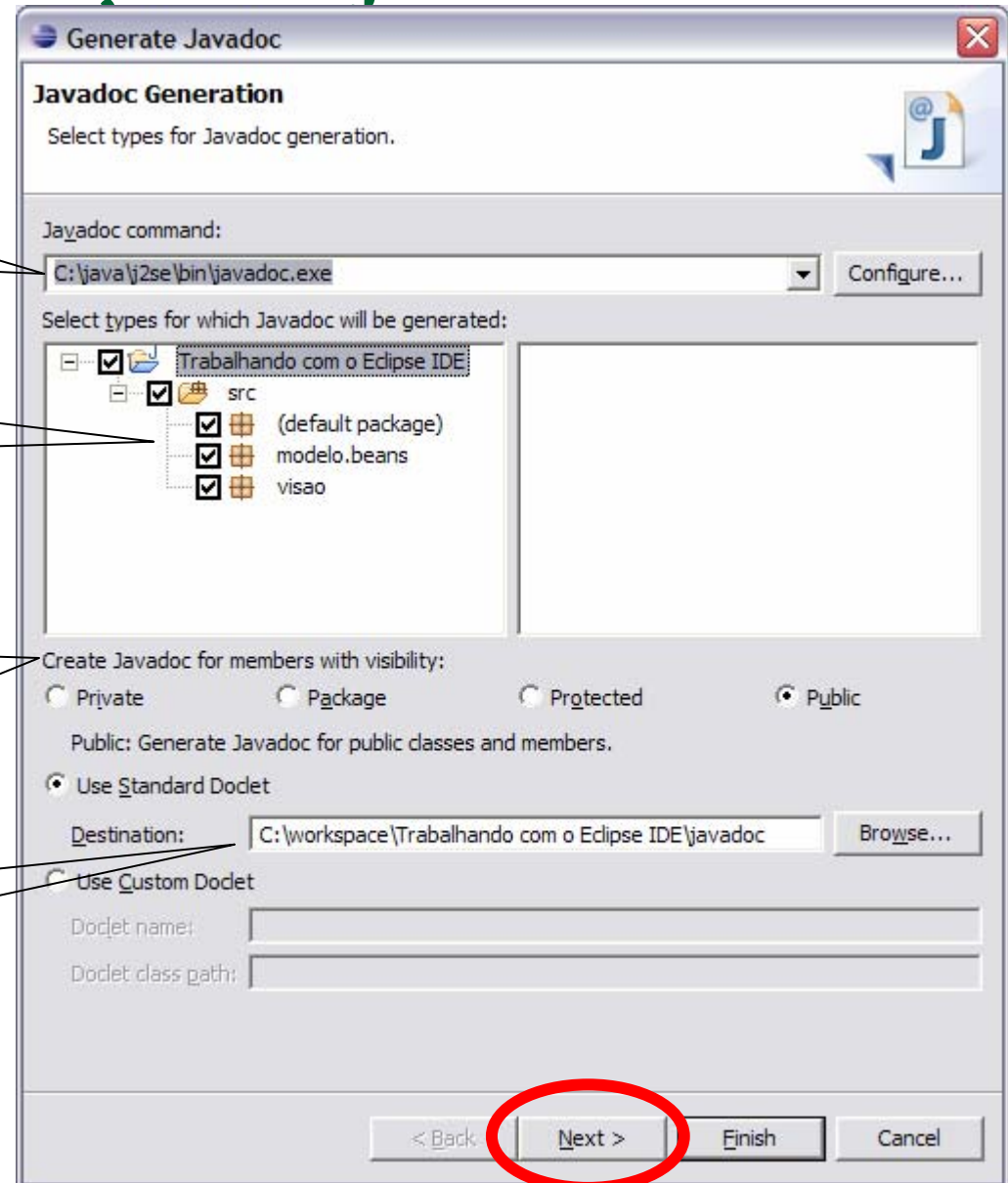
# Configurando a Geração do Javadoc no Eclipse IDE

Localização do arquivo  
“javadoc.exe” (Windows)

Pacotes para os quais  
será gerado o Javadoc

Tipo de elemento das  
classes que será gerado o  
Javadoc

Local onde será gerado o  
Javadoc do projeto



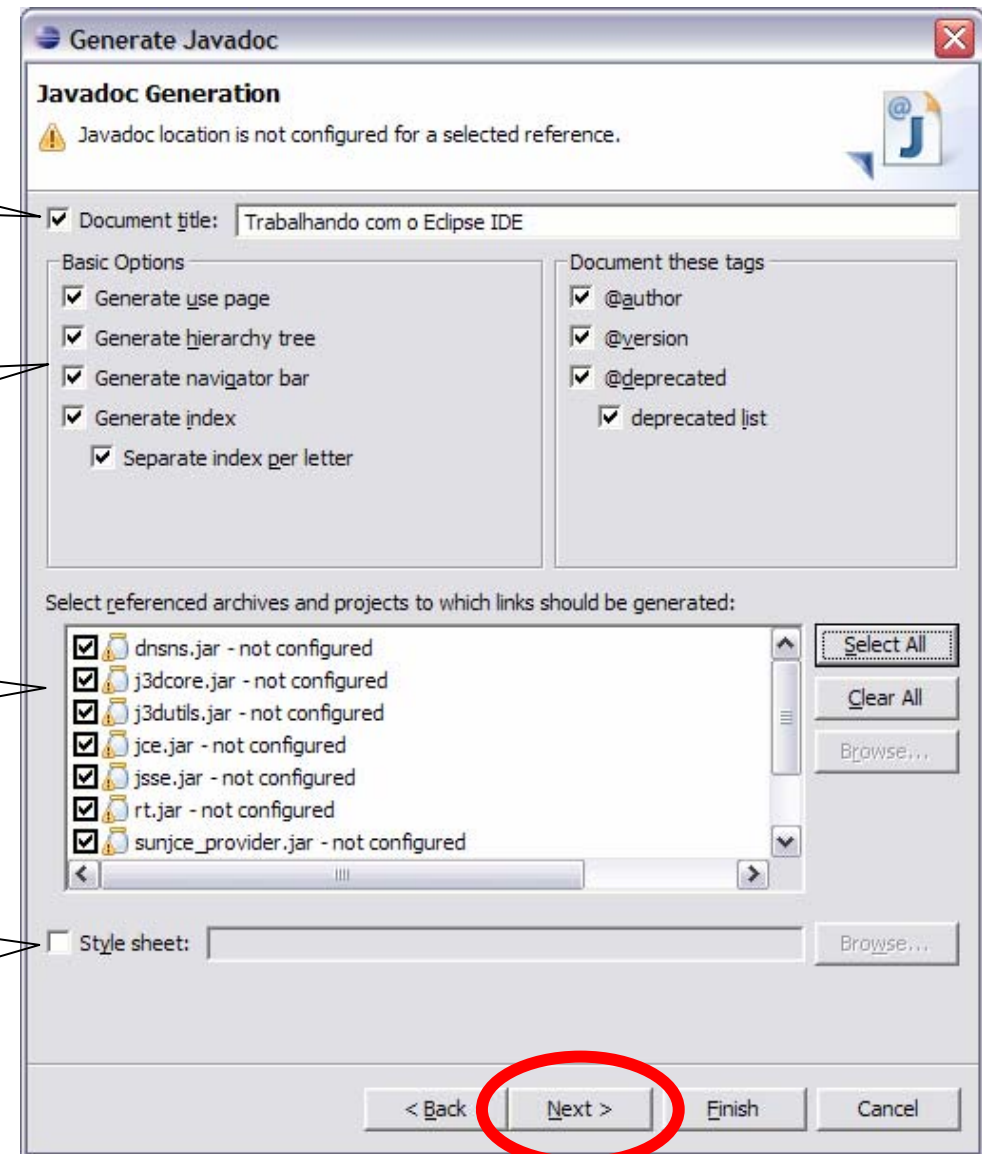
# Configurando a Geração do Javadoc no Eclipse IDE

Título do Javadoc

Algumas opções de elementos a serem gerados

Classes que podem ser referenciadas no Javadoc

Pode-se utilizar uma folha de estilos diferente do padrão do Javadoc



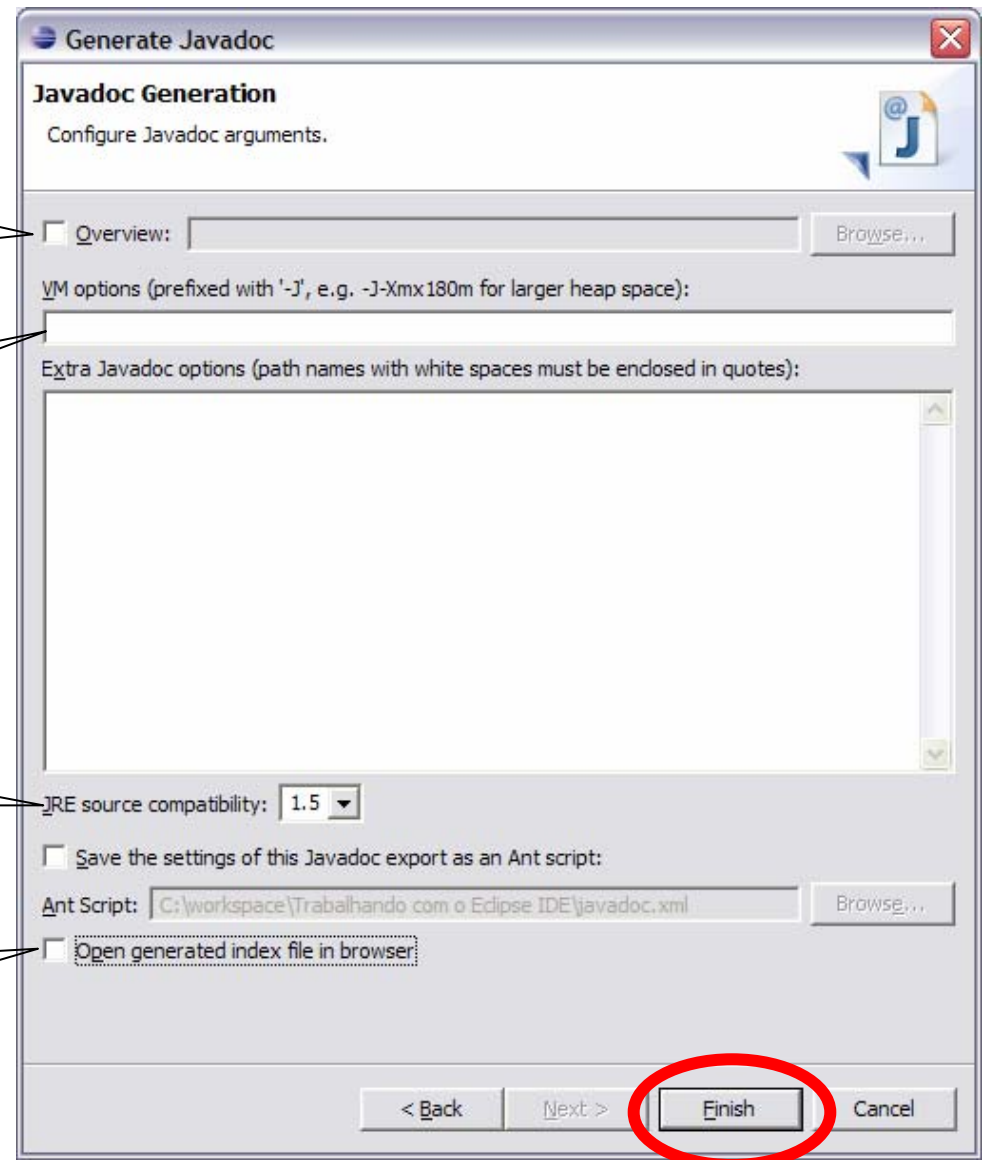
# Configurando a Geração do Javadoc no Eclipse IDE

Pode-se incluir uma página HTML inicial com um resumo do Javadoc

Argumentos para a JVM na geração do Javadoc

Versão do Java para a geração do Javadoc

Abrir o arquivo "index.html" em um navegador



# Gerando o Javadoc

## Páginas HTML do Javadoc

The screenshot displays the Eclipse IDE interface during Javadoc generation. On the left, the Project Explorer shows the 'javadoc' folder containing various HTML files like 'index.html', 'allclasses-frame.html', and 'overview-summary.html'. The central Editor window shows the source code of 'JanelaPrincipal' with a Javadoc comment for the constructor. The bottom Console window shows the progress of Javadoc generation, listing the files being generated and the final warning count.

```
import java.awt.Dimension;

public class JanelaPrincipal extends JFrame {

 /**
 * Utilizado para controle de versão, por exemplo pelo CVS
 */
 private static final long serialVersionUID = -3829956601547781628L;

 private String tituloJanela = "Trabalhando com o Eclipse IDE...";
 private int larguraJanela = 400;
 private int alturaJanela = 200;
 private boolean janelaRedimensionavel = false;

 public JanelaPrincipal(String tituloJanela, int larguraJanela,
 int alturaJanela, boolean janelaRedimensionavel) throws HeadlessException {
 super();
 }
}
```

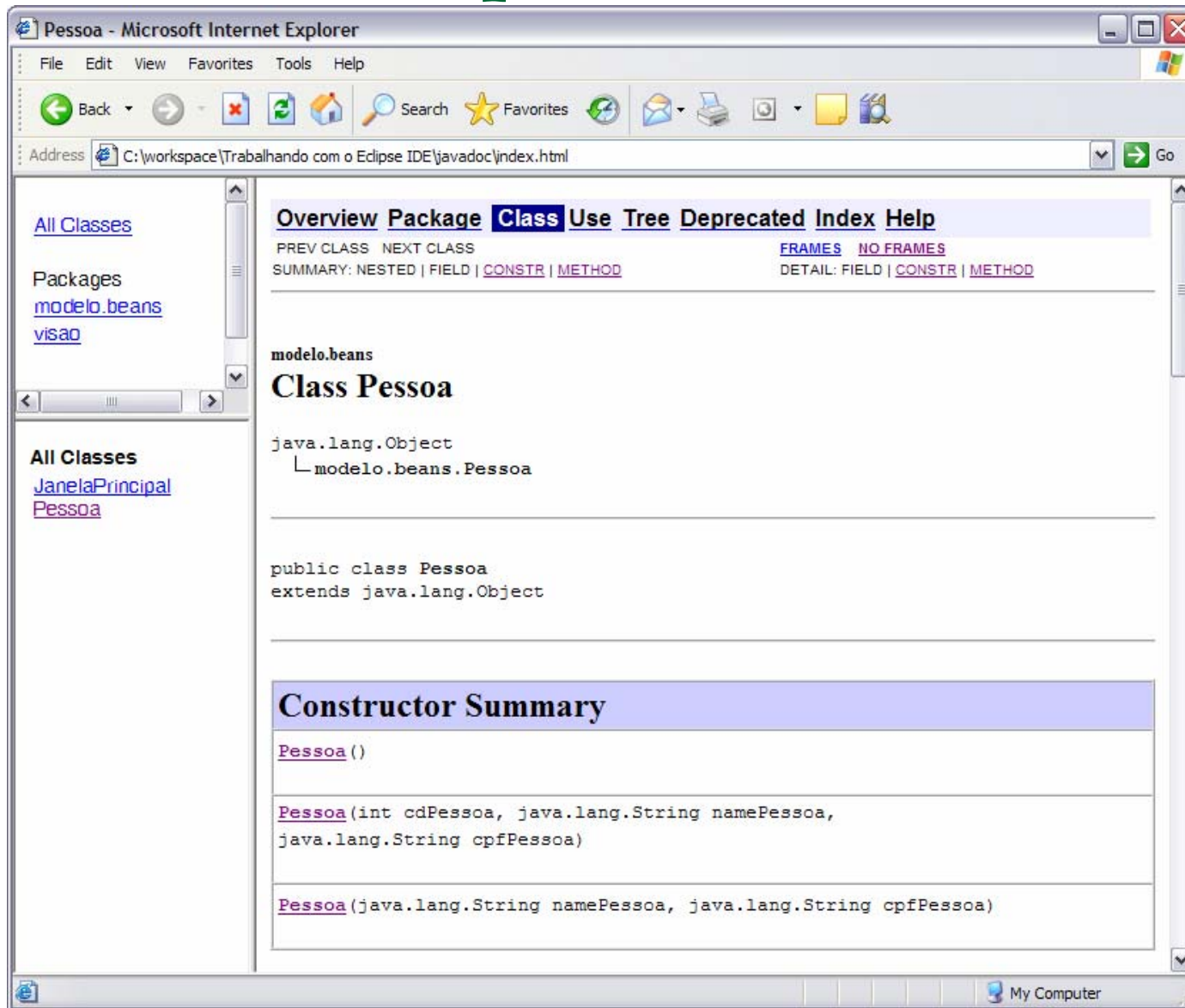
Problems Javadoc Declaration Console

<terminated> Javadoc Generation  
Generating C:\workspace\Trabalhando com o Eclipse IDE\javadoc\index-files\index-5.html...  
Generating C:\workspace\Trabalhando com o Eclipse IDE\javadoc\index-files\index-6.html...  
Generating C:\workspace\Trabalhando com o Eclipse IDE\javadoc\index-files\index-7.html...  
Generating C:\workspace\Trabalhando com o Eclipse IDE\javadoc\index-files\index-8.html...  
Generating C:\workspace\Trabalhando com o Eclipse IDE\javadoc\deprecated-list.html...  
Building index for all classes...  
Generating C:\workspace\Trabalhando com o Eclipse IDE\javadoc\allclasses-frame.html...  
Generating C:\workspace\Trabalhando com o Eclipse IDE\javadoc\allclasses-noframe.html...  
Generating C:\workspace\Trabalhando com o Eclipse IDE\javadoc\index.html...  
Generating C:\workspace\Trabalhando com o Eclipse IDE\javadoc\overview-summary.html...  
Generating C:\workspace\Trabalhando com o Eclipse IDE\javadoc\help-doc.html...  
Generating C:\workspace\Trabalhando com o Eclipse IDE\javadoc\stylesheet.css...  
1 warning

index.html - Trabalhando com o Eclipse IDE/javadoc

## Log de geração do Javadoc

# Javadoc Gerado para a Classe Pessoa



---

# Exercício Futuro...

- Leia sobre o Javadoc no site:

<http://java.sun.com/j2se/1.5.0/docs/guide/javadoc/index.html>

- Aprenda as principais características e *tags* dessa ferramenta e documente todas as classes do projeto implementadas até agora

---

# *Plugins* do Eclipse

---

Ferramentas de Programação Java






---

# *Plugins* no Eclipse IDE

- Os *plugins* são aplicações desenvolvidas por terceiros e que adicionam funcionalidades a um determinado ambiente, como é o caso do Eclipse IDE
- Existem dezenas de *plugins* desenvolvidos para o Eclipse:
  - Estão divididos em várias categorias, como em <http://eclipse-plugins.2y.net/eclipse/index.jsp>
- Além de ser possível adicionar *plugins* de terceiros ao Eclipse, o ambiente permite também o desenvolvimento de *plugins*




# Categorias de *Plugins*



[home](#)  
[what's new](#)  
[categories](#)  
[search](#)  
[promotion progr.](#)  
[contact](#)  
[links](#)  
[login / register](#)  
  
[Eclipse Wiki](#)

registered users:  
12529  
active visitors: 1091  
pages per minute: 56

[RSS](#)

hosting provided by  


## Plugin categories

|                                                |                                            |                                                  |
|------------------------------------------------|--------------------------------------------|--------------------------------------------------|
| <a href="#">All (1174)</a>                     | <a href="#">Languages - Latex (3)</a>      | <a href="#">Source Code Formatter (4)</a>        |
| <a href="#">Ant (11)</a>                       | <a href="#">Languages - Macromedia (5)</a> | <a href="#">Team (8)</a>                         |
| <a href="#">AspectJ (6)</a>                    | <a href="#">Languages - others (25)</a>    | <a href="#">Testing (49)</a>                     |
| <a href="#">Bug Tracker (8)</a>                | <a href="#">LDAP (7)</a>                   | <a href="#">Tomcat (7)</a>                       |
| <a href="#">Business Process Tools (11)</a>    | <a href="#">Logging (11)</a>               | <a href="#">Tools (72)</a>                       |
| <a href="#">Code Generation (19)</a>           | <a href="#">Misc (21)</a>                  | <a href="#">Tools (jar,classpath) (7)</a>        |
| <a href="#">Code Generation/Modelling (13)</a> | <a href="#">Mobile/PDA (13)</a>            | <a href="#">Tools - ContextMenu (20)</a>         |
| <a href="#">Code mngt (37)</a>                 | <a href="#">Modelling (28)</a>             | <a href="#">Tools - Editor Enhancements (26)</a> |
| <a href="#">Com,Corba,Idl,... (10)</a>         | <a href="#">Network (10)</a>               | <a href="#">Tools - for eclipse (30)</a>         |
| <a href="#">Database (18)</a>                  | <a href="#">Obsolete (29)</a>              | <a href="#">Tools - Math (4)</a>                 |
| <a href="#">Database Persistence (21)</a>      | <a href="#">Patterns (6)</a>               | <a href="#">Tools - Navigation (20)</a>          |
| <a href="#">Decompiler (7)</a>                 | <a href="#">Profiling (14)</a>             | <a href="#">Tools - special editors (13)</a>     |
| <a href="#">Deployment (22)</a>                | <a href="#">Project management (8)</a>     | <a href="#">Tools - WebSearch (4)</a>            |
| <a href="#">Distribution Package (5)</a>       | <a href="#">Report (6)</a>                 | <a href="#">Tutorial (37)</a>                    |
| <a href="#">Documentation (15)</a>             | <a href="#">Rich Client (24)</a>           | <a href="#">UI (37)</a>                          |
| <a href="#">Entertainment (39)</a>             | <a href="#">RSS (5)</a>                    | <a href="#">UI components (23)</a>               |
| <a href="#">Info - Website (9)</a>             | <a href="#">SAP (3)</a>                    | <a href="#">UML (28)</a>                         |
| <a href="#">J2EE development platform (60)</a> | <a href="#">SCM (23)</a>                   | <a href="#">Web (47)</a>                         |
| <a href="#">Languages (36)</a>                 | <a href="#">SCM - CVS (10)</a>             | <a href="#">Web Service (28)</a>                 |
| <a href="#">Languages - C#.Net (13)</a>        | <a href="#">Source Code Analyzer (33)</a>  | <a href="#">XML (40)</a>                         |
| <a href="#">Languages - Javascript (5)</a>     |                                            |                                                  |

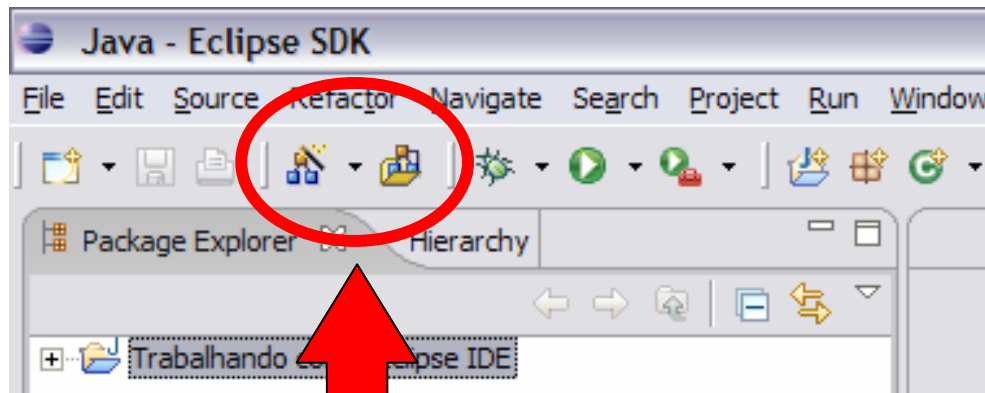
# Adicionando *Plugins* de Terceiros

- Para tanto é necessário primeiramente fazer o download do *plugin*
- Vamos utilizar o *plugin* **Omondo** como exemplo:
  - Download:  
[http://www.eclipsedownload.com/eclipseUML\\_E312\\_freeEdition\\_2.1.0.20060320.jar](http://www.eclipsedownload.com/eclipseUML_E312_freeEdition_2.1.0.20060320.jar)
  - O arquivo jar quando executado instala o *plugin* automaticamente
  - Siga as instruções até finalizar a instalação
  - Reinicie o Eclipse
  - O *plugin* está incorporado ao Eclipse e pronto para ser utilizado

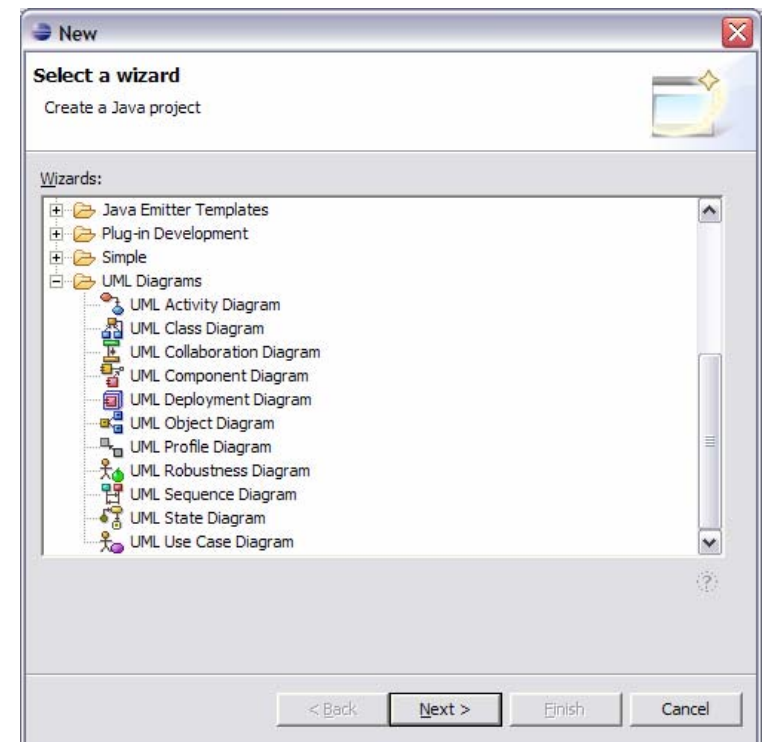
# Instalação do *Plugin* Omondo



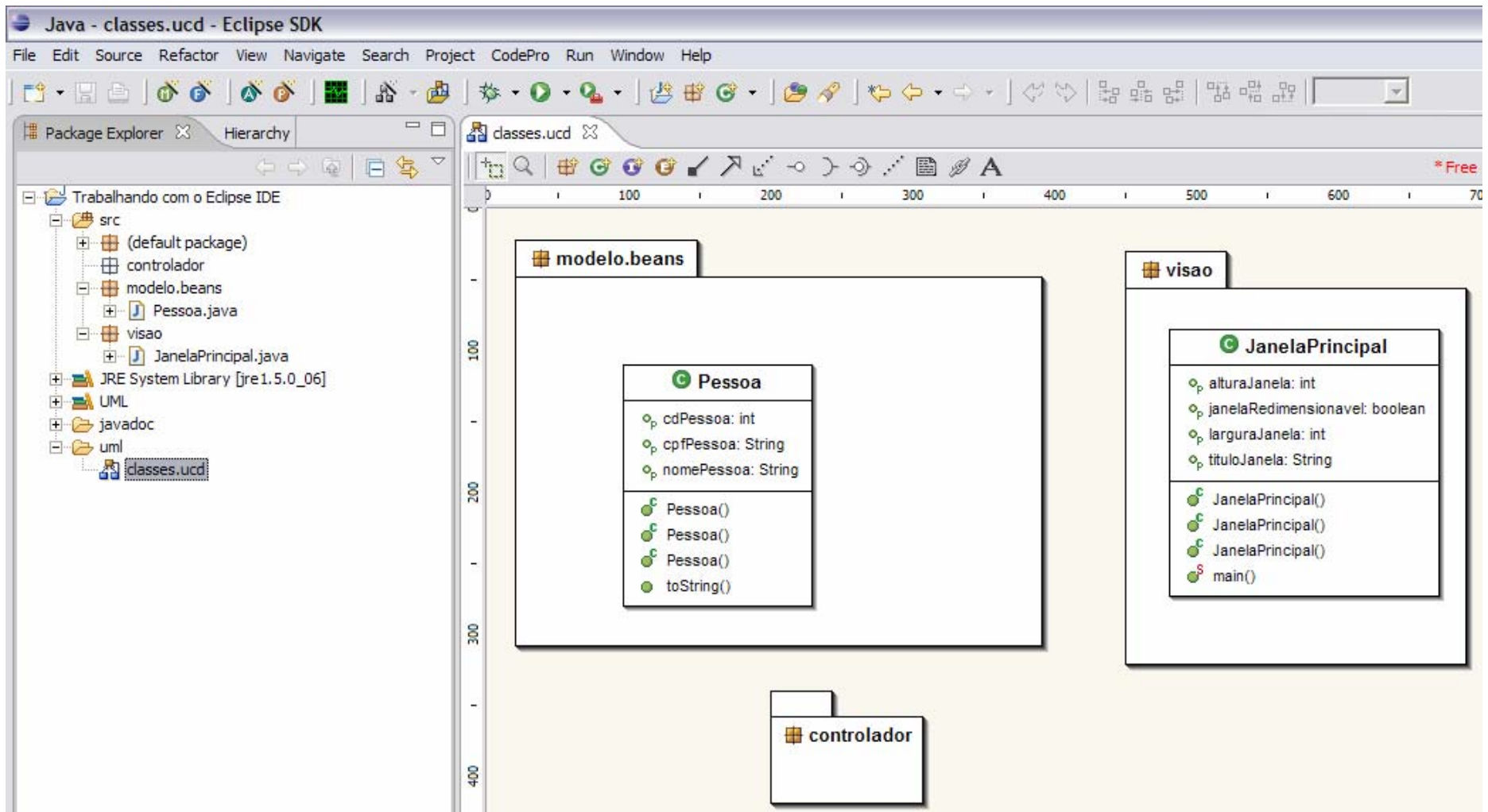
# Utilizando o *Plugin* Omondo



Indica a correta  
instalação do plugin  
***Omondo***



# Utilizando o *Plugin* Omondo



# Adicionando *Plugins* de Terceiros

- Alguns *plugins* não possuem instalação automática, por exemplo **CodePro AnalytiX**
- Assim, para instalá-los é necessário:
  - ❑ Fazer download do(s) *plugin(s)*
  - ❑ Copiar o conteúdo da pasta ***plugins*** para ***eclipse/plugins***
  - ❑ Copiar o conteúdo da pasta ***features*** para ***eclipse/features***
- Alguns *plugins* podem possuir somente a pasta ***plugins*** ou somente a pasta ***features***

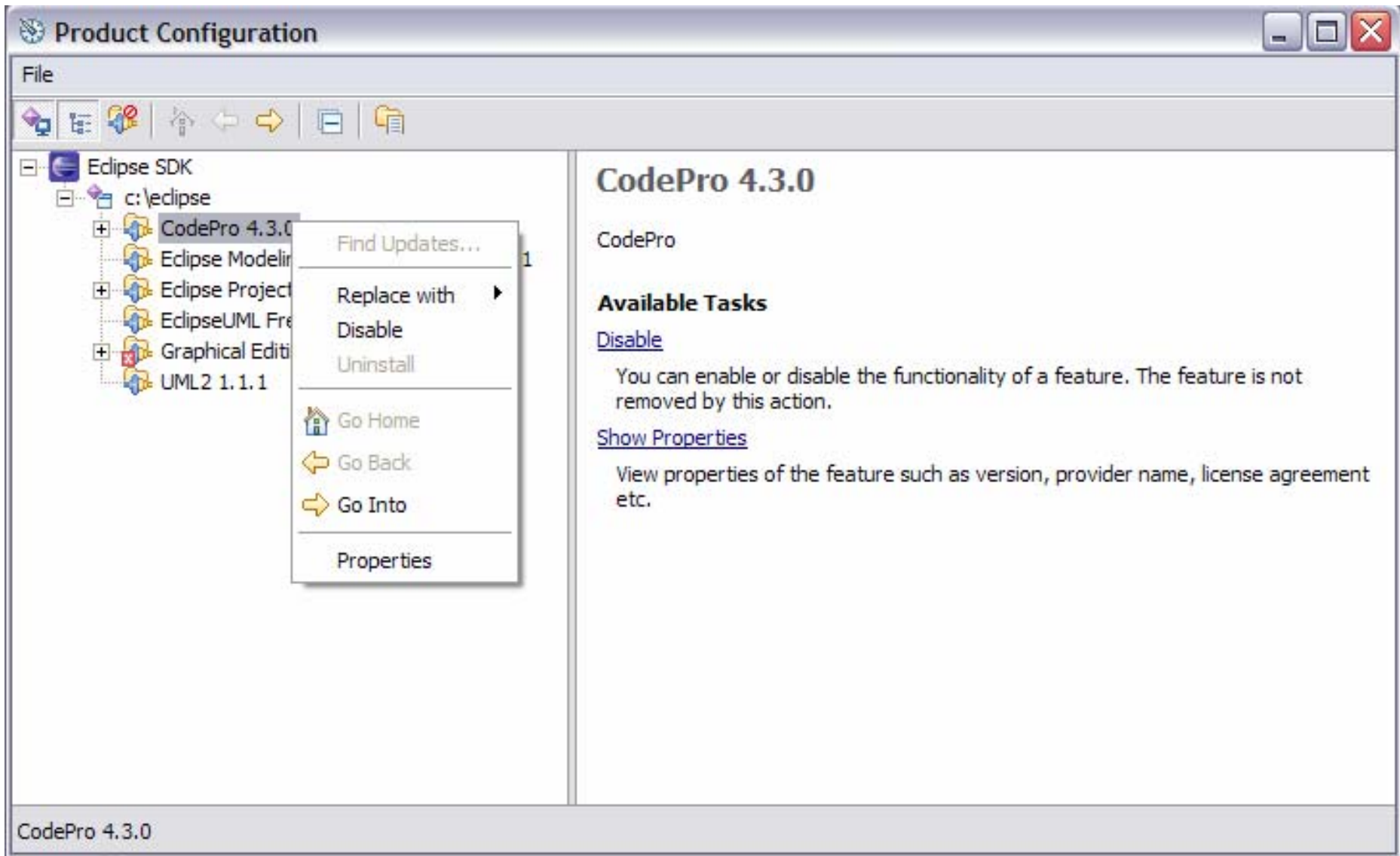


---

# Gerenciando *Plugins* no Eclipse

- É possível gerenciar os *plugins* instalados no Eclipse IDE:
  - O gerenciamento consiste em:
    - Visualizar as propriedades dos *plugins*
    - Habilitar ou desabilitar a utilização dos *plugins* no Eclipse
- O gerenciamento de plugins está disponível no menu ***Help / Software Updates / Manage Configuration***

# Gerenciando *Plugins* no Eclipse





---

# Aplicações Web com o Eclipse

---

Ferramentas de Programação Java

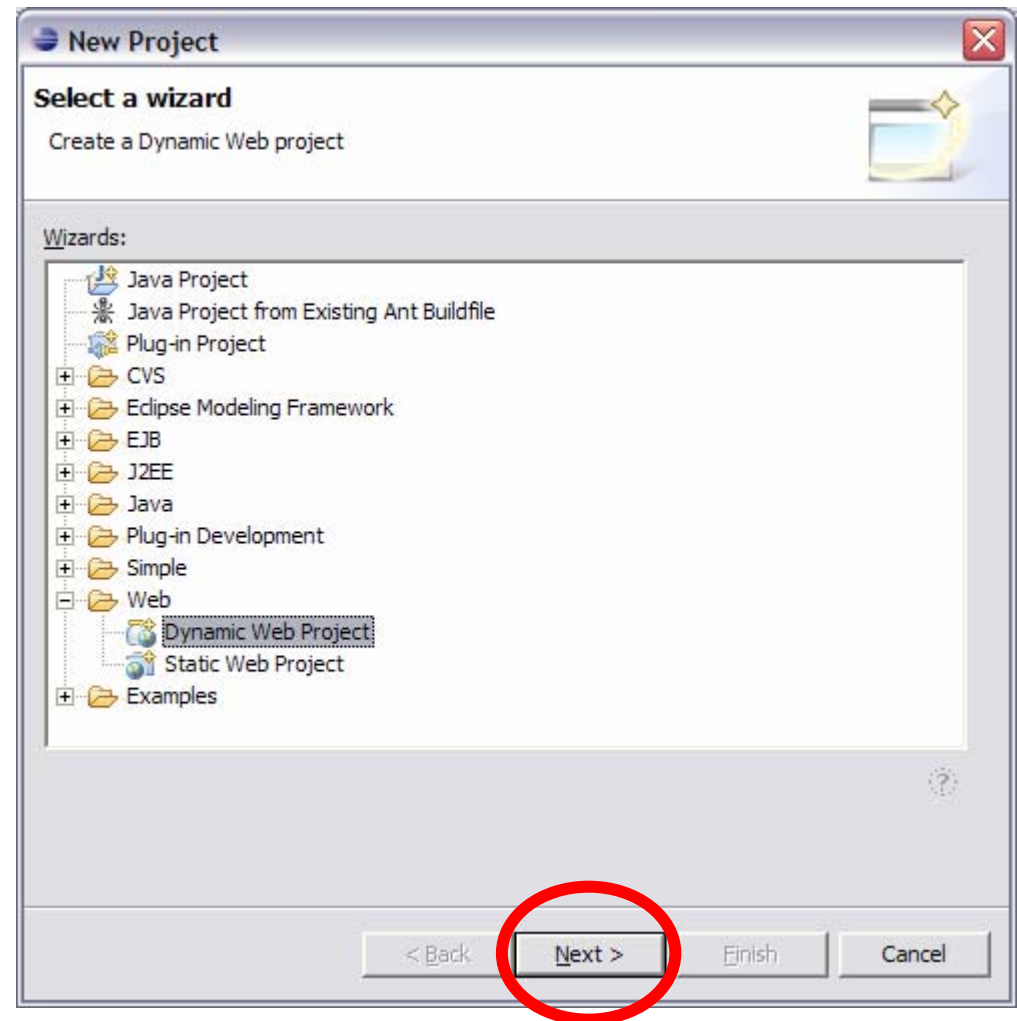


# Aplicações Web

- As aplicações Web podem ser desenvolvidas de duas maneiras utilizando o Eclipse IDE:
  - ❑ Baixando *plugins* de terceiros específicos para Web, por exemplo MyEclipse - <http://www.myeclipseide.com>
  - ❑ Utilizar os recursos do projeto *Web Tools Platform Project (WTP)* desenvolvido pela *Eclipse Foundation*
- ***Web Tools Platform (WTP):***
  - ❑ Desenvolvimento de aplicações Web, EJB, Web Services, Conectores J2EE, Persistência, etc...
  - ❑ Home page: <http://www.eclipse.org/webtools/jst/main.html> e <http://www.eclipse.org/webtools/wst/main.html>
  - ❑ Recursos: <http://download.eclipse.org/webtools/downloads>

# Eclipse WTP

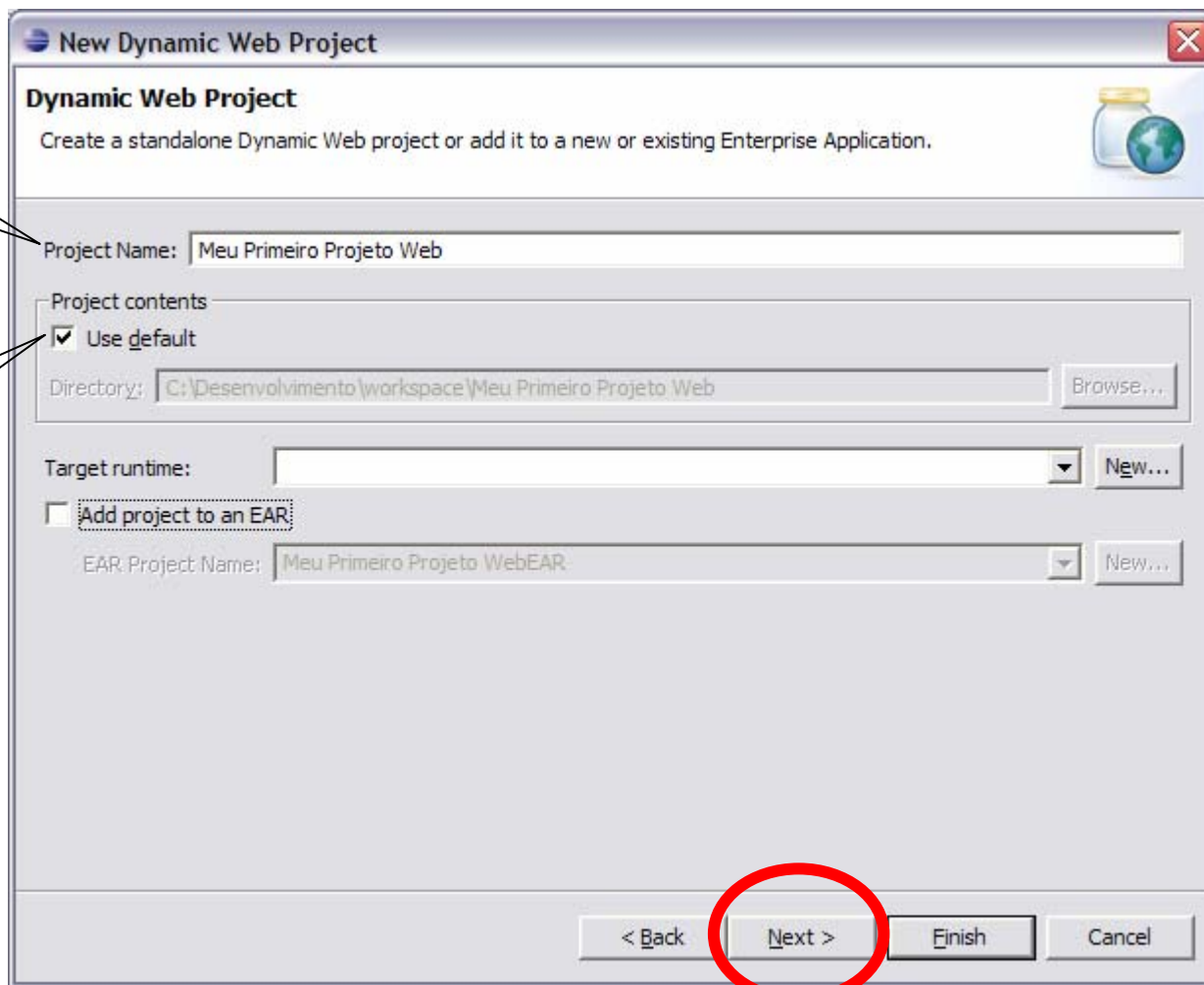
- O ambiente do Eclipse WTP é semelhante ao Eclipse IDE, porém a diferença está nas funcionalidades de cada um
- O Eclipse WTP possui perspectivas e tipos de projetos diferentes
- Os projetos Web são criados por meio do menu **File / New / Project / Web / Dynamic Web Project**



# Criando um Projeto Web (Etapa 1)

Título do projeto Web

Local do projeto Web



**New Dynamic Web Project**

**Dynamic Web Project**  
Create a standalone Dynamic Web project or add it to a new or existing Enterprise Application.

Project Name:

Project contents  
☒ Use default

Directory:

Target runtime:

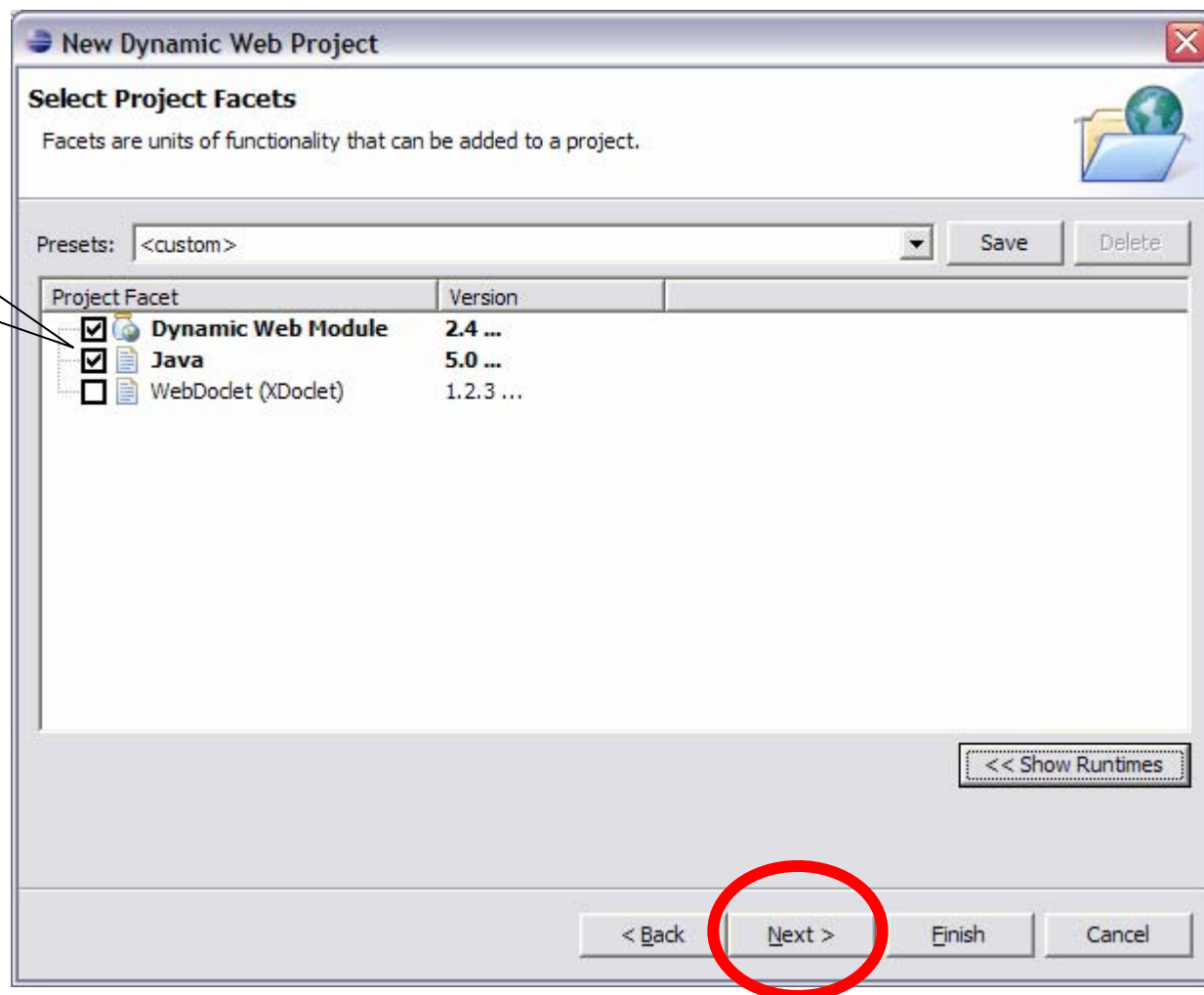
☐ Add project to an EAR

EAR Project Name:

< Back **Next >** Finish Cancel

# Criando um Projeto Web (Etapa 2)

Definição das  
versões do Java  
utilizadas



# Criando um Projeto Web (Etapa 3)

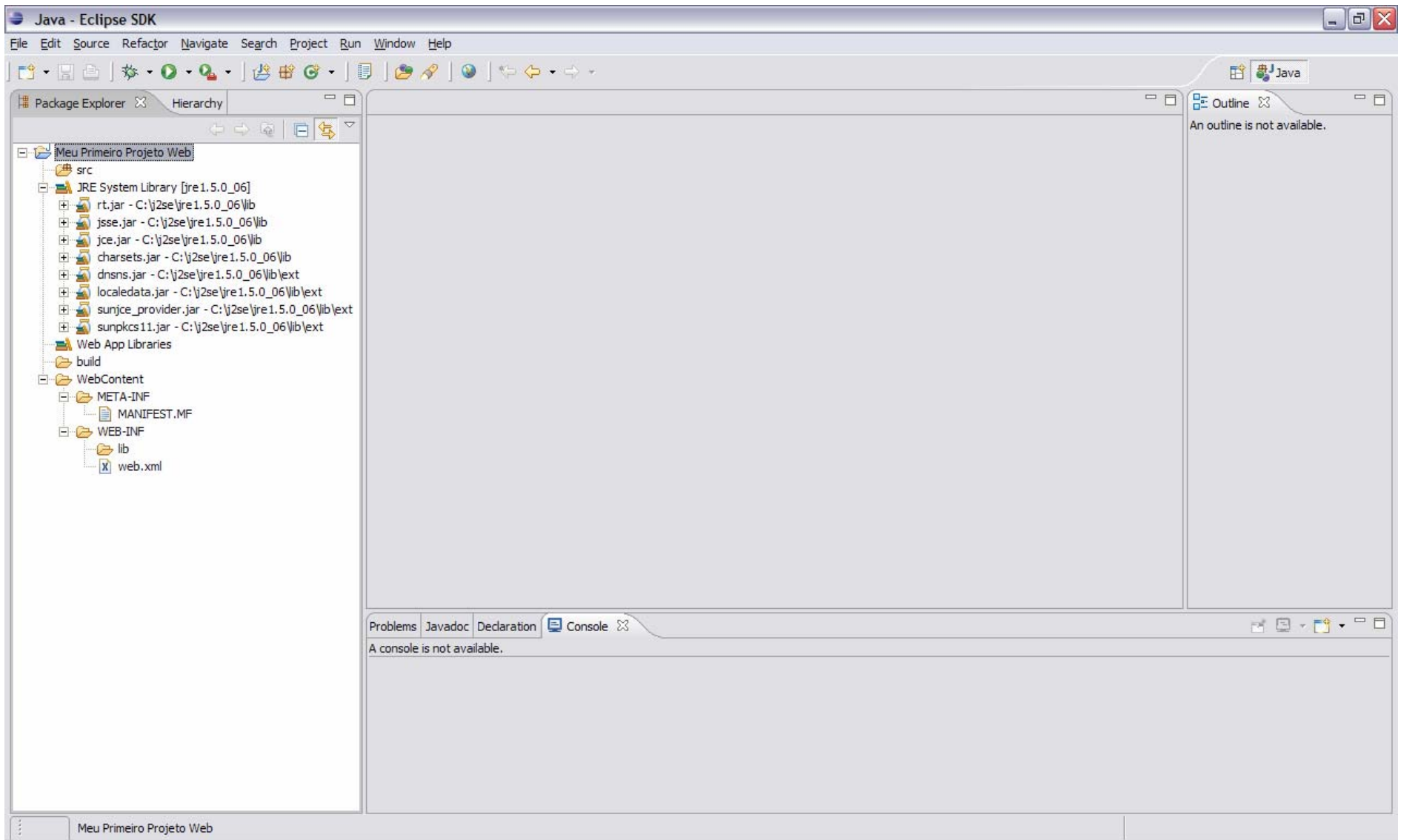
Contexto do projeto.  
Faz parte da URL  
para acessar o  
sistema.  
`http://localhost:8080/  
projetoWeb`

Pasta onde serão  
armazenados os  
arquivos da  
aplicação Web:  
JSPs, HTMLs, etc.

Armazena as classes  
Java criadas.

The screenshot shows the 'New Dynamic Web Project' dialog box in the Eclipse IDE. The dialog has a title bar with a close button. Below the title bar, it says 'Web Module' and 'Configure web module settings.' There are three input fields: 'Context Root:' with the value 'projetoWeb', 'Content Directory:' with the value 'WebContent', and 'Java Source Directory:' with the value 'src'. At the bottom right, there are four buttons: '< Back', 'Next >', 'Finish', and 'Cancel'. The 'Finish' button is circled in red.

# Meu Primeiro Projeto Web

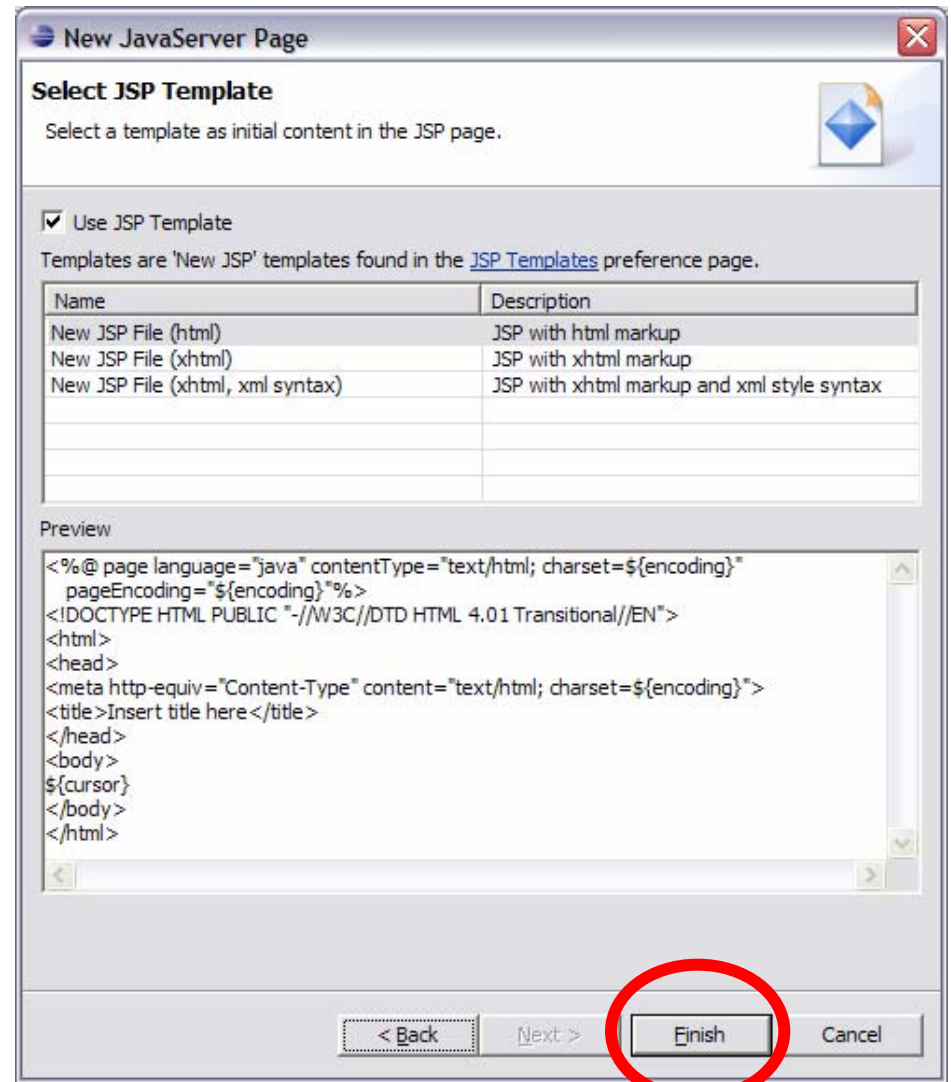
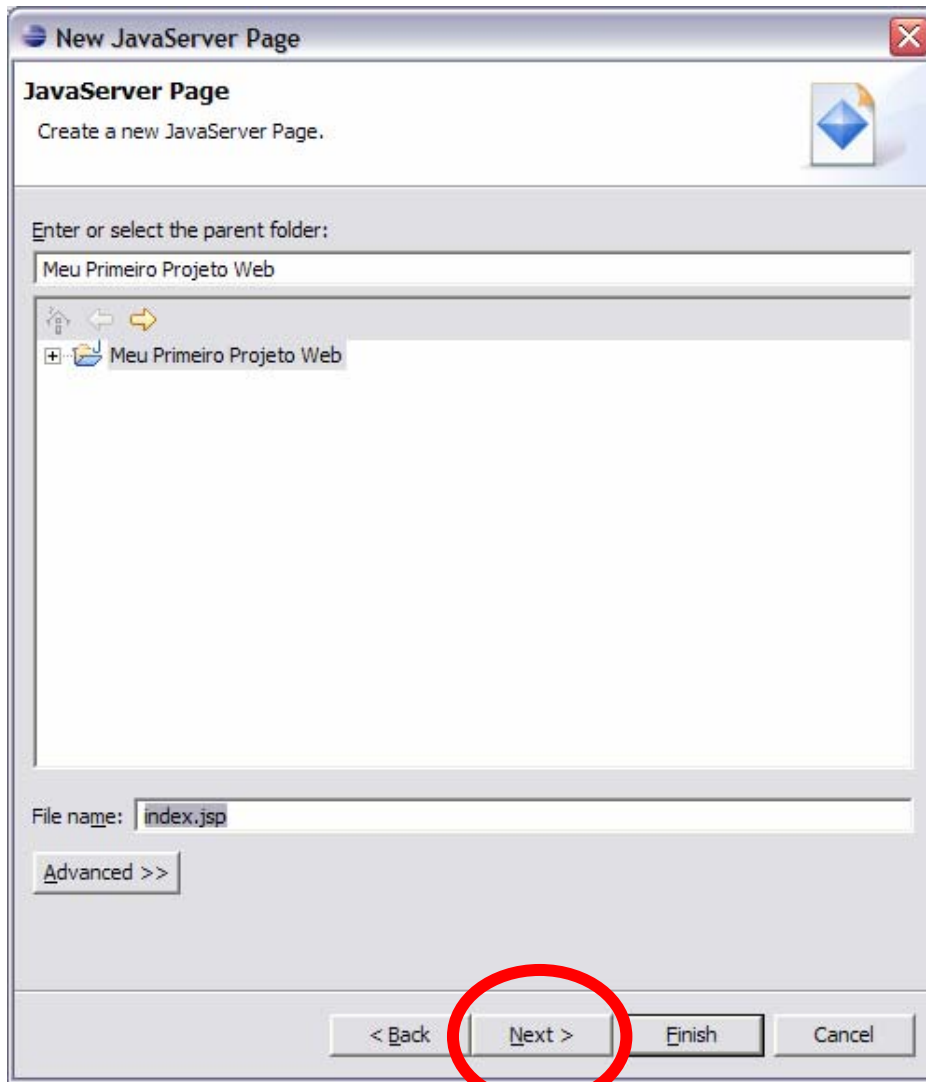


# Criando uma Página JSP

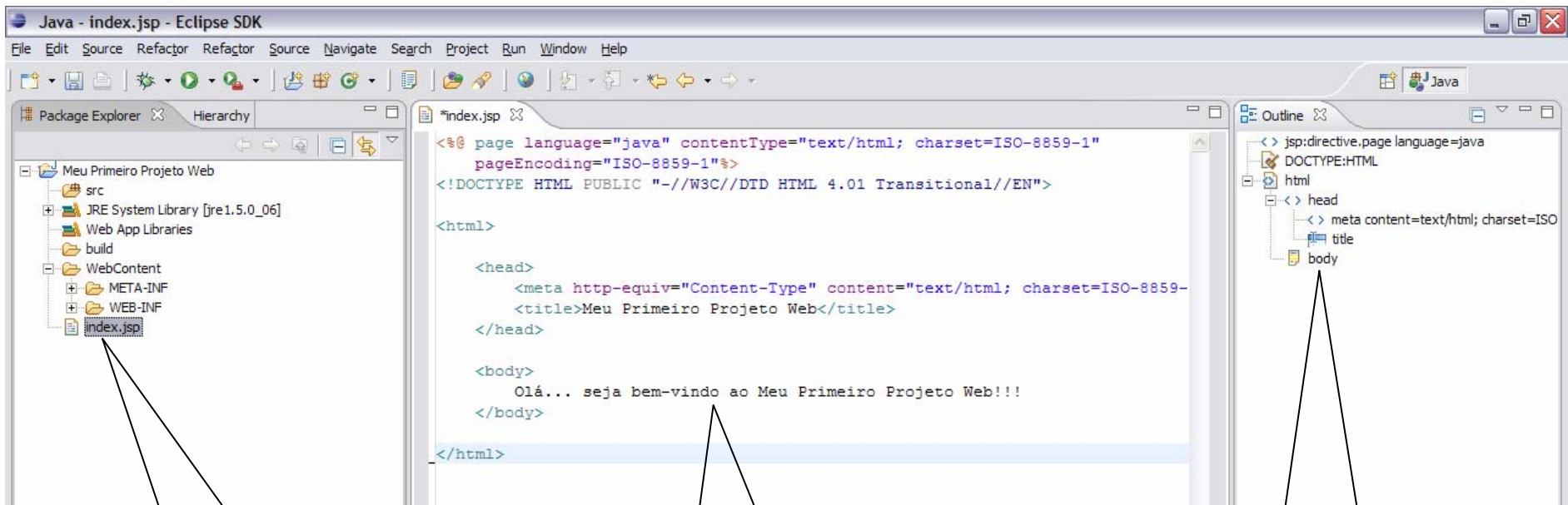
- Selecione a pasta **WebContent**
- Vá no menu **File / New / Other... / Web / JSP**
- Selecione **“Meu Primeiro Projeto Web”**
- Informe o nome da página JSP: **index.jsp**
- Escolha o Template JSP a ser utilizado. O mais comum é **JSP with html markup**



# Criando uma Página JSP



# Criando uma Página JSP



**Nova página  
JSP**

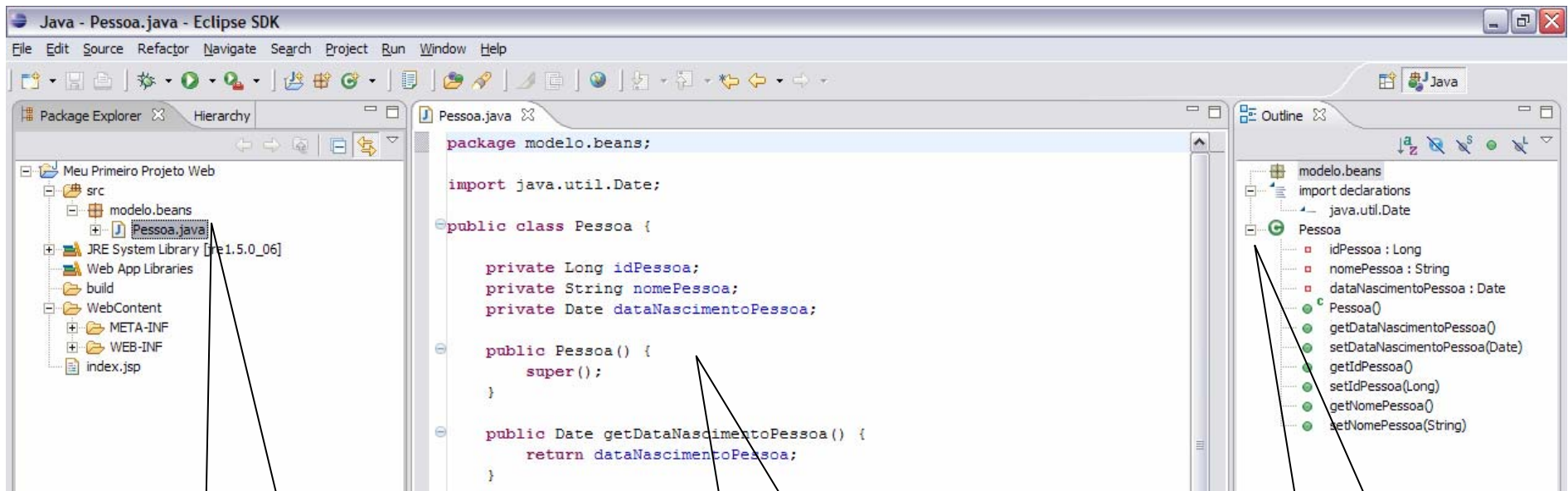
**index.jsp**

**Marcações JTML  
da página JSP**

# Criando Pacotes e Classes Java

- A criação de pacotes e classes Java para projetos Web acontece da mesma forma como no Eclipse IDE
- **Criando Pacotes:**
  - ❑ Selecione a pasta **src**
  - ❑ Vá no menu **File / New / Package**
  - ❑ Informe o nome do pacote: **modelo.beans**
  - ❑ Clique em **Finish**
- **Criando Classes:**
  - ❑ Selecione o pacote **modelo.beans**
  - ❑ Vá no menu **File / New / Class**
  - ❑ Informe o nome da classe (**Pessoa**) e demais elementos como é feito no Eclipse IDE

# Criando Pacotes e Classes Java



**Nova classe:  
Pessoa**

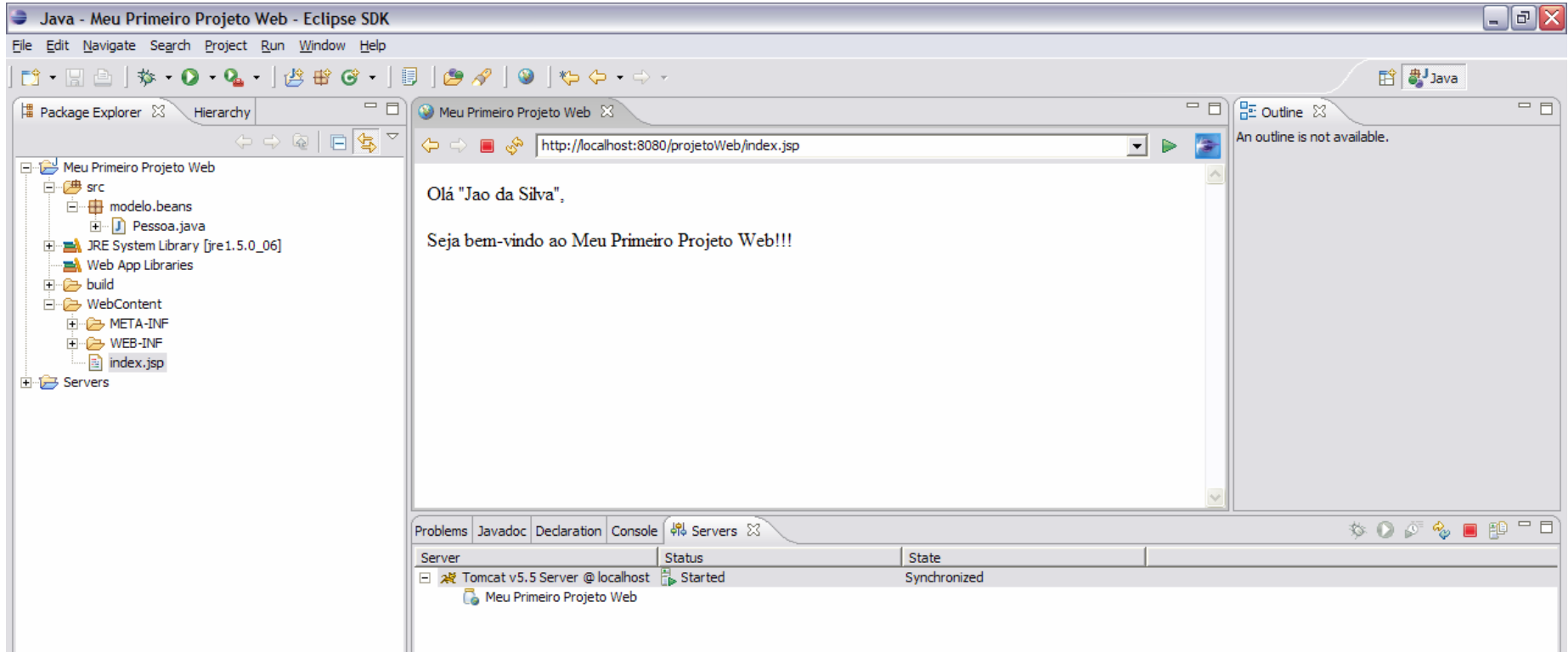
**Código da  
classe Pessoa**

**Membros da  
classe Pessoa**

# Executando uma Aplicação Web

- Selecione o projeto
- Vá no menu **Run / Run As / Run on Server**
  - ❑ Na primeira vez será necessário configurar o servidor Web em que a aplicação será executada
  - ❑ Escolha a opção **Manually define a new server**
  - ❑ Escolha o servidor **Apache / Tomcat v5.5 Server**
  - ❑ Em seguida, indique o diretório de instalação do tomcat
- A aplicação Web será iniciada
- O Eclipse WTP possui um navegador próprio, mas é possível visualizar a aplicação em outros navegadores

# Executando uma Aplicação Web



# Exportando Aplicações Web

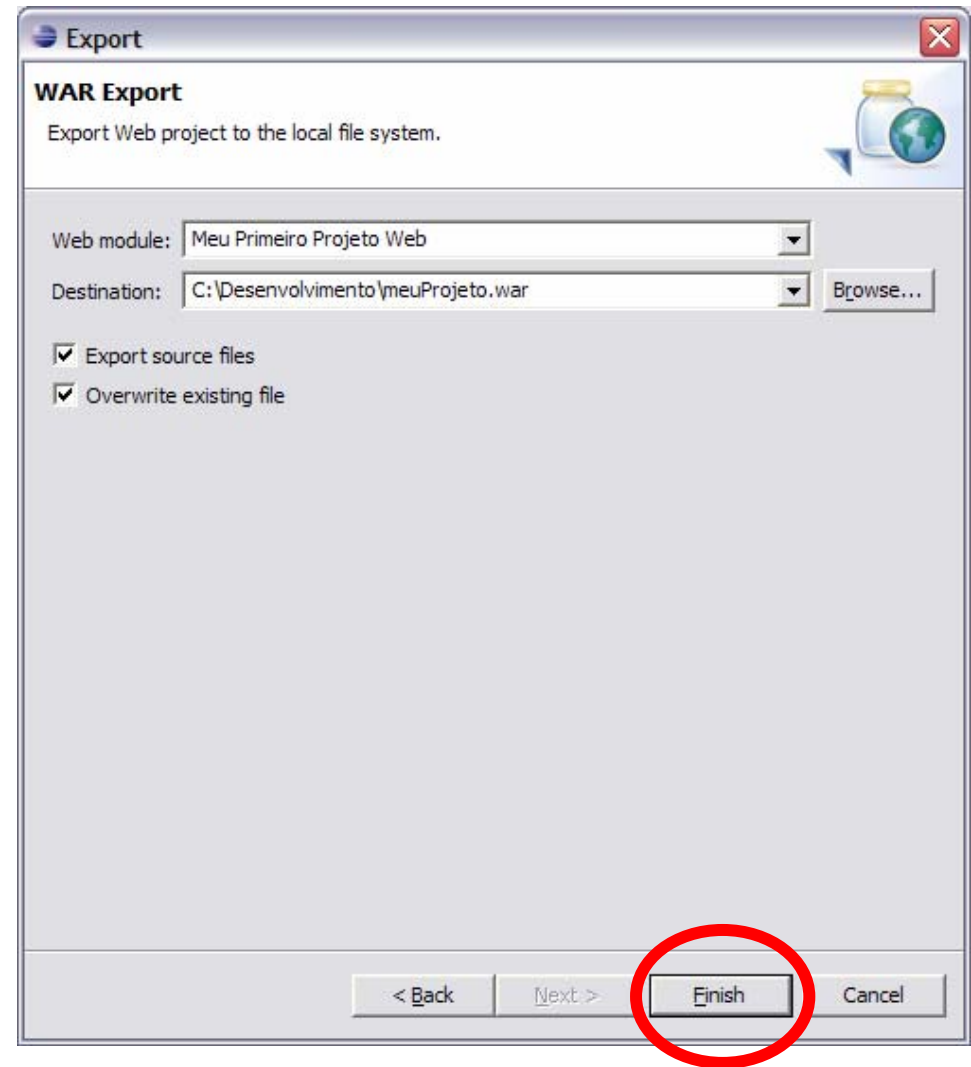
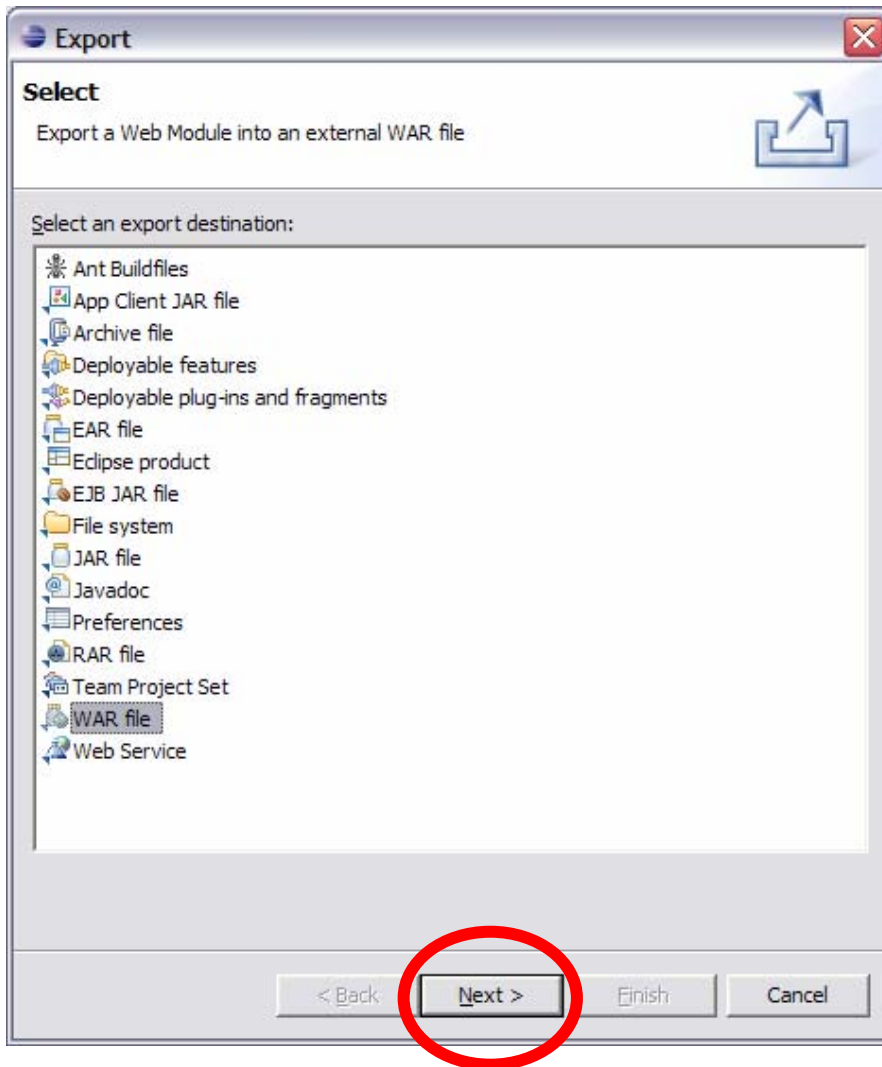
- Assim como aplicações desktop e conjuntos de classes Java podem ser distribuídos em arquivos JAR (Java Archive):
  - ❑ As aplicações Web podem ser distribuídas, porém no formato de arquivos WAR (Web Archive)
  - ❑ Arquivos WAR são criados da mesma forma que os arquivos JAR, por meio da ferramenta jar, incluída no J2SE SDK
  - ❑ Estes arquivos podem ser implantados em qualquer servidor (container) Web que siga os padrões de aplicações Java Web como, por exemplo, Apache Tomcat e Resin

# Exportando Aplicações Web

- Para gerar um arquivo WAR a partir de um projeto Web no Eclipse WTP é necessário:
  - Acessar o menu **File / Export**
  - Escolha a opção **WAR file** na tela
  - Em seguida, indique:
    - qual projeto deverá ser exportado (Web module)
    - a pasta de destino em que será gerado o arquivo
    - se deseja exportar os arquivos .java
    - se deseja sobrescrever arquivos existentes
  - Agora a aplicação Web desenvolvida já pode ser implantada em um container Web



# Exportando Aplicações Web



---

# Referências

**Slides:** [www.edsonjr.pro.br/eclipse.zip](http://www.edsonjr.pro.br/eclipse.zip)

- **Wiki do Eclipse:**

- <http://wiki.eclipse.org>

- **Documentação Geral do Eclipse:**

- Guia do usuário, desenvolvimento de plugins, documentação HTML, etc...

- <http://www.eclipse.org/documentation>

- **Artigos Técnicos:**

- <http://www.eclipse.org/articles>

- **Newsgroups:**

- <http://www.eclipse.org/newsgroups>

- **FAQ sobre as licenças:**

- <http://www.eclipse.org/legal/eplfaq.php>