



Flávio Sousa

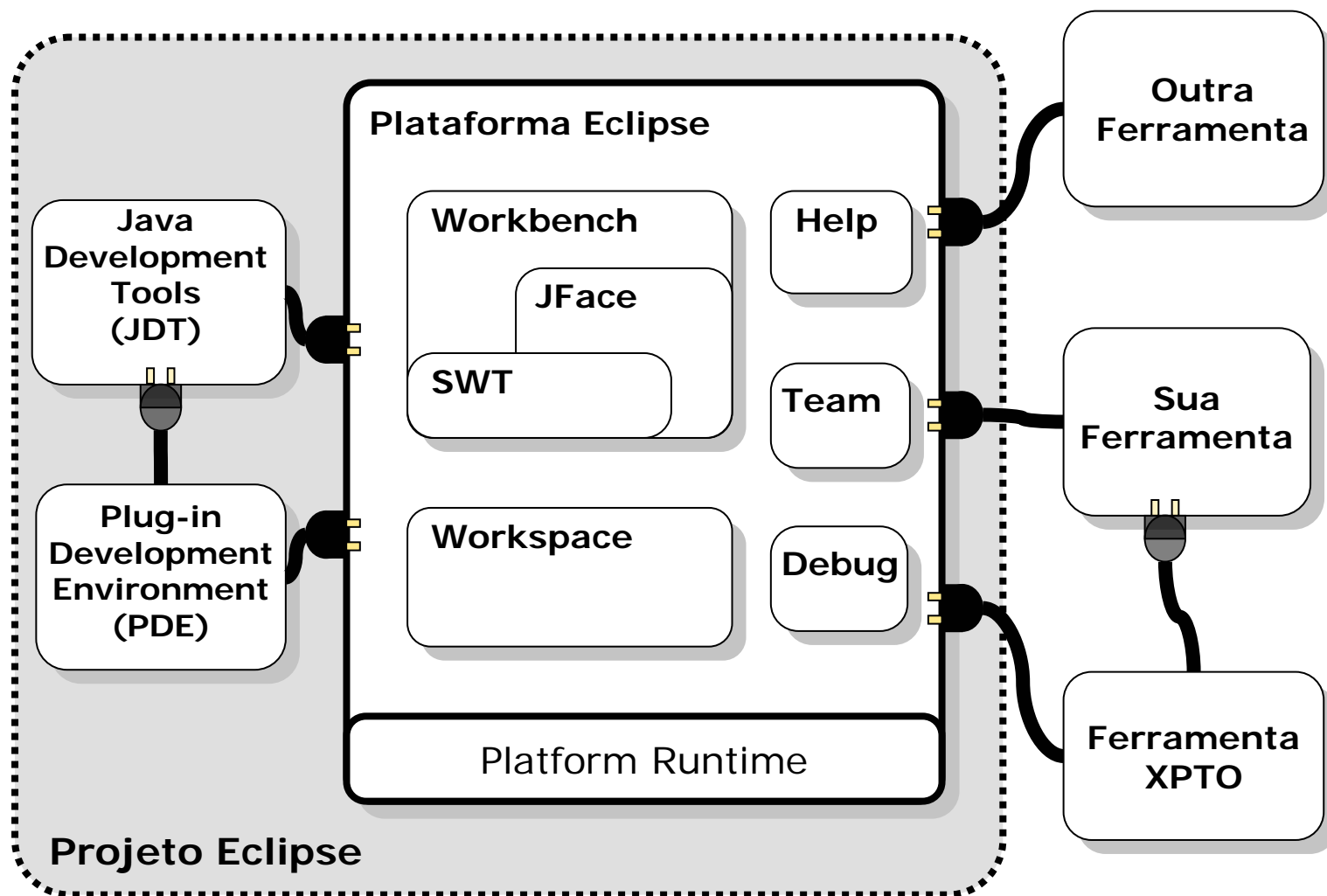
- ✓ Descrever a motivação por trás do Eclipse
- ✓ Usar os recursos confortavelmente
- ✓ Desenvolver e depurar aplicações Java
- ✓ Trabalhar com aplicações J2EE
- ✓ Trabalhar com UML
- ✓ Encontrar e instalar plugins

- ✓ Introdução
- ✓ Arquitetura
- ✓ Ambiente
- ✓ Desenvolvendo aplicações
- ✓ Depurando aplicações
- ✓ Plugin OMondo
- ✓ Plugin Lombok
- ✓ Conclusões

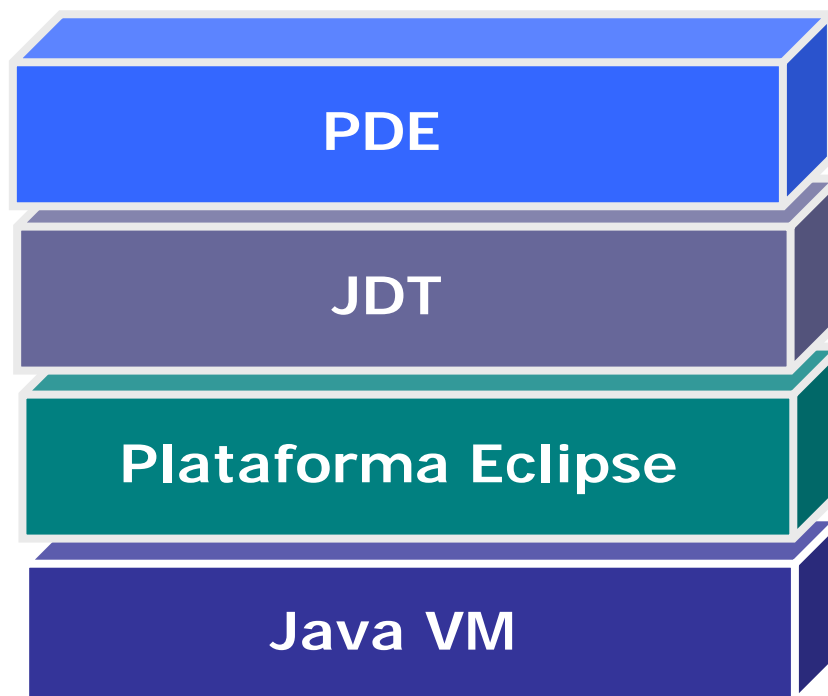
- ✓ O Eclipse é uma plataforma para a integração de ferramentas de desenvolvimento
- ✓ Extensível, aberto e portátil. Através de *plugins*, diversas ferramentas podem ser combinadas criando um ambiente de desenvolvimento integrado
- ✓ Suporte para desenvolvimento de novas ferramentas

➤ Características

- ✓ Independência de sistema operacional
- ✓ Neutralidade de linguagens
- ✓ Update automático
- ✓ Controle de versão
- ✓ Escolha de idioma



➤ Camadas



➤ Instalação

- ✓ Download : <http://www.eclipse.org>
- ✓ Necessita da JVM instalada.
- ✓ Descompactar o arquivo. Pronto.

➤ Plataformas

- ✓ Windows
- ✓ Linux
- ✓ Unix em geral

➤ Perspectivas

- ✓ Define um conjunto de **editores** e **visões** organizadas de uma forma visual tal que auxilie o trabalho de um determinado papel (projetista, designer, etc).

➤ Perspectivas mais utilizadas

- ✓ **Java:** geração de código em Java.
- ✓ **Java browsing:** visualizar projeto em Java.
- ✓ **Debug:** para uso do debug.
- ✓ **Recursos:** exibe todos os recursos do workspace em um único local.
- ✓ **CVS:** trabalho em equipe.

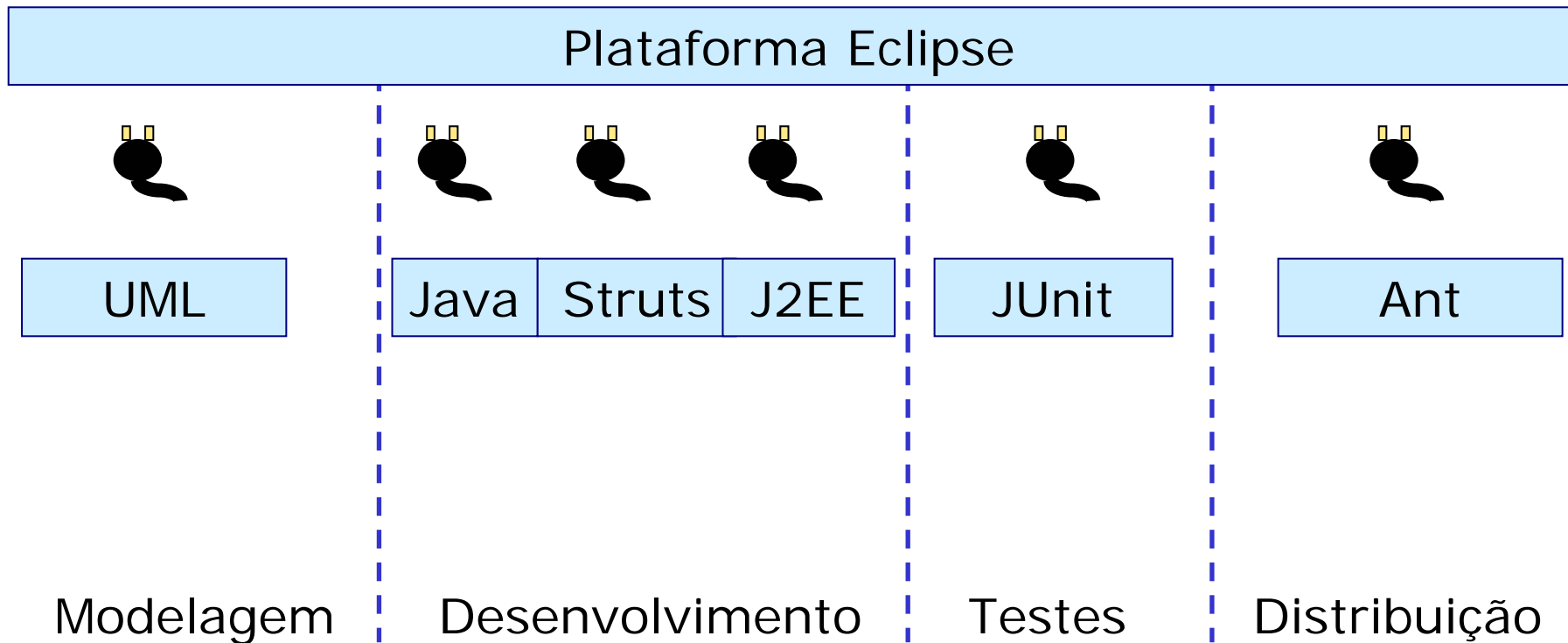
➤ Plugins

- ✓ As aplicações são desenvolvidas e instaladas no Eclipse sob a forma de **plugins**. Plugins são automaticamente reconhecidos e integrados à plataforma.
- ✓ São os reais responsáveis pelas funcionalidades do ambiente.

➤ Plugins

- ✓ Para instalar novas aplicações, basta copiar os plugins para a pasta \$ECLIPSE/plugins e reiniciar o Eclipse.
- ✓ Dependendo do plugin, novas **perspectivas** ou opções no menu estarão disponíveis.

➤ Plugins



➤ Workspace

- ✓ **Recursos** geralmente são arquivos no HD. Eles ficam no **workspace**, uma pasta especial no sistema de arquivos.
- ✓ O **workspace** é o local onde ficam os **recursos**, organizados em **projetos**.

➤ Arquivos e Pastas

- ✓ Um projeto Java pode ter arquivos e pastas, porém uma pasta pode ser de duas naturezas:
- ✓ **Folder**: contém recursos quaisquer
- ✓ **Source folder**: contém código Java

➤ Folder x Source

- ✓ O compilador não tenta compilar o que estiver presente numa pasta comum. O seu conteúdo é tratado como pastas e arquivos comuns.
- ✓ A Source folder é compilada. Seu conteúdo é tratado como uma estrutura de pacotes.

➤ Compilador

- ✓ Os .class apareceram na pasta de compilação, sem nenhuma operação de compilação.
- ✓ O Eclipse compila as classes em tempo de criação. Dessa forma, erros podem ser detectados antes da compilação.

➤ Compilador

- ✓ O Eclipse exibiria, o erro ocorrido, bem como uma sugestão para correção.
- ✓ Não há mais um passo de geração de código e um passo de compilação. Ambos estão condensados em um único passo.

➤ Editor Java

- ✓ Formatação de código
- ✓ Assistente de importações
- ✓ Depuração integrada (erros de compilação são marcados e entram na lista de tarefas)
- ✓ Sugestões para consertar erros rapidamente
- ✓ Atalho para linhas com problemas

➤ Importação

- ✓ Permite importar projetos do Workspace
- ✓ Arquivos .zip
- ✓ Outras

➤ Exportação

- ✓ Arquivos .jar
- ✓ Javadoc
- ✓ Outras

➤ Exportação

✓ JavaDoc → Documentação

➤ Formato do Comentário

```
/**
```

```
 * @exception
```

```
 */
```

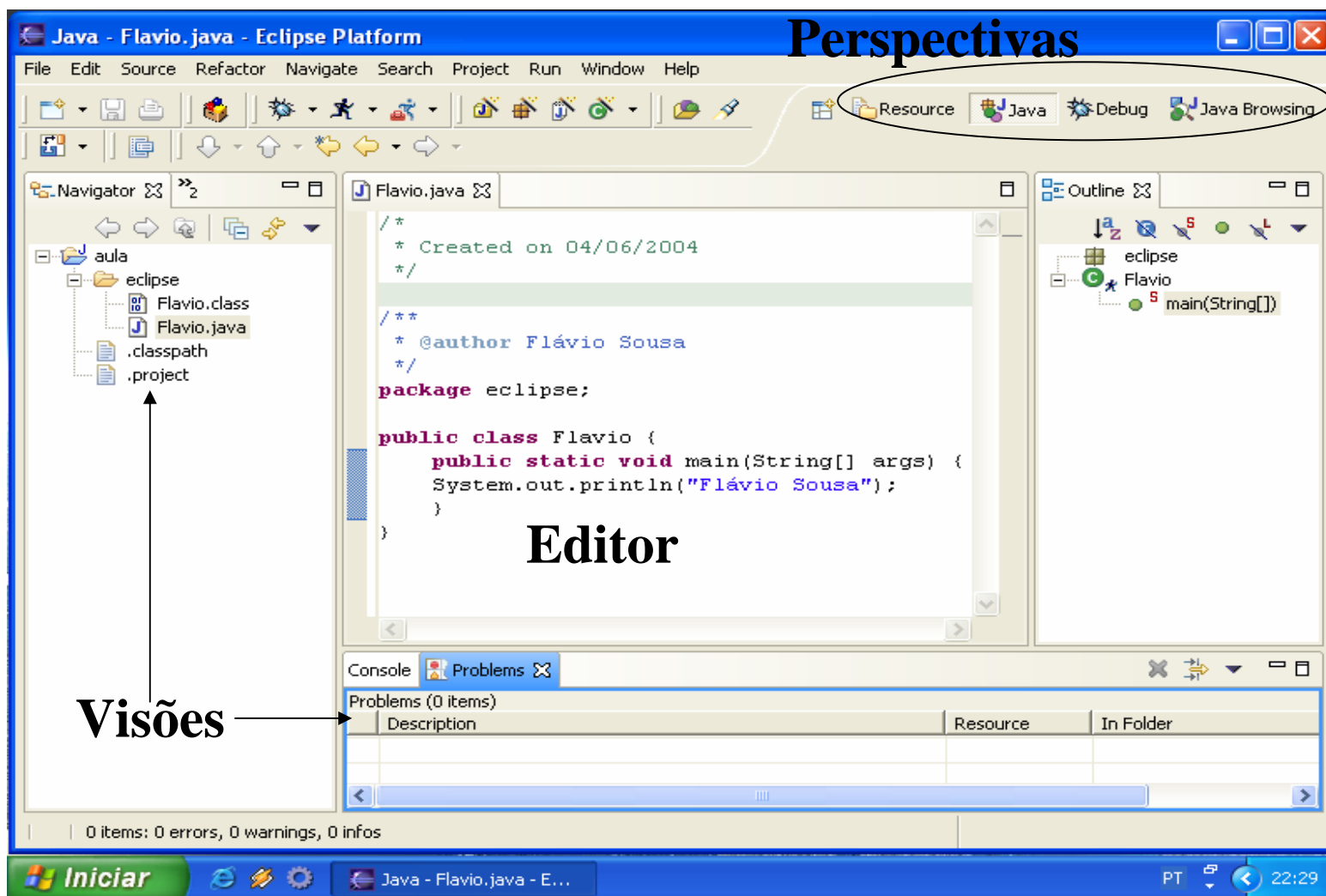
➤ Outros : @return , @param, etc

➤ Exportação

- ✓ File → Export → Javadoc
- ✓ Informe o local do Javadoc (J2SDK/bin)
- ✓ Informe o título. Finish. Pronto
- ✓ A documentação gerada segue as especificações da SUN

➤ Tipos de Projeto

- ✓ Projeto Java: projeto de uma aplicação Java
- ✓ Projeto Simples: projeto não Java
- ✓ Projeto de plugin: para desenvolver plugins para o eclipse



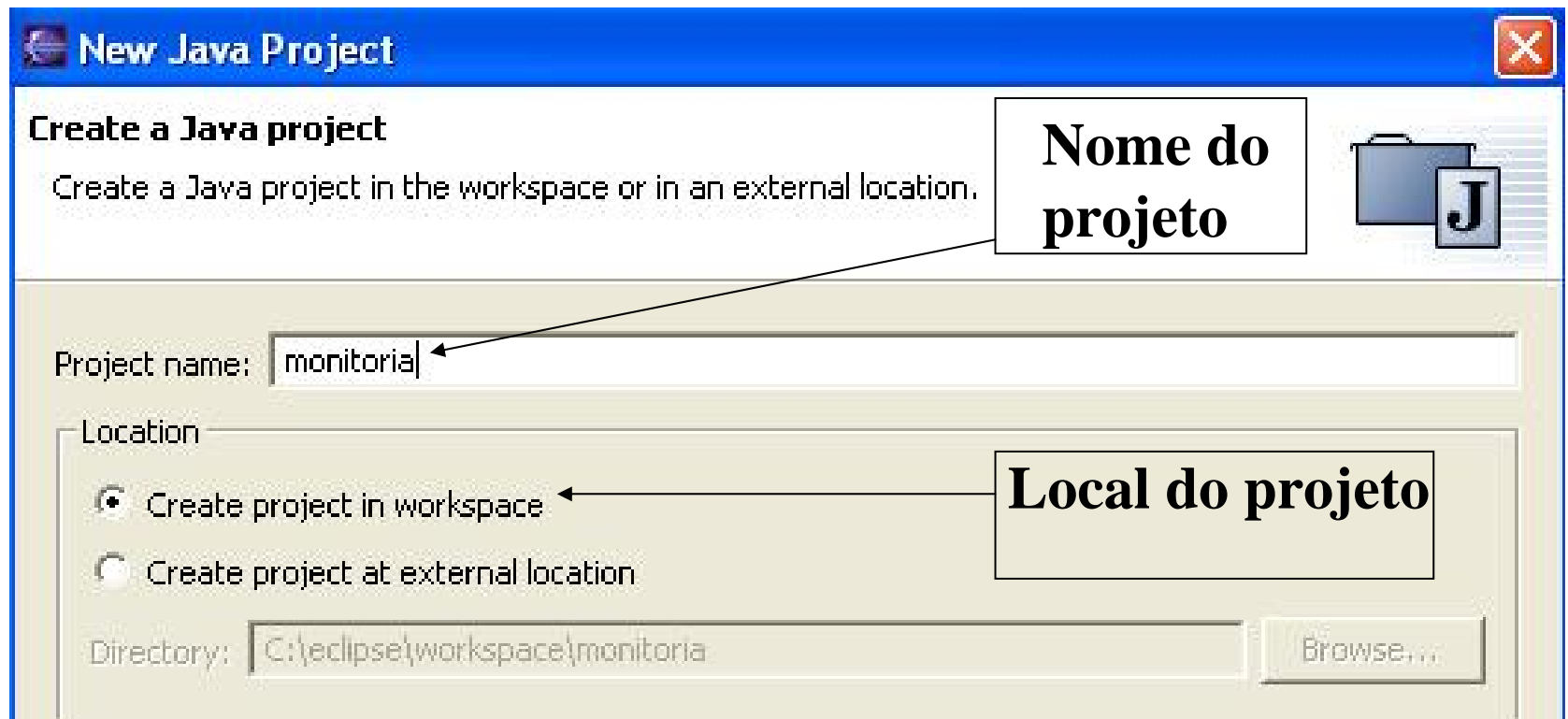
- ✓ Cria um Projeto Java
- ✓ Criar classes
- ✓ Criar pacotes
- ✓ Executar a aplicação

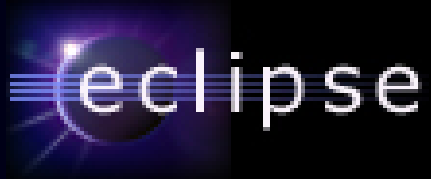
File → New → Project





Desenvolvendo aplicações





Desenvolvendo aplicações

Selecione → Finish





Desenvolvendo aplicações

File → New → Class

Nome do Source

Adicionar
pacote

Source Folder: Browse...

Package: Browse...

☐ Enclosing type: Browse...

Name:

Modifiers: ☒ public ☐ default ☐ private ☐ protected
☐ abstract ☐ final ☐ static

Superclass: Browse...

Interfaces: Add... Remove

Which method stubs would you like to create?

☒ public static void main(String[] args)
☐ Constructors from superclass
☒ Inherited abstract methods

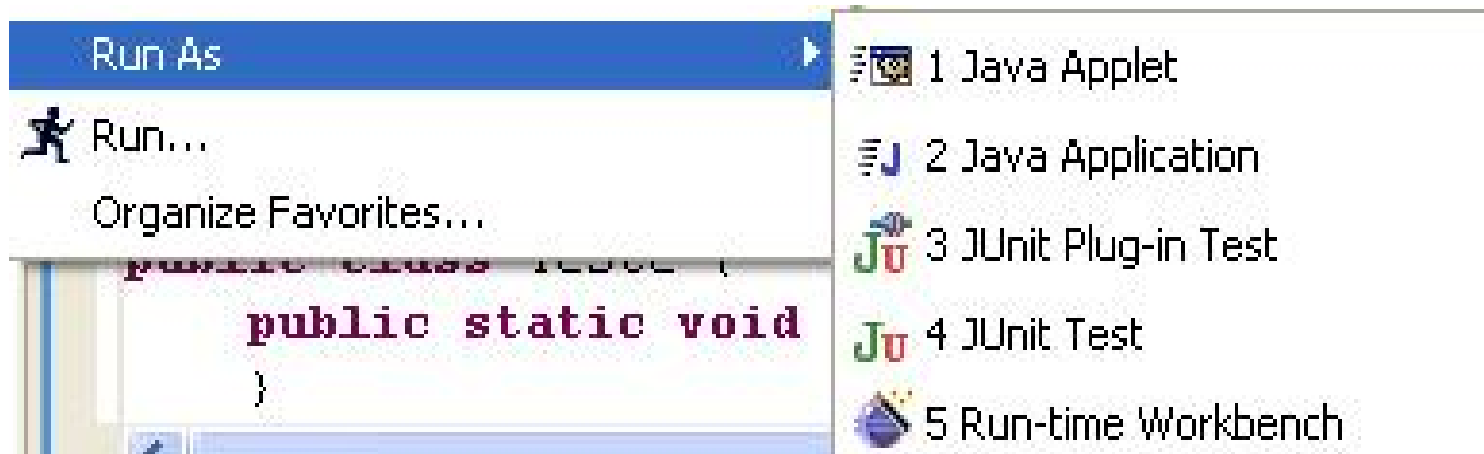
Nome da
Classe

Classe
Principal



Desenvolvendo aplicações

Run → Run As → Java Application



➤ Depurador

- ✓ Permite depurar programas locais ou remotos e também multithreaded
- ✓ Suspende a execução, inspecionar e modificar variáveis
- ✓ Não é necessário recompilar o código para depurar
- ✓ Mudança do valor de variáveis enquanto caminha pelo código
- ✓ Alteração do próprio código durante a depuração

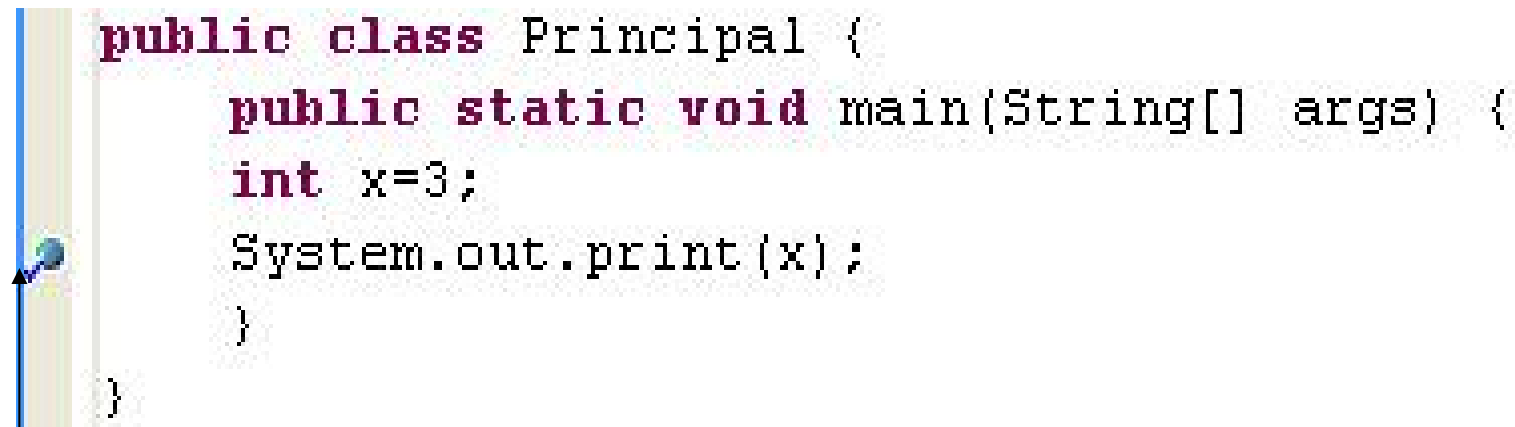
➤ Breakpoints

- ✓ São marcadores que suspendem a execução do programa
- ✓ Quando um breakpoint é acionado o Eclipse abre a perspectiva de depuração
- ✓ Breakpoints ficam ativos até serem removidos ou desabilitados
- ✓ Para adicionar um breakpoint de um clique duplo em qualquer linha do editor

➤ Pilha

- ✓ O depurador apresenta a pilha de execução logo antes do breakpoint ser atingido ou da exceção ser lançada.
- ✓ Entradas na pilha correspondem a chamadas de método em ordem cronológica reversa (o topo da pilha foi o último a ser executado).

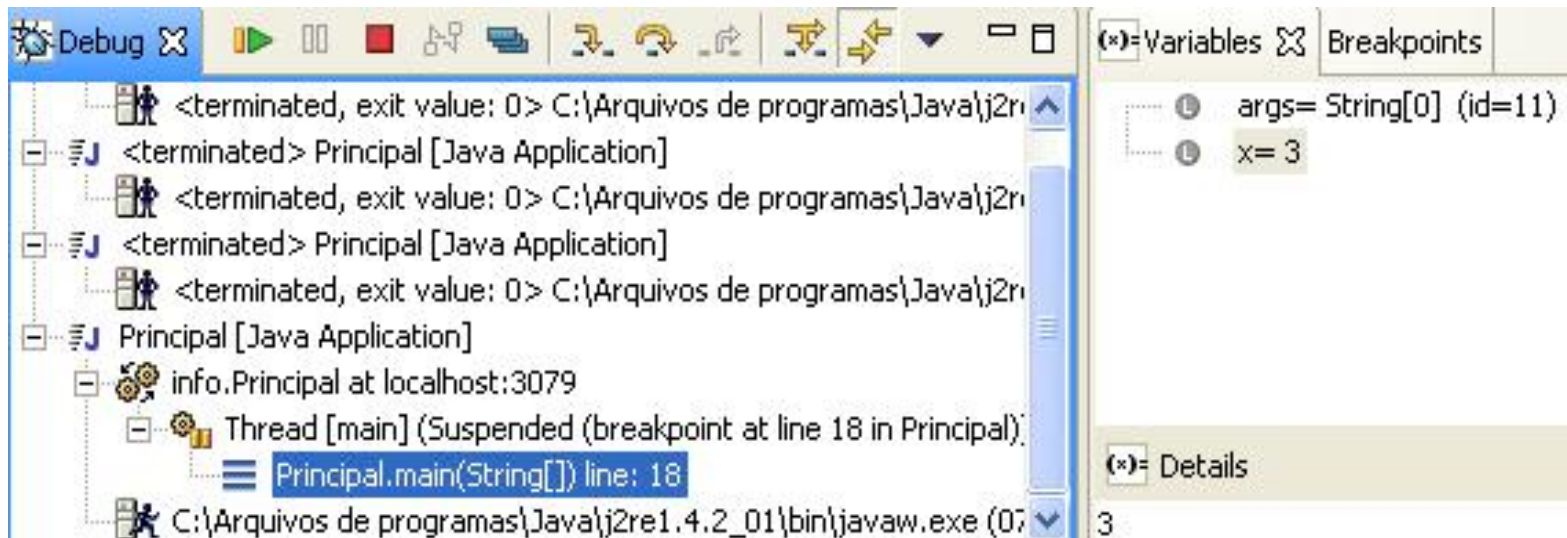
➤ Depurando um exemplo



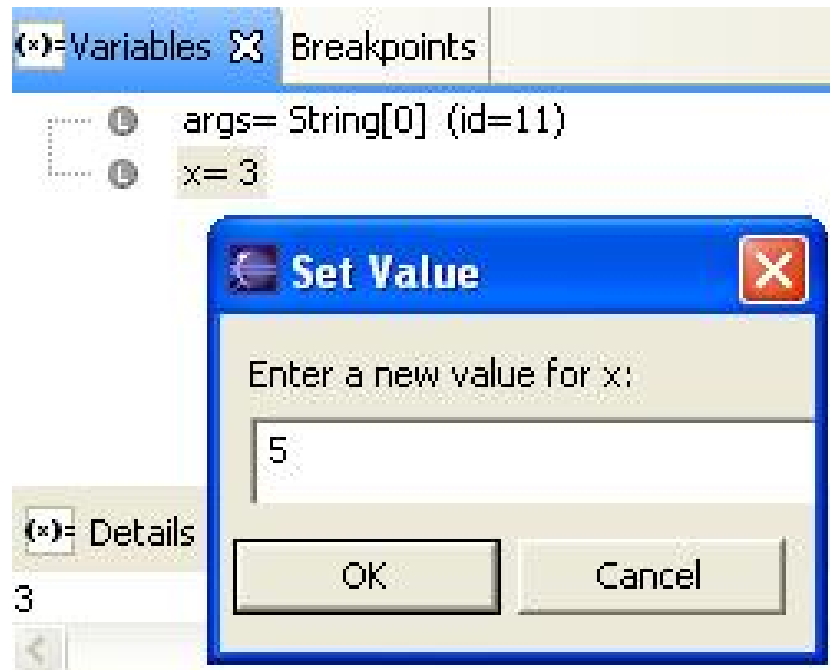
```
public class Principal {  
    public static void main(String[] args) {  
        int x=3;  
        System.out.print(x);  
    }  
}
```

Indica o Breakpoint

➤ Perspectiva Debug



- Alterando valores de variáveis
- ✓ Duplo click sobre o variável



➤ OMONDO

- ✓ Plugin para desenvolvimento UML.










➤ Características

- ✓ Permite construir os principais diagramas UML
- ✓ Gerar código a partir de diagramas
- ✓ Engenharia Reversa do diagramas

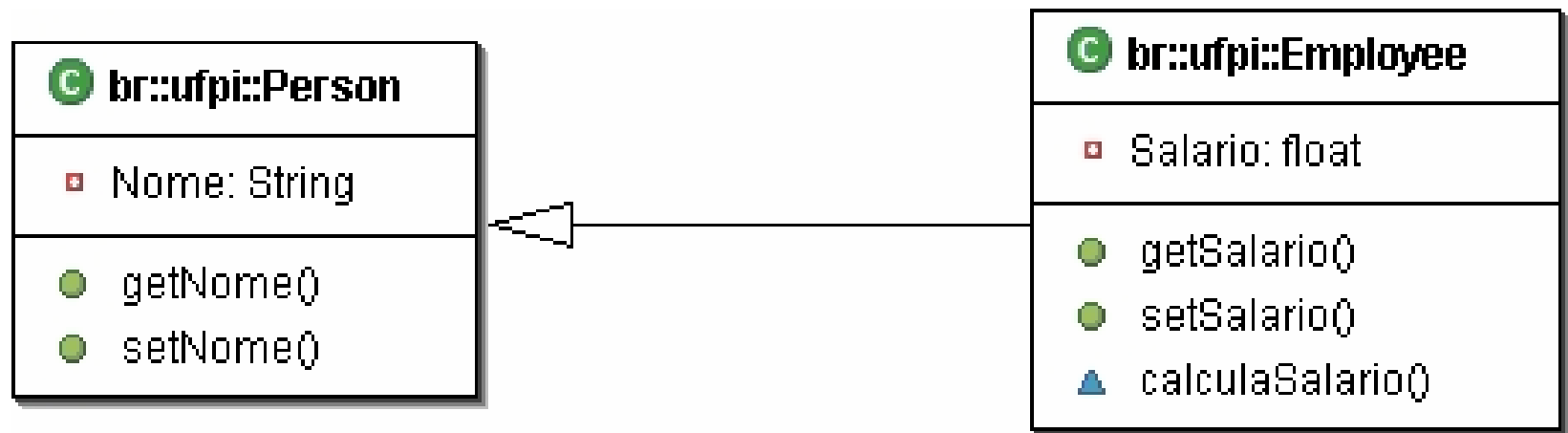
➤ Instalação

- ✓ **Pré-Requisito:** Eclipse Modeling Framework
- ✓ Download : <http://www.eclipse.org/emf>
- ✓ OMONDO : <http://www.omondo.com>
- ✓ Descompactar os arquivos na pastas
ECLIPSE/plugins e Eclipse/features

➤ Opções de diagramas

-
-  UML Class Diagram
 -  UML Sequence Diagram
 -  UML State Diagram
 -  UML Use Case Diagram
 -  UML Collaboration Diagram
 -  UML Activity Diagram
 -  UML Object Diagram
 -  UML Component Diagram
 -  UML Deployment Diagram

➤ Diagrama de Classe



➤ Classe Person

```
package br.ufpi;  
  
public class Person {  
  
    private String Nome;  
  
    public String getNome() {  
        return Nome;  
    }  
  
    public void setNome(String Nome) {  
        this.Nome = Nome;  
    }  
  
}
```

➤Classe Employee

```
package br.ufpi;  
  
public class Employee extends Person {  
  
    private float Salario;  
    public float getSalario() {  
        return Salario;  
    }  
    public void setSalario(float Salario) {  
        this.Salario = Salario;  
    }  
    float calculaSalario() {  
        return 0;  
    }  
}
```

➤ Lomboz

- ✓ Plugin para desenvolvimento J2EE.

➤ Características

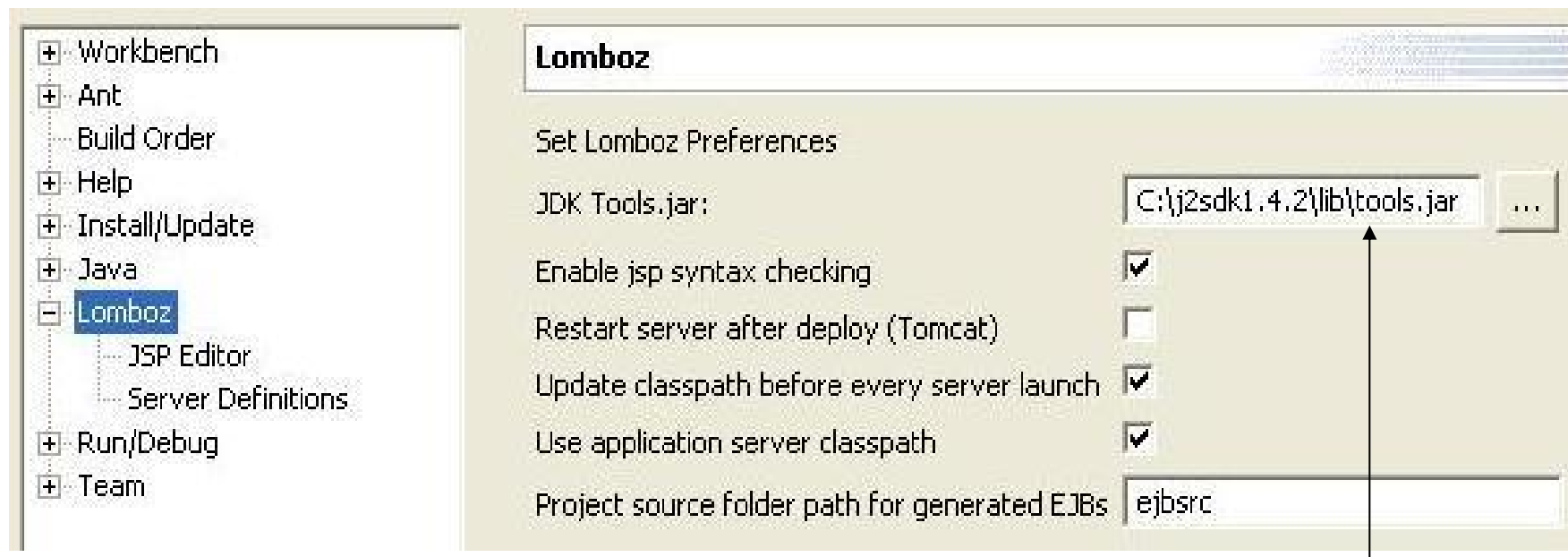
- ✓ Integração com vários servidores
- ✓ Verificação de Sintaxe JSP
- ✓ Produtividade utilizando wizards e geradores de código

➤ Instalação

- ✓ Download : <http://www.objectlearn.com>
- ✓ Verificar se as versões são compatíveis
- ✓ Copiar as pastas abaixo para a pasta plugins
 - [com.objectlearn.jdt.j2ee](#)
 - [com.objectlearn.jdt.j2ee.editors](#)

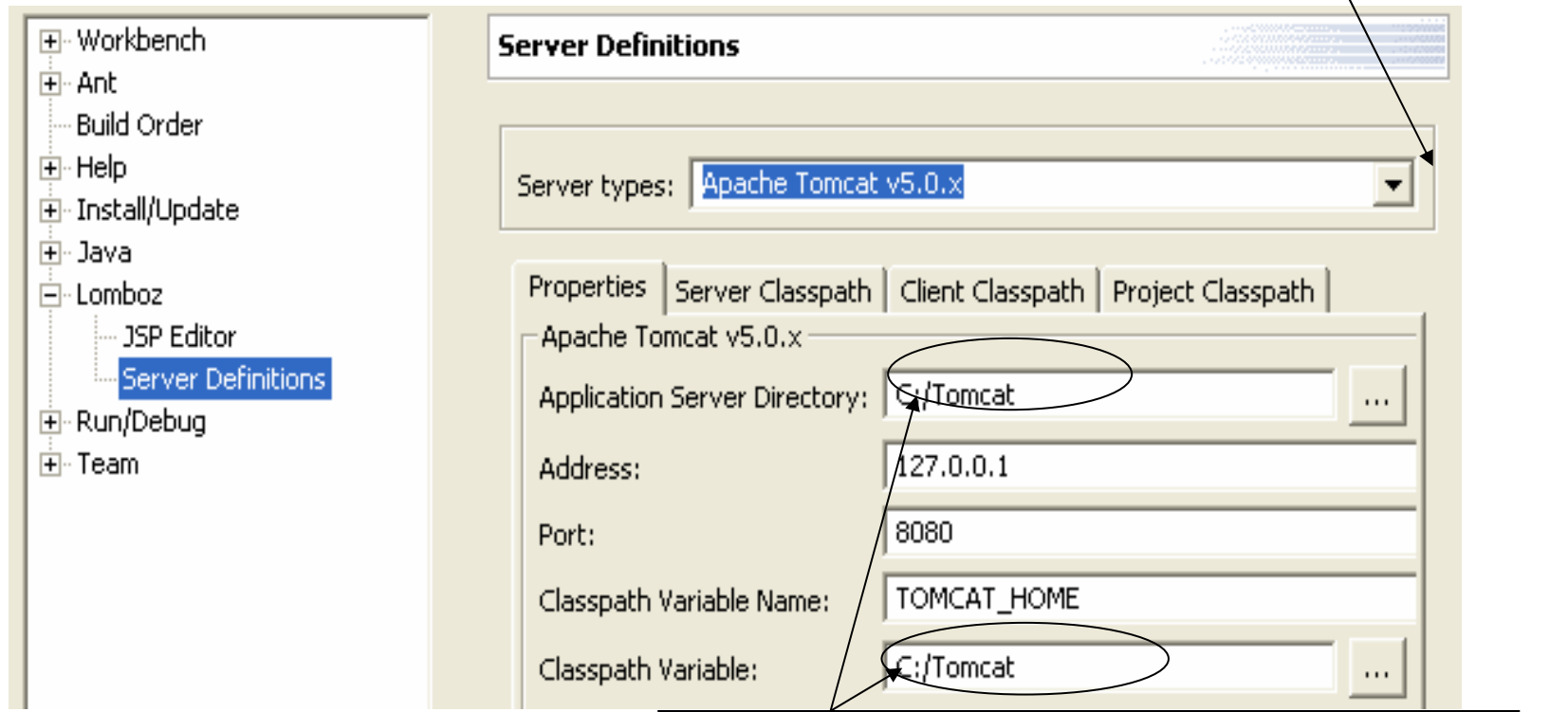
➤ Configuração

Window → Preferences → Lomboz



Indicar o arquivo `tools.jar`

➤ Configuração



Tipo de servidor

Local de instalação do servidor

Server Definitions

Server types: Apache Tomcat v5.0.x

Properties | Server Classpath | Client Classpath | Project Classpath

Apache Tomcat v5.0.x

Application Server Directory: C:/Tomcat

Address: 127.0.0.1

Port: 8080

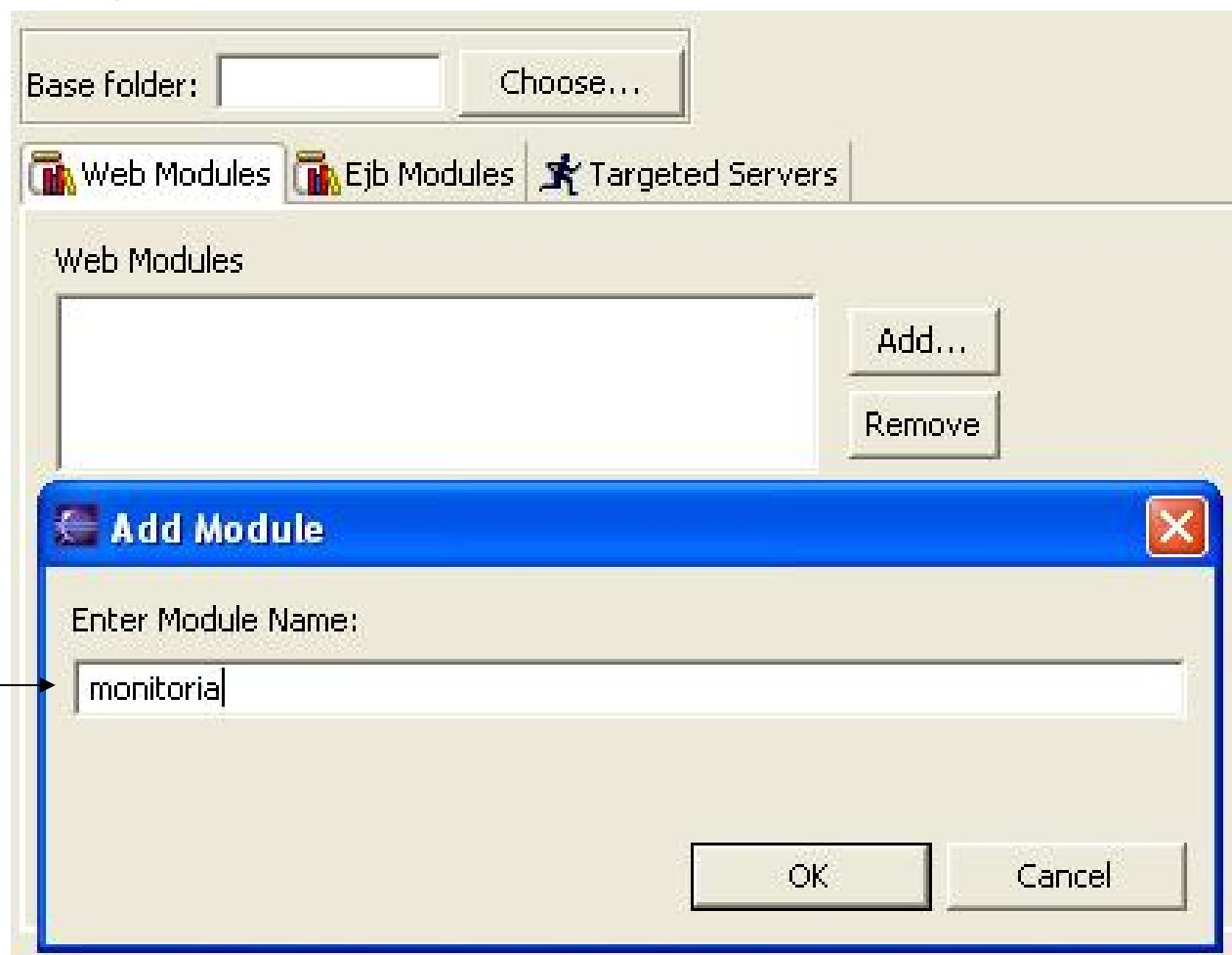
Classpath Variable Name: TOMCAT_HOME

Classpath Variable: C:/Tomcat

➤ Criar Projeto Lombok

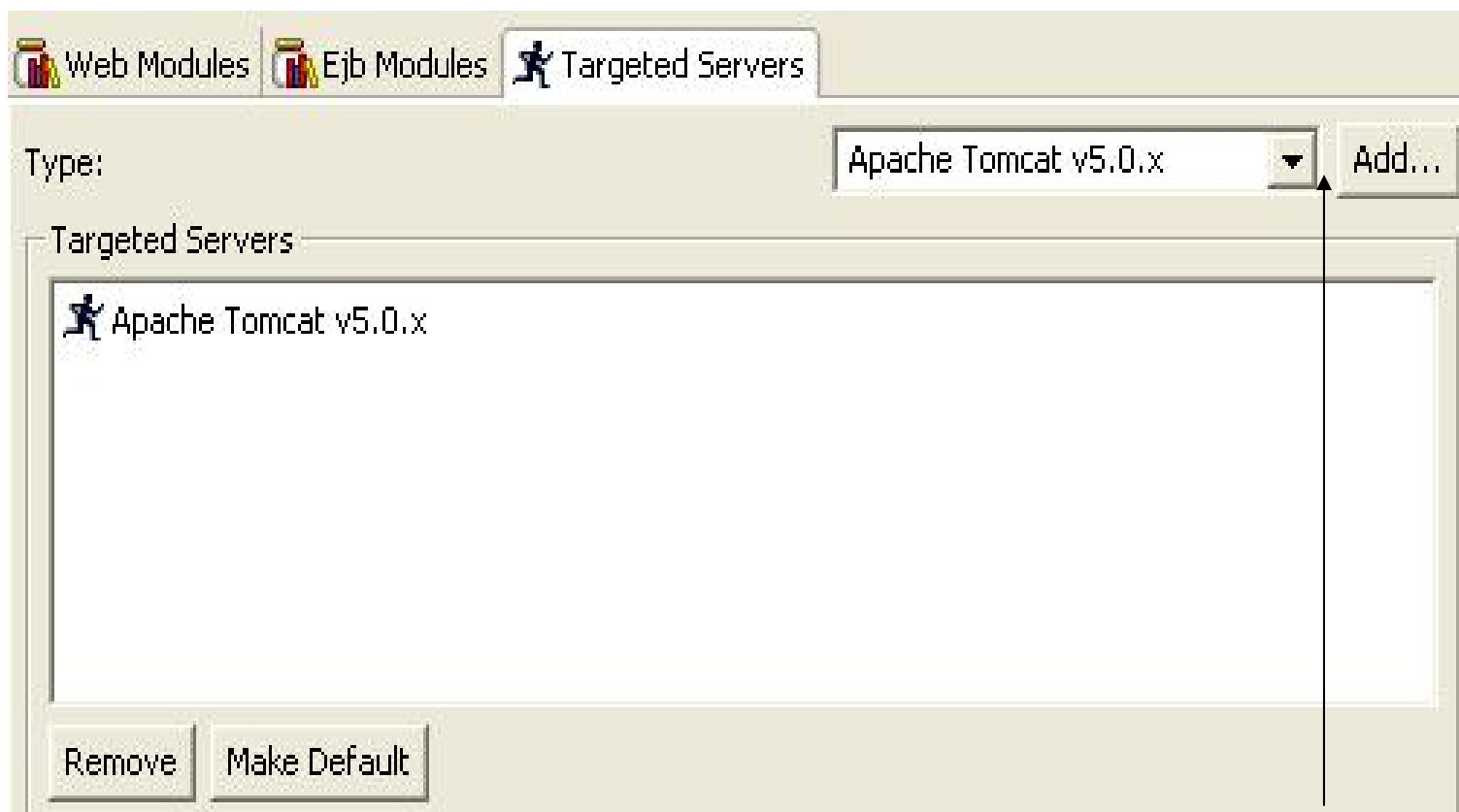
- ✓ Seleccione File → New → Project
- ✓ Seleccione Lombok J2EE Project
- ✓ Informe o nome do Projeto

➤ Criar Projeto Lomboz

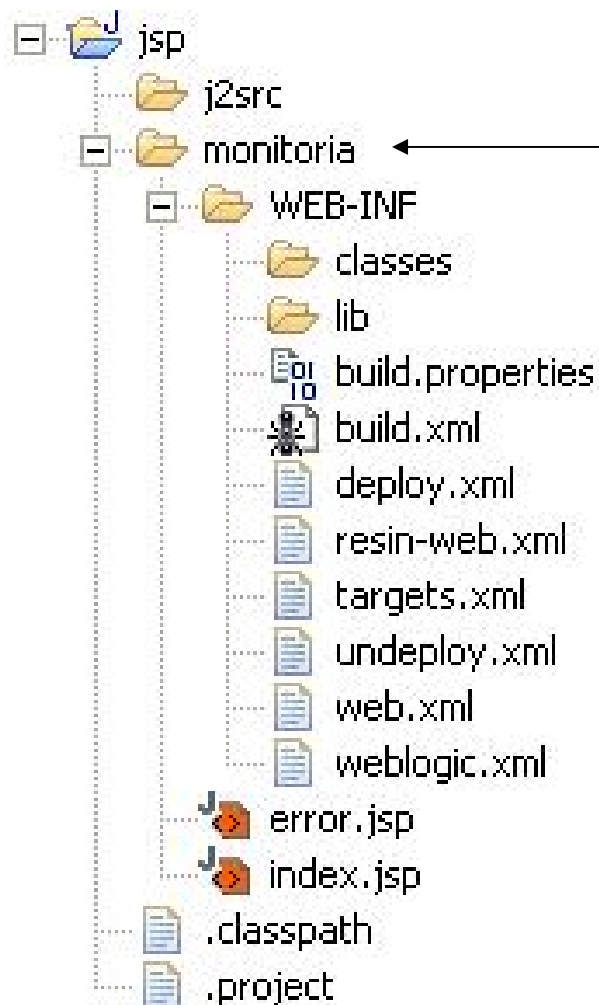


**Informe o
nome do
módulo
Web**

➤ Criar Projeto Lomboz



Informe o tipo de Servidor



**Estrutura de diretórios necessária
foi gerada**

Plugin Lomboz

 Lomboz J2EE Project

 Lomboz J2EE Module

 Lomboz EAR Module

 Lomboz EJB Creation Wizard

 Lomboz JSP Wizard

 Lomboz HTML Wizard

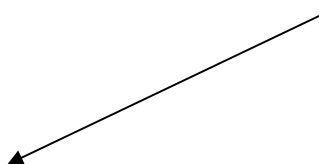
 Lomboz Servlet Wizard

 Lomboz Filter Wizard

 Lomboz EJB Test Client Wizard

 Lomboz EJB Method Wizard

Alternativas de escolha



➤ Adicionando Servlet

Crie um Source Folder

Nome do Servlet

Source Folder: Browse...

Package: (default) Browse...

Name:

Modifiers: ☒ public ☐ default ☐ private ☐ protected
☐ abstract ☐ final ☐ static
☐ Use single thread model

Superclass: Browse...

Interfaces: Add... Remove

Which method stubs would you like to create?

<input type="checkbox"/> init()	<input type="checkbox"/> getServletInfo()	<input type="checkbox"/> service()
<input checked="" type="checkbox"/> doGet()	<input type="checkbox"/> doPost()	<input type="checkbox"/> doDelete()
<input type="checkbox"/> doHead()	<input type="checkbox"/> doOptions()	<input type="checkbox"/> doTrace()
<input type="checkbox"/> destroy()		

Opções de métodos

➤ Mapeando Servlet

☐ Do NOT add deployment details to web.xml

Web module:

Servlet name:

Display name:

Description:

Mapping URL:

Adiciona no web.xml

Selecione o módulo

URL Servlet

➤ Testando Servlet

✓ <http://localhost:8080/servlet/codigo/Teste>

➤ Executando a aplicação

The screenshot shows the Eclipse IDE context menu for a J2EE project. The menu items are as follows:

- Delete
- Source
- Refactor
- Import...
- Export...
- Refresh
- Run
- Team
- Compare With
- Restore from Local History...
- Lomboz J2EE...** (highlighted)

The Lomboz J2EE... menu item is expanded, showing the following options:

- Add New EJB
- Generate EJB Classes
- Change default server
- Stop Server
- Run Server
- Debug Server
- Check All JSP Syntax
- Make default web module
- Undeploy Module
- Deploy Module
- Show in Browser
- Add WEB-INF/lib jars to classpath

Annotations with arrows point to specific menu items:

- Start o Servidor (1)** points to **Run Server**.
- Checa Sintaxe JSP** points to **Check All JSP Syntax**.
- Exporta o Módulo (2)** points to **Deploy Module**.
- Exibe no Browser (3)** points to **Show in Browser**.

➤ Alguns Plugins interessantes

- ✓ VEP: Eclipse Visual Editor
- ✓ XMLBuddy: editor de XML
- ✓ DBEdit : conecte-se a banco de dados
- ✓ Easy Struts : Framework Struts

➤ Mais Plugins

- ✓ <http://eclipse-plugins.2y.net/eclipse/>

- ✓ Ambiente simples de usar
- ✓ Permite ganho na produtividade
- ✓ Possibilidade de automatizar atividades
- ✓ Integração com várias ferramentas

- ✓ <http://www.eclipse.org>
- ✓ <http://www.objectlearn.com>
- ✓ <http://gsd.ime.usp.br/eclipse>
- ✓ <http://web.teccomm.les.inf.puc-rio.br/eclipse/>
- ✓ <http://www.guj.com.br>

Dúvidas e Sugestões



Flávio Sousa