

PROJETO DE SISTEMAS III

Prof. Maíra Gatti

Tutorial Eclipse

Eclipse User Guide

1. Instalação do JDK
2. Introdução
3. Perspectiva
4. Plugins
5. Workspace
 1. Adicionando views para uma perspectiva
 2. Arrumando o workbench
 3. Manipulando recursos com ferramentas externas
6. Instalação
7. Configuração Básica
 1. Estrutura das pastas nos projetos Java
 2. Use JDK ao invés de JRE (ou instale um novo JRE/JDK)
8. Configurando classpath variables
9. O primeiro projeto Java
 1. A primeira classe Java
10. Configurando o Launcher - Executando um programa
11. Importando um projeto existente
12. Configurando o Java Buildpath
13. Rodando em modo Debug
 1. Debugger
 1. Breakpoints
 2. Remote Debugging
 3. Connection
 2. Debug View
 3. Realizando debugging nos programas
14. PLUGIN'S
 1. Lombok
 1. Configuração
 2. Ativação
15. Eclipse e Junit
 1. Configurando o projeto
 2. O primeiro teste
 3. Executando o primeiro teste:
16. Eclipse e Ant
 1. Executando Ant
 2. Clear Output

Instalação do JDK

A primeira coisa a fazer para utilizar a linguagem Java é instalar o Java Development Kit, ou JDK.

Observe a estrutura de diretórios do drive C: se já houver um diretório JDK* (JDK1.2, JDK1.2.2, JDK1.2.1, JDK1.3, JDK1.3.1, JDK1.4), o ambiente já está instalado. Se não houver, será necessário instalar uma das versões. Recomenda-se a instalação do JDK 1.4, disponível na página da Sun: selecione a versão adequada para seu sistema operacional (Windows, Linux, Solaris, etc), aceite a licença de uso, informe o diretório adequado (c:\jdk1.4) e confirme. Após a instalação, a seguinte estrutura estará disponível:

```
c:\jdk1.4
```

```
    bin
    demo
    include

    jre

        bin

        lib
    lib
```

- bin - contém todos os programas utilitários necessários para compilação e execução
- demo - contém diversos programas em Java para demonstrar alguns recursos da linguagem
- include - contém os headers para interface de programas Java e C (se você não entendeu para que serve isso, provavelmente não irá precisar)
- jre (Java Run-time Environment) - contém os programas e bibliotecas necessários para executar código Java.
- jre\lib - contém as bibliotecas necessária para a execução e compilação de programas Java

Observação: o instalador do JDK normalmente inclui o caminho c:\jdk1.4\bin no PATH. Para testar, abra uma janela DOS e digite:

```
c:\winnt> java -version
java version "1.4.0"
Java(TM) 2 Runtime Environment, Standard
Edition (build 1.4.0-b92)
```

```
Java HotSpot(TM) Client VM (build 1.4.0-b92,  
mixed mode)
```

Se aparecer uma mensagem de erro informando que não foi possível achar o arquivo, ajuste corretamente o PATH. No Windows NT isso é feito pelo painel de controle, system properties. No Windows 95/98, deve ser alterado o arquivo AUTOEXEC.BAT e reinicializada a máquina.

Introdução

O Eclipse é um *framework* para integrar diferentes tipos de aplicações. Uma de suas aplicações é a JDT(Java Development Tooling), a qual já vem com o Eclipse.

Essas aplicações são oferecidas em forma de plugins e automaticamente reconhecidas e integradas pela plataforma. Tendo seus próprios recursos para gerenciamento de mecanismo, que são geralmente arquivos no seu Hard Disk. Eles residem no seu workspace, uma pasta especial localizada no seu sistema de arquivos. As aplicações instaladas comunicam-se entre si, com isso, se uma aplicação altera um recurso qualquer, todas as outras aplicações instaladas serão notificadas sobre essa mudança, garantindo uma consistência e integração em todo o seu ambiente de desenvolvimento.

Um usuário sempre trabalha no workbench, a parte "visível" da plataforma (GUI). A perspectiva escolhida determina a aparência atual do workbench. A perspectiva é uma coleção conhecida como "views e editores" que contribuem com ações especiais para o menu e a toolbar.

A maioria das views mostra informações especiais sobre os recursos. Dependendo da view somente partes ou relacionamentos internos dos recursos poderão ser mostrados. Como no caso de arquivos do tipo XML.

Um editor trabalha diretamente sobre um recurso (classes Java como exemplo). O Eclipse segue um ciclo de carrega-altera-salva que somente se um editor salvar suas alterações, a plataforma será capaz de notificar as outras aplicações.

Views especiais podem ser conectadas diretamente a um editor, adicionando recursos complementares ao manuseamento dos recursos. Por exemplo a Outline View da perspectiva Java é conectada diretamente ao Editor Java.

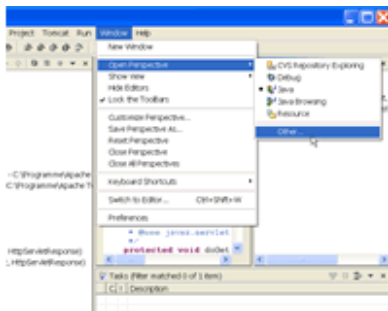
O que torna o Eclipse uma IDE especial é a extrema flexibilidade na qual podem ser combinadas views e editores. Dessa forma o workbench pode ser arrumado de uma

forma livre e que melhor adapte o desenvolvedor. As views e editores podem ser adicionados em uma perspectiva aberta (mesmo se eles foram definidos em um plugin totalmente diferente). Portanto é possível ter a total liberdade para criar o ambiente de desenvolvimento que melhor agrade ao desenvolvedor, de uma forma agradável e customizada.

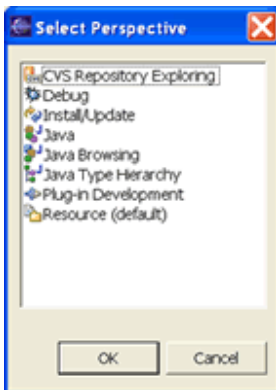
Perspectiva

Para abrir uma nova perspectiva no seu worbench faça:

1. Escolha 'Window -> Open Perspective' no menu principal:



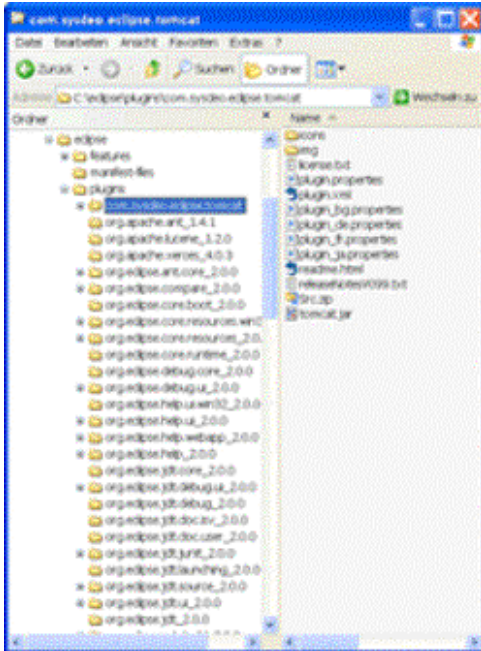
2. Escolha 'Other' para ver todas as perspectivas instaladas:



Plugins

Para instalar novas aplicações, simplesmente copie os plugins para dentro da pasta `$ECLIPSE_HOME/plugins` como mostrado com o Tomcat plugin. Será preciso reiniciar o Eclipse para tornar a nova aplicação "ativa". Dependendo plugin uma nova

perspectiva poderá ser escolhida, sendo encontradas novas opções no menu e toolbar.



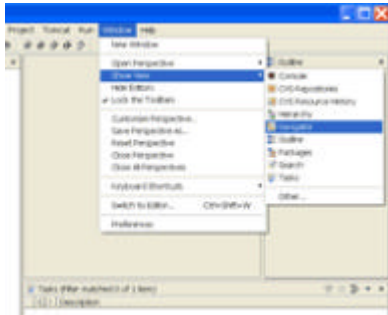
Workspace

Como default a pasta root do workspace é `$ECLIPSE_HOME/workspace`. Caso seja necessário atuar em vários projetos, ou mesmo compartilhar uma máquina para armazenamento dos projetos, faz mais sentido separar o workspace. Pode ser escolhido um diretório arbitrário para o workspace, para ter uma pasta específica, como root, basta usar a opção `-data` na inicialização do Eclipse. Por exemplo "eclipse -data c:\MeuWorkspace".

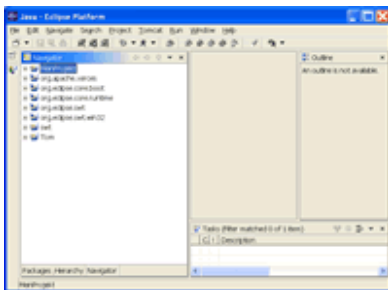
Adicionando views para uma perspectiva

1. Abra a perspectiva, ex. perspectiva Java.

2. Escolha 'Show View -> Navigator' no menu:

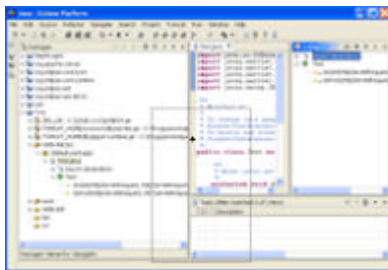


A navigator view será adicionada na perspectiva Java.

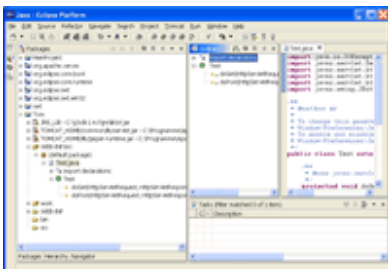


Arrumando o workbench

1. Clique na barra de título da view ou do editor e arraste para outra posição dentro do workbench

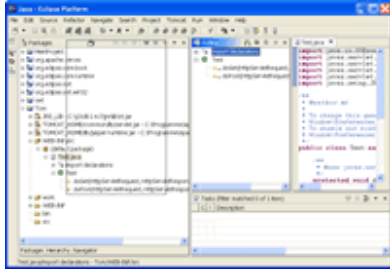


2. A aparência do workbench será ajustada automaticamente:

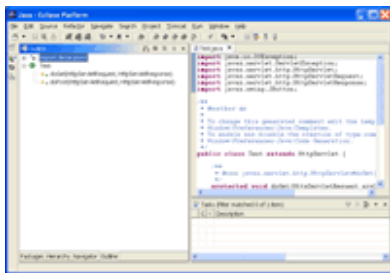


Para adicionar uma view ou um editor como uma tab faça:

3. Clique na barra de título da view ou do editor e arraste até o cursor mudar dentro do símbolo da tab.

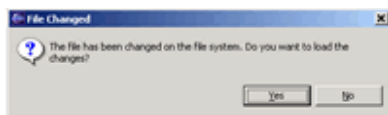


4. A aparência do workbench será ajustada automaticamente:



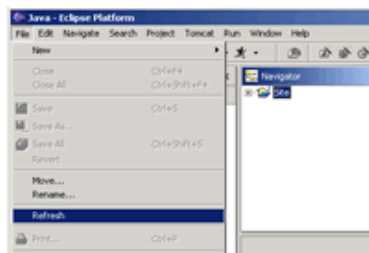
Manipulando recursos com ferramentas externas

Se um recurso existente no workspace for alterado por uma ferramenta externa, a notificação não chegará ao Eclipse automaticamente. Mas se o recurso alterado externamente estiver aberto em um editor no Eclipse, será recebida uma notificação perguntando se é desejado carregar as alterações ocorridas.



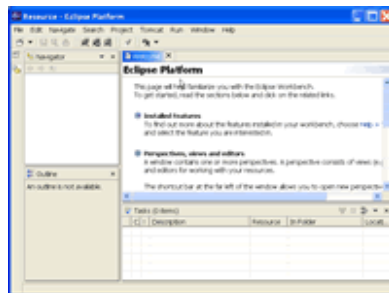
Caso sejam criados arquivos ou pastas para o seu projeto externamente, será preciso notificar ao Eclipse:

1. Selecione a pasta na qual foram criados novos arquivos / pastas na view (ex. Navigator)
2. Escolha 'File -> Refresh'



Instalação

- Sistema operacional:
 - Windows
 - Linux
 - Solaris
 - QNX
 - Mac OS/X
- Recomendável 256 MB RAM.
- Java 2 runtime environment (JRE) ou Java 2 Software Development Kit (J2SDK).
- Eclipse necessita da versão 1.3 ou superior.
- Eclipse 2.0.2-archive ou superior.
 1. Instale o respectivo JRE.
 2. Descompacte o Eclipse-archive para uma pasta arbitrária. No Windows o Eclipse acha a instalação do JRE/SQK automaticamente, através do registry.
 3. Inicialize o Eclipse executando eclipse.exe (Windows) ou eclipse.sh (Linux) localizado no diretório de instalação. Após isso, o workbench aparecerá:



Configuração Básica

Após a inicialização do Eclipse, serão necessárias algumas configurações da ferramenta.

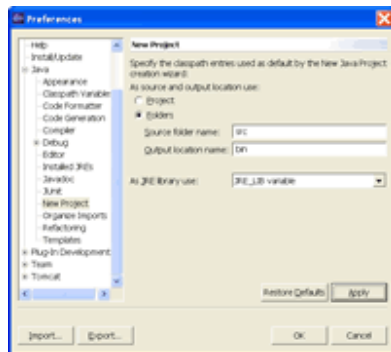
Estrutura das pastas nos projetos Java

Por padrão os arquivos fonte e compilados serão salvos diretamente na pasta do projeto. É melhor separar o código fonte do código compilado (arquivos .class). Com isso devemos ter uma pasta especial (source) que contem somente o código fonte.

Um projeto pode crescer consideravelmente através da adição de novas pastas de código fonte.

1. Escolha 'Window -> Preferences' no menu.

2. Expanda o nó 'Java' e escolha 'New Project'.
3. Escolha o radio-button 'Folders'.

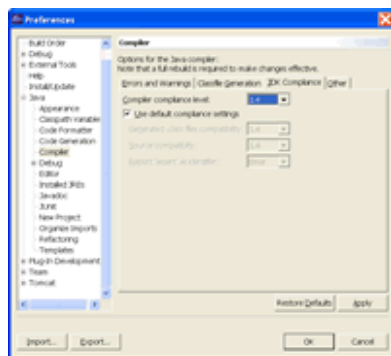


4. Pressione 'Apply'.
5. Pressione 'OK'.

Selecionar compilador e editor para o JDK

O compilador Java e editor do Eclipse podem ser configurados para trabalhar com diferentes versões de JDK.

1. Escolha 'Window -> Preferences' no menu.
2. Expanda o nó 'Java' e escolha 'Compiler'.
3. Escolha 'JDK Compliance'.
4. Escolha '1.4' na lista de seleção 'Compile compliance level'.



5. Pressione 'Apply'.
6. Pressione 'OK'.

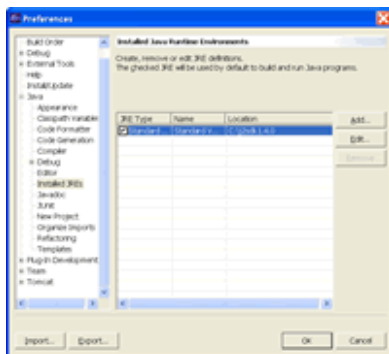
Use JDK ao invés de JRE (ou instale um novo JRE/JDK)

Para ver o código fonte das classes padrão do Java, o Eclipse precisa ser "avisado" para usar o JDK correspondente. Por padrão o Eclipse utiliza o JRE, o qual não possui código fonte (o arquivo src.zip).

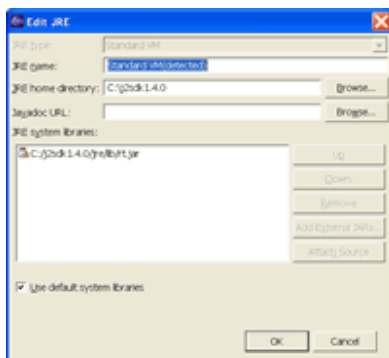
Ao usar o JDK, ajudas adicionais serão oferecidas: Code Assist pode mostrar nomes e retornos dos métodos existentes no JavaDoc e comentários através de um hover help.

Pré-requisito: O JDK precisa estar instalado no Sistema Operacional

1. Escolha 'Window -> Preferences' no menu.
2. Expanda o nó 'Java' e escolha 'Installed JREs'.
3. Escolha a primeira linha. (Standard VM).



4. Pressione o botão 'Edit' (ou 'Add'). O diálogo 'Edit JRE' (ou 'Add JRE') aparecerá:



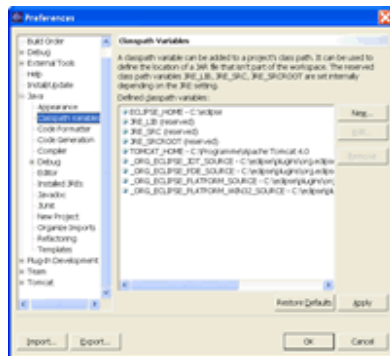
5. Pressione o botão 'Browse' na linha 'JRE home directory'.
6. Escolha a pasta no diretório de instalação no diálogo de seleção. No Windows deve ser C:\jdk1.4.1.
7. Pressione 'OK' para a pasta selecionada.
8. Pressione 'OK' para o diálogo 'Edit JRE'.
9. Pressione 'OK' para o diálogo 'Preferences'.

Configurando classpath variables

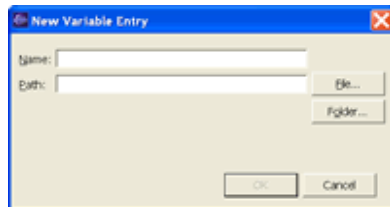
Classpath variables são definidas a nível do workbench. Pastas e arquivos .jar ou .zip podem receber nomes, fazendo com que seja fácil incluí-los no buildpath.

1. Escolha 'Window -> Preferences'.

2. Expanda o nó 'Java' e escolha 'Classpath Variables'. O seguinte diálogo aparecerá:



3. Pressione o botão 'New'. O diálogo 'New Variable Entry' aparecerá:



4. Especifique o nome da variável no textfield 'Name'.
5.
 - a) Se a variável apontar para um arquivo (.zip ou .jar), pressione o botão 'File'.
 - b) Se a variável apontar para uma pasta, pressione o botão 'Folder'.

Variáveis apontam para uma pasta que deve complementar o path para dentro do projeto que irá utilizá-la. Isto significa que tanto pastas .zip ou .jar serão adicionados ao buildpath do projeto.

6. Pressione 'OK' no diálogo 'New Variable Entry'.
7. Pressione 'OK' no diálogo 'Preferences' Dialog.

O primeiro projeto Java

Como criar um projeto Java simples, onde o código fonte esta separado na sua pasta de "saída".

Criação do projeto Java:

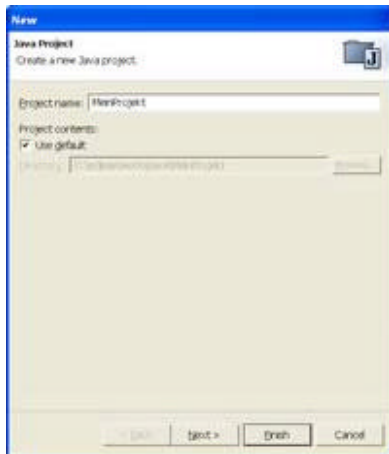
Não importa se será criado um novo projeto ou será importado um projeto existente, é necessário criar um projeto Java no Eclipse antes.

1. Abrir a perspectiva Java
2. Abrir o assistente 'New Java Project'.

Existem várias maneiras de iniciar esse assistente. A mais fácil é pressionando o botão na toolbar.

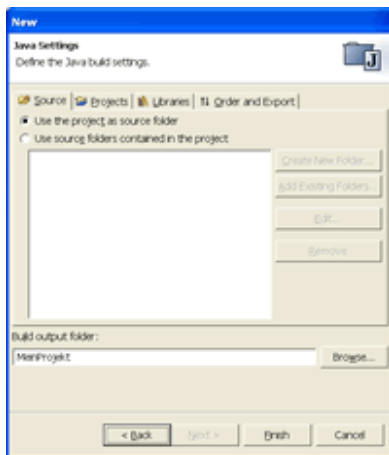
Também pode ser escolhido 'File -> New -> Project' para obter um diálogo que deixa escolher 'Java project'.

O diálogo 'New Java Project':



3. Escolha um nome para o seu projeto, ex. "MeuProjeto". O padrão para a pasta do projeto é workspace_home/MeuProjeto .
4. Pressione 'Next'. O diálogo 'New Java Settings' aparecerá:

Com esse diálogo será editado o arquivo .classpath do projeto.

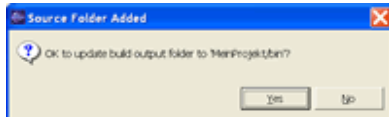


A tab 'Source' serve para especificar pastas que contenham recursos Java (arquivos .java).

5. Selecione o radio button 'Use source folders contained in the project'.
6. Pressione o botão 'Create New Folder'. O diálogo 'New Source Folder' aparecerá:

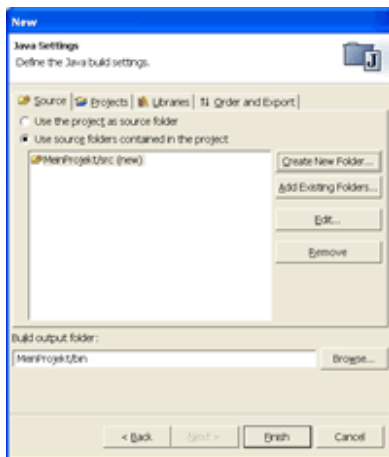


7. Digite "src" no textfield.
8. Pressine 'OK'. O diálogo 'Source Folder Added' aparecerá:



9. Pressione 'Yes'. A pasta de saída para as classes compiladas será especificada como workspace_home/MeuProjeto/bin

O diálogo 'New Java Settings' aparecerá:




10. Pressione 'Finish'.

A primeira classe Java

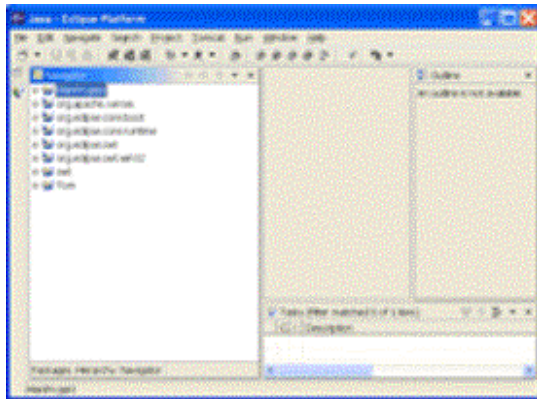
Criação da classe Java:

1. Abra a perspectiva Java
2. Abra o assistente 'New Java Class'.

Existem várias formas de iniciar o assistente, a mais fácil é pressionando o botão  na toolbar.

Também pode ser escolhido 'File -> New -> Class' e será mostrado o diálogo para escolher 'Java project'.

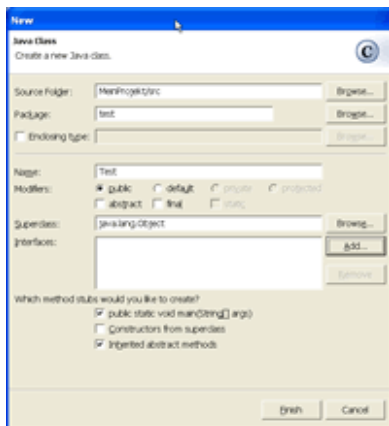
O diálogo 'New Java Class' aparecerá:



O primeiro botão chamado 'Browse' serve para selecionar a pasta para os códigos fontes, no qual as classes serão criadas, nesse caso 'MeuProjeto/src'.

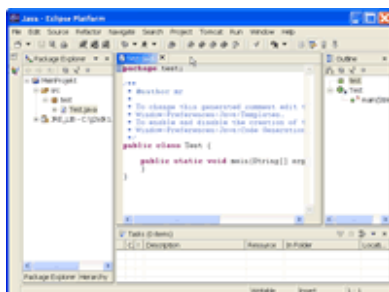
3. Digite "test" como o nome do package Java. (Package)
4. Digite "Test" como o nome da classe. (Name)
5. Marque o checkbox 'public static void main(String[] args)' Podem ser especificadas superclasses e interfaces para implementação. Para o primeiro teste será criado um simples método main.

O diálogo será:



6. Pressione 'Finish'

A nova classe será aberta no editor Java:



7. Digite 'System.out.println("Hello world!");' no corpo do método main.

8. Salve as mudanças para a nova classe. (Ctrl+S ou no menu)

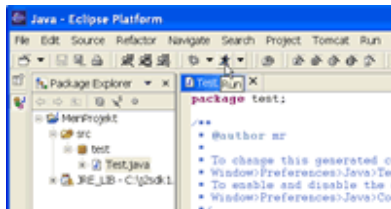
Assim que ocorrer o salvamento, o código fonte será compilado. (Essa ação pode ser alterada em preferencias) Para executar a classe, será necessário configurar o launcher.

Configurando o Launcher - Executando um programa

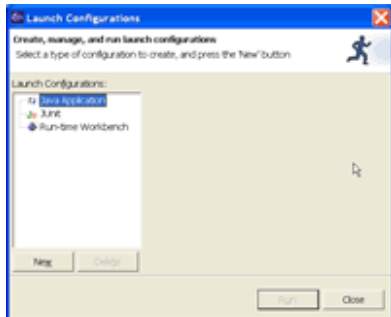
Para executar o método main, será preciso configurar o launcher:

Como atalho pode ser escolhido 'Run As -> Java Application' e no menu dropdown 'Run'. Isso é totalmente suficiente para executar um programa simples (sem parâmetros etc.)

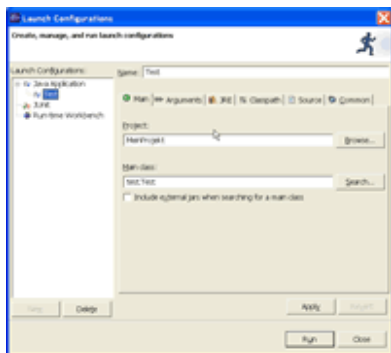
1. Pressione o botão 'Run' na toolbar:



O diálogo 'Launch Configurations' aparecerá:

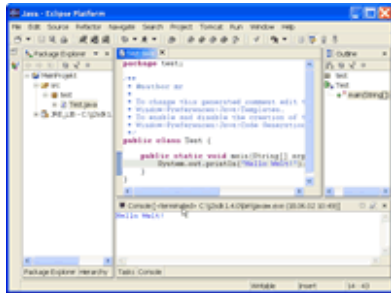


2. Selecione 'Java Application' e pressione 'New'. A segunda parte do diálogo 'Launch Configurations' aparecerá:



Várias configurações podem ser informadas (parâmetros para a VM, classpath, ações do workbench quando inicializado etc.).

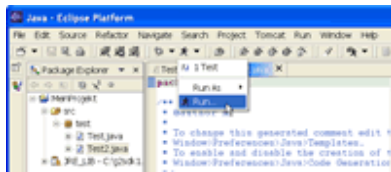
3. Pressione 'Run'. A Console view da perspectiva Java aparecerá contendo a saída padrão:



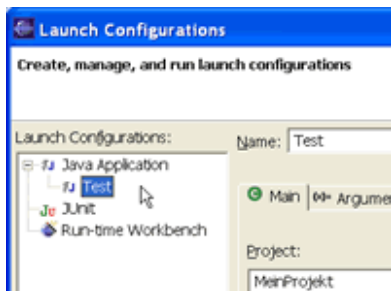
Pressionando o botão 'Run', a última ação será executada novamente, nesse caso a classe 'Test'.

Será necessário criar uma nova configuração:

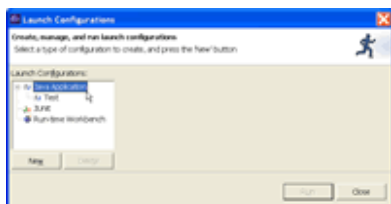
1. Escolha 'Run' no menu dropdown:



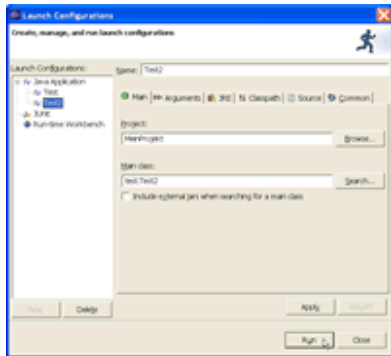
O seguinte diálogo 'Launch Configurations' aparecerá para completar a configuração:



Selecione 'Java Application':



Pressione 'New'. A nova classe aparecerá no diálogo:



Pressione 'Run'.

Importando um projeto existente

Para importar um projeto existente, será necessário:

1. Criar um projeto Java.
2. Escolha 'File -> Import'. O diálogo 'Import Select' aparecerá.
3. Selecione 'File System'.



4. Pressione 'Next'. O diálogo 'Import File System' aparecerá
5. Especifique o diretório que conterá o projeto importado com o primeiro botão 'Browse'. Esse diretório será o da raiz para o seguinte diálogo de seleção.
6. Selecione as pastas e arquivos que deverão ser importados para dentro do projeto.
7. Especifique a pasta que na qual serão armazenados os arquivos importados, com o segundo botão 'Browse'.

Poderão ser escolhidas pastas de projetos existentes e seus subdiretórios. A opção 'Create complete folder structure' também cria pastas hierarquicamente no projeto.

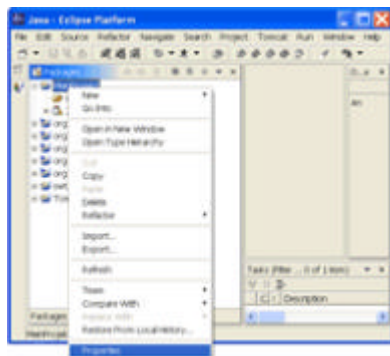


8. Pressione 'Finish'. As pastas e arquivos selecionados serão adicionados ao projeto.

Configurando o Java Buildpath

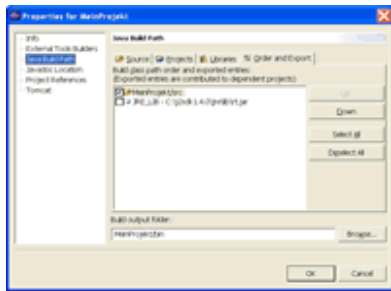
O buildpath avisa ao compilador onde procurar por vários arquivos e recursos para realizar o seu trabalho. Isso significa que devem ser compilados o código fonte e as classes definidas através das implementações. Essas informações são armazenadas no arquivo `.classpath` do projeto Java. Não é aconselhável editar esse arquivo diretamente, mas sim usar um assistente especial.

1. Selecione o projeto no Package Explorer ou Navigator view.
2. Abra o menu para esse projeto (right-click):
3. Escolha 'Properties' no menu:



O diálogo 'Properties for ProjektName' aparecerá.

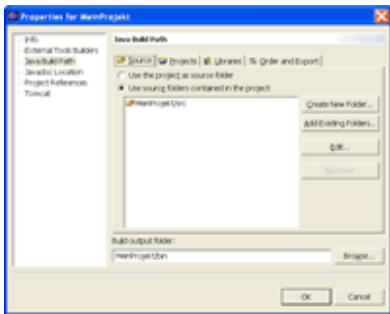
4. Selecione 'Java Build Path'. O diálogo aparecerá:



5. Configure o buildpath com a seguinte descrição.

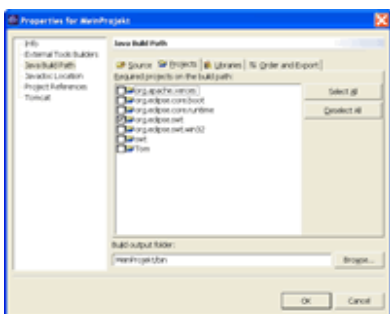
Significado das várias tabs:

Source:



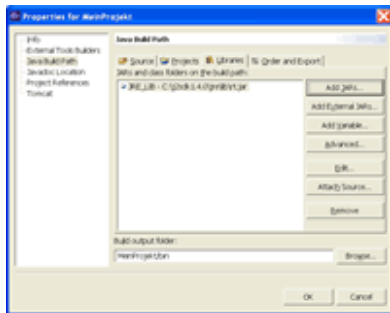
Aqui serão especificadas as pastas nas quais os códigos fonte serão armazenados. Somente nessas pastas novas classes e packages poderão ser criados. Por padrão novas classes serão criadas diretamente nessas pastas dos projetos. Para separar o código fonte dos arquivos compilados, precisa ser especificado aqui.

Projects:



Aqui projetos existentes podem ser adicionados ao buildpath. O compilador procura por classes definidas nos projetos especificados.

Libraries:



Referências para arquivos .jar e .zip dentro do workspace ou em qualquer lugar do seu HardDisk pode ser especificado aqui.

Detalhes:

- Add JARs: Arquivos que existem dentro de projetos no workspace serão adicionados aqui.
- Add External JARs: Arquivos arbitrários no seu HardDisk serão adicionados aqui.
- Add Variable: Adiciona uma classpath variable.
- Advanced: O diálogo auxilia a criação de pastas para classes. Essas pastas contem definições de classes descompactadas (arquivos .class, não .jar ou .zip).
- Edit: Muda entradas existentes.
- Attach Source: Adiciona o código fonte para uma referência existente. O código fonte será mostrado no editor Java, mas não poderá ser alterado.
- Remove: Seleciona uma entrada e remove do buildpath.
- Change the order of the entries. O compilador sempre utiliza a definição de classes que encontrar primeiro.
- Textfield 'Build output folder': A pasta que recebe as classes compiladas. Essa pasta não deve conter código fonte.

Rodando em modo Debug

O Eclipse permite rodar programas em modo debug, ou seja, habilitar ferramentas como breakpoints e variable tracking.

Debugger

O debugger permite detectar e diagnosticar erros em programas sendo executados localmente ou remotamente. Torna-se fácil controlar a execução dos programas, através da adição de breakpoints, os quais suspendem a inicialização, possibilitando analisar o código por "dentro" e examinar o conteúdo das variáveis.

A utilização dessa perspectiva é através de um design cliente/servidor, sendo assim os programas podem rodar remotamente em outros sistemas operacionais na rede, como rodariam localmente na estação de trabalho do desenvolvedor. O servidor para debugger roda no mesmo sistema operacional que o programa a ser analisado, o qual pode estar na máquina do desenvolvedor (local debugging) ou um sistema operacional que esteja acessível através da rede (remote debugging).

BREAKPOINTS

Breakpoints são marcos colocados no programa, que avisam ao debugger onde parar. Quando o workbench está rodando um programa e encontra um breakpoint, a execução é suspensa. O thread correspondente é suspenso (temporariamente para de rodar) permitindo que seja vista toda a stack para o thread em questão.

A execução é suspensa antes do statement que contém o breakpoint. Os conteúdos das variáveis e a stack podem ser checados nesse momento, além de realizar step over, step into em métodos ou classes, a execução rodará até encontrar o próximo breakpoint ou o fim do programa.

Os breakpoints podem ser desabilitados sem suspender a execução, e habilitados em outro momento.

REMOTE DEBUGGING

Remote debugging permite rodar aplicações em um sistema operacional e realizar o debug em outro sistema operacional. O sistema local roda o debugger, e o sistema remoto roda o debugging engine e o programa.


CONNECTION

O principal requerimento para remote debugging é acessar a máquina remota, na qual os arquivos onde ocorrerá o debug devem residir. Ao realizar o remote debugging em um programa, o debug engine daemon inicia escutando a conexão, após realizada pode ser feito o debug do programa.

Debug View

Esta view permite o gerenciamento para realizar o debugging de um programa no workbench. Ela mostra uma janela com a stack de cada thread suspenso para cada

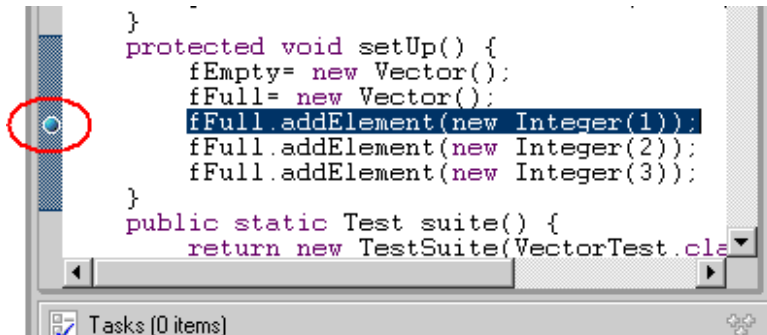
tarefa que esteja realizando o debugging. Cada thread aparece como um nó da árvore.

Comandos da Debug View		
Botões da Toolbar	Comando	Descrição
	Resume	Este comando executa o thread suspenso.
	Suspend	Este comando suspende o thread da tarefa, podendo navegar ou modificar o código, inspecionar os dados, etapas e assim por diante.
	Terminate	Este comando termina a tarefa do debug selecionado.
Somente no menu	Terminate & Remove	Este comando termina todas as tarefas de debug selecionadas e remove-as da janela.
Somente no menu	Terminate All	Este comando termina todas as atividades iniciadas na janela.
	Disconnect	Este comando fecha todas as conexões das tarefas de debug
selecionadas quando executado remote debugging.		
	Remove All Terminated Launches	Este comando limpa da janela todas as tarefas de debug terminadas.
	Step Into	Este comando executa um steps into no statement selecionado.
	Step Over	Este comando executa um step over no statement selecionado. A execução continuará até a próxima linha no caso do mesmo método, ou voltará para o método de origem, caso o método corrente tenha sido chamado por outro método. O cursor é posicionado no método e seleciona a linha.
	Run to Return	Este comando sai do método corrente. Esta opção para a execução após sair do método corrente.
	Show/Hide Qualified Names	Esta opção pode ser selecionada para mostrar ou ocultar qualified names.
Somente no menu	Relaunch	Este comando reinicia a tarefa de debug selecionada.
Somente no menu	Properties	Este comando mostra as propriedades do processo selecionado, também permitindo visualizar as linhas de comando do processo selecionado.

Realizando debugging nos programas

1. Abra alguma classe na perspectiva Java.

2. Coloque o cursor na barra do lado esquerdo da janela na qual esta aberta a classe e de um duplo clique ao lado do statement desejado, adicionando um breakpoint.

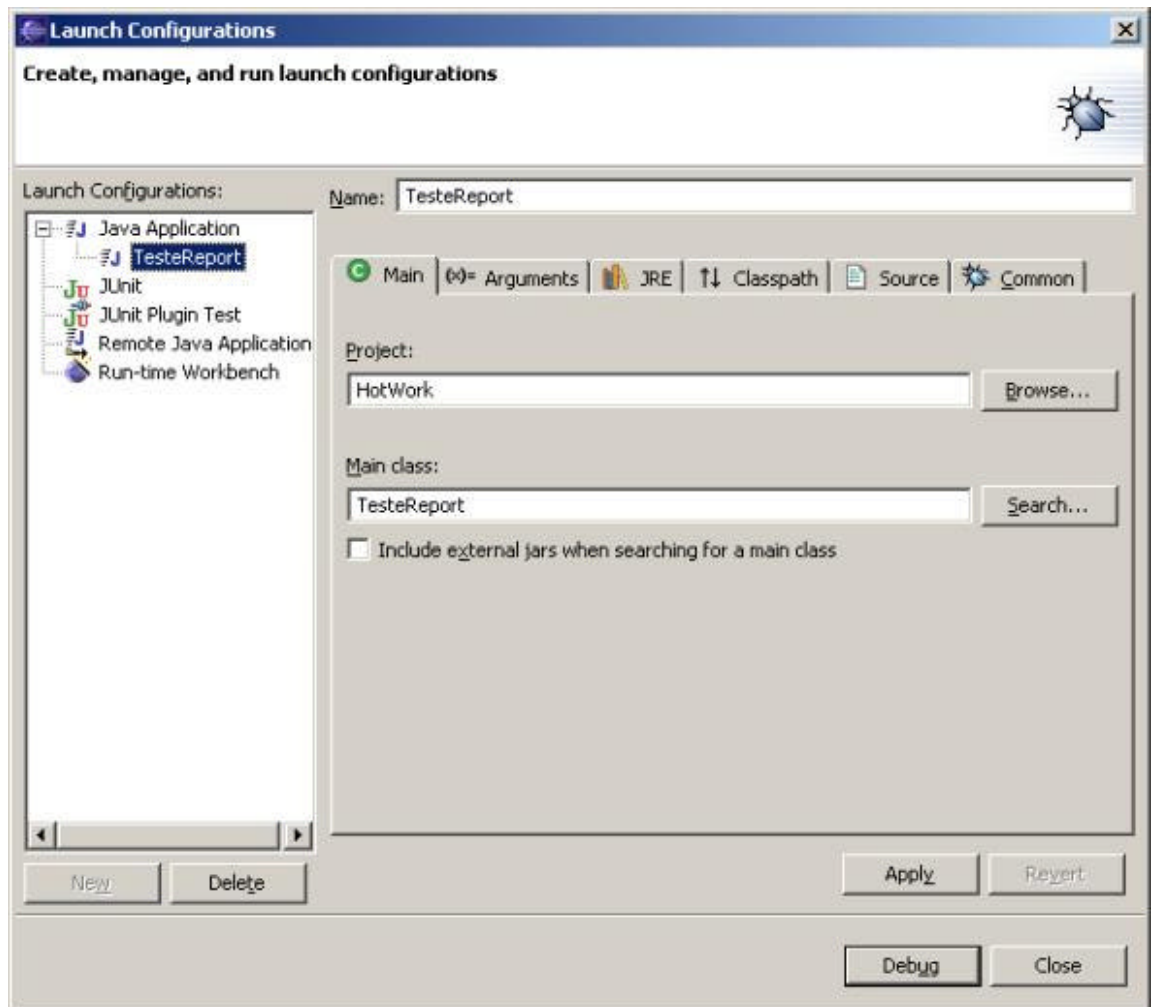


Adição de um Breakpoint.

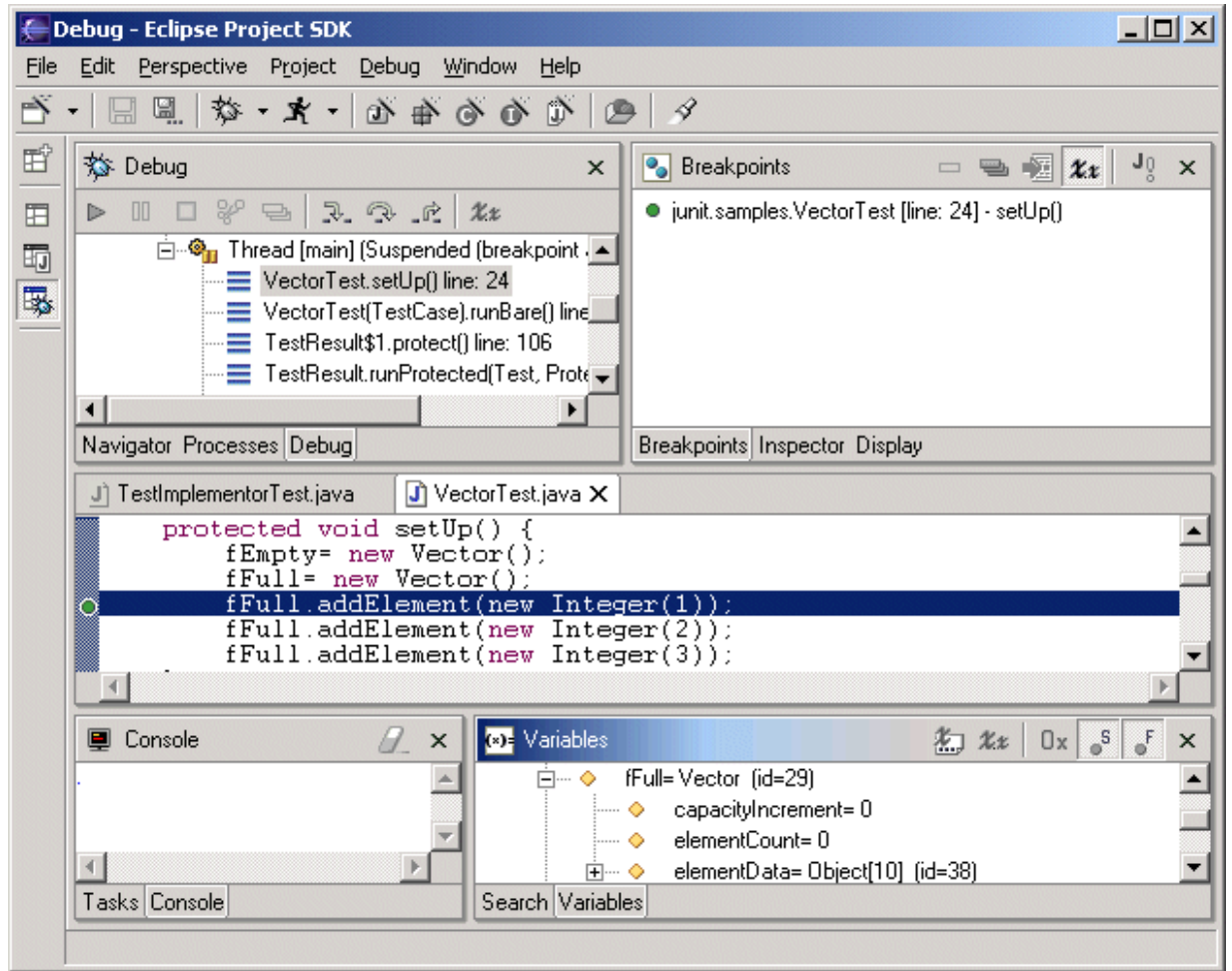
O breakpoint está em azul porque ainda não foi verificado, significando que a classe que o contém ainda não foi carregada pela Java VM.

3. Clique no botão Debug na toolbar.

4. Selecione a classe na caixa de diálogo e pressione Debug.

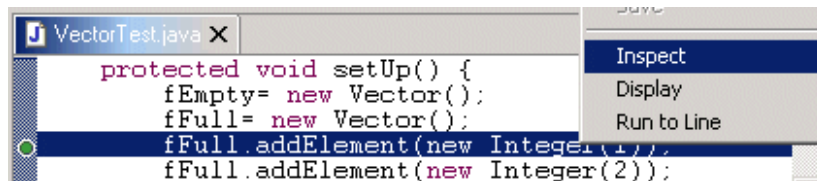


5. Tão logo tenha adicionado um breakpoint, a perspectiva Debug abre automaticamente, e a execução é suspensa. Perceba que o processo ainda está ativo (não foi terminado) na Processes view. Outros threads ainda devem estar rodando.

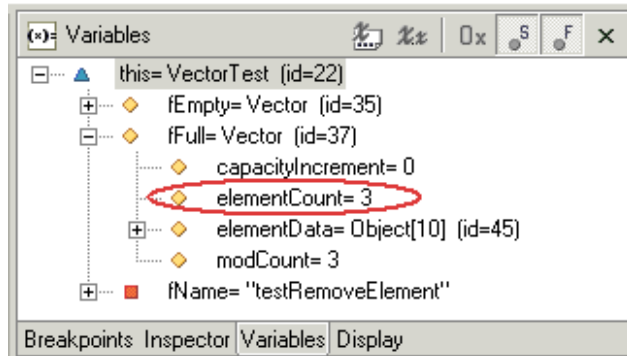


O breakpoint agora esta verde, pois foi verificado pela Java VM.

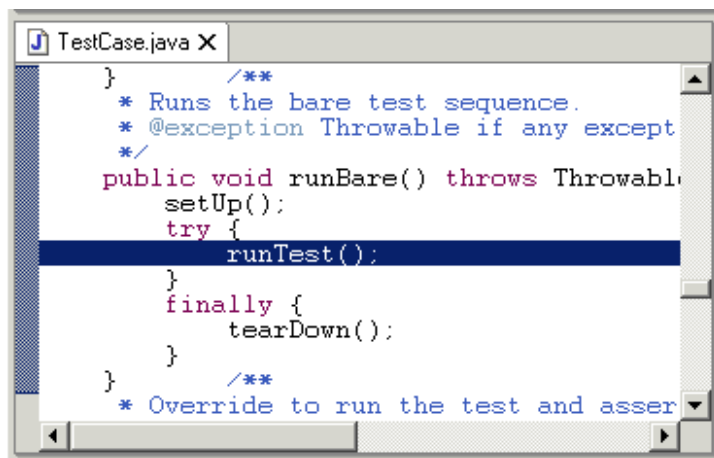
- No editor na perspectiva Debug, selecione a linha onde esta o breakpoint e no menu selecione Inspect.



A Variables view mostra os valores das variaveis no stack frame selecionado.



7. Pressione Step Over para executar a linha de código selecionada.



Se o programa não terminou totalmente após o termino do debugging, selecione Terminate no menu para o programa iniciado, na Processes view ou Debug view.

PLUGIN'S

Além dos plug-ins como o JDT para editar, compilar e debugar aplicações, outros plug-ins estão disponíveis para suportar completamente o processo de desenvolvimento através de modelagem, automação de deployment, teste unitário, teste de performance, controlador de versão e gerencia de configuração.

Dica: Adicionando novas aplicações ao Eclipse

Para instalar novas aplicações, simplesmente copie os plugins para dentro da pasta \$ECLIPSE_HOME/plugins. Será preciso reiniciar o Eclipse para tornar a nova aplicação "ativa". Dependendo do plugin uma nova perspectiva poderá ser escolhida, sendo encontradas novas opções no menu e toolbar.

Lomboz

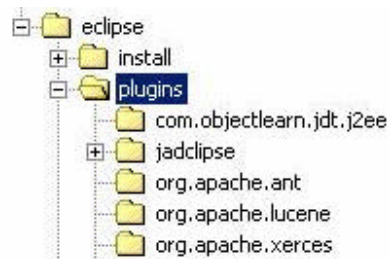
Esse plugin auxilia aos programadores Java realizarem build, teste e deploy das aplicações Java nos Servidores de aplicação J2EE.

Ele oferece uma série de funcionalidades, tais como:

- Editor de JSP com sintaxe highlighting e code assist
- Verificador de sintaxe JSP
- Wizards para geração de módulos Web e Ejb
- Wizards para geração de Ejb test clients
- Suporte a deployment de WAR e Ejb(Jar)
- Produtividade usando wizards e geradores de código
- Desenvolvimento integrado de EJB 1.1 e 2.0 com Xdoclet (Session, Entity and Message Driven Beans)

Instalação

- Execute o unzip do arquivo lomboz.zip para o diretório <Eclipse_Home>

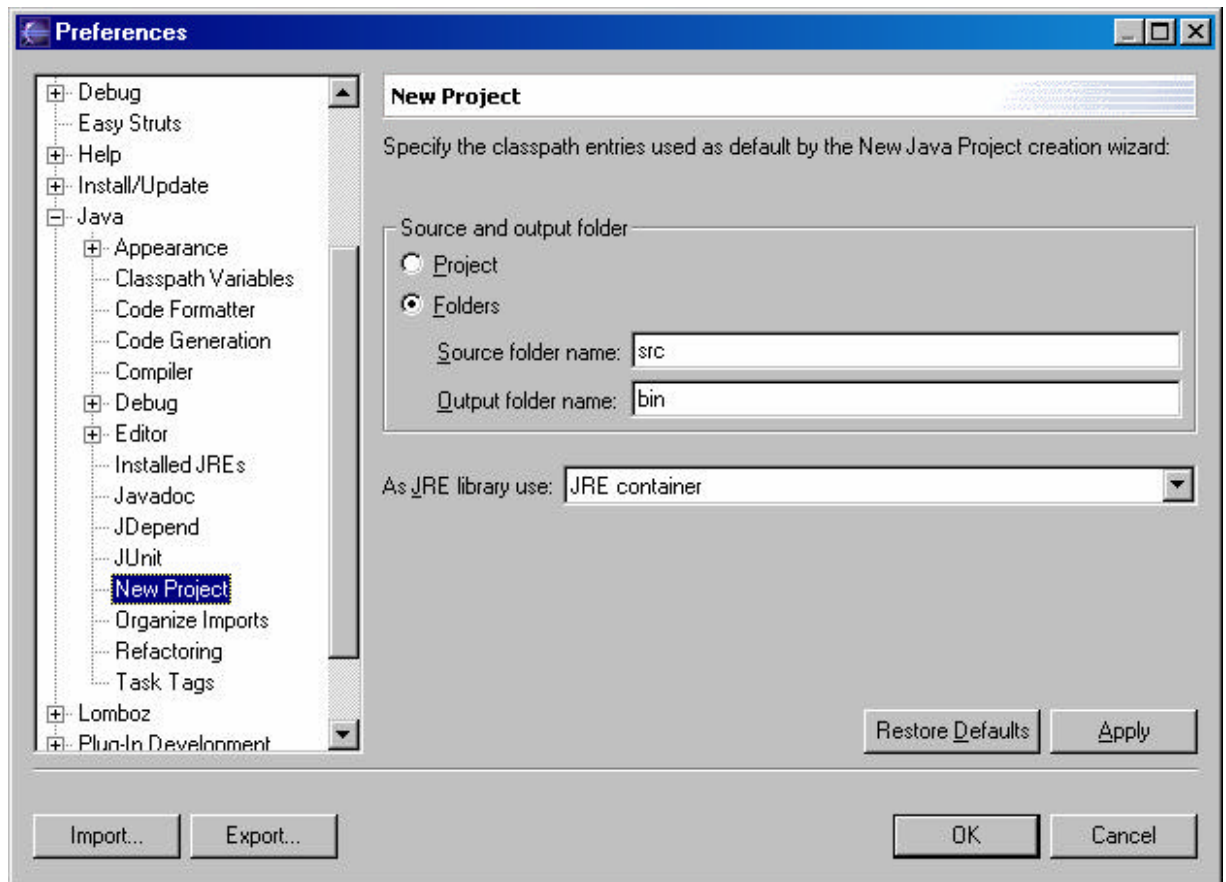


- Será criado um novo diretório chamado com.objectlearn.jdt.j2ee no diretório plugin
- Agora inicialize o Eclipse

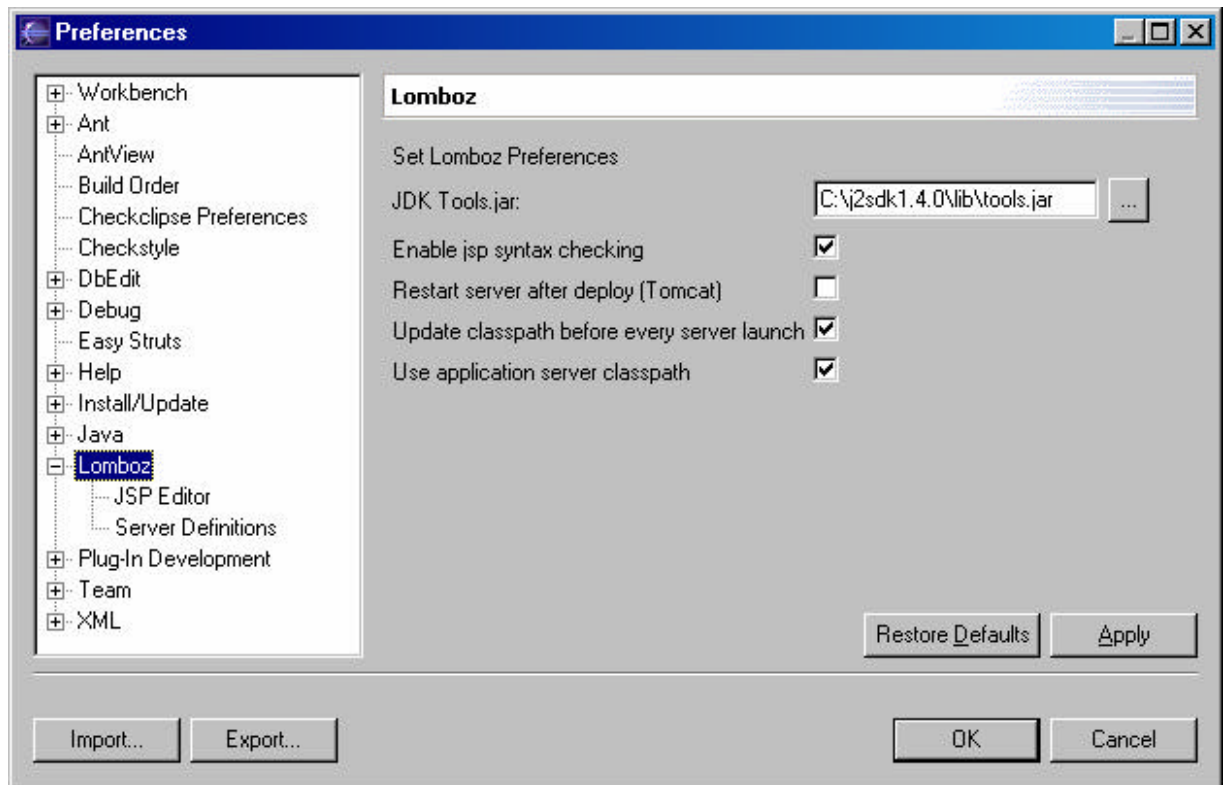
Configuração

- Caixa de diálogo no menu -Preferences e expanda o item Java> New Project.

- As configurações devem estar igual à figura.

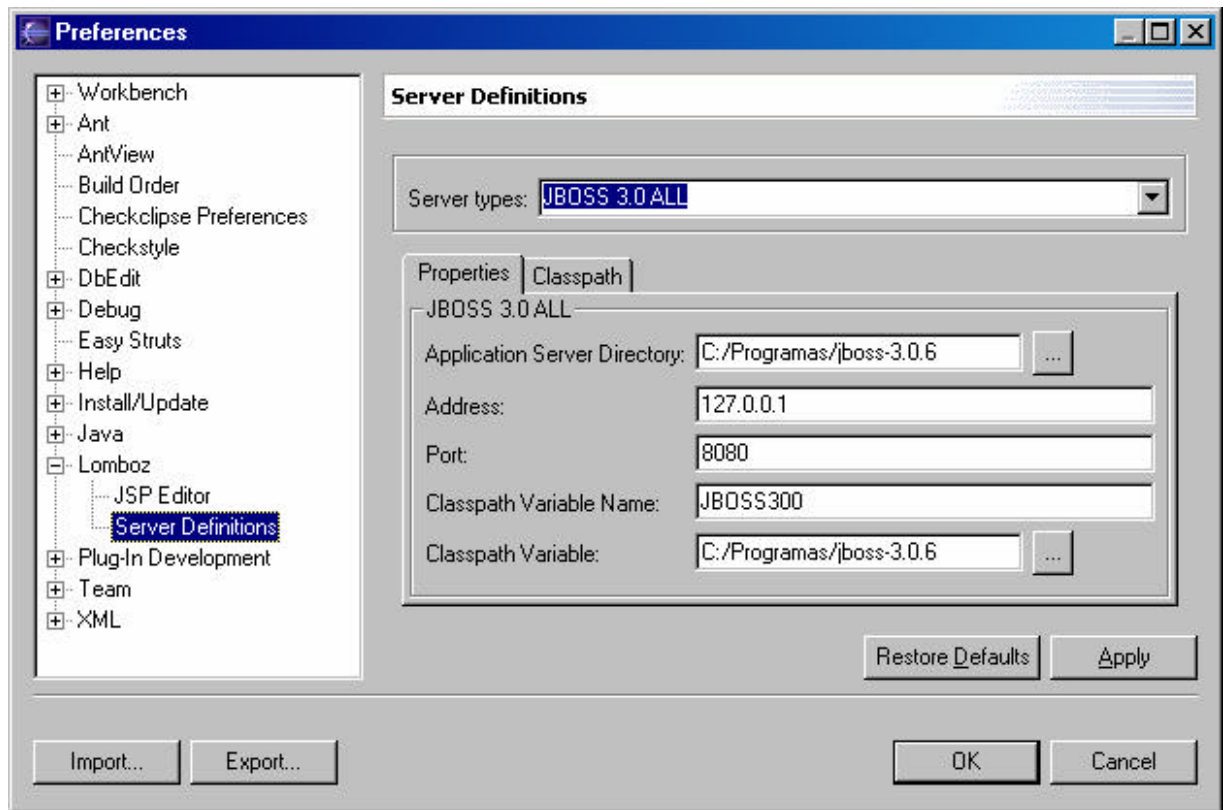


- A maioria dos Applications Servers utilizam o compilador Java padrão (javac), o mesmo é localizado em tools.jar que vem com a distribuição padrão do Java



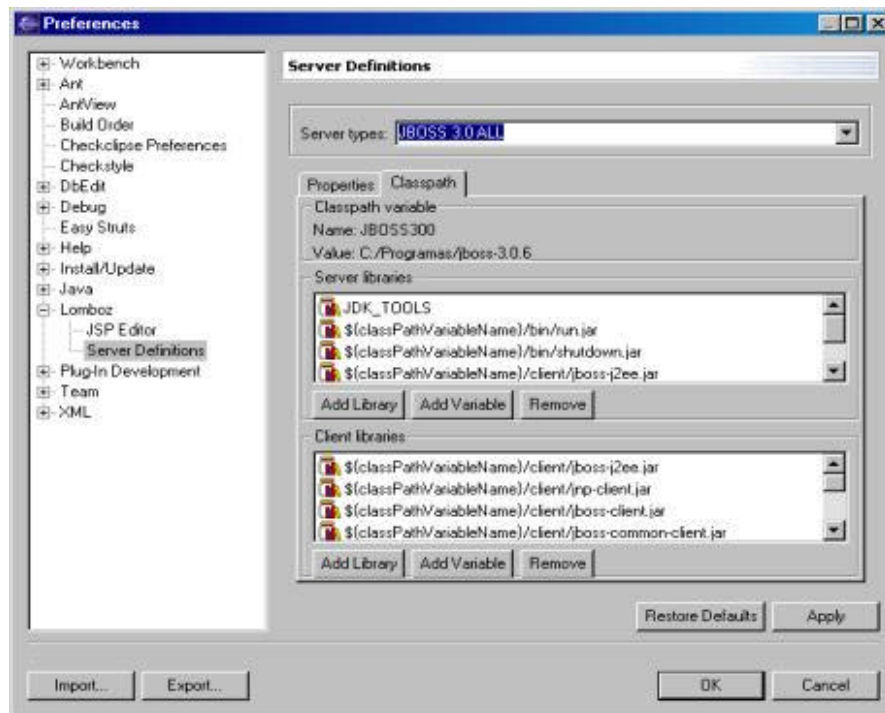
- o Enable JSP Syntax checking - Permite a compilação dos JSP's
- o Server restart- Toda vez que for feito um novo deploy de um módulo, o server será reiniciado
- o Update server classpath and deployment parameters - O Lombok verifica o classpath do projeto e as propriedades de deployment.
- o Use application server classpath - Utiliza o classpath definido na configuração do server para inicia-lo e não o classpath do projeto. Evita problemas de class loaders

- Definindo o Application Server



- Classpath variable name - O nome da variável que será usada para referenciar as java libraries
- Classpath variable - O caminho que será usado para armazenar a variável. As java libraries serão um caminho relativo
- Server Home Directory - O caminho para o diretório de instalação do Application Server

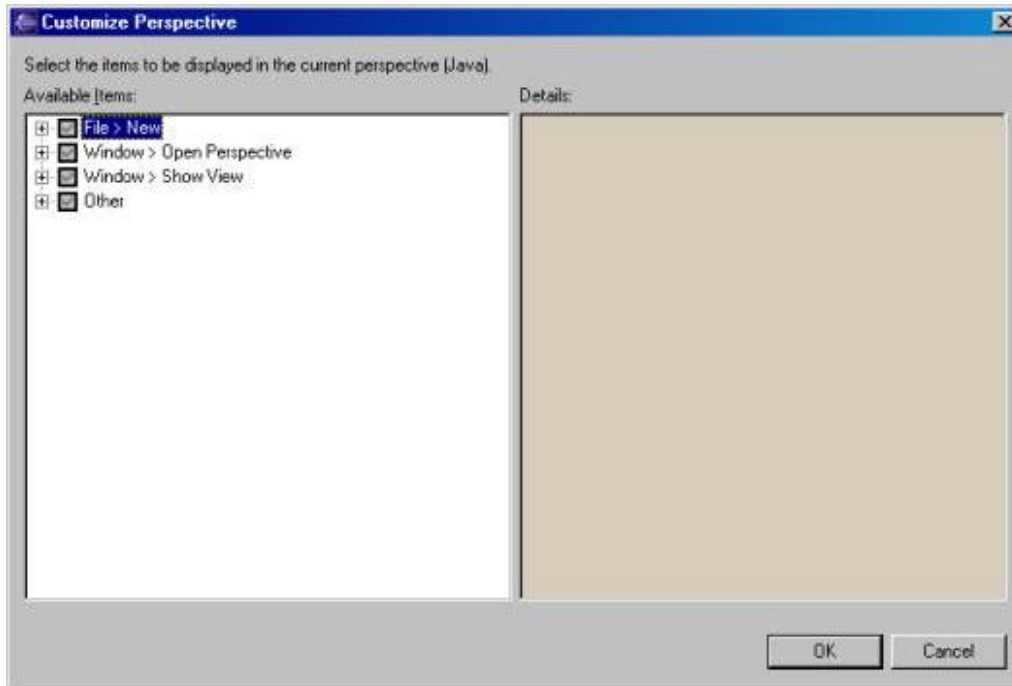
- o Classpath Page



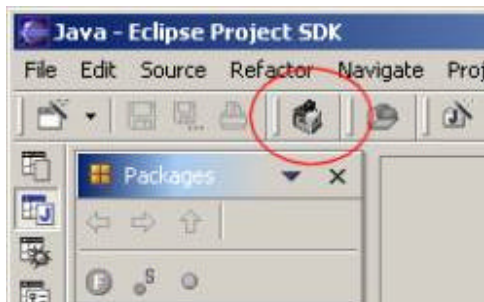
- Java libraries que serão utilizadas para iniciar o Application Server
- Server Classpath é utilizado para iniciar o Application Server
- Client path utilizado por aplicações cliente (ex.: Ejb Test Clients)

Ativação

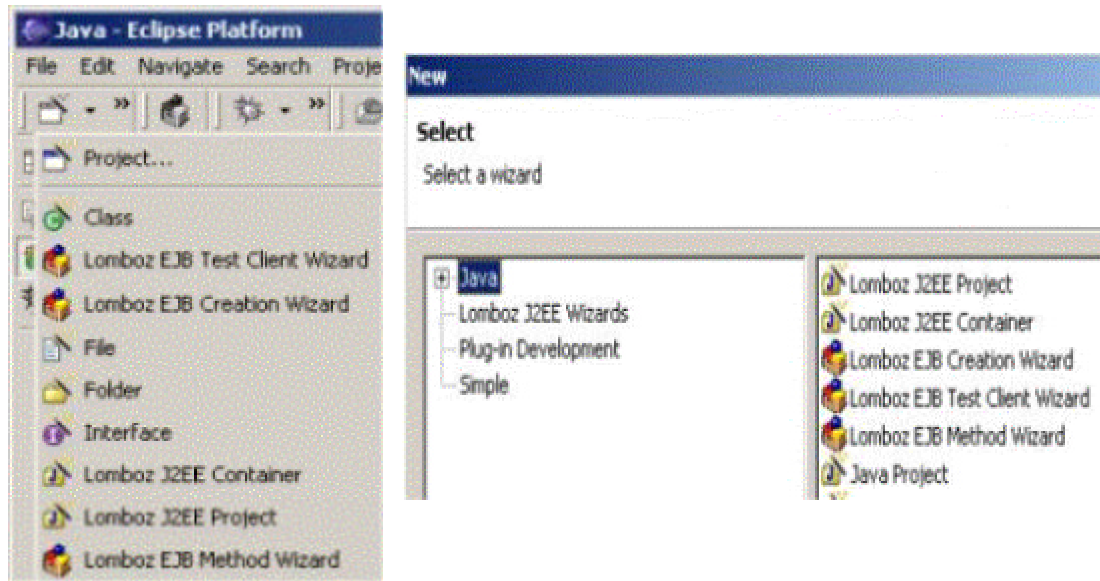
Adicionar ações, wizards e views para o Eclipse.



- Selecione no menu 'Window>Customize Perspective...', expanda 'File>New' e marque todos os itens relacionados ao Lombok.
- Selecione no menu Window>Customize Perspective...', expanda 'Other' e marque 'Lombok Actions'
- Selecione no menu Window>Customize Perspective...', expanda 'Window>Show View' e marque 'Lombok J2EE View'
- Após essas seleções, será adicionado um novo botão a toolbar



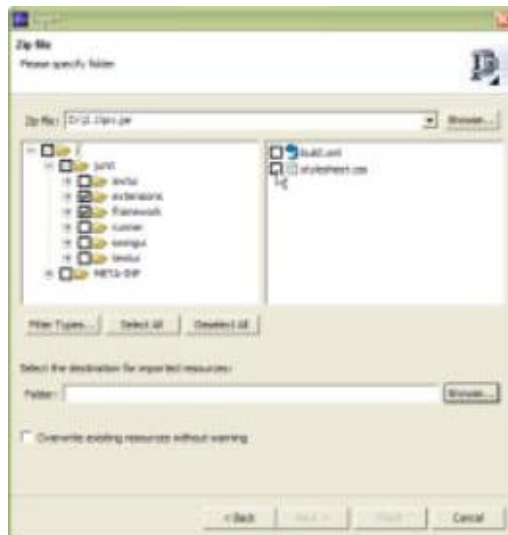
- Novos wizards estarão disponíveis no diálogo New Project e no item de menu New



Eclipse e Junit

Através do uso do Eclipse com o Junit, torna-se fácil desenvolver códigos de boa qualidade.

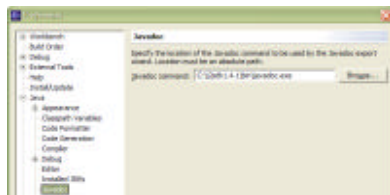
1. Download JUnit em www.junit.org. Descompacte src.jar a partir do arquivo .zip.
2. Crie um novo projeto Java Junit.
3. Escolha 'File- >Import- > Zip file- > Browse- > src.jar' .



4. Selecione a pasta src (Pressione o botão 'Browse'):



5. Pressione Finish.
6. Suporte opcional ao Javadoc: Selecione o caminho para o Javadoc Path em 'Preferences':



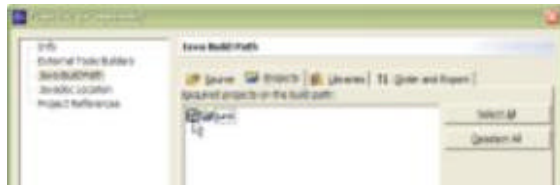
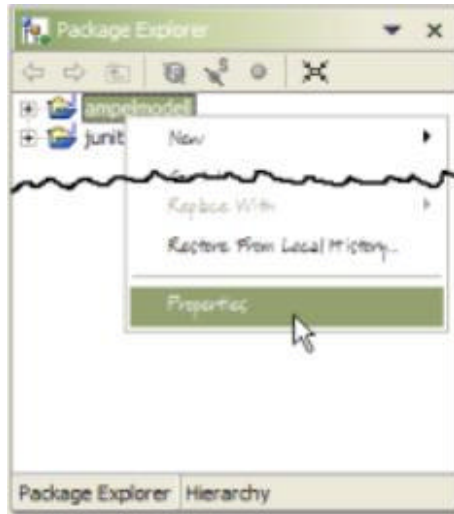
e selecione a documentação em junit3.8.1.zip :



O local no qual o arquivo .zip foi extraído.

Configurando o projeto

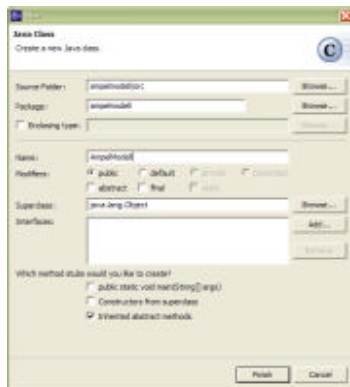
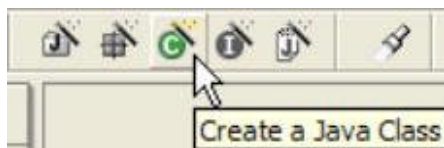
1. Adicione o projeto Junit ao 'Java build path' na 'Properties' do projeto:



O primeiro teste

O primeiro teste é para uma instancia da classe AmpeImodell que deve estar no estado vermelho. O estado dessa instancia é acessado por três métodos getters, getRed(), getYellow() e getGreen() cada um retornando valores do tipo boolean.

1. Criar uma nova classe AmpeImodellTest:



2. Selecione junit.framework.TestCase as superclass ('Browse'):



3. Pressione Finish.

Eclipse cria uma nova classe:



Métodos que contenham os testes serão adicionados. JUnit procura automaticamente por métodos que iniciam com 'test':

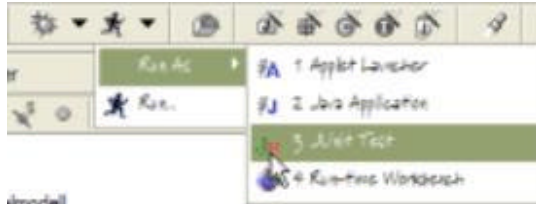
```
public class AmpelModellTest extends TestCase {
    public void testNewAmpel() {
        AmpelModell a = new AmpelModell();
        assertTrue(a.getRed());
        assertFalse(a.getYellow());
        assertFalse(a.getGreen());
    }
}
```

```
public class AmpelModell {
    public boolean getRed() {
        return false;
    }
    public boolean getYellow() {
        return false;
    }
    public boolean getGreen() {
```

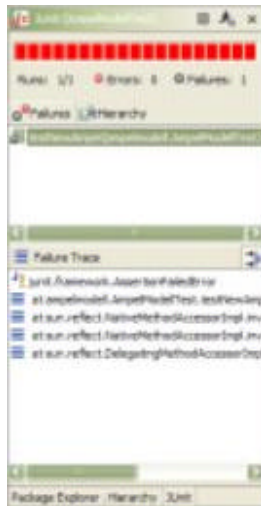
```
return false;
}
}
```

Executando o primeiro teste:

1. Escolha 'Run As -> Unit Test' no menu:



2. A view JUnit aparecerá. A barra vermelha indica que algo deu errado:



É mostrado o total dos testes executados, as exceções lançadas e o testes que falharam. Tão logo algum teste falhe, a barra de status aparecerá vermelha. A área do meio da view JUnit tem duas abas.

- o 'Failures'
 - Lista os testes que falharam.
- o 'Hierarchy'
 - Mostra a visão geral de todos os testes que foram executados e é extremamente usual quando executando um conjunto de testes (TestSuite).

A view abaixo mostra o 'Failure Trace'. Pode ser visto a call stack da falha do teste. Um clique duplo localiza o código fonte onde foi ocorrido a falha.

FIM