Desenvolvimento de Software Utilizando o Eclipse e Ferramentas de Software Livre.

Alexandre Freire [alex@arca.ime.usp.br]

IME/USP

Objetivos

Após este curso você vai saber:

- Descrever a motivação por trás do Eclipse
- Usar os recursos do Eclipse confortavelmente
- Desenvolver e depurar aplicações Java
- Escrever testes com JUnit e aplicar outras técnicas de XP
- Compartilhar código com CVS
- Conhecer o Ant
- Trabalhar com aplicações J2EE no Tomcat
- Achar e instalar os plugins mais legais do Eclipse

Oque é o Eclipse afinal?

Um IDE comum para todos (ou para o programador esquizofrênico...)

- O Eclipse não é uma IDE propriamente dita, é um arcabouço para desenvolvimento de ferramentas
- Estensível, aberto e portável. Através de *plugins*, diversas ferramentas podem ser combinadas criando um ambiente de desenvolvimento integrado
- Uma mesma plataforma para vários papéis de desenvolvimento: programador de componentes, integrador, responsável por testes, web designer...
  - O time inteiro usa a mesma aplicação
- Suporte para desenvolvimento de novas ferramentas
- Suporte para desenvolvimento de aplicações Java na Web

Ferramentas disponíveis

Desenvolvimento integrado de ponta-a-ponta para Java

- Compilação incremental e automática + suporte a Ant
- Depuração flexível
- Ambiente de testes (JUnit)
- Interpretador Java
- Substituição Dinâmica de código
- Suporte a múltiplos ambientes java (JREs)
- Assistente de código
- Refatoração
  - Permite renomear/mover métodos/classes/pacotes
  - Conserta todas as dependências automaticamente

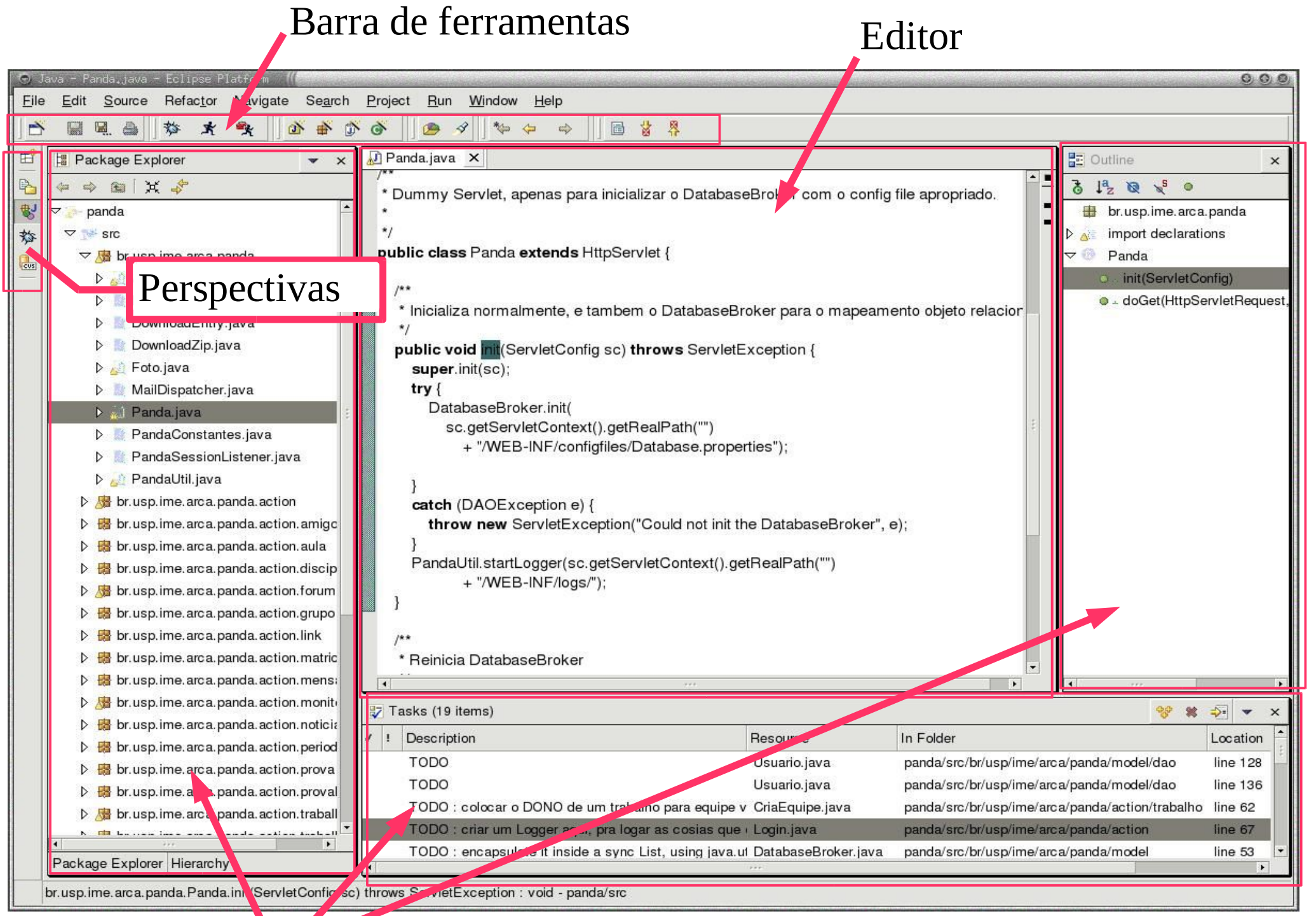
Mais ferramentas...

- Lista de tarefas e erros
- Controle de versões
- Controle de recursos
- Buscas avançadas
- Ajuda *on-line*
- Arcabouço dinâmico: deteção e carga automática
- PDE: Ambiente de desenvolvimento de plugins
  - Novas tecnologias para criar interfaces gráficas: SWT e JFace
- Muitos plugins na comunidade
  - Editores de HTML, XML, JSP...
  - Interfaces para Banco de Dados
  - Editores de UML

A bancada de trabalho

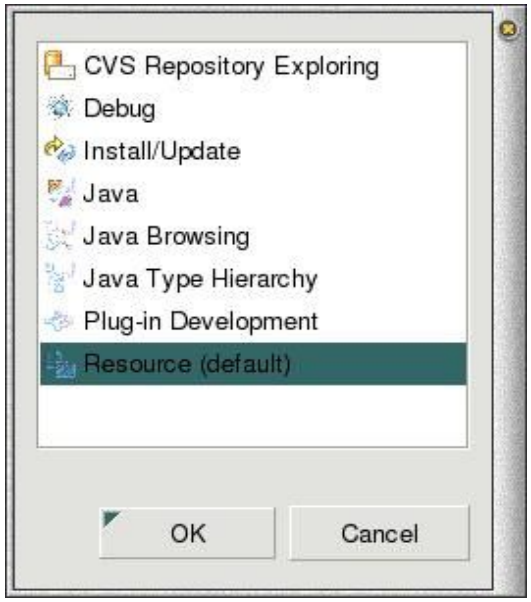
O iluminismo nas IDEs, é tudo uma questão de perspectiva

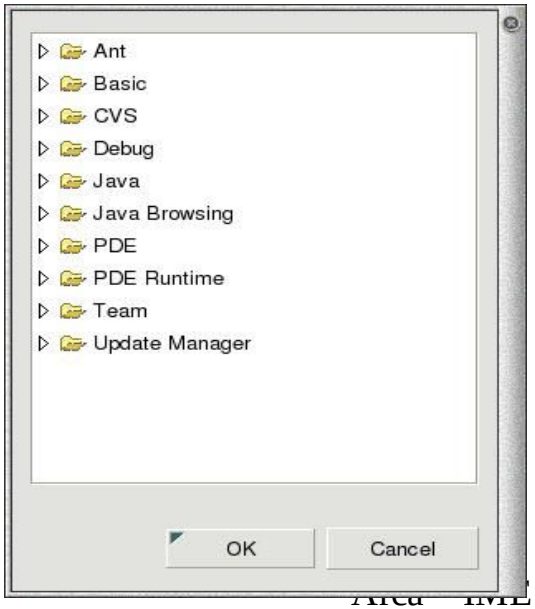
- A bancada contém menus, barras de ferramentas, editores e visualizadores
- A bancada pode ter várias perspectivas, cada perspectiva organiza diferentes editores e visualizadores e contribui com menus e barras de ferramentas, determinando a aparência da bancada
- Visualizadores oferecem diversas informações sobre os recursos de um projeto. Podem apresentar somente partes ou atributos internos de um recurso ou podem ser ligadas a um editor.
- Perspectivas diferentes para desenvolvedores diferentes
- A disposição dos elementos da bancada pode ser totalmente reorganizada, e é armazenada em um diretório do usuário



Perspectivas, Visualizadores

- Window->Open Perspective->Other
  - Java, Debug, CVS
- Window->Show View->Other
  - Podem ser empilhados
  - Têm menus contextuais
  - Atualizações são salvas automaticamente
  - Podem ser reorganizadas





Editores e o Workspace

- Editam ou visualizam um recurso
- Contribuem para o menu da bancada
- Recursos são armazenados no workspace
  - Diretório comum
  - Armazena meta-dados

Notificação de mudanças

- Mudanças em um recurso feitas em um visualizador ou editor da bancada são notificadas para todas aplicações registradas no Eclipse
- Mudanças externas não são detectadas
- Novos recursos não são detectados automaticamente
  - Selecione a pasta na qual você criou o novo recurso em um visualizador (ex. Navigator)
  - Escolha 'File->Refresh'

Customizando a Bancada

- Várias opções podem ser personalizadas para aumentar a produtividade individual
- Clique duplo na barra de título de qualquer componente maximizá-o
- Visualizadores tem o recurso ¨Fast View¨
- Clique com o botão esquerdo na barra de título de um componente para movê-lo
- Ao mover um componente repare no cursor do mouse
  - Se o cursor se transforma em uma seta você pode encaixar o componente nesta posição da bancada
  - Ao colocar o cursor na barra de título de outro componente, ele se transforma numa pastinha e você pode então empilhar componentes

A perspectiva Java

Lista de visualizadores e editores

- Package Exlporer -> lista hierárquica de pacotes
- Hierarchy -> hierarquia de classes ou interfaces
- Outline -> resumo de importações, métodos e váriaveis do recurso sendo editado
- Editor -> ligado ao Outline, navegue pelo código e edite-o
- Tasks -> listas de tarefas (comentários com TODO)
- Console -> saída do console (stdout, stderr)

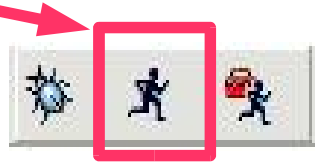
O editor Java

- Sintaxe colorida
- Ajuda flutuante para métodos com javadoc
- Assistente de código/conteúdo (CTRL-espaço)
- Formatação de código
- Assistente de importações
- Esconde código para ajudar na visualização
- Depuração integrada (erros de compilação são marcados e entram na lista de tarefas)
- Sugestões para consertar erros rapidamente
- Atalho para linhas com problemas

Meu primeiro projeto Java

Exercicio 1: Olá mundo!

- Abra a perspectiva Java
- Crie um novo projeto*
- Crie um novo pacote
- Crie uma nova classe
- Rode o programa
- O Eclipse cuida de tudo para você
  - Classpath
  - Compilação
  - Ele até escreve o código...



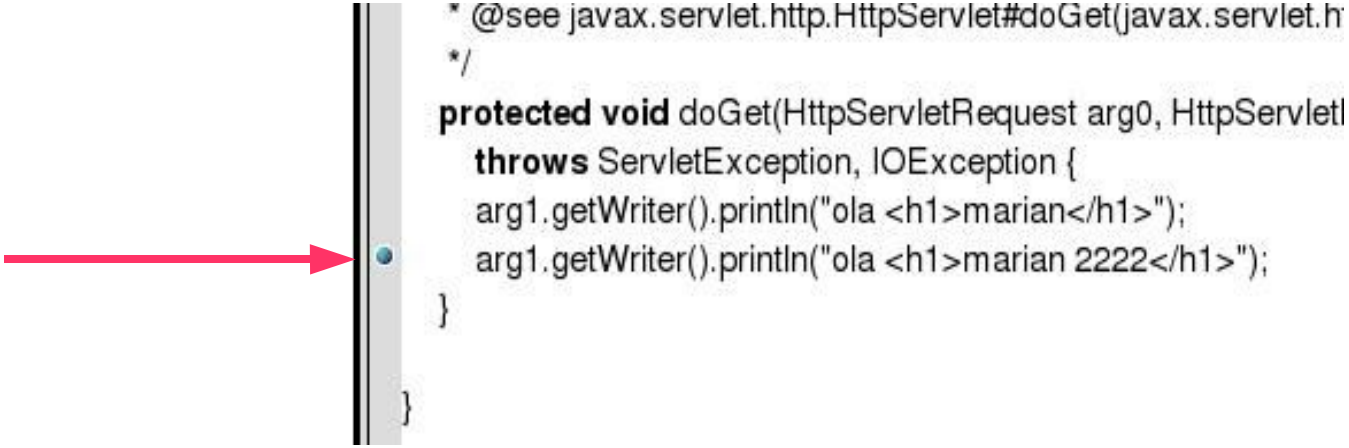
* com os fontes em um diretório separado

Depurando programas

- Depure programas locais ou remotos e também *multi-threaded*
- Depurador visual de código roda automaticamente ao atingir um breakpoint ou ao ser lançada uma exceção
- Suspenda a execução, inspecione e modifique variáveis e caminhe pelo código
- Não é necessário recompilar o código para depurar
- Esqueça de System.outs e editar arquivos para depurar seu código
- Breakpoints no editor ou depurador
- Mude o valor de variáveis enquanto caminha pelo código
- Mude o próprio código durante a depuração!

Breakpoints

- *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, eles podem ser desabilitados temporariamente
- Para adcionar um breakpoint de um clique duplo em qualquer linha do editor

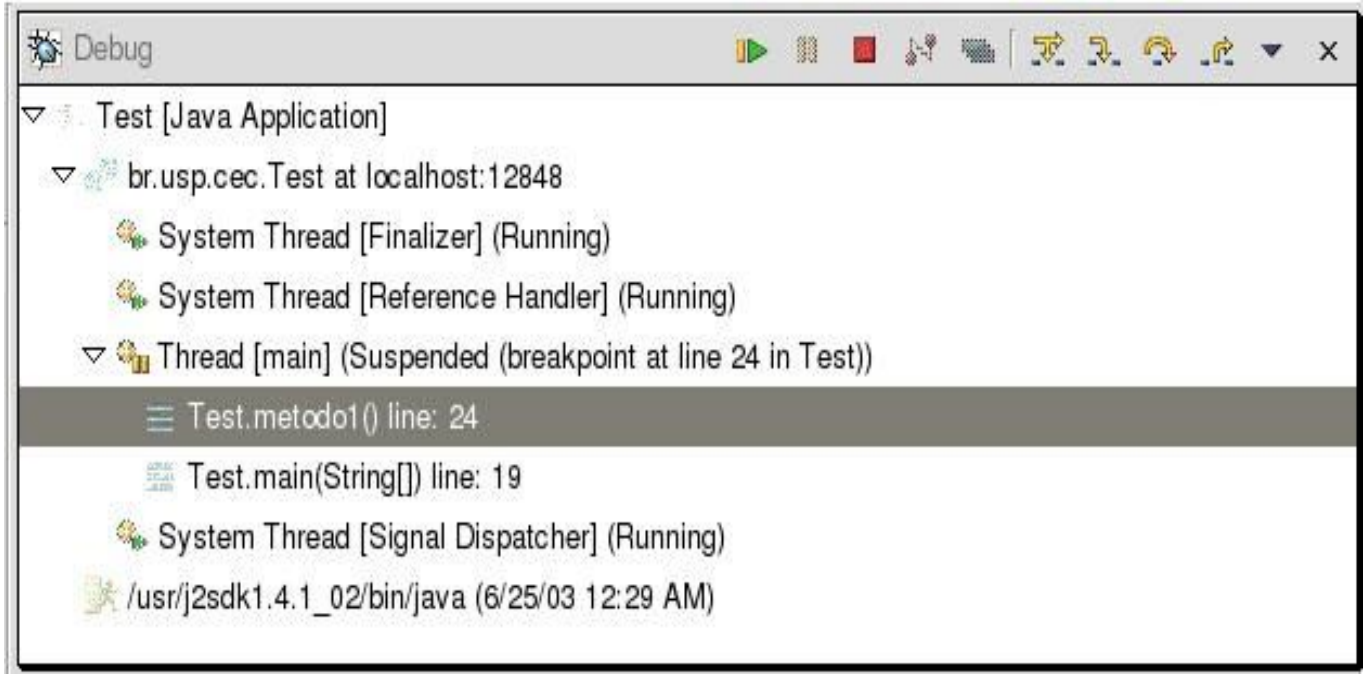


Mais breakpoints...

- Temos alguns breakpoints especiais, para habilitá-los basta acessar o menu de propriedades do breakpoint
- Hit-Count breakpoints
  - A execução do thread é suspendida somente após a n-ésima passada pelo breakpoint
  - O breakpoint fica desativado até ser reativado ou mudar seu hit-count
- Breakpoints condicionais
  - Só param a execução se a condição for verdadeira
  - Qualquer expressão booleana válida no contexto pode ser avaliada

Pilha de execução

- 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)



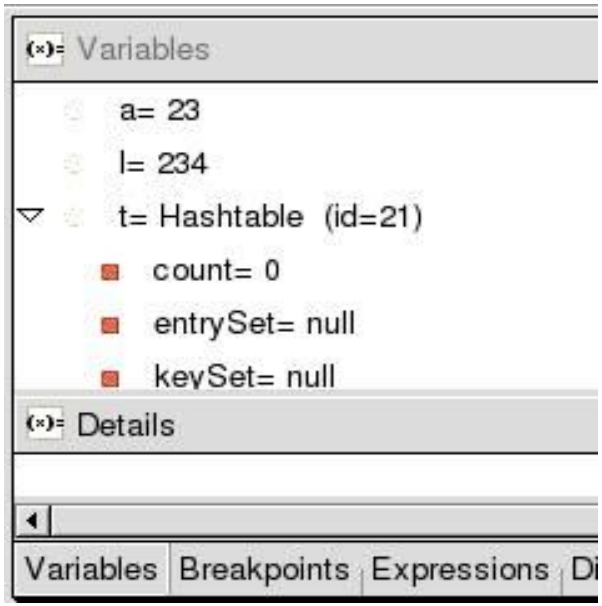
Manipulando a pilha

Brincando de hacker madrugada adentro...

- Você pode escolher um elemento qualquer da pilha para reiniciar uma aplicação (menu contextual->Relaunch)
  - A aplicação recomeça em um novo thread
  - A aplicação original continua existindo, suspensa no breakpoint
- Menu contextual->Drop Frame permite que você volte na pilha para o método em questão
  - Volte no passado e continue como se nada tivesse acontecido...
  - Mas cuidado, dados globais (ex. Variáveis de classe) continuam com seu valor intacto

A perspectiva de depuração

- Visualizador de breakpoints
  - Lista breakpoints dos projetos da bancada
  - breakpoints de exceções java (podem parar a execução se a exeção for pega, jogada, ou ambos)
- Visualizador de variáveis
  - Ligado ao contexto da pilha
  - Mostra todas as variáveis e objetos
  - Permite alterar o valor das variáveis

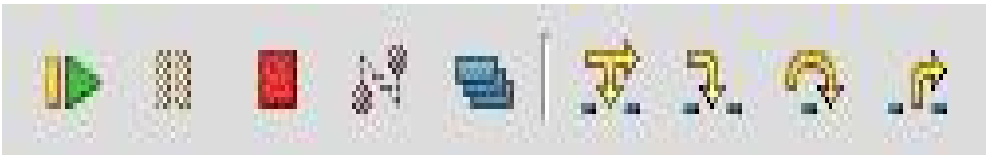


Scrap pages e mais

- Um interpretador permite com que você teste pedaços de código
- Crie uma nova Scrapbook Page ou depure um programa
- Você pode usar os seus objetos e inspecioná-los, basta selecionar o texto do código ou nome da variável no editor
- Menu contextual -> Display : executa o código e imprime o resultado no visualizador de Display ou na própria scrapbook page (você pode chamar métodos no contexto)
- Menu contextual-> Inspect : abre o visualizador de expressões
  - Similar ao visualizador de variáveis

Walk on the park

Exercicio 2: Depurar um algoritmo recursivo!



Continua

Pausa

Termina

*

**

Depurar

Entra no método

Executa o método

Retorna

* depuração remota

** limpa visualizador

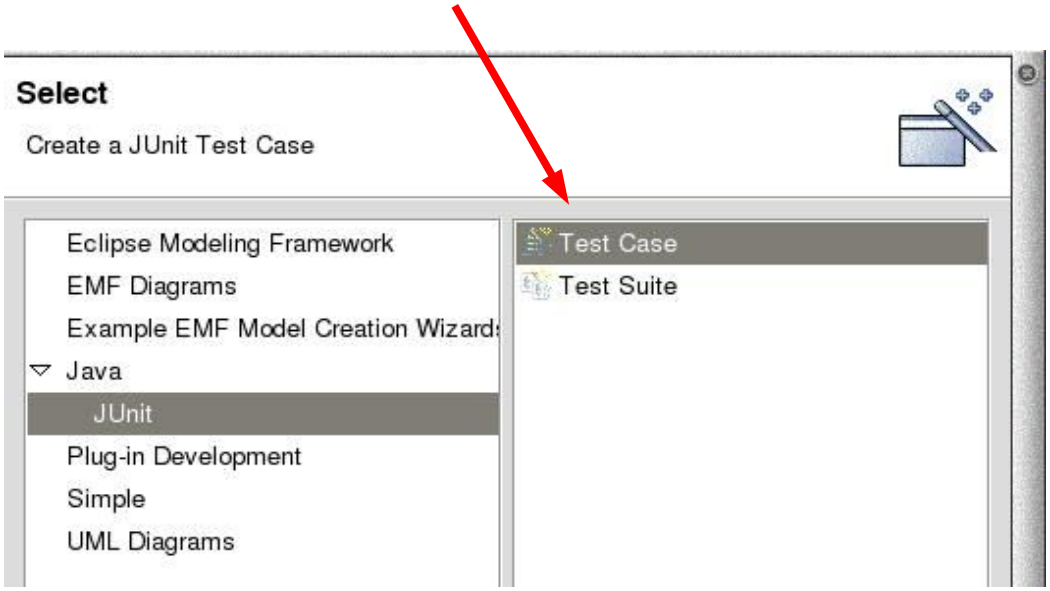
Programação eXtrema

Eclipse e XP, tudo a ver!

- XP é simplicidade, comunicação, feedback, e coragem!
- Teste primeiro, teste sempre.
- Refatoração
  - Código simples e legível, o código é a documentação
  - Elimina duplicação de código
- No Eclipse
  - Testes com Junit
  - Ferramentas de refatoração inteligentes

Meu primeiro teste

- Vamos escrever um teste para um Farol
- Crie um novo projeto, adicione um caso de testes
- Nomeie a classe FarolTest
- Como testar o farol:




```
public class FarolTest
       extends TestCase {

 public void testNovoFarol() {
  Farol f = new Farol();
  assertTrue(f.getVermelho());
  assertFalse(f.getAmarelo());
  assertFalse(f.getVerde());
 }

}
```

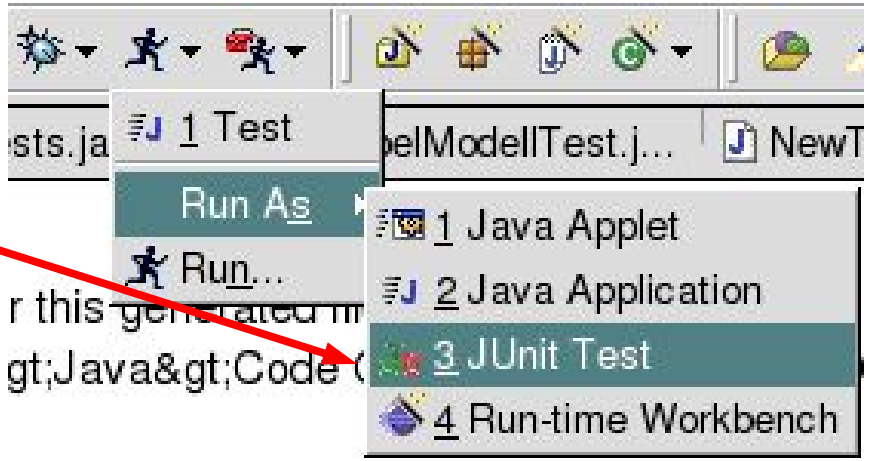
Escrevendo código...

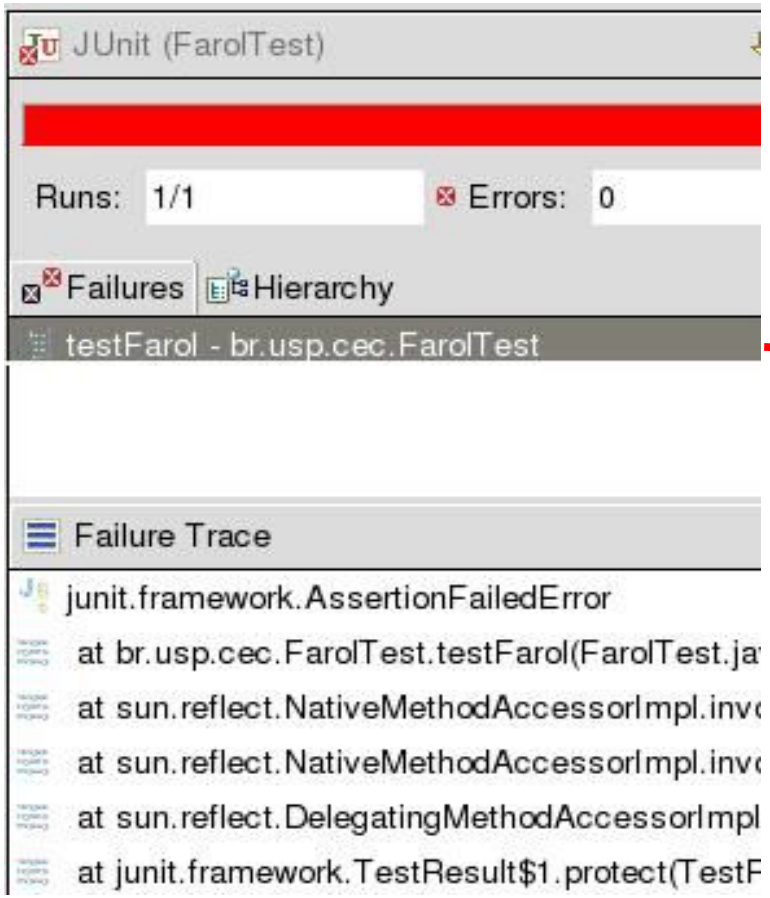
- O Eclipse nos ajuda a escrever o código para passar no teste
- Selecione 'Farol' no editor e clique em Edit->Quick Fix
- Escolha a opção de criar a classe Farol
- Implemente os métodos necessários para começar a testar a classe:

```
public boolean getVermelho() {
 return false;
}
public boolean getAmarelo() {
 return false;
}
public boolean getVerde() {
 return false;
}
```

Rodando o teste

- Rode o teste
- O visualizador JUnit aparece





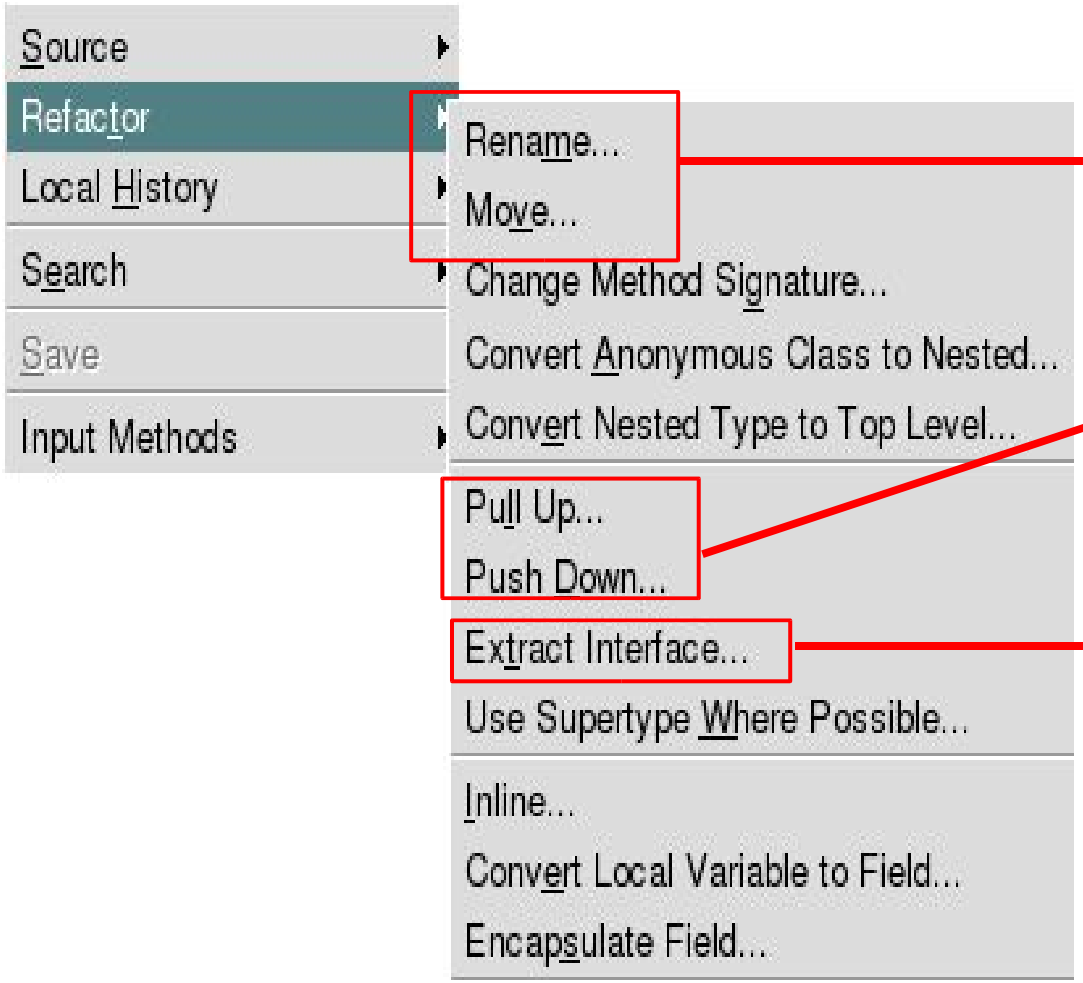
Clique duplo nos leva para a asserção que falhou

Melhorando o teste

- Mude o código para a menssagem de erro ser mais explicativa, em vez de assertTrue(), use assertEquals()
- Rode o teste novamente
- Clique duplo para achar a causa do erro
- Posicione o cursor do mouse sobre o método getVermelho() e pressione F3 para ir direto para a implementação e corrigi-la
- Rode o teste novamente
- Sucesso!
- Você pode rodar múltiplos testes usando TestSuites, visualize-os usando a hierarquia do visualizador JUnit

Refatorando código

- No Editor e no Package Explorer você tem acesso ao menu de refatoração



Renomeie e mova classes, variáveis, métodos...

Mova métodos na hierarquia de classes...

Crie uma interface baseada na classe atual...

- Olhe também o menu Source

Controle de versões

RMS says: We have to share with our neighboors!

- Com CVS podemos controlar as diferentes versões da nossa aplicação e as revisões dos arquivos fonte
- Se algum erro ocorrer sempre podemos voltar a uma versão antiga.
- Podemos trabalhar em grupo em um mesmo projeto, todos são notificados das mudanças ocorridas, pode-se fazer uma mescla de mudanças concorrentes
- Existe a possibilidade de controlar diferentes árvores de um mesmo projeto (uma ramificação só para “patches” de uma versão velha por exemplo)
- O Eclipse vem com um cliente de CVS embutido

Importando projetos

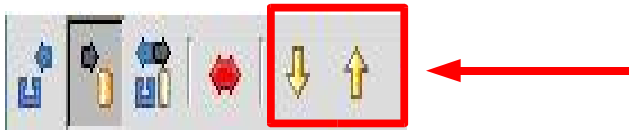
- Abra a perspectiva do CVS
- Ache o visualizador CVS Repositories
- Menu contextual -> New ->Repository location
- Conecte-se ao servidor de CVS da Arca:

Host:	arca.ime.usp.br
Repository path:	/var/lib/cvs
User:	anonymous

- Ache a versão atual do Panda e de Checkout
- Dê uma olhada no projeto
- Menu contextual -> Team

Trabalhando com mudanças

- Edite um arquivo qualquer (dica: faça algun TODO da lista de tarefas*)
- Menu contextual -> Team -> Synchronize with repository
- Menu contextual -> Show content comparison



Navegue pelas mudanças

Veja somente os conflitos

Veja mudanças no diretório local

Veja mudanças no repositório remoto

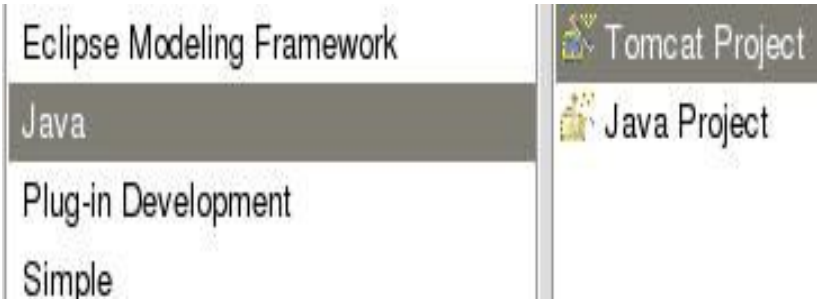
* para salvar suas mudanças no repositório remoto você tem que dar commit.

Ant, facilitando nossas vidas

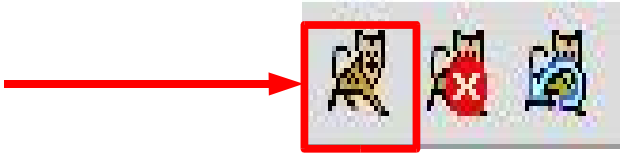
- Ant é uma ferramenta que automatiza tarefas comuns no desenvolvimento de uma aplicação
- Parecido com Make
- Usa um arquivo XML (build.xml) que define quais tarefas são automatizadas em um projeto
- Provê um ambiente comum para todos desenvolvedores independente de plataformas
- Usado para compilar, criar estruturas de diretórios, gerar arquivos, gerar javadoc e colocar no site, implantar uma aplicação J2EE, automatizar a execução de testes, etc...
- Como exemplo vamos olhar o build.xml do Panda

Tomcat, aplicações na web

- O Tomcat é um servlet container que permite a implantação de aplicações J2EE na Web
- Crie um novo Tomcat Project (plugin Sysdeo)
- Crie um HttpServlet
- Implemente o método doGet()



- Registre o projeto no servidor (menu contextual -> Tomcat Project-> update context in server.xml)
- Rode o tomcat



- Acesse a página, experimente brincar com depuração remota e JSPs

Estendendo o Eclipse

Gotta catch'em all!

- Existem muitos plugins disponíveis na comunidade
- http://eclipse-plugins.2y.net/eclipse/
- Instalar um plugin é fácil, basta colocá-lo no diretório de plugins na pasta de instalação do eclipse
- Alguns plugins interessantes
  - XMLBuddy: editor de XML
  - OMONDO: diagramas UML no seu projeto
  - DbEdit e QuantumDB: conecte-se a um banco de dados
  - Sangam: programação pareada à distância!

Muito obrigado!

gsd.ime.usp.br
arca.ime.usp.br

alex@arca.ime.usp.br