

IDE - Eclipse Breve Tutorial

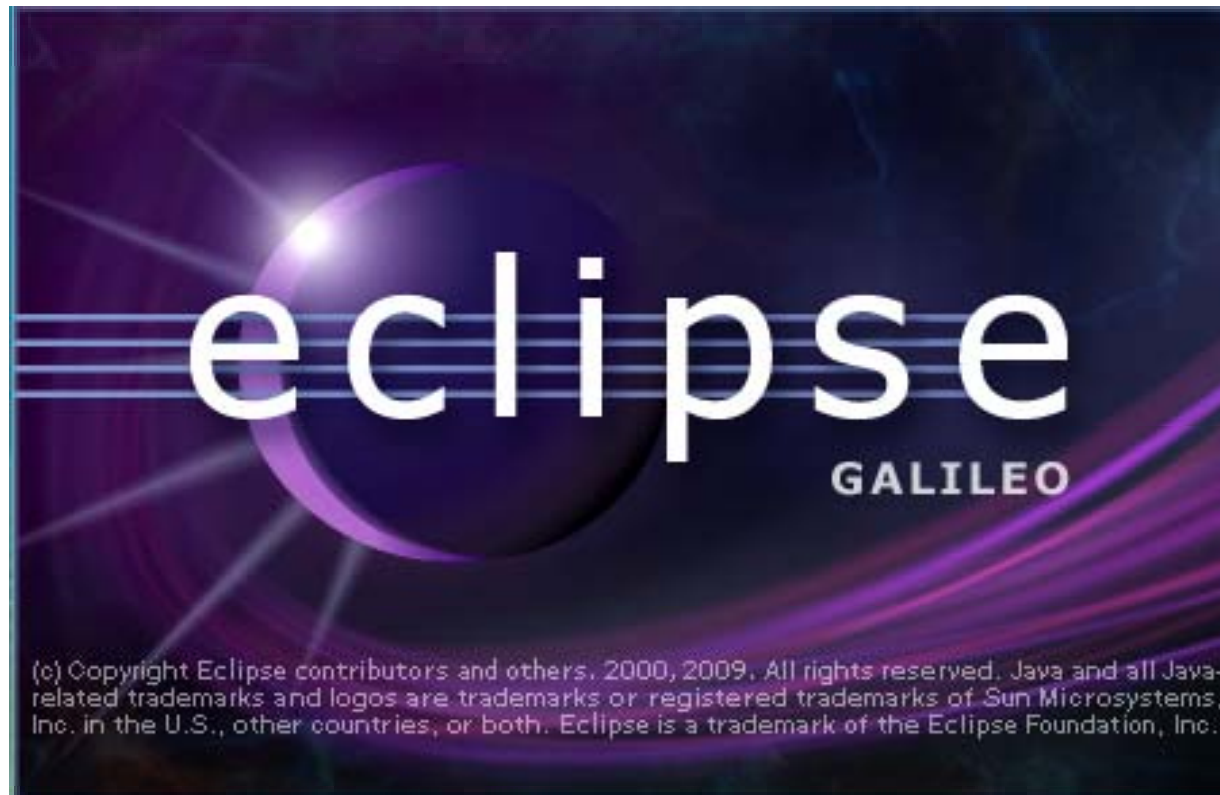
Michel Leles

SUMÁRIO

▶ IDE – Eclipse.....	3
▶ Onde salvar projetos e programas.....	4
▶ Eclipse - Tela de Boas Vindas.....	6
▶ Eclipse pronto para iniciar um projeto.....	11
▶ Criar um projeto simples no Eclipse.....	21
▶ Criação de um programa simples.....	25
▶ Imprimindo mensagens na tela.....	31
▶ Executar um programa no Eclipse.....	36
▶ Como gerar um executável.....	39
▶ Executar um programa fora do Eclipse.....	48
▶ Para “debugar” um programa.....	56

IDE - Eclipse

- ▶ **Ambiente de Desenvolvimento Integrado**
 - ▶ IDE – *Integrated Development Enviroment*



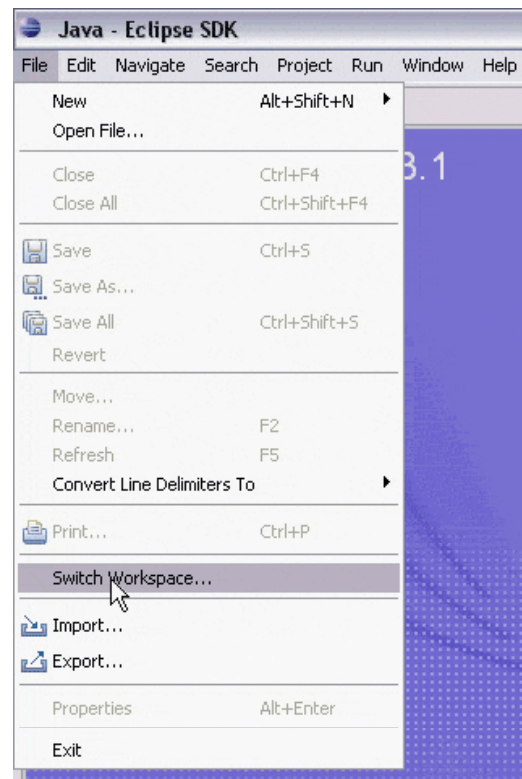
Onde salvar projetos e programas

- ▶ No momento da inicialização, o Eclipse faz o questionamento sobre onde salvar seus projetos e arquivos.
- ▶ Selecione uma pasta exclusiva para maior segurança.
- ▶ Clique no botão *Browse* para selecionar a pasta destino de sua preferência.



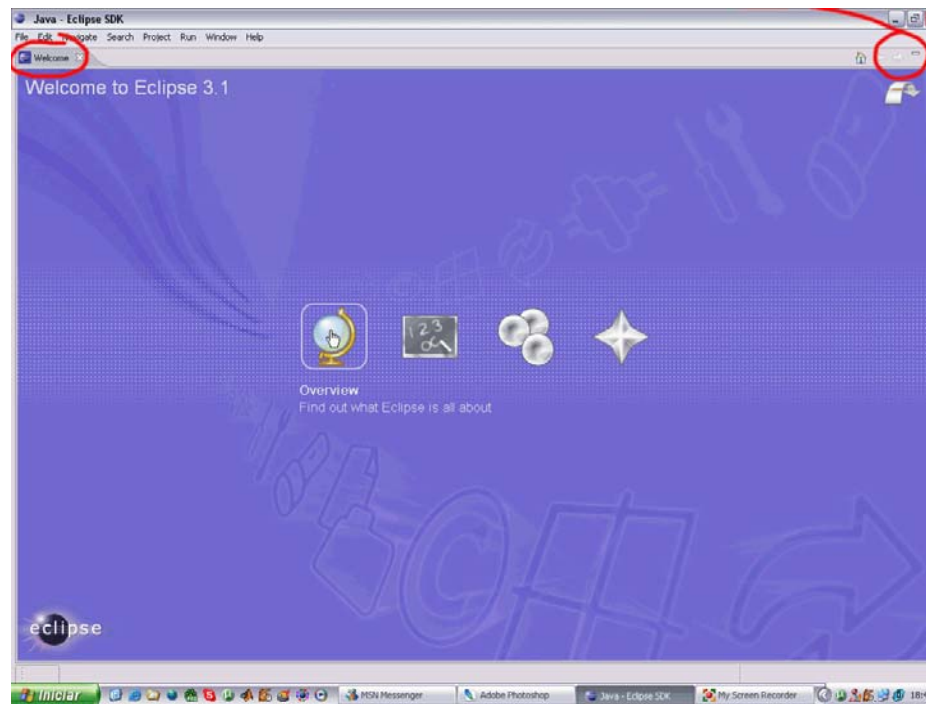
Onde salvar projetos e programas

- ▶ Se por algum motivo o Eclipse não questionar sobre onde salvar os arquivos e projetos podemos forçá-lo a isso.
- ▶ Vá até menu principal em *File >> Switch Workspace* e aparecerá a janela do PASSO 1.



Eclipse – Tela de Boas Vindas

- ▶ Tela Inicial do Eclipse (*Welcome*).
 - ▶ Pode ser ignorada minimizando-a como indicado na figura, porém ela contém atalhos para dicas e informações úteis sobre o funcionamento do Eclipse.



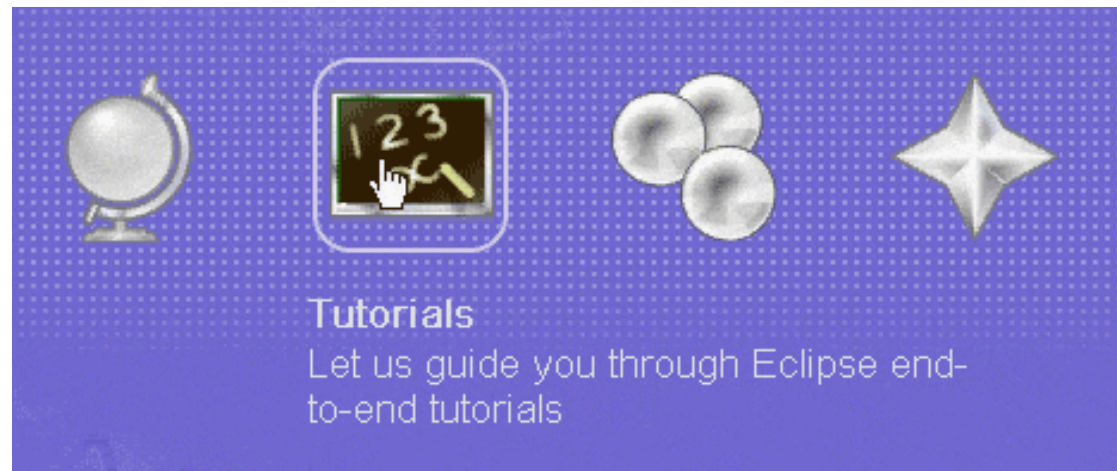
Eclipse – Tela de Boas Vindas

- ▶ **Apresentação (Overview).**
 - ▶ Contém uma apresentação das funcionalidades, vantagens, suporte e comparações de desempenho do Eclipse.



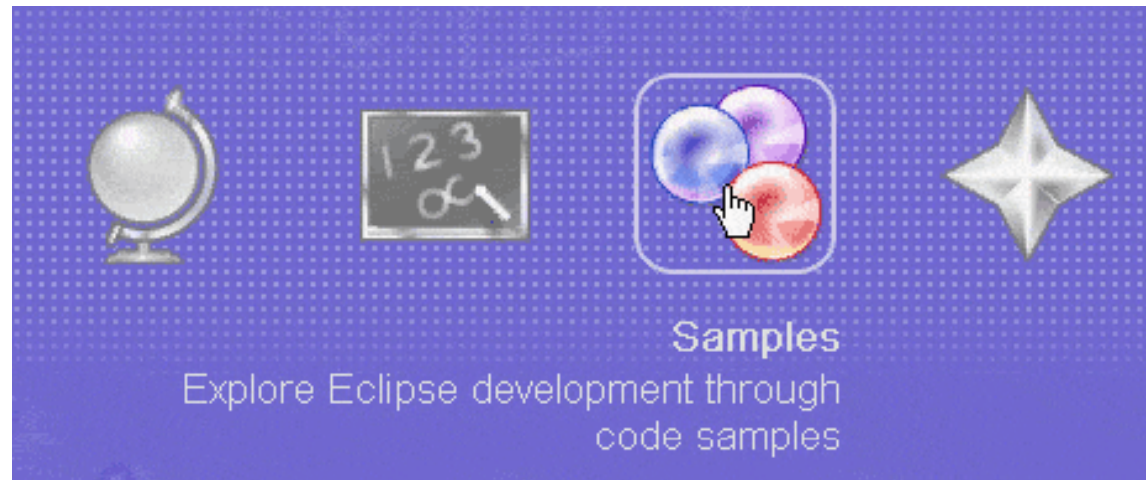
Eclipse – Tela de Boas Vindas

- ▶ Lições (*Tutorials*).
 - ▶ Contém exemplos para iniciante de aplicações que podem ser desenvolvidas no ambiente Eclipse.



Eclipse – Tela de Boas Vindas

- ▶ **Exemplos (*Samples*).**
 - ▶ Contém alguns programas exemplos do e permite baixar exemplos do site Eclipse.org.



Eclipse – Tela de Boas Vindas

- ▶ **Novidades (*What's New*).**
 - ▶ Contém um resumo de funcionalidades adicionadas na versão atual em comparação com versões anteriores.

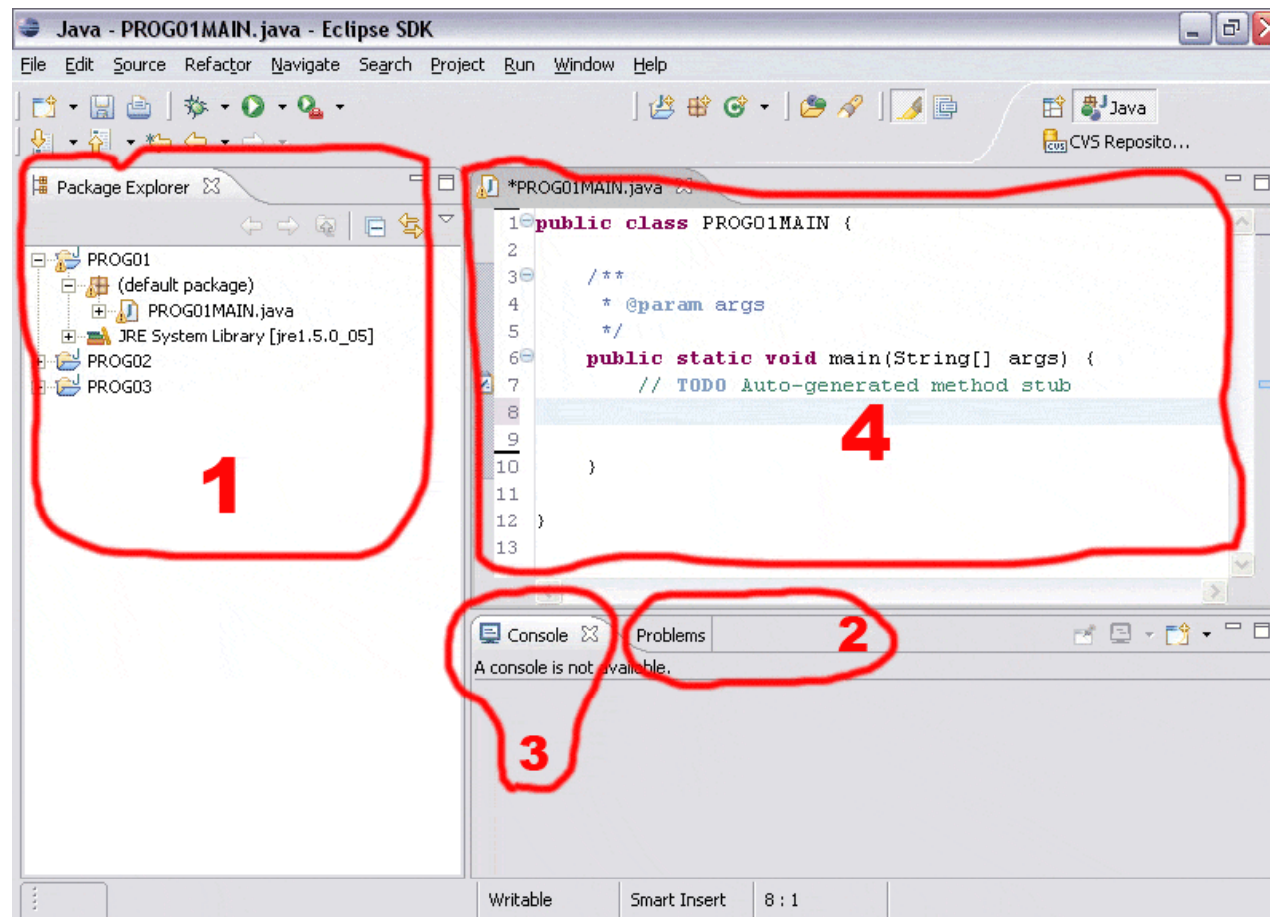


Eclipse pronto para iniciar um projeto

- ▶ Visão de programação (Perspectiva JAVA).
 - ▶ A tela inicial que deveria aparecer para conseguirmos executar mais facilmente as tarefas deve ser a exemplificada na figura abaixo.
 - ▶ São elas:
 - **Package Explorer** - janela contendo todos os projetos desenvolvidos, cada projeto representa um programa ou aplicação;
 - **Console** - janela responsável pela saída de padrão para mensagens proveniente qualquer programa Java;
 - **Problems** - janela que indica erros contidos no código ordenados pela linha em que acontece;
 - **Janela com código** - janela onde aparecerá o código fonte do arquivo .java selecionado no momento. Tal janela aparece automaticamente ou dando 2 cliques em algum arquivo .java presente na janela **Package Explorer**.

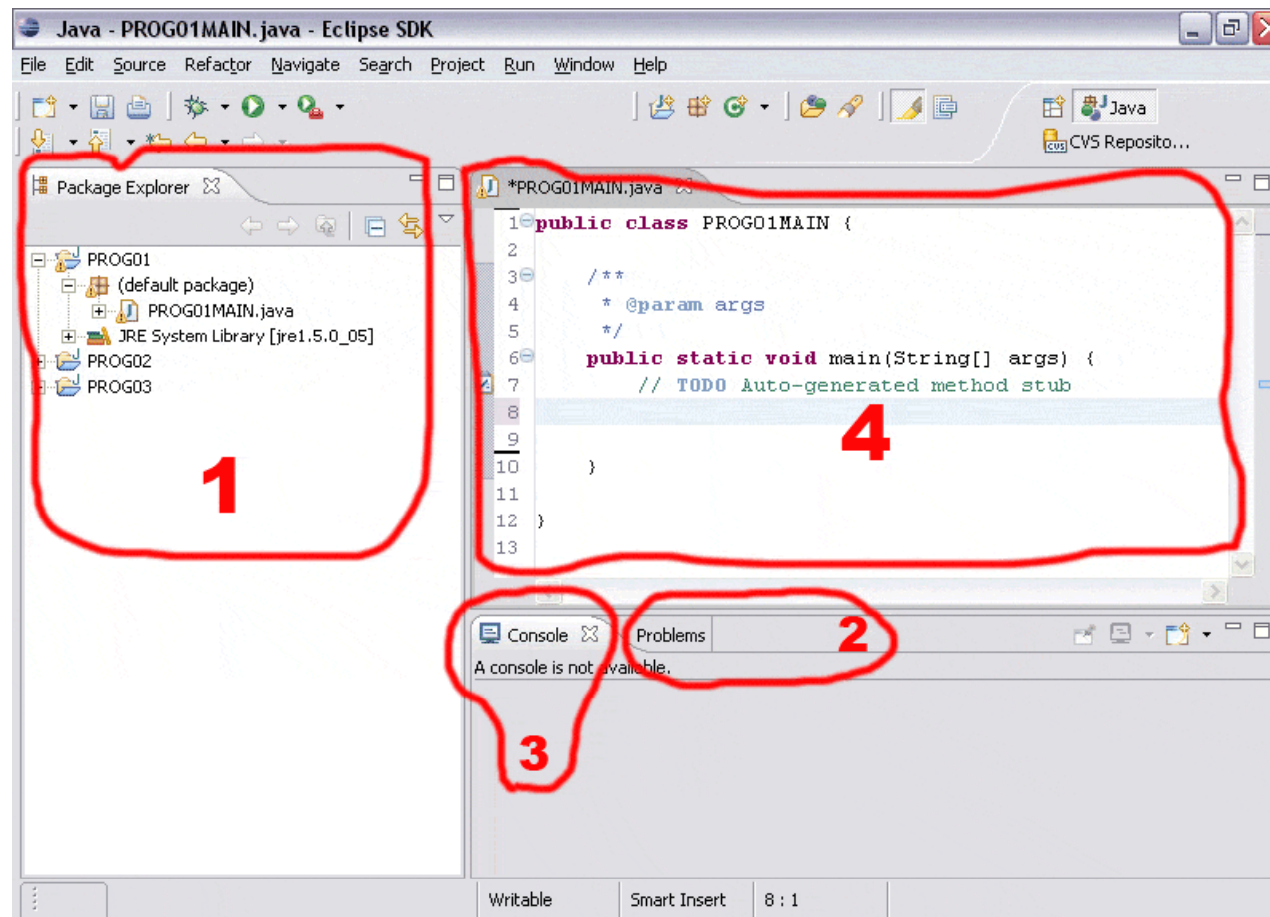
Eclipse pronto para iniciar um projeto

- ▶ Visão de programação (Perspectiva JAVA).



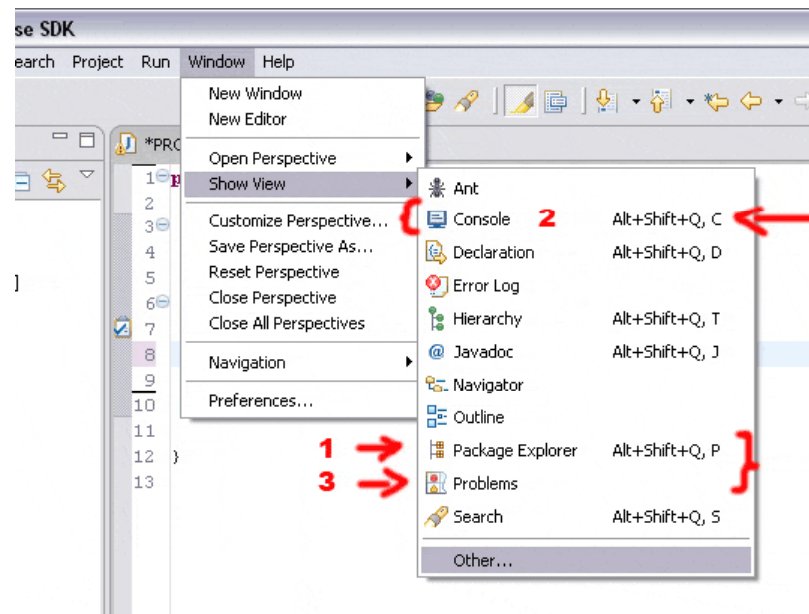
Eclipse pronto para iniciar um projeto

- ▶ Visão de programação (Perspectiva JAVA).



Eclipse pronto para iniciar um projeto

- ▶ **Habilitando a visão de programação (Perspectiva JAVA).**
 - ▶ Se por algum motivo a configuração inicial do Eclipse não conter todas as janelas necessárias devemos habilitá-las.
 - ▶ No menu *Window* >> *Show View* podemos ativar qualquer janela informativa que desejemos.

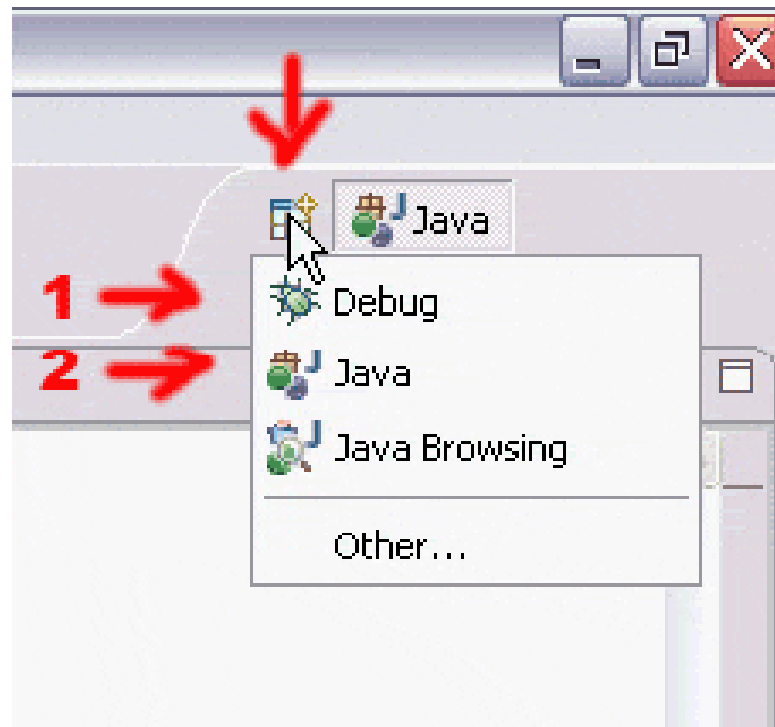


Eclipse pronto para iniciar um projeto

- ▶ **Ativando outras perspectivas (conjuntos de janelas).**
 - ▶ O conjunto de janelas disponíveis ou perspectivas podem ser diferentes dependendo da tarefa que estejamos dispostos a realizar.
 - ▶ A mudança de perspectiva pode ser feita pelo botão encontrado na parte superior direita da tela.
 - ▶ As 2 perspectivas essenciais em qualquer ambiente de programação são as seguintes:
 - ▶ **Visão de programação (Java)** - a perspectiva Java seleciona todas as janelas necessárias para se começar a construir uma aplicação, nela encontramos tudo que está relacionado à criar e executar um código;
 - ▶ **Visão de testes (Debug)** - a perspectiva de Debug é necessária quando a ação desejada é a de testar possíveis erros de uma aplicação, nessa tela conseguimos executar programas passo a passo.

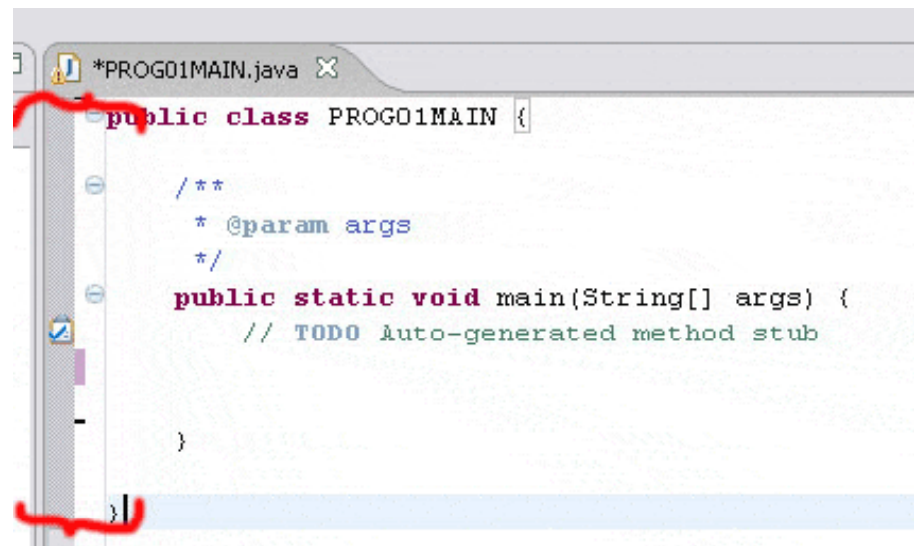
Eclipse pronto para iniciar um projeto

- ▶ Ativando outras perspectivas (conjuntos de janelas).
 - ▶ Para mudarmos de uma perspectiva para outra basta clicar no botão correspondente no alto da tela.



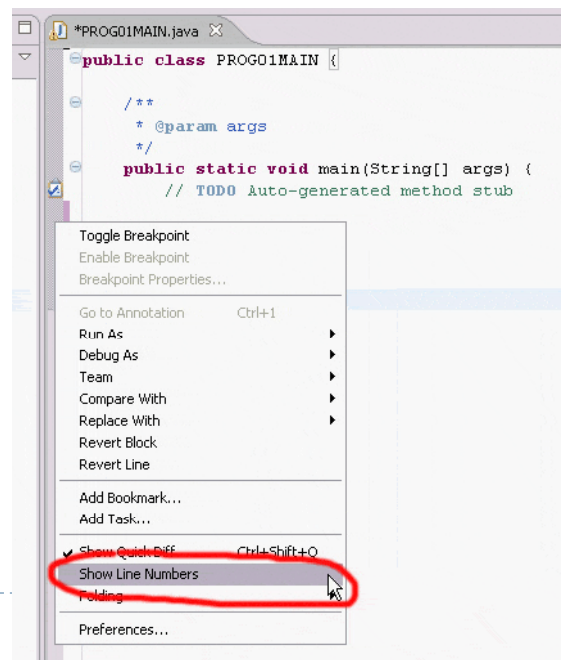
Eclipse pronto para iniciar um projeto

- ▶ Usando a ajuda da barra lateral.
 - ▶ Às vezes pode ser muito útil prestar atenção na barra posicionada à esquerda de todo código.
 - ▶ Ela pode fornecer informações como pontos de parada (*breakpoints*), avisos de variáveis não inicializadas, número das linhas, etc.



Eclipse pronto para iniciar um projeto

- ▶ Ativando número nas linhas no código.
 - ▶ Uma informação muito comum em todo bom compilador é a de numerar as linhas de código para que tarefas como testes e procuras por erros sejam simplificadas.
 - ▶ Clicando com o botão direito na barra cinza à esquerda do código podemos habilitar essa opção.

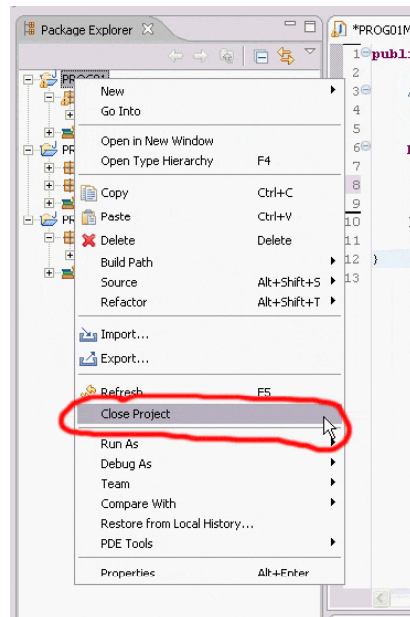


Eclipse pronto para iniciar um projeto

- ▶ **Fechando projetos não utilizados.**
 - ▶ Nas tarefas iniciais definimos onde será criada nossa Área de trabalho (workspace).
 - ▶ Muitas vezes possuímos diferentes projetos criados.
 - ▶ No caso da figura a seguir podemos notar a presença de 3 projetos (PROG01, PROG02 e PROG03).
 - ▶ Para evitar transtornos futuros e ter certeza que projetos sejam modificados por acidente é recomendado que se desabilite ou feche projetos não utilizados no momento, ou seja, apenas 1 projeto deve ficar ativo.

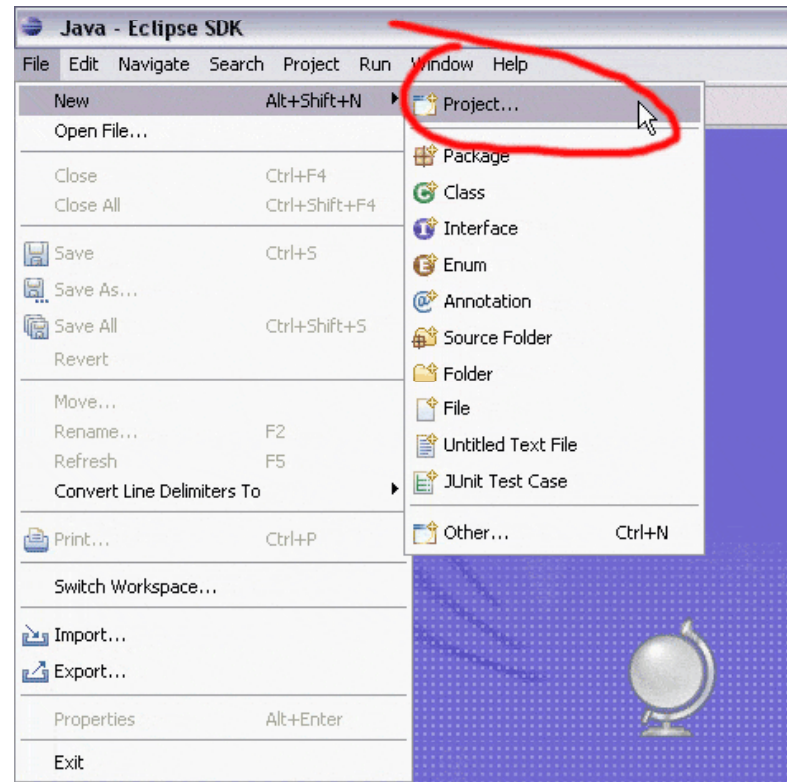
Eclipse pronto para iniciar um projeto

- ▶ Fechando projetos não utilizados.
 - ▶ Clicando com o botão direito no ícone do projeto na janela do explorador de pacotes (Package Explorer) podemos fechar o projeto.
 - ▶ Observe o ícone do projeto fechado sendo modificado.



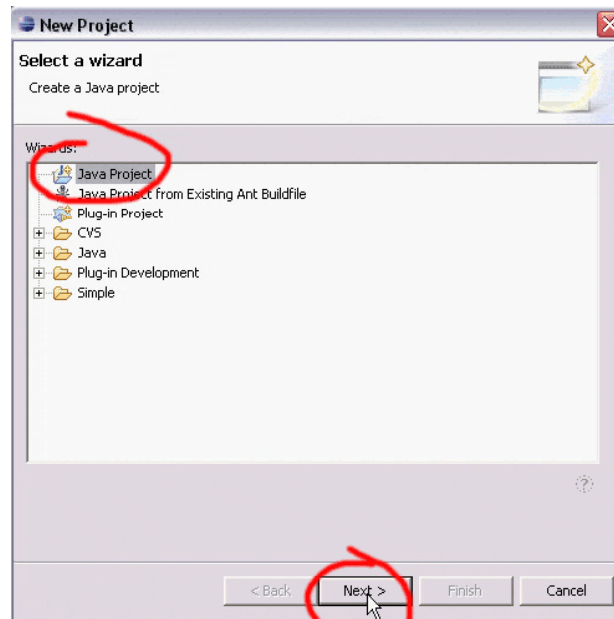
Criar um projeto simples no Eclipse

- ▶ Criando um projeto.
 - ▶ No menu Principal selecione:
 - ▶ File >> New >> Project.



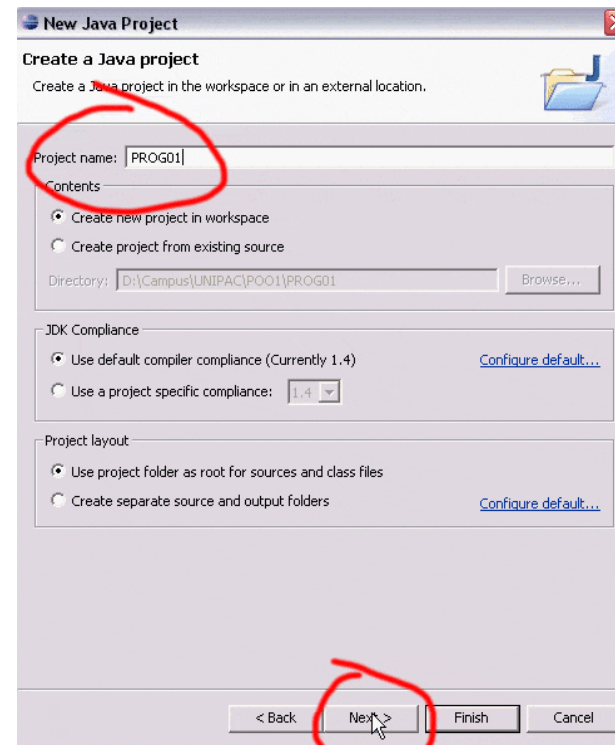
Criar um projeto simples no Eclipse

- ▶ **Selecionando o tipo do projeto.**
 - ▶ Uma caixa de diálogo do tipo "Mágico que faz tudo" (Wizard) será aberta, sua função é permitir que o programador escolha dentre os diversos tipos de aplicação que o Eclipse suporta.
 - ▶ Inicialmente iremos trabalhar apenas com programas mais básicos.



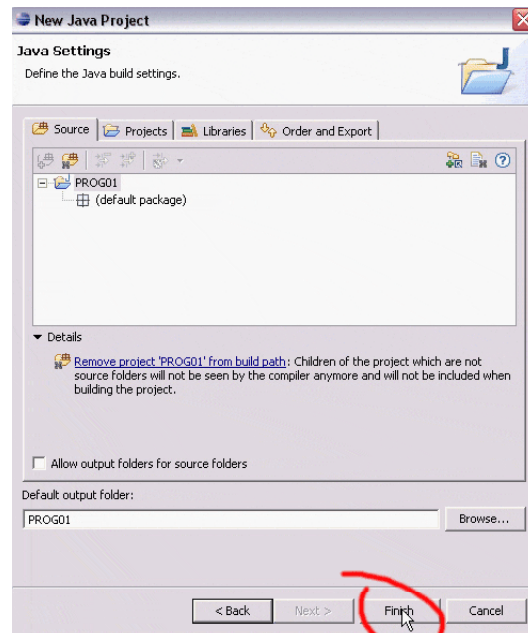
Criar um projeto simples no Eclipse

- ▶ Definindo as opções do projeto.
 - ▶ No campo *Project name* escreva o nome que desejar para seu projeto (sugestão: use nomes intuitivos e que tenham a ver com a finalidade do programa).
 - ▶ Clique em *Next* para prosseguir.



Criar um projeto simples no Eclipse

- ▶ Outras propriedades e configurações.
 - ▶ Eventualmente podemos fazer uso de bibliotecas ou classes que não sejam parte da linguagem, é nesse passo que definimos isso.
 - ▶ Por enquanto deixaremos todas as opções padrão configuradas e finalizaremos a criação clicando em *Finish*.

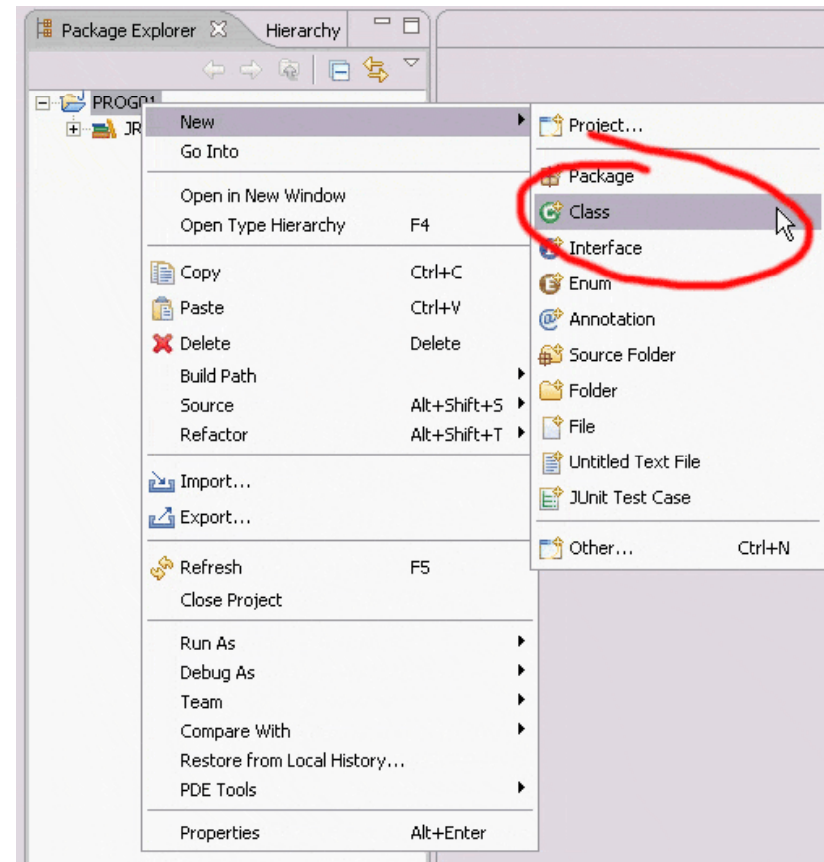


Criação de um programa simples

- ▶ **Criando o programa principal.**
 - ▶ Linguagens não orientadas a objeto definem uma função principal, que segue um formato fixo e é responsável pela execução do programa (ex: em C, C++ essa função é chamada de *main()*).
 - ▶ Em Java, tudo está relacionado a objetos, então era de se esperar que existisse um objeto que executasse a tarefa de ser a função principal do programa.

Criação de um programa simples

- ▶ Criando o programa principal.
 - ▶ Clique com o botão direito do mouse sobre o nome do seu projeto. Selecione a opção:
 - ▶ New >> Class



Criação de um programa simples

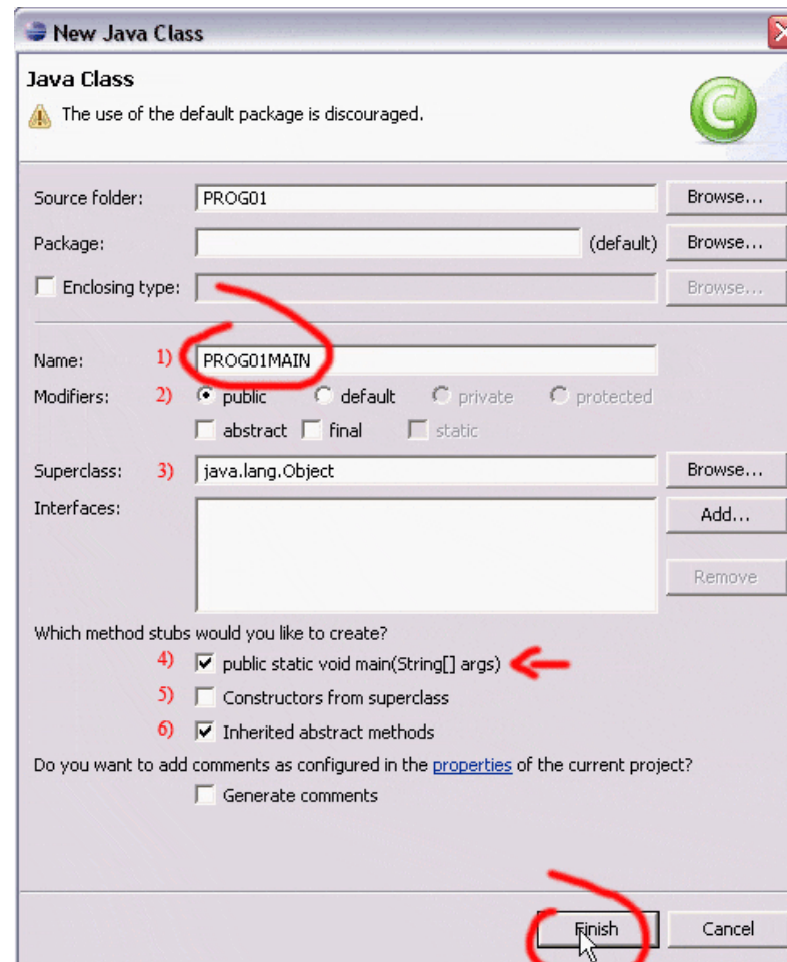
- ▶ Criando uma Classe.
- ▶ São 6 opções de interesse maior:
 - ▶ 1 - Onde definimos o nome da Classe;
 - ▶ 2 - Onde definimos os especificadores de acesso da Classe (veremos mais especificamente para que serve cada um deles no decorrer do curso);
 - ▶ 3 - Definição de qual seria o PAI do objeto que criamos (vide o conceito de herança);

Criação de um programa simples

- ▶ Criando uma Classe (cont).
 - ▶ 4 - Opção que habilita a criação automática do método main() na Classe.
 - ▶ Essa opção que define qual classe que será responsável pelo nosso programa principal (IMPORTANTE: só pode existir 1 Classe "MAIN" por projeto);
 - ▶ 5 - Opção que habilita a criação automática do método Construtor (método com mesmo nome da classe);
 - ▶ 6 - Opção que habilita a herança de métodos abstratos da classe PAI (vide conceito de polimorfismo).
 - ▶ Como desejamos um programa simples com 1 Classe apenas então será habilitada a opção do item 4 e finalizaremos clicando no botão *Finish*.

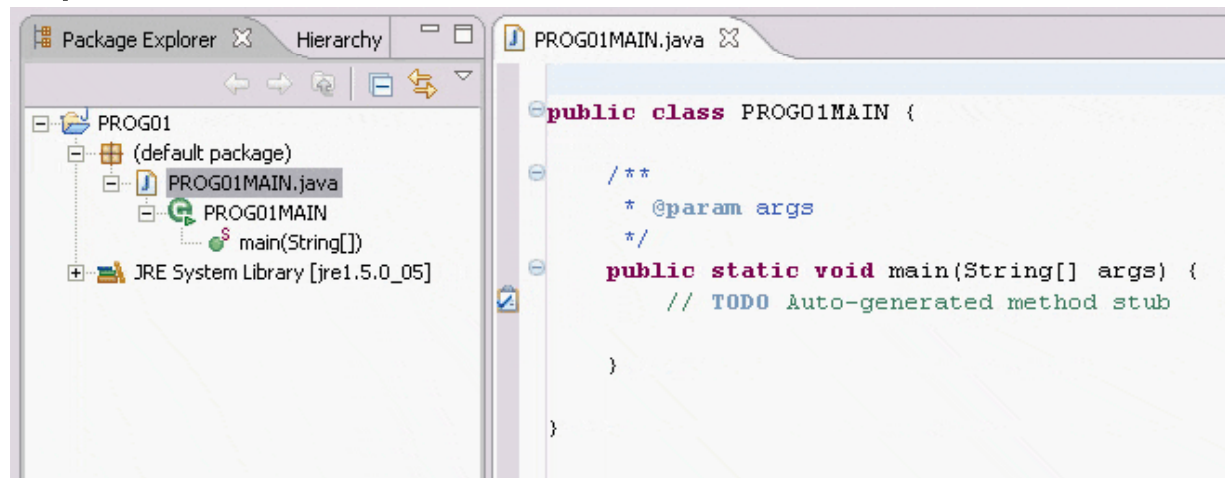
Criação de um programa simples

► Criando uma Classe (cont).



Criação de um programa simples

- ▶ Código gerado automaticamente.
 - ▶ Qual a função do "Mágico" (Wizard)?
 - ▶ É de facilitar a vida do programador inserindo código repetitivo automaticamente.
 - ▶ Pela figura podemos perceber que o código da Classe principal (PROG01MAIN) e do método principal (public static void main(String[] args)) foram gerados conforme as opções selecionadas nos passos anteriores.



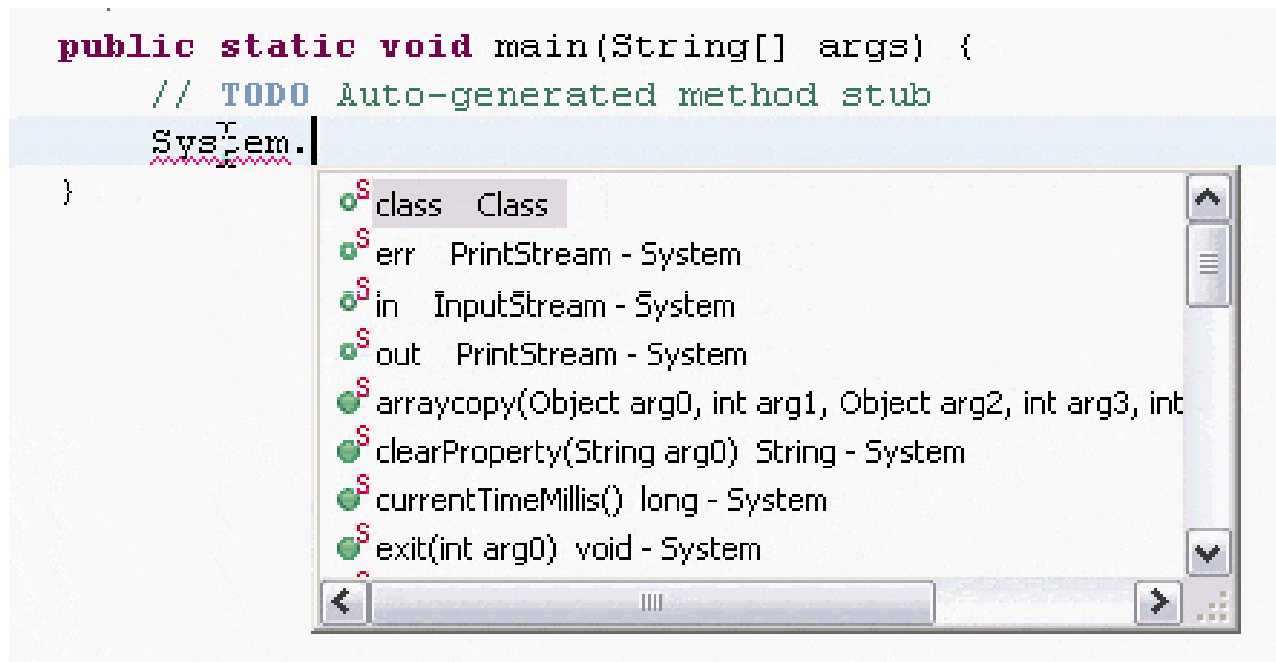
Imprimindo mensagens na tela

- ▶ Usando objetos nativos da linguagem para executar funções básicas.
 - ▶ Em Java, assim como em outras linguagens, existem funcionalidades básicas que esperamos ter para facilitar o trabalho do programador.
 - ▶ A mais comum e trivial seria a tarefa de impressão na tela. Como Java é uma linguagem totalmente orientada a objetos não é de assustar se uma Classe ou objeto fosse responsável por essa tarefa.
 - ▶ A Classe System é responsável pela tarefa de escrita de dados na tela e entrada via teclado, entre outras funcionalidades.
 - ▶ 1 - Saída de dados - System.out;
 - ▶ 2 - Entrada de dados - System.in;

Imprimindo mensagens na tela

- ▶ Digite `System` dentro do corpo do seu programa principal e utilize o operador `.` para visualizar seus métodos e atributos. Selecione com as setas do teclado o *PrintStream out* ou apenas digite *out*.

```
public static void main(String[] args) {  
    // TODO Auto-generated method stub  
    System.  
}
```

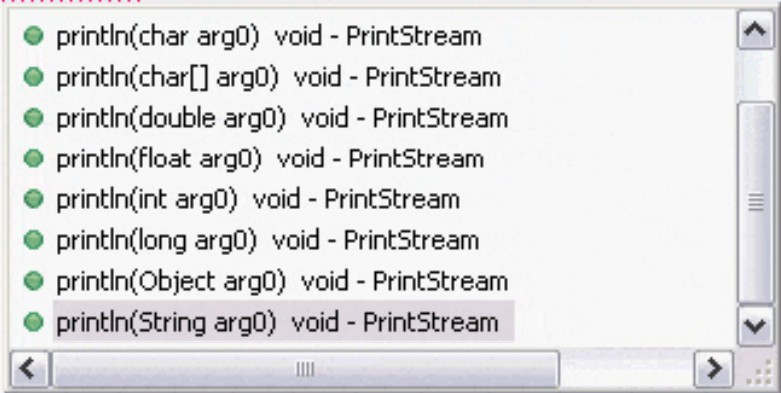
A screenshot of the Eclipse IDE showing a code editor with a Java main method. The cursor is positioned after the dot in the `System.` line. A completion list is displayed, showing various classes and methods from the `System` class. The list includes `class Class`, `err PrintStream - System`, `in InputStream - System`, `out PrintStream - System`, `arraycopy(Object arg0, int arg1, Object arg2, int arg3, int`, `clearProperty(String arg0) String - System`, `currentTimeMillis() long - System`, and `exit(int arg0) void - System`. The `out PrintStream - System` option is highlighted.

```
class Class  
err PrintStream - System  
in InputStream - System  
out PrintStream - System  
arraycopy(Object arg0, int arg1, Object arg2, int arg3, int  
clearProperty(String arg0) String - System  
currentTimeMillis() long - System  
exit(int arg0) void - System
```


Imprimindo mensagens na tela

- ▶ Escolhendo a função de imprimir adequada.
 - ▶ Para imprimir uma linha na tela que já contenha um caracter "\n" que indica final de linha podemos chamar a função println().
 - ▶ Dica:
 - O compilador tende a autocompletar o que escrevemos, se digitarmos "System.out.p" já serão ofertadas todas as opções que autocompletem esse comando.

```
*/  
public static void main(String[] args) {  
    // TODO Auto-generated method stub  
    System.out.println  
}
```



The screenshot shows an IDE window with a code editor. The code is a Java main method. The line 'System.out.println' is highlighted, and an autocomplete menu is open below it. The menu lists several overloaded println methods from the PrintStream class, each with a green circular icon to its left. The methods listed are: println(char arg0) void - PrintStream, println(char[] arg0) void - PrintStream, println(double arg0) void - PrintStream, println(float arg0) void - PrintStream, println(int arg0) void - PrintStream, println(long arg0) void - PrintStream, println(Object arg0) void - PrintStream, and println(String arg0) void - PrintStream. The last option, println(String arg0), is currently selected and highlighted in a light blue color.

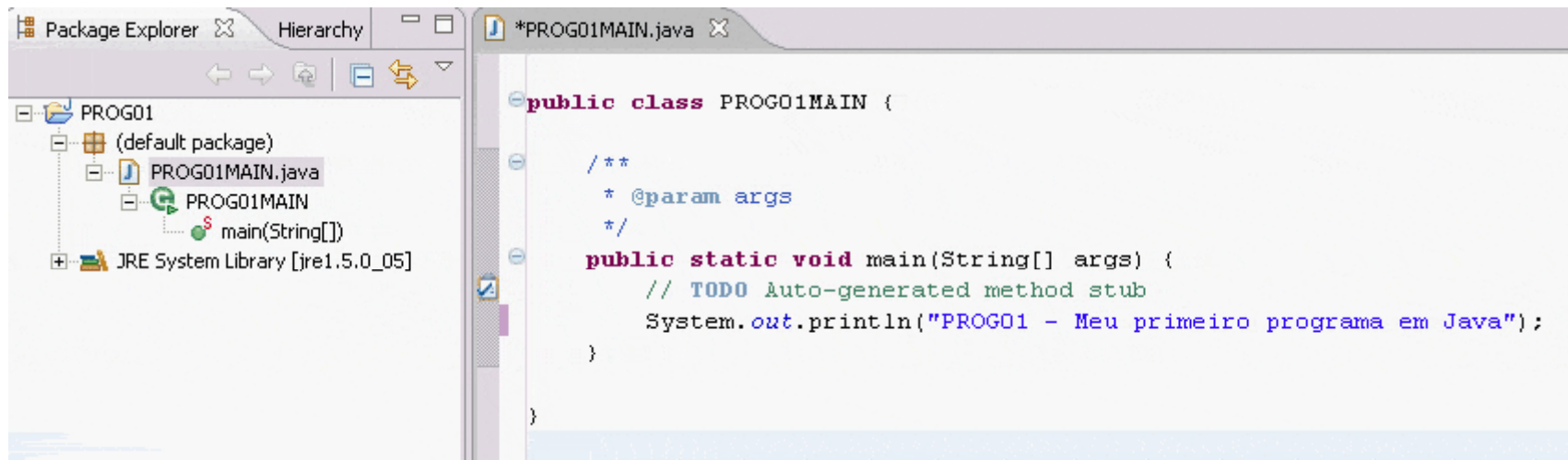
Imprimindo mensagens na tela

- ▶ Escolhendo parâmetros para o método.
 - ▶ Quando escolhemos um método de um objeto conhecido pela linguagem o compilador nos informa qual parâmetro necessário para a execução correta.
 - ▶ No presente caso o compilador espera um objeto String que seria um texto delimitado entre aspas (exemplo: "Exemplo de texto").

```
public class PROG01MAIN {  
    /**  
     * @param args  
     */  
    public static void main(String[] args) {  
        // TODO Auto-generated stub  
        System.out.println();  
    }  
}
```

Imprimindo mensagens na tela

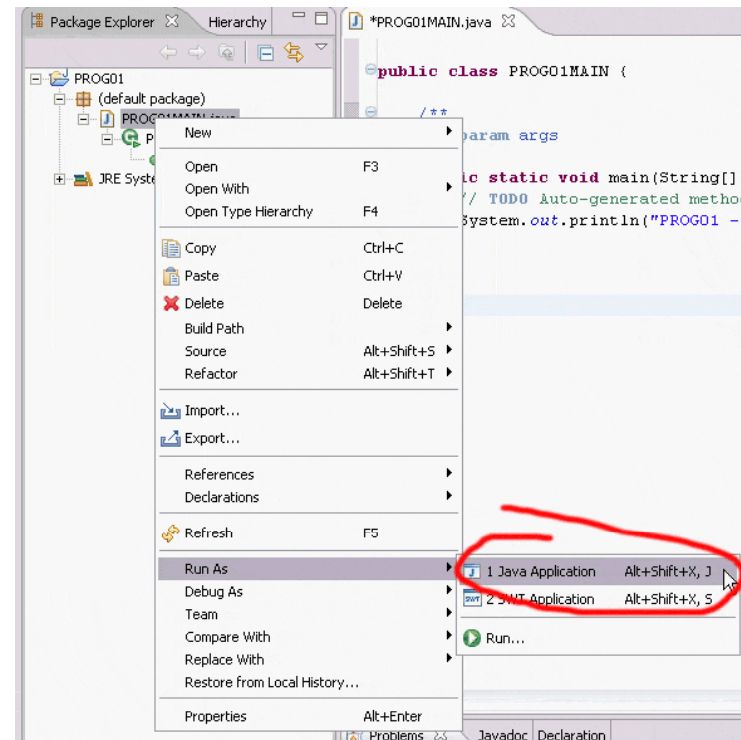
- ▶ Escolhendo o parâmetro adequado.
 - ▶ Se escolhermos o texto de maneira correta não haverá indicativo de erro do compilador, geralmente indicativos de erros podem ser vistos como palavras ou pedaços de códigos sublinhados.
 - ▶ O programa agora estaria pronto para ser executado.



```
public class PROG01MAIN {  
  
    /**  
     * @param args  
     */  
    public static void main(String[] args) {  
        // TODO Auto-generated method stub  
        System.out.println("PROG01 - Meu primeiro programa em Java");  
    }  
}
```

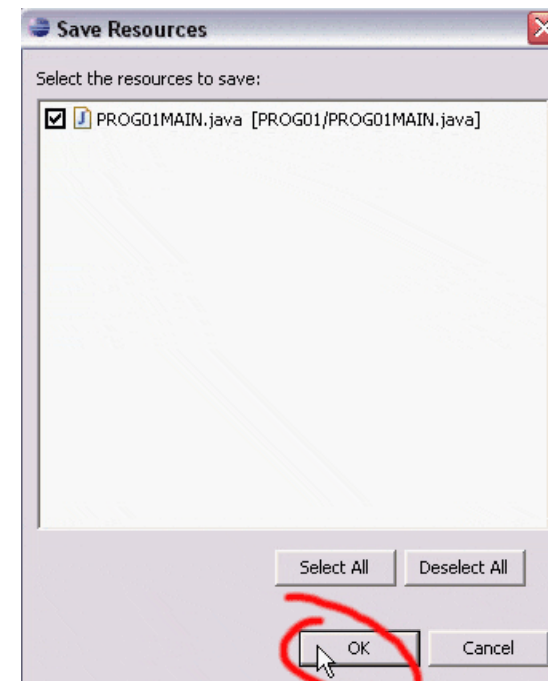
Executar um programa no Eclipse

- ▶ Executando a Classe Principal (aquela que possui o método *main()*).
- ▶ Clique com o botão direito na Classe que possui o programa principal. Selecione a opção:
 - ▶ *Run As >> java Application*



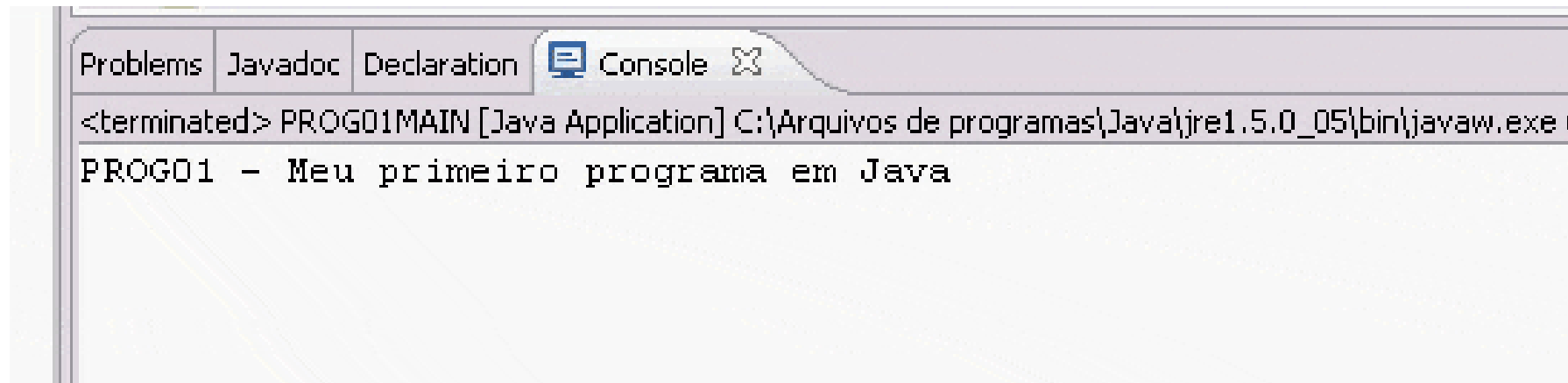
Executar um programa no Eclipse

- ▶ Salvando arquivos antes da execução.
 - ▶ Antes de executar seu código o compilador pode solicitar que salvemos a versão corrente do programa.
 - ▶ Verificamos se todas as classes do programa estão selecionadas e clicamos em OK para prosseguir.



Executar um programa no Eclipse

- ▶ Saída na janela Console.
 - ▶ Após a execução correta aparecerá uma janela de nome Console que mostrará a saída do programa com as Strings e/ou variáveis que mandamos imprimir.

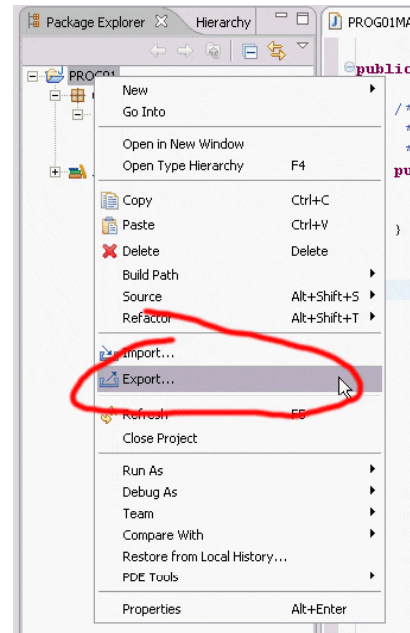


The screenshot shows the Eclipse IDE's Console window. The window title bar includes tabs for 'Problems', 'Javadoc', 'Declaration', and 'Console'. The console output displays the following text:

```
<terminated> PROG01MAIN [Java Application] C:\Arquivos de programas\Java\jre1.5.0_05\bin\javaw.exe  
PROG01 - Meu primeiro programa em Java
```

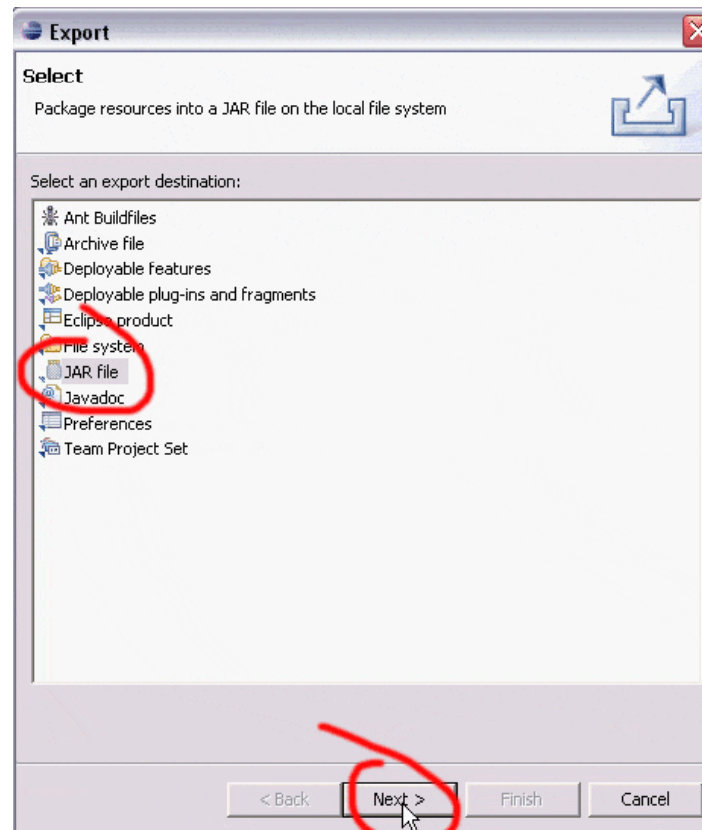
Como gerar um "executável"

- ▶ Empacotando a aplicação.
 - ▶ Em Java não geramos arquivos executáveis (com extensão .exe). Entretanto podemos organizar nossa aplicação em um arquivo único que será interpretado pela Máquina Virtual Java.
 - ▶ Clique com o botão direito sobre o nome do projeto.
 - ▶ Escolha a opção *Export* .



Como gerar um "executável"

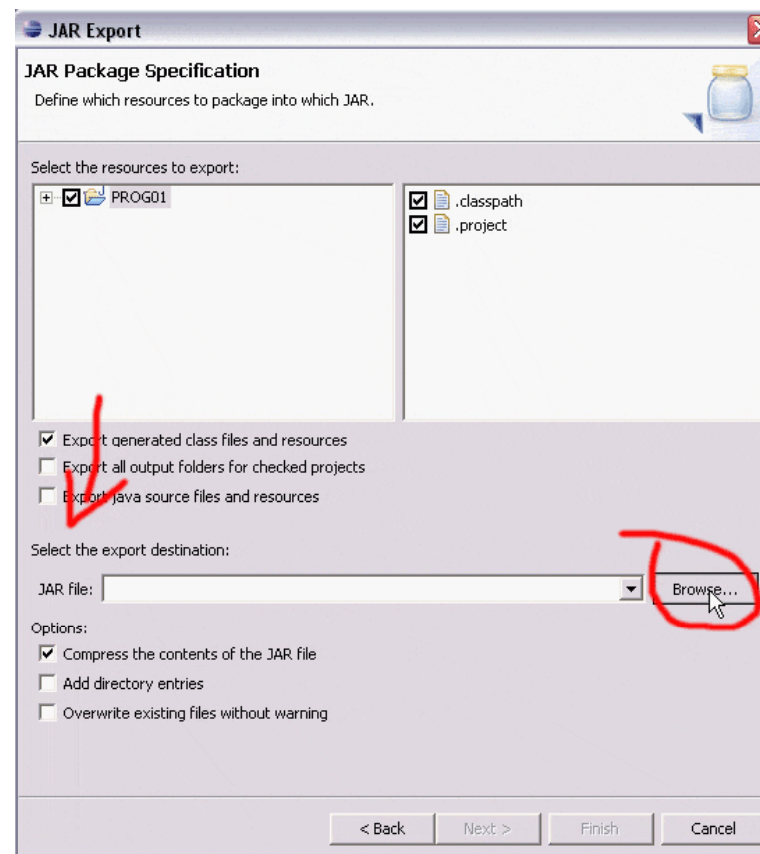
- ▶ Selecionando o tipo de arquivo a ser exportado.
 - ▶ Escolha a opção *Jar File* e prossiga clicando em *Next*.



Como gerar um "executável"

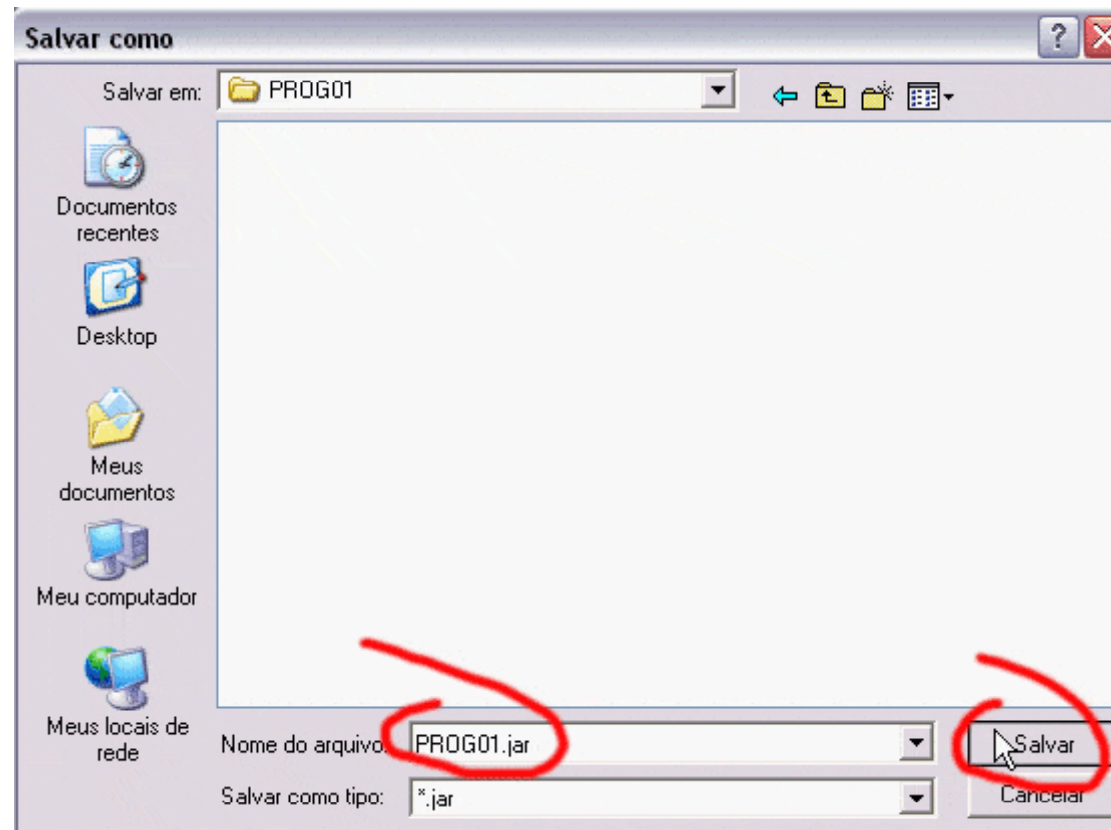
▶ Opções do arquivo JAR.

- ▶ Clique no botão *browse* e para selecionar onde o arquivo final será salvo.



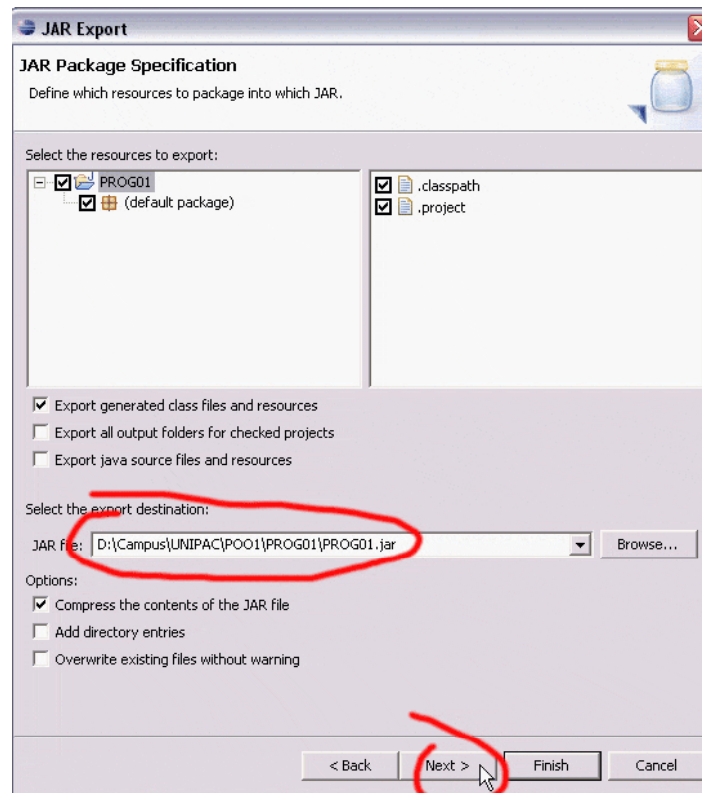
Como gerar um "executável"

- ▶ Escolhendo um nome para o arquivo.
 - ▶ Escolha um nome qualquer para o arquivo e clique em Salvar.



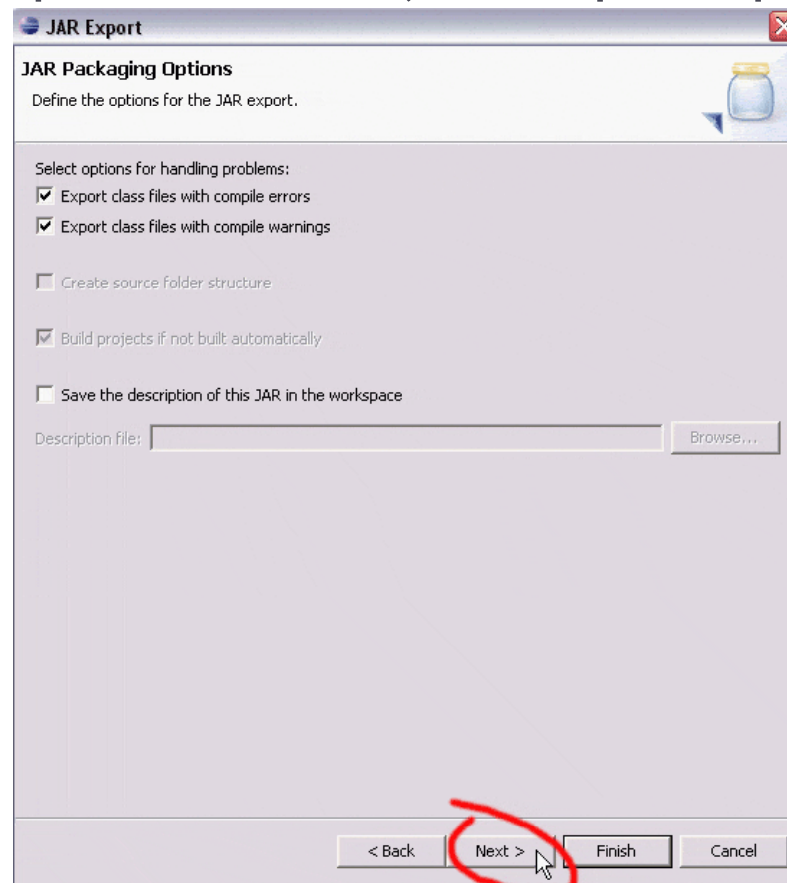
Como gerar um "executável"

- ▶ Verificando onde o arquivo JAR será salvo.
 - ▶ Verifique se o diretório escolhido está corretamente explicitado no campo *Jar File:* e clique em *Next* para prosseguir.



Como gerar um "executável"

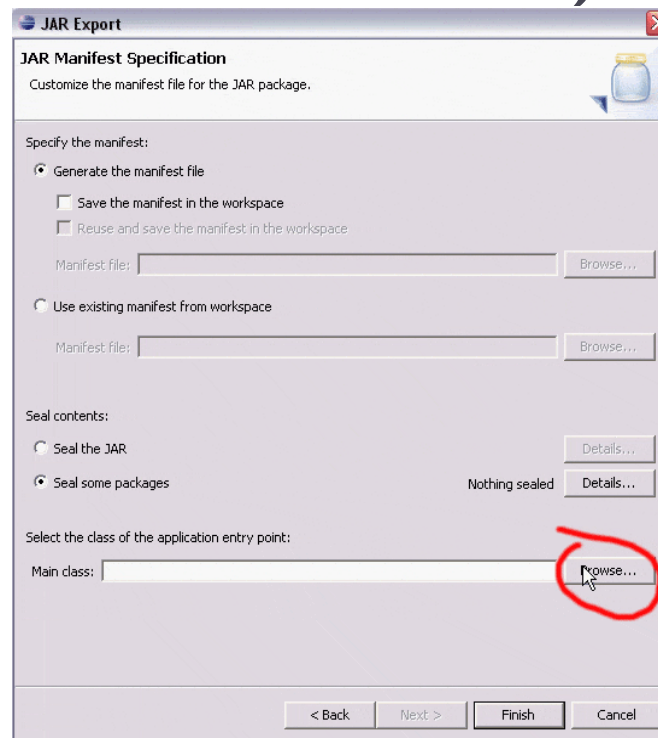
- ▶ Outras opções de empacotamento.
 - ▶ Não modifique nada nessa janela, apenas prossiga.



Como gerar um "executável"

▶ Escolhendo a Classe Principal.

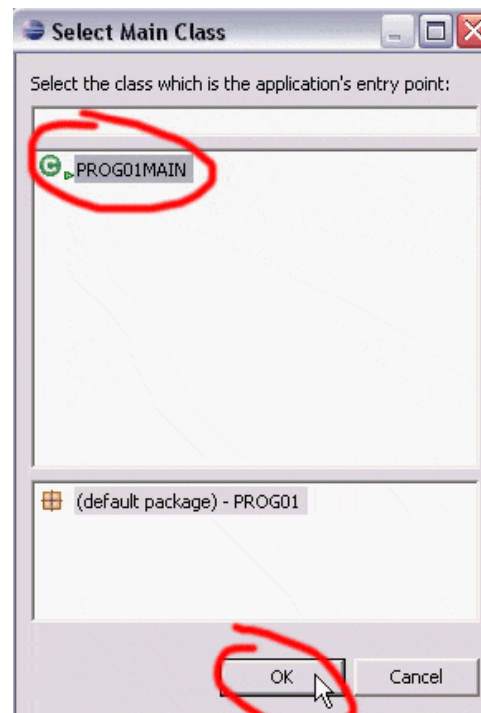
- ▶ Clique em *browse* para selecionar qual das classes presentes no projeto possui uma função *main* que deverá ser chamada.
(PASSO MUITO IMPORTANTE!!!)



Como gerar um "executável"

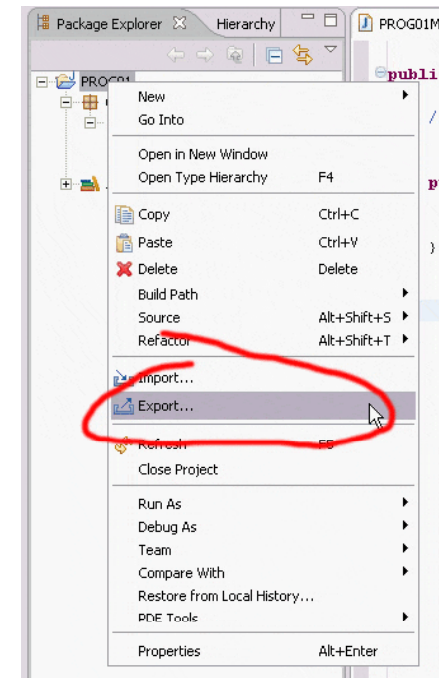
▶ Seleção da Classe *Main*

- ▶ Selecione corretamente a Classe que possui a função *main* definida e clique em OK para prosseguir.
- ▶ No exemplo abaixo só existe uma classe então fica fácil.



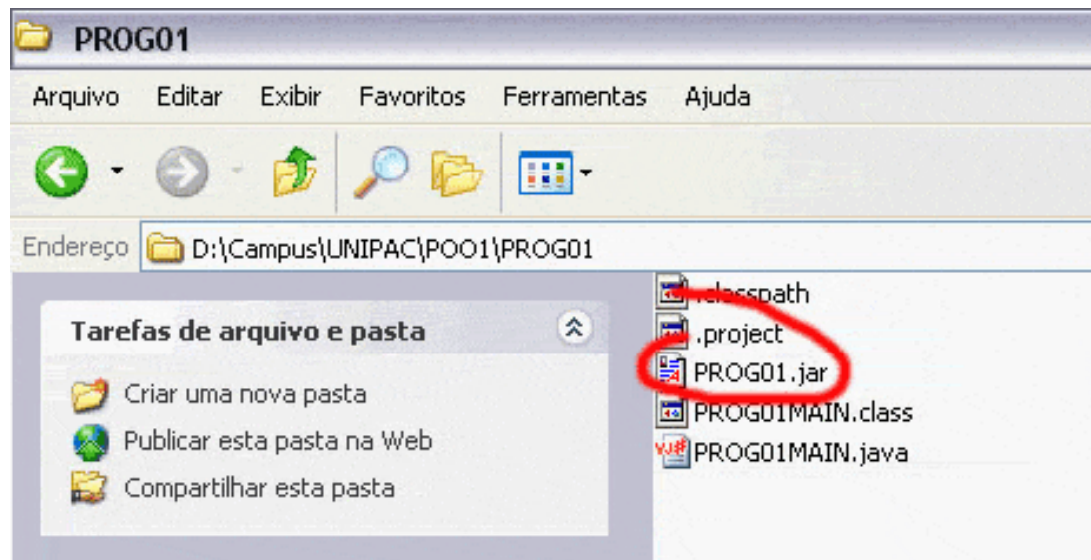
Como gerar um "executável"

- ▶ Verificação das opções de empacotamento.
- ▶ Verifique se no nome da Classe principal está correto e finalize clicando em *Finish*.
- ▶ Seu programa empacotado será salvo no diretório selecionado no passo:
 - ▶ Escolhendo um nome para o arquivo.



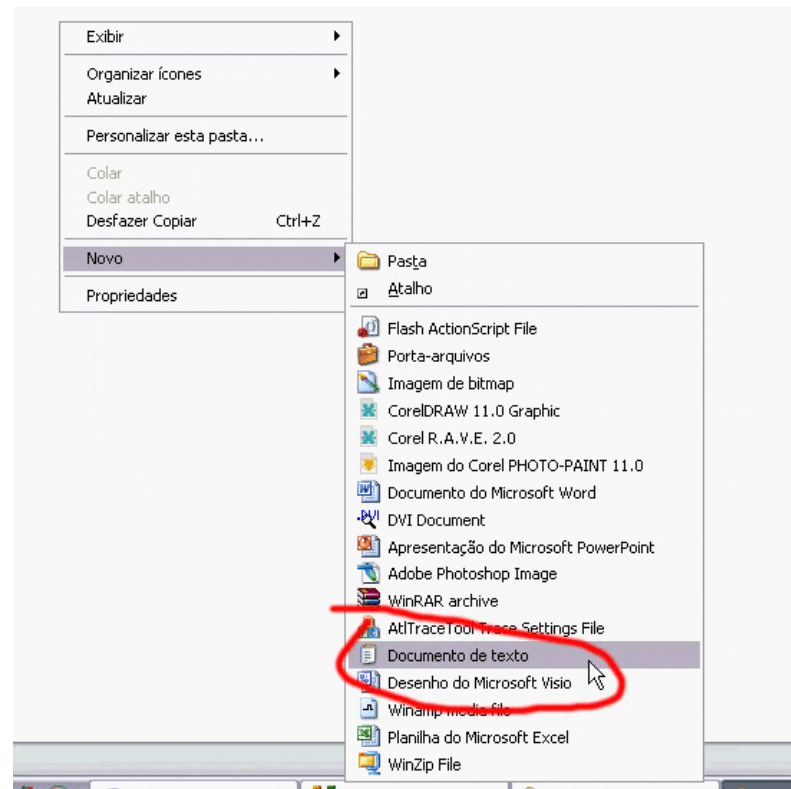
Executar um programa fora do Eclipse

- ▶ Localizando um arquivo JAR.
- ▶ Após gerar um arquivo JAR procure no gerenciador de arquivos em que local ele foi salvo.
- ▶ No exemplo abaixo temos um arquivo salvo no mesmo local do seu projeto.



Executar um programa fora do Eclipse

- ▶ Criando um arquivo texto.
 - ▶ Clique com o botão direito na pasta onde o JAR localiza-se e crie um novo Documento de Texto.

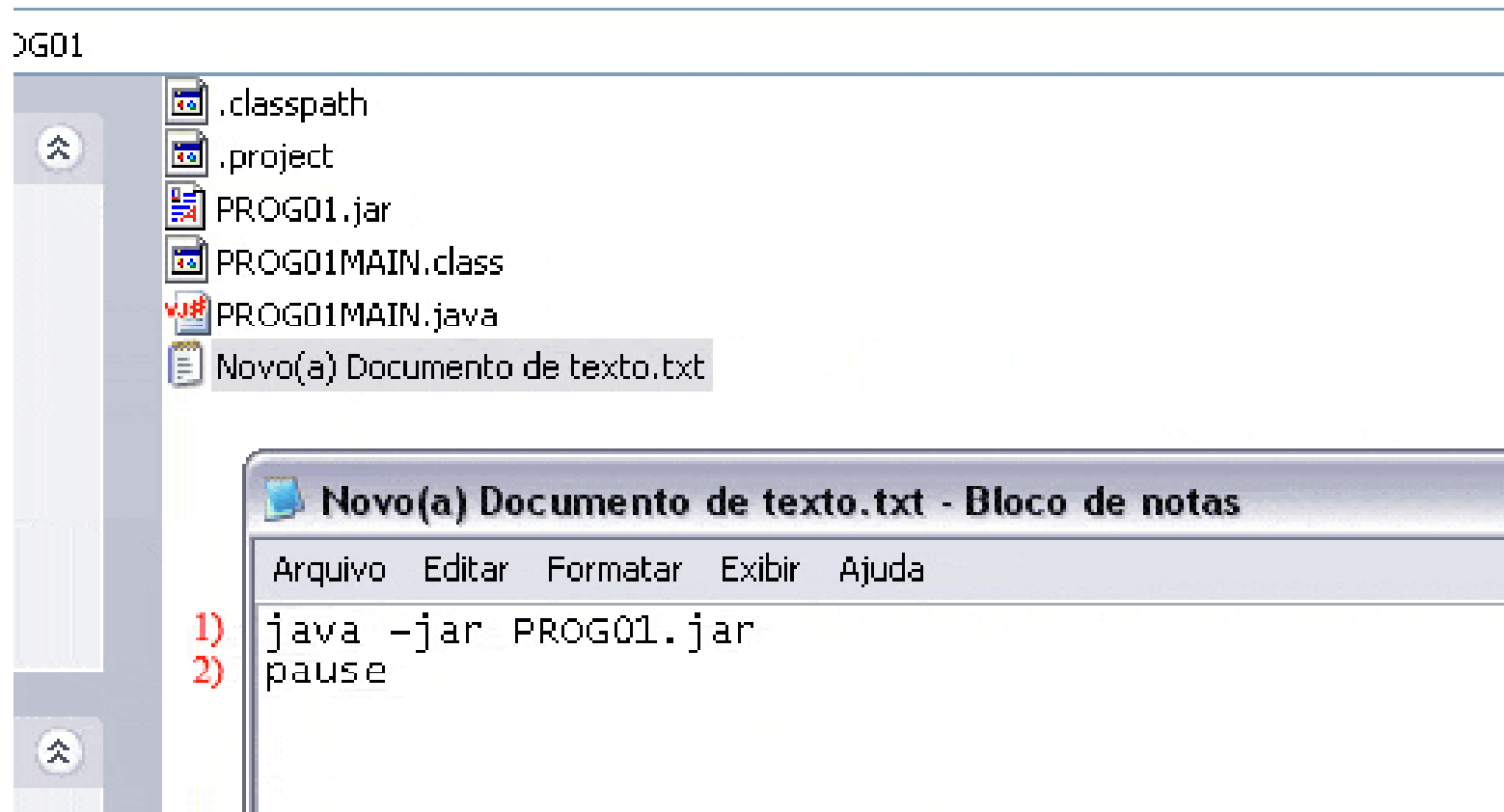


Executar um programa fora do Eclipse

- ▶ Abra o arquivo texto criado.
- ▶ Insira as seguintes linhas no arquivo:
 - ▶ 1 - `java -jar NOME_DO_ARQUIVO.jar`
 - ▶ o comando 1 se for executado no *prompt* de comando (DOS) chama a máquina virtual java e manda ela executar o conteúdo do arquivo especificado.
 - obs: troque NOME_DO_ARQUIVO pelo nome correto do arquivo JAR. presente no diretório.
 - ▶ 2 - `pause`
 - ▶ o comando `pause` é um comando do *prompt* de comando e ele pausa a execução de um arquivo `.bat` para que possamos ver os resultados na tela.
 - Sugestão: retire o comando `pause` para testar seu efeito.

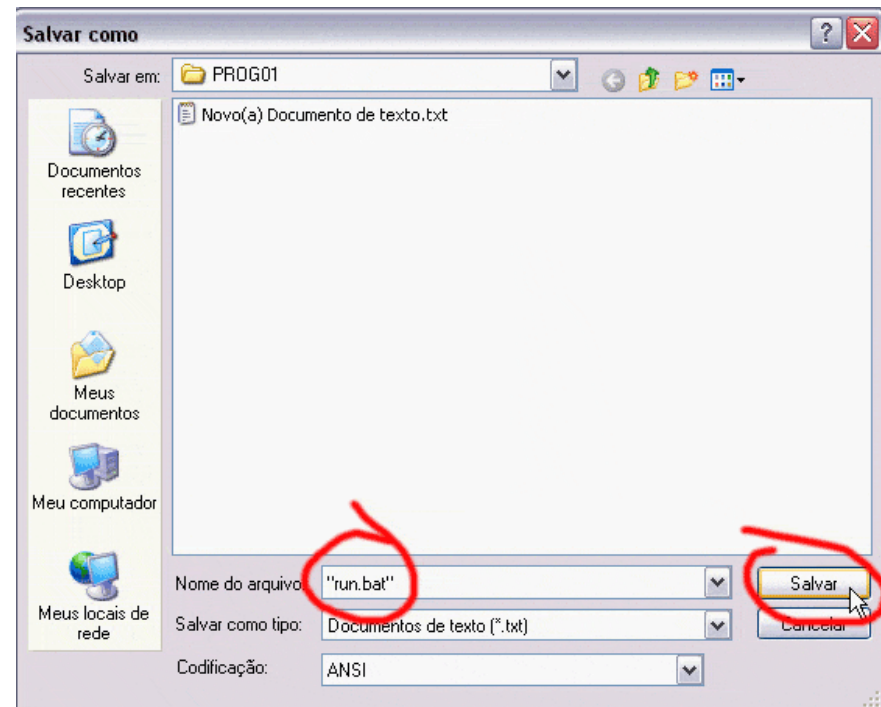
Executar um programa fora do Eclipse

- ▶ Deve-se ter obtido algo como:



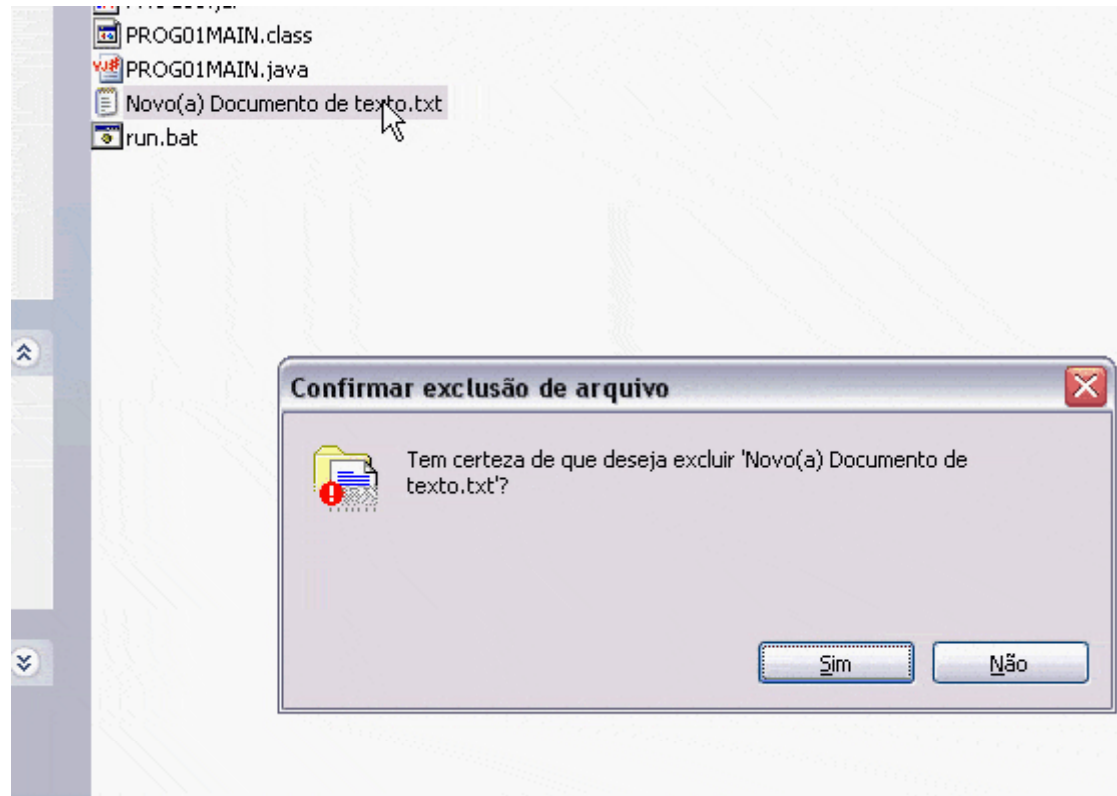
Executar um programa fora do Eclipse

- ▶ Salvando o arquivo texto.
 - ▶ Um arquivo texto com tais comandos não serviria para executar nosso programa.
 - ▶ Para ser capaz de executar o arquivo JAR ele deve ser salvo com outra extensão → .bat.
 - ▶ Na caixa de diálogo de salvamento escolha um nome de arquivo com a extensão .bat e coloque esse nome entre aspas (").
 - ▶ Clique em salvar em seguida.



Executar um programa fora do Eclipse

- ▶ Apagando o arquivo texto original.
 - ▶ O arquivo texto criado inicialmente já não é mais necessário, então já pode ser apagado.



Executar um programa fora do Eclipse

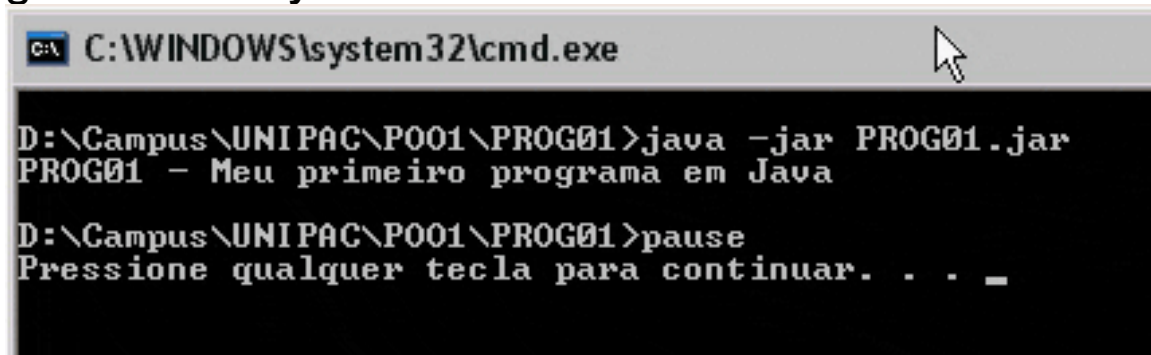
- ▶ Executando o arquivo JAR.

- ▶ Dê um duplo clique no arquivo .bat criado e ele executará sua aplicação em Java. Na figura abaixo vemos a saída do programa.

- ▶ Pergunta: Vemos que o programa executou, mas por quê?

- ▶ Resposta: O arquivo .bat tem como função executar todos os comandos DOS inseridos nele de forma seqüencial (comandos em batelada).

- O programa executou do mesmo jeito que no Eclipse, pois executamos comandos dizendo à Máquina Virtual Java (JVM) para que interpretasse nosso código contido no JAR.



```
C:\WINDOWS\system32\cmd.exe
D:\Campus\UNIPAC\PO01\PROG01>java -jar PROG01.jar
PROG01 - Meu primeiro programa em Java
D:\Campus\UNIPAC\PO01\PROG01>pause
Pressione qualquer tecla para continuar. . . _
```

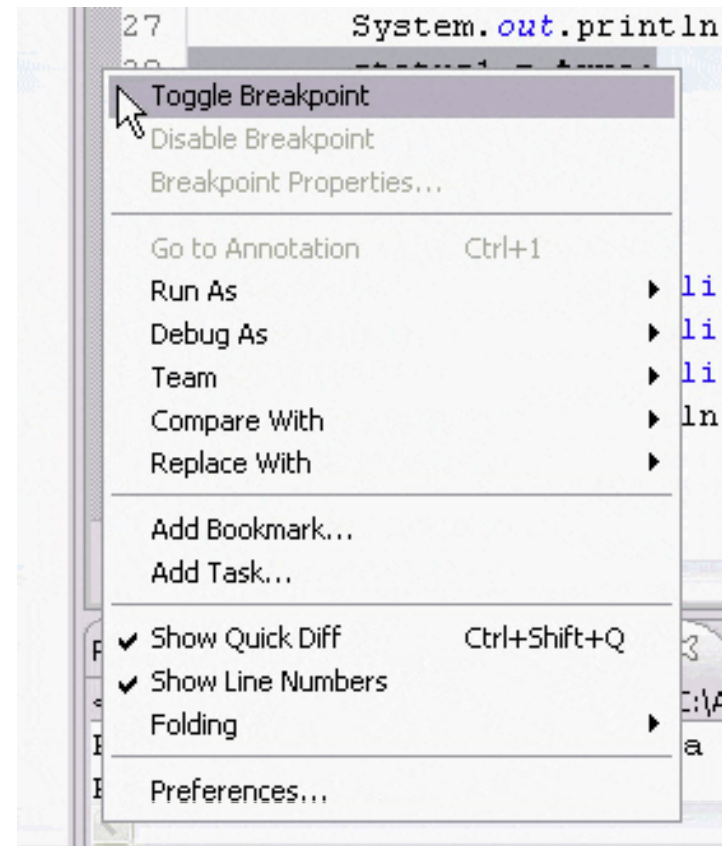
Para "debugar" um programa

- ▶ Gere um código primeiramente.
 - ▶ A figura abaixo demonstra um código exemplo.

```
5      * @param args
6      */
7      public static void main(String[] args) {
8          // TODO Auto-generated method stub
9          System.out.println("PROG01 - Debugando um programa");
10
11         /**
12          * Definindo variáveis sem INICIALIZÁ-LAS
13          */
14         System.out.println("PROG01 - Definindo variáveis");
15         boolean status1, status2;
16
17         int inteiro1, inteiro2, inteiro3;
18
19         String texto1, texto2;
20         String texto3 = new String();
21
22         System.out.println("PROG01 - Variáveis definidas corretamente");
23
24         /**
25          * Inicializando variáveis
26          */
27         System.out.println("PROG01 - Inicializando variáveis");
28         status1 = true;
29         status2 = false;
30         inteiro1 = 15;
31         inteiro2 = 75;
32         inteiro3 = 94;
33         texto1 = "Inicializando texto 1";
34         texto2 = "Inicializando texto 2";
35         texto3 = "Inicializando texto 3";
36         System.out.println("PROG01 - Variáveis inicializadas corretamente");
37
38     }
39
```

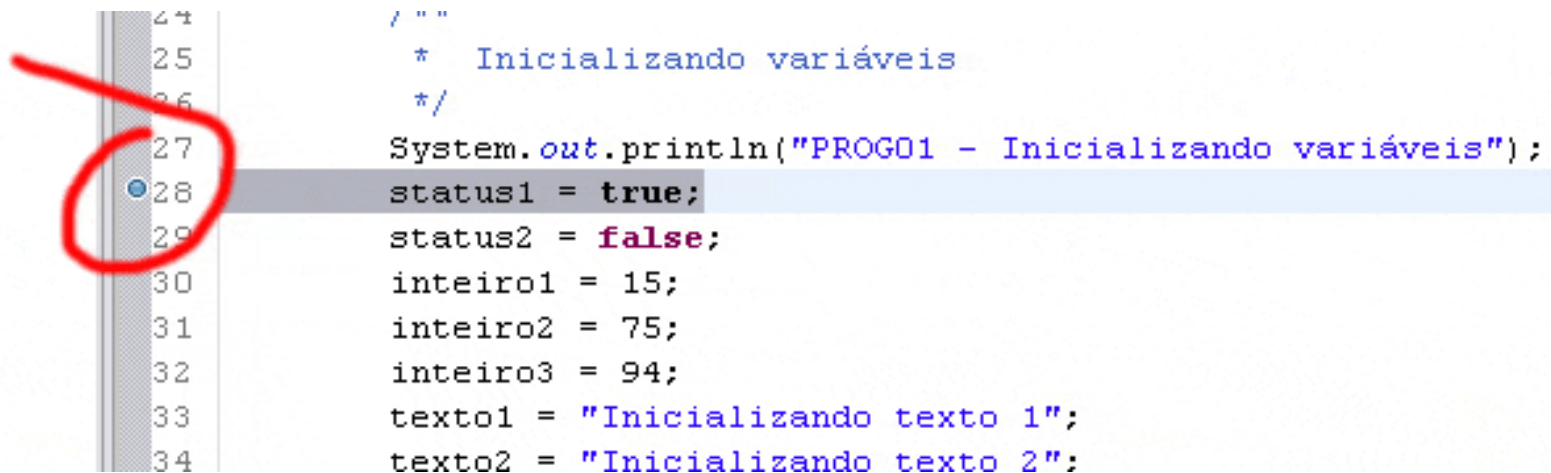
Para "debugar" um programa

- ▶ Criando pontos de parada (*breakpoints*) .
 - ▶ Clique com o botão direito linha onde você deseja suspender a execução.
 - ▶ Selecione a opção *Toggle Breakpoint*.



Para "debugar" um programa

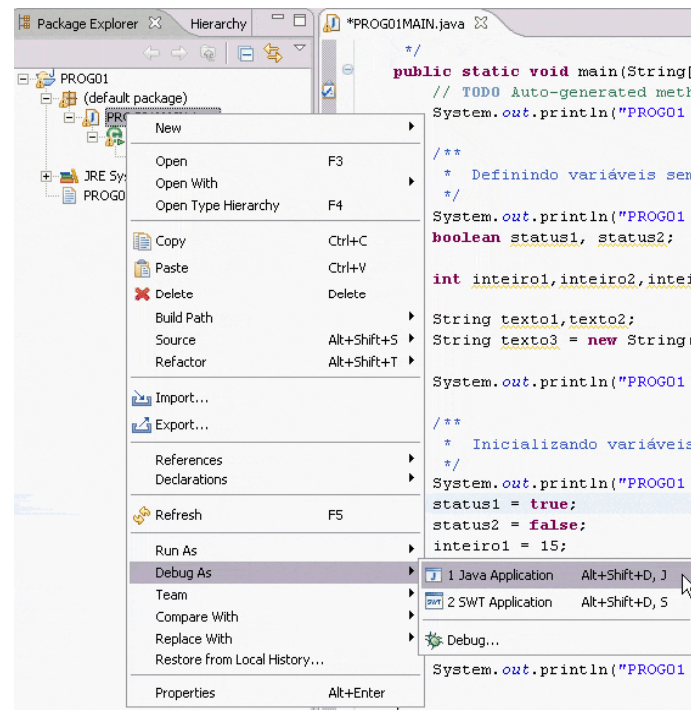
- ▶ Visualização do *breakpoint*.
 - ▶ O ponto de parada aparece à esquerda do código como um ponto azul.



```
24  
25     * Inicializando variáveis  
26     */  
27     System.out.println("PROG01 - Inicializando variáveis");  
28     status1 = true;  
29     status2 = false;  
30     inteiro1 = 15;  
31     inteiro2 = 75;  
32     inteiro3 = 94;  
33     texto1 = "Inicializando texto 1";  
34     texto2 = "Inicializando texto 2";
```

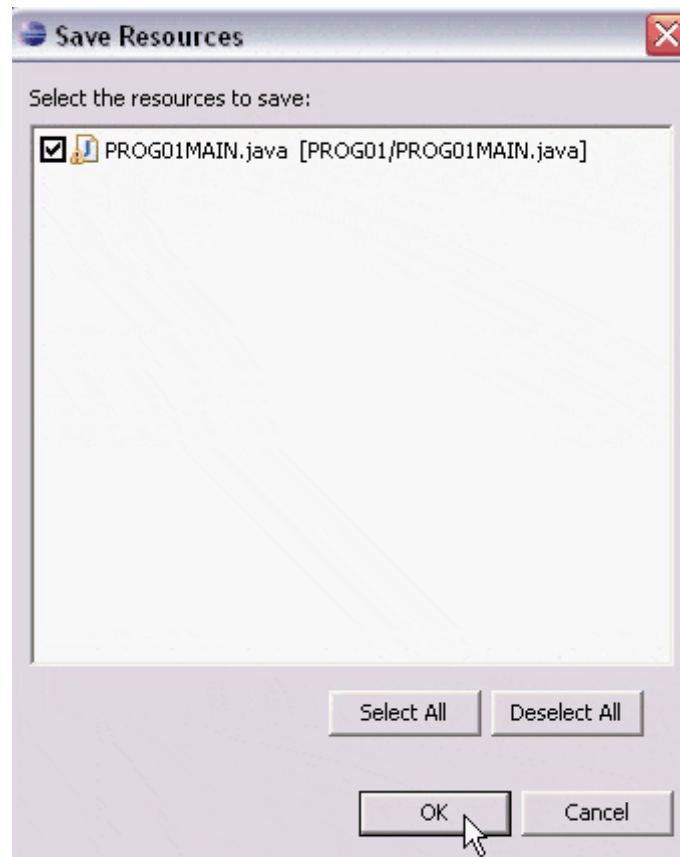
Para "debugar" um programa

- ▶ Executando em modo DEBUG.
 - ▶ Clique com o botão direito na Classe Principal (classe que possui o método `main()`). Selecione a opção:
 - ▶ *Debug As >> Java Application*



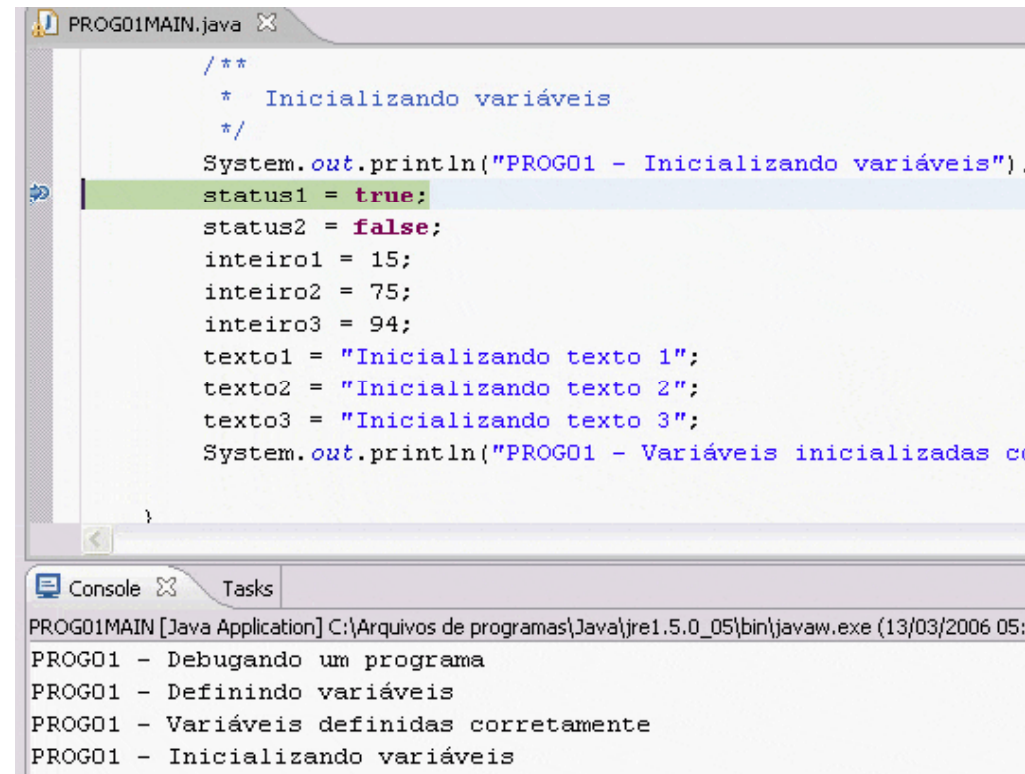
Para "debugar" um programa

- ▶ **Solicitação de salvamento.**
 - ▶ Antes de executar o projeto deve ser salvo.



Para "debugar" um programa

- ▶ Entrando em modo Debug.
 - ▶ O programa muda sua aparência e entra em modo Debug (modo de testes).
 - ▶ Perceba que a execução do programa foi pausada exatamente onde o breakpoint foi habilitado.
 - ▶ A linha corrente de execução aparece selecionada.



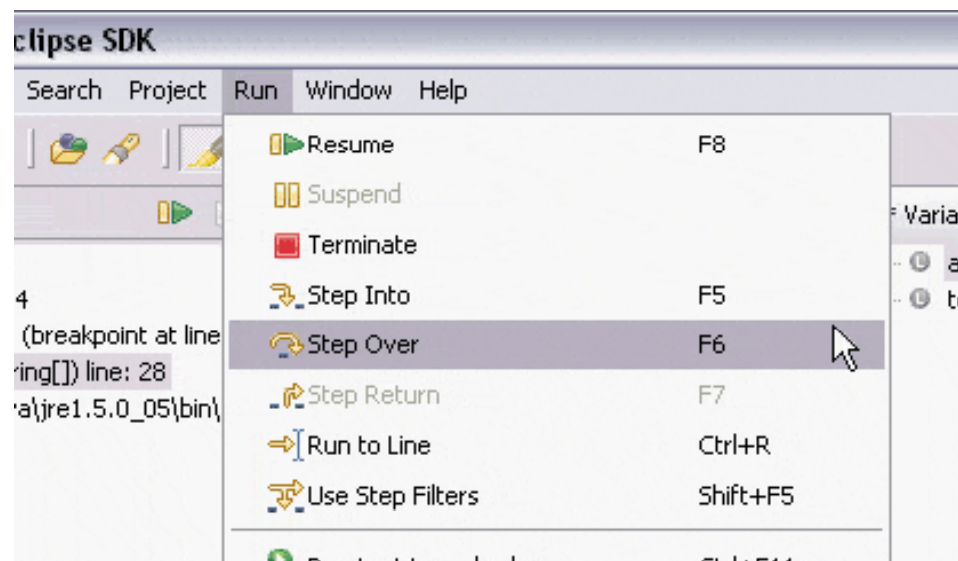
```
PROG01MAIN.java X
/**
 * Inicializando variáveis
 */
System.out.println("PROG01 - Inicializando variáveis");
status1 = true;
status2 = false;
inteiro1 = 15;
inteiro2 = 75;
inteiro3 = 94;
texto1 = "Inicializando texto 1";
texto2 = "Inicializando texto 2";
texto3 = "Inicializando texto 3";
System.out.println("PROG01 - Variáveis inicializadas co

}

Console X Tasks
PROG01MAIN [Java Application] C:\Arquivos de programas\Java\jre1.5.0_05\bin\javaw.exe (13/03/2006 05:
PROG01 - Debugando um programa
PROG01 - Definindo variáveis
PROG01 - Variáveis definidas corretamente
PROG01 - Inicializando variáveis
```

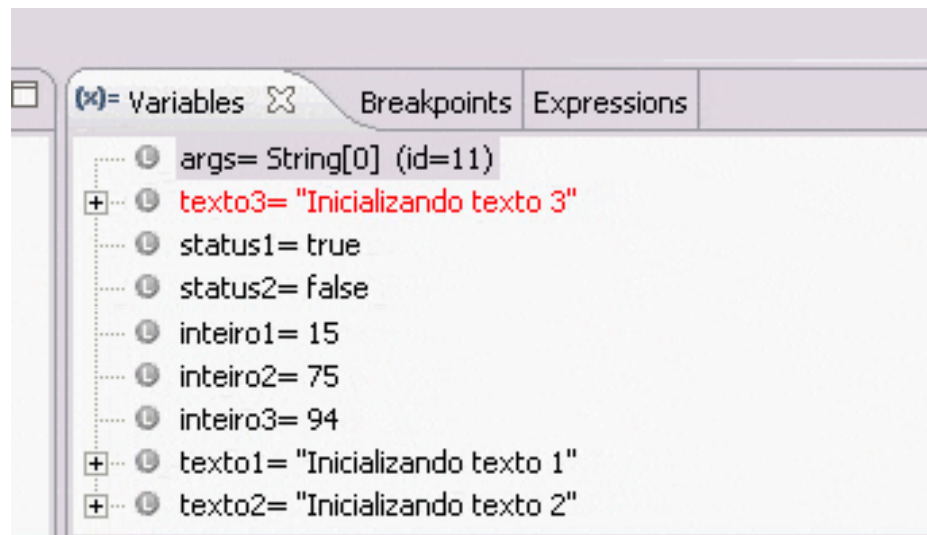
Para "debugar" um programa

- ▶ Executando passo a passo.
 - ▶ A partir do ponto de parada (breakpoint) podemos escolher executar linha por linha de código (*Step Over* → F6), entrar dentro de algum método (*Step Into* → F5), entre outras ações.
 - ▶ Escolheremos executar passo a passo com o *Step Over* (passada por cima).



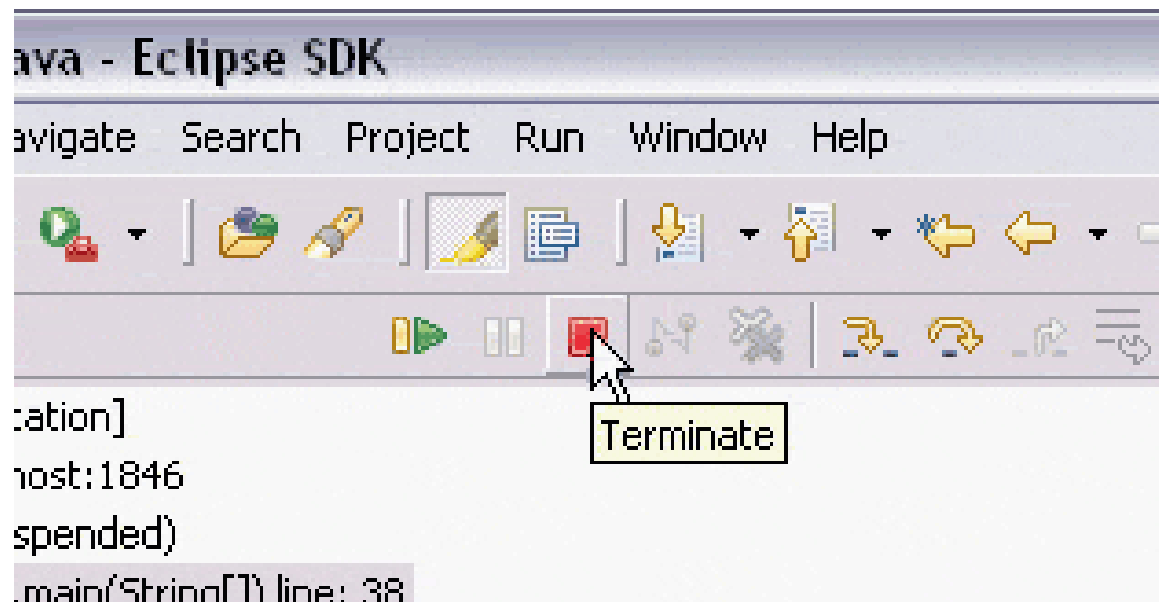
Para "debugar" um programa

- ▶ Verificando valores de variáveis.
 - ▶ Surge no modo Debug uma janela de nome *Variables*.
 - ▶ Tal janela mostra ao usuário as variáveis correntes e seus valores.
 - ▶ Conforme executamos passo a passo os valores das variáveis vão se modificando.



Para "debugar" um programa

- ▶ Terminando o modo Debug.
 - ▶ Sair do modo Debug é possível se o programa terminou sua execução ou se clicarmos no botão *Terminate*.



DÚVIDAS

