O IDE baseado em

Eclipse

para desenvolvimento em Java

João Paulo Barros
Instituto Politécnico de Beja
Escola Superior de Tecnologia e Gestão
Beja, Portugal

Temos estado a utilizar um ambiente de desenvolvimento (em inglês, IDE para *Integrated Development Environment*) que tem fins pedagógicos: o BlueJ. O BlueJ é um excelente ambiente para o fim a que se destina: a aprendizagem da programação orientada pelos objectos utilizando a linguagem Java. No entanto e especialmente quando é utilizada uma linguagem muito popular como a linguagem Java, o desenvolvimento de programas é feito utilizando IDEs muito mais complexos e sofisticados. Tal é necessário para lidar com a complexidade de muitos dos programas actuais. Estes atingem frequentemente dezenas de milhares de linhas de código e, nalguns casos, muitas, muitas mais. Tal implica a utilização de várias ferramentas de auxílio ao desenvolvimento de código. Estas não se resumem ao editor de texto e ao compilador, incluem também coisas como ferramentas de suporte à geração de testes (*unit testing frameworks*) depuradores (*debuggers*), controlo de versões, editores para construção de interfaces gráficas e editores para linguagens gráficas (e.g. UML).

Todas estas ferramentas são importantes pelo que os IDEs modernos têm vindo a incluir todas elas. Ou seja, integram cada vez mais ferramentas. O eclipse é uma resposta a esta necessidade de integrar ferramentas distintas para o desenvolvimento de software. Em http://wiki.eclipse.org/index.php/FAQ_What_is_Eclipse%3F encontramos esta definição:

> *Eclipse is an open (IDE) platform for anything, and for nothing in particular. Eclipse is open because its design allows for easy extension by third parties. It is an Integrated Development Environment (IDE) because it provides tooling to manage workspaces; to build, launch and debug applications; to share artifacts with a team and to version code; and to easily customize the programming experience. Eclipse is a platform because it is not a finished application per se but is designed to be extended indefinitely with more and more sophisticated tooling. Eclipse is suitable for anything because it has been used successfully to build environments for wide-ranging topics, such as Java development, Web Services, embedded device programming, and game-programming*

> *contests. Eclipse has no particular focus on any vertical domain. The dominance of Java development tooling in Eclipse is merely historical. The platform has no explicit or implicit support whatsoever for Java development as provided by the Java development tools (JDT). The JDT has to play according to the same rules as all the other plug-ins that use the platform.*

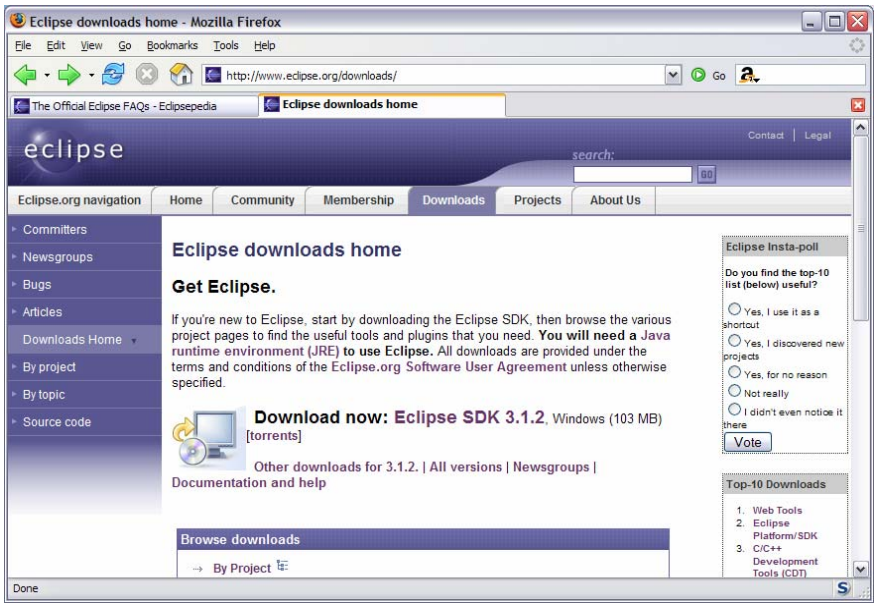
Pois é. O Eclipse dá para muita coisa. Apesar de todas estas ferramentas terem a sua utilidade, por agora vamos centrar-nos na utilização mais popular do eclipse: o seu IDE para Java. Para muito mais informação do que aquela que aqui será dada, aconselham-se as referências na Leitura Complementar no final deste texto.

Vamos então seguir o caminho necessário até pôr o eclipse a funcionar. Para tal, vamos considerar quatro passos: **obtenção**, **instalação**, **execução** e **configuração**.

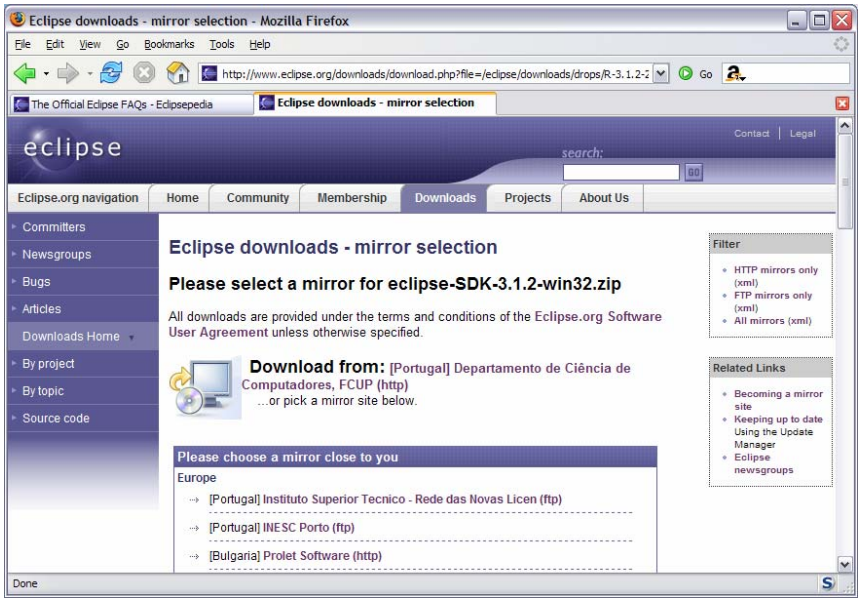
No código seguinte utiliza-se **esta fonte** para texto que surge na interface do eclipse.

1 Obtenção

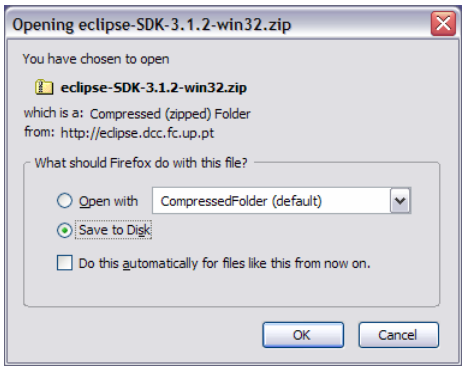
O Eclipse é disponibilizado a custo zero na Internet. Pode fazer o *download* do Eclipse SDK que inclui o IDE Java em http://www.eclipse.org/downloads/:



Clicando em Eclipse SDK 3.1.2 irá para a seguinte janela:



Clicando no *link* sugerido ou num dos *mirrors* deverá fazer *save* para o disco do seu computador:



O *download* deverá demorar vários minutos. Na mesma página encontra muito outros links que lhe possibilitam várias formas distintas de fazer o *download*. Se a ligação que está a utilizar for lenta ou pouco fiável pode ser boa ideia utilizar um software para gerir *downloads* (o http://www.freedownloadmanager.org/ é um dos melhores).

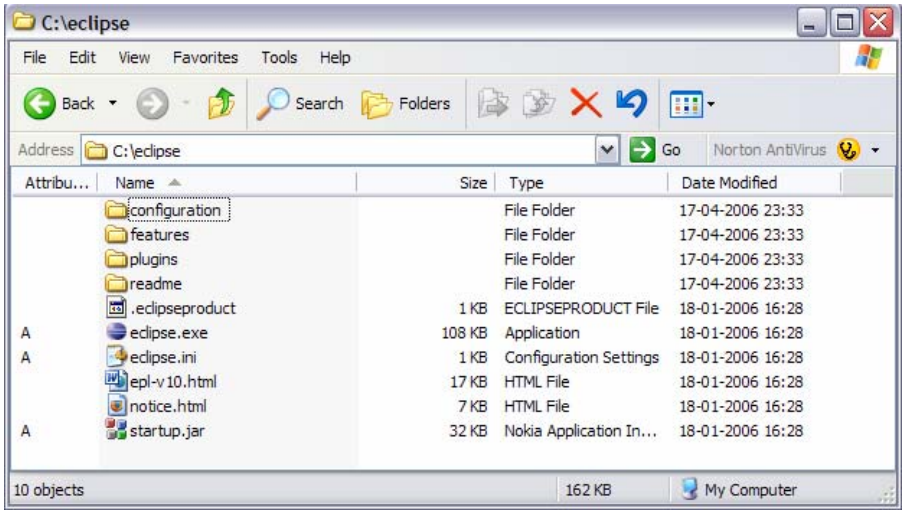
2 Instalação

Antes de instalar o eclipse deve instalar o Java SDK 1.5. Em princípio já o deve ter instalado, pois ele também é utilizado pelo BlueJ que temos estado a utilizar. Vamos assumir que o SDK está instalado na directoria `C:\Program Files\Java\jdk1.5.0_06` e vamos descrever a instalação no sistema operativo Microsoft Windows XP.

A primeira tarefa é a de descompactar o ficheiro `eclipse-SDK-3.1.2-win32.zip` para uma directoria. Vamos assumir a directoria `C:\eclipse`. Na descompactação basta

abrir o ficheiro `zip`, que contem uma directoria eclipse, e copiá-la para a directoria `C:\`. Tal deve demorar alguns minutos pois são muitos ficheiros. Note que desde que tenha permissões de escrita na directoria escolhida para destino, neste caso a directoria `C:\`, não necessita de privilégios de administrador para fazer esta operação[1].

Logo que a descompactação tiver terminado, deve ter uma directoria `C:\eclipse` com o seguinte conteúdo:

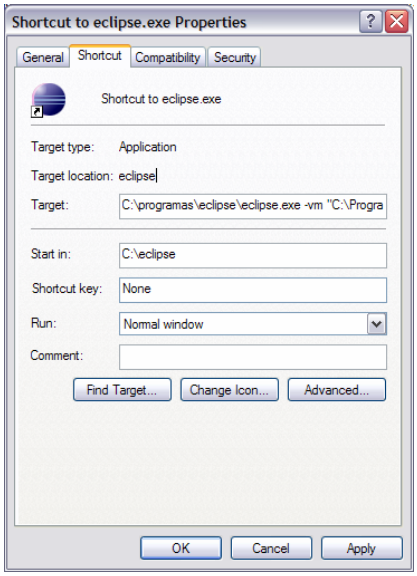


Agora é aconselhável fazer mais uma configuração: adicionar um *shortcut* para o ficheiro eclipse.exe. Tal pode ser feito em três passos: (1) seleccionando o ficheiro, (2) fazendo `Edit->Copy`, (3) `Edit->Paste Shortcut`.

Seguidamente, deve clicar com o botão do lado direito do rato no shortcut criado e colocar o seguinte texto em `Target` (*vide* imagem seguinte):

```
C:\eclipse\eclipse.exe -vm "C:\Program Files\Java\jdk1.5.0_06\bin\javaw" -vmargs -
Xmx256M
```

[1] É boa ideia trabalhar sempre como utilizador restrito (*restricted user*) no Windows XP e só utilizar a conta de utilizador quando é mesmo necessário.



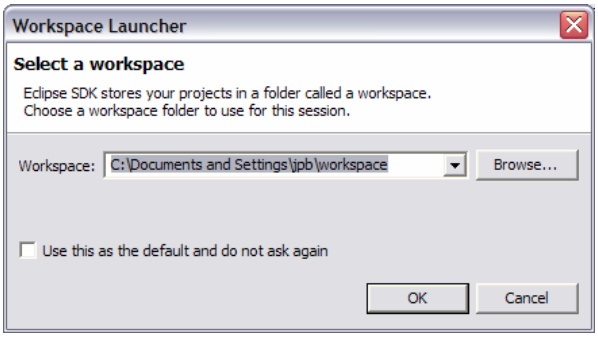
Esta linha especifica explicitamente qual a máquina virtual Java que será utilizada para executar o eclipse e qual a quantidade de memória que deve ser especificada. Isto é aconselhado no ficheiro `C:\eclipse\readme\readme_eclipse.html`.

Finalmente, o *shortcut* pode ser copiado para o *desktop* ou para outros locais da sua preferência.

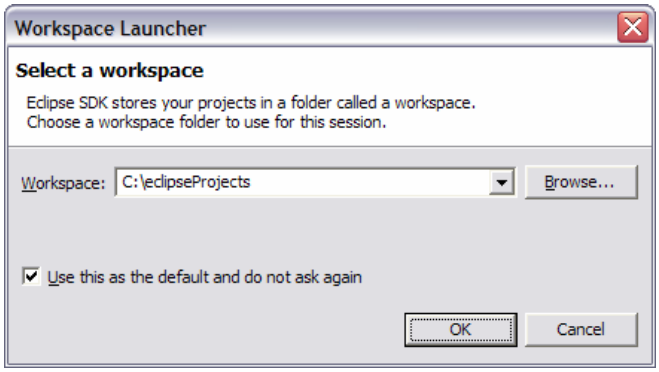
Agora já pode executar o eclipse.

3 Execução

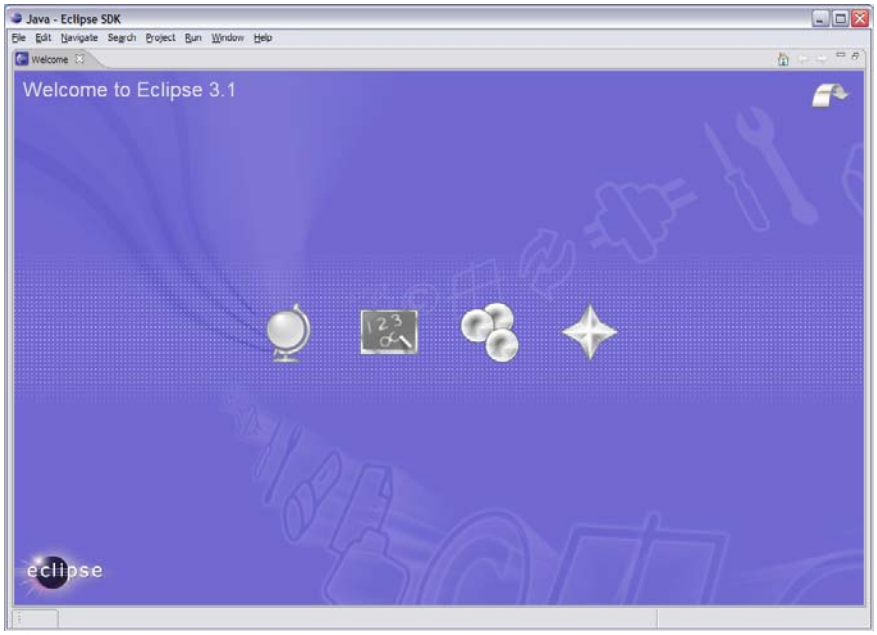
Fazendo duplo *click* no *shortcut* criado deverá obter a seguinte caixa de diálogo:



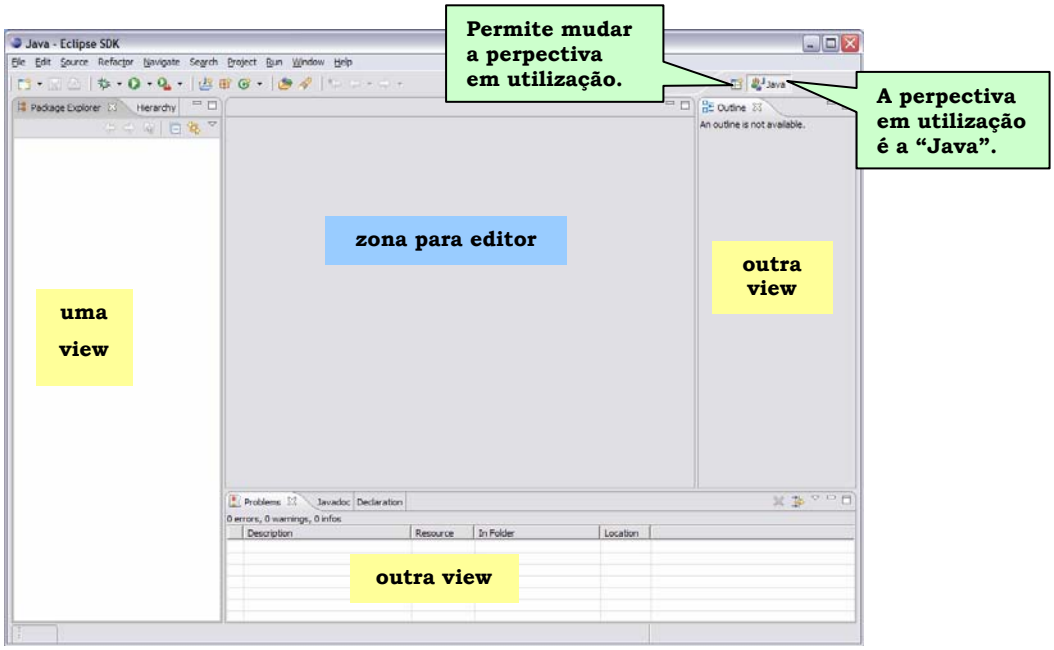
É boa ideia evitar a localização por omissão (*default*). Em vez disso crie uma nova directoria e passe a utilizá-la por omissão. Por exemplo, considerando a directoria `C:\eclipseProjects` teremos:



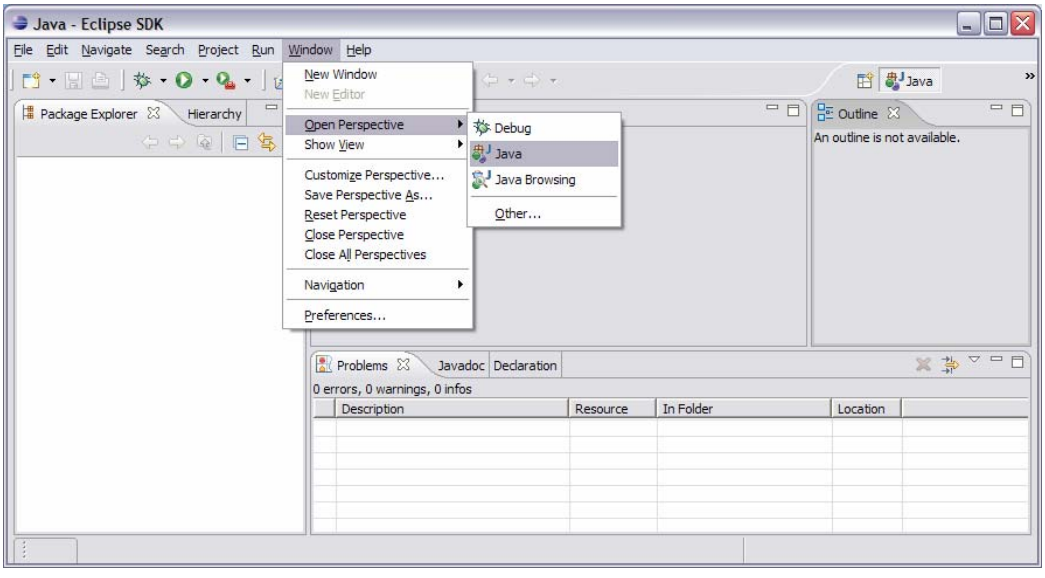
Clicando em OK deverá obter uma janela introdutória.



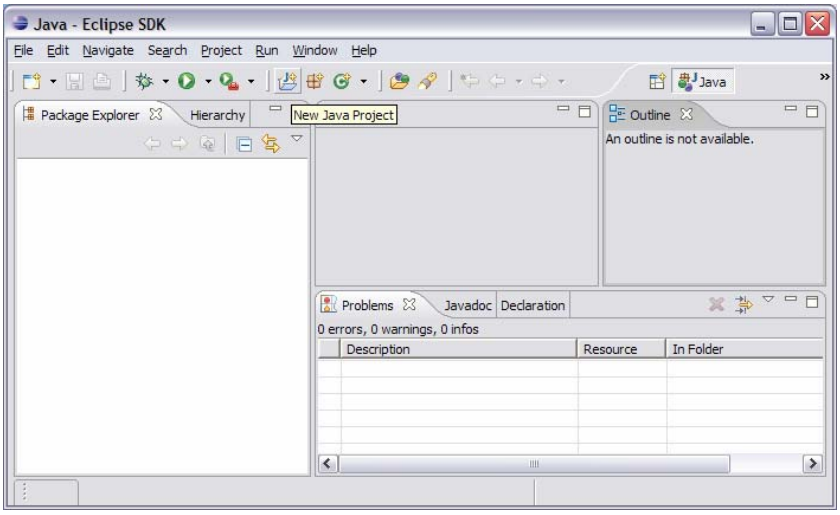
Mais tarde pode seguir as tutorias (segundo icon a contar da esquerda) mas por agora vamos apenas fechar esta janela **Welcome**. Para tal, deve clicar no símbolo × no campo superior esquerdo, logo após a palavra **Welcome**, ou seja, fechar a `tab` **Welcome**. Chegamos assim à ***resource workbench*** do Eclipse, ou seja, a bancada de trabalho de recursos. Este nome soa algo estranho pelo que vamos utilizar o nome em inglês: *resource workbench.*



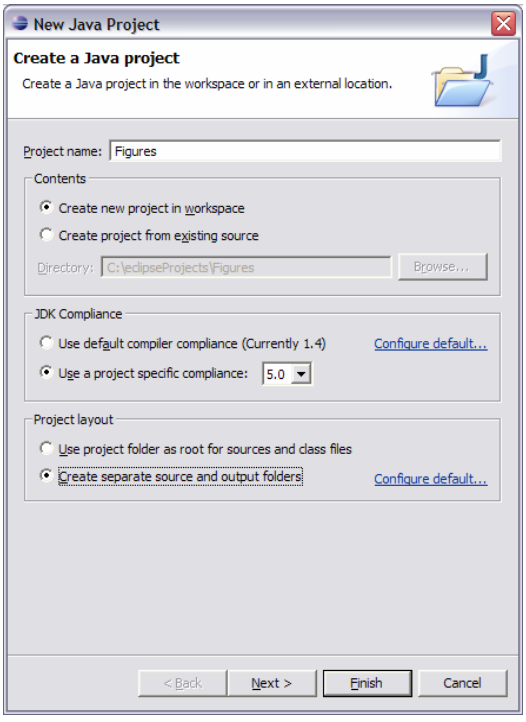
O conceito de ***perspective*** é muito importante no Eclipse. Uma *perspective* define quais e como surgem no bancada (na *workbench*) as ***views*** que lhe estão associadas. A razão para este funcionamento resulta do facto de ser conveniente uma arrumação diferente (uma diferente perspectiva) para cada tipo de tarefa que possamos estar a realizar. E note-se que o Eclipse suporta muitos tipos de tarefas. A perspectiva apresentada está feita para programação em Java e denomina-se simplesmente Java. Note-se que as perspectivas estão disponíveis no menu **Window->Open Perspective**, conforme ilustrado na figura seguinte:



Vamos agora criar um projecto que irá suportar o nosso programa. Primeiro vamos espreitar três botões que estão na *toolbar*. A figura seguinte mostra um desses botões já seleccionado: o **New Java Project**. Os outros dois são os que estão à direita: **New Java Package** e **New Java Class**.



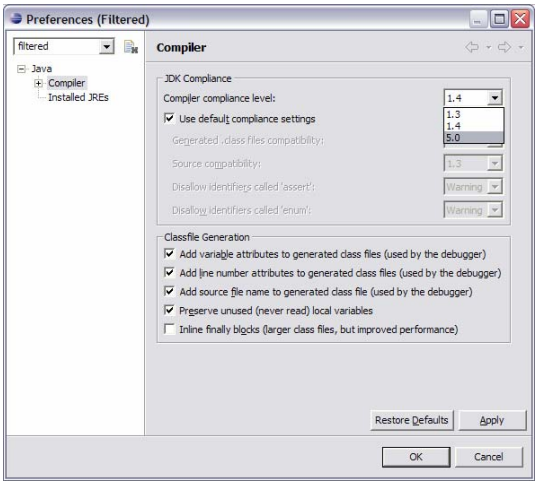
Vamos então criar um novo **Java Project** para o nosso programa de exemplo, o `Figures`. Temos então o seguinte diálogo que pode ser visto como um *wizard*:



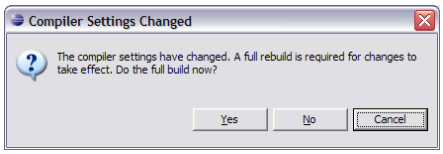
Aqui há várias coisas a fazer:

1. Dar um nome ao projecto (no nosso exemplo é `Figures`);

2. Especificar o **JDK Compliance** para a versão 5.0 do Java e **Configure Default**. Esta opção apresenta o seguinte diálogo.



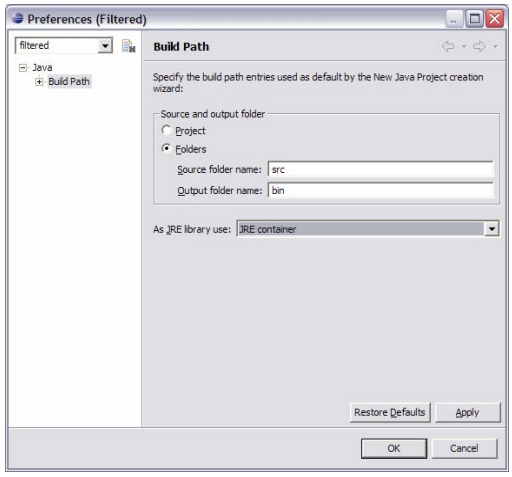
Aí deve escolher 5.0, tal como indicado na figura, e fazer **OK**. Tal origina a seguinte pergunta:



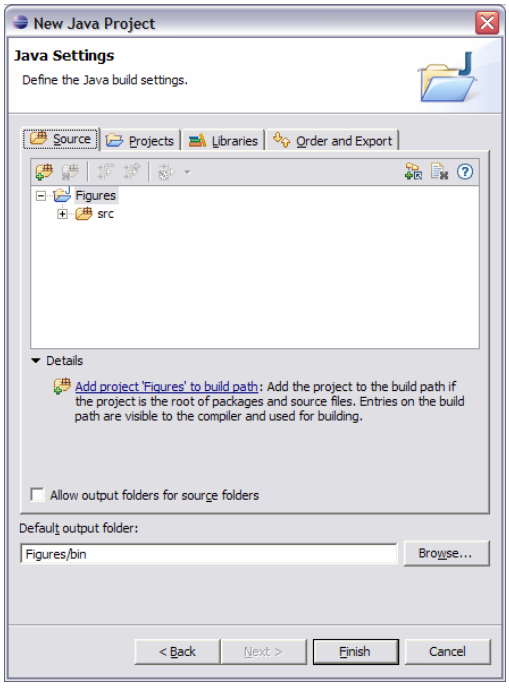
Responda **Yes**. Parece coisa complicada mas é muito rápido.

Estas opções ficam disponíveis para futuras consultas ou alterações em **Project->Properties**.

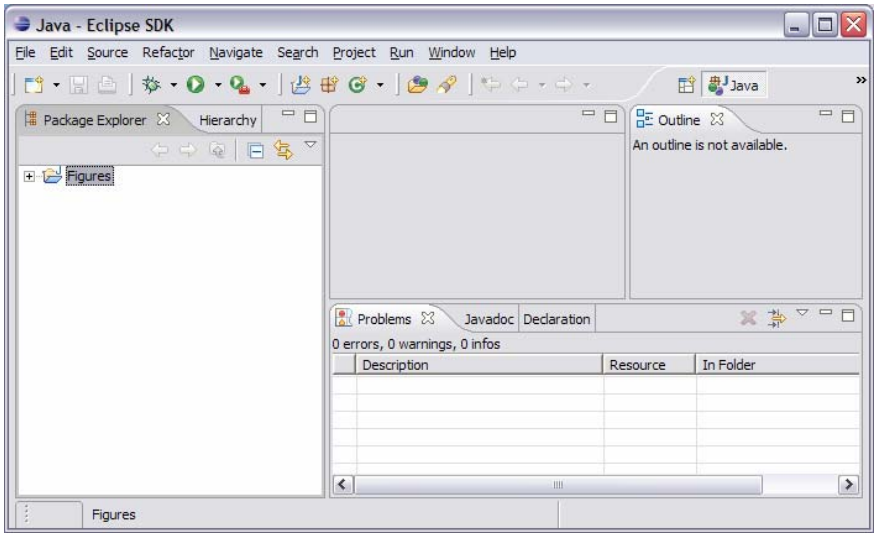
3. No **Project layout** deve especificar diferentes directorias para **source** e **output files**. Tal originará diferentes directorias para os ficheiros `.java` e `.class`. É uma questão de arrumação. Estabeleça também tal como *default*. Na janela seguinte escolha **Folders**.



Voltando à janela **New Java Project**, podemos agora clicar em **Next**. Na janela seguinte basta clicar em **Finish**. Note que nesta janela também pode clicar em **Back** para voltar atrás. É a este tipo de "janelas em cascata" que é costume chamar um *wizard*.



Agora, o projecto criado deverá aparecer na *view* da esquerda, na tab **Package Explorer**:



Podemos agora começar a criar o programa propriamente dito! Para tal vamos começar por criar uma ***package***.

É boa ideia colocar as nossas classes em packages (pacotes) tal como sucede com as classes da biblioteca do Java. Por exemplo, o verdadeiro nome da classe ArrayList é java.util.ArrayList. Para utilizarmos o nome curto (ArrayList) temos de colocar uma instrução import java.util.ArrayList no ficheiro onde a pretendemos utilizar.

As packages permitem evitar colisões de nomes. Por exemplo, é possível que façamos uma classe chamada Worker que queremos adicionar a um programa de gestão de salários. Acontece que já lá pode existir uma classe com esse nome. Se colocarmos a nossa dentro de uma *package* denominada salary então o nome completo da nossa classe passa a ser salary.Worker. Isso é feito adicionado a seguinte directiva como a primeira instrução no ficheiro Worker.java:

```
package salary;
import java.util.ArrayList;
public class Worker
{
   …
}
```

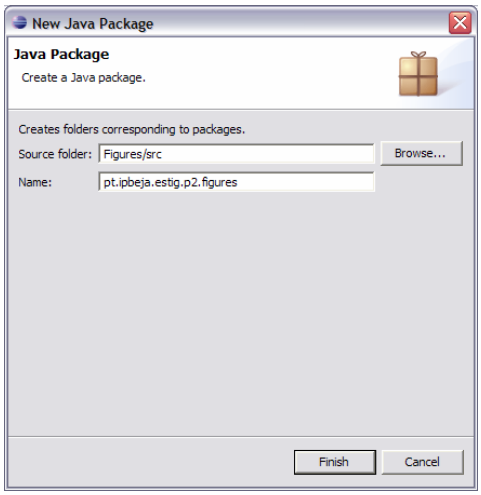
Claro que podemos ter mais azar ainda e já existir uma *package* com uma classe Worker! Como podemos então garantir que os nomes das classes nunca colidem com outras que já existam? A Sun recomenda que utilizemos a seguinte convenção para os nomes das *packages*: colocamos o inverso do nosso nome de domínio na Net. Por exemplo:

pt.ipbeja.estig.p2.salary.Worker

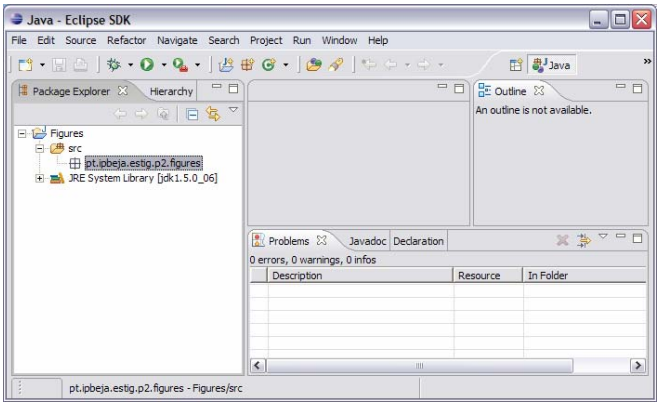
Desta forma garantimos que a classe Worker definida na disciplina de programação 2 da ESTIG tem um nome diferente de outras também denominadas Worker. Note que se utilizam letras minúsculas para os nomes das *packages*, mas os nomes das classes, como sempre, surgem em letra maiúscula.

A organização das classes em packages é muito semelhante à organização dos ficheiros em directorias. Na verdade tal é aproveitado para guardar os ficheiros. Na verdade temos de colocar o ficheiro Worker.java na directoria pt\ipbeja\estig\p2\salary. Felizmente, há várias ferramentas que tratam disso automaticamente e o Eclipse é uma delas.

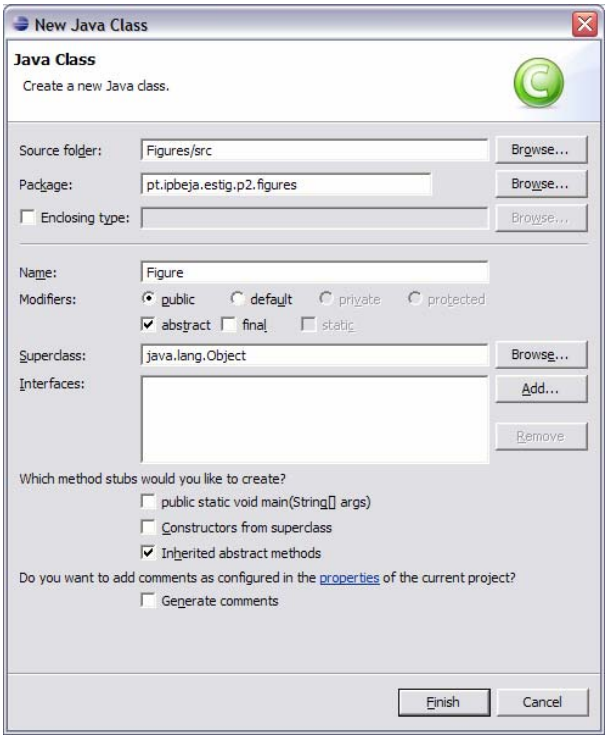
Vamos então criar uma *package* com o nome `pt.ipbeja.estig.p2.figures`. Para tal deve premir o botão **New Java Package** mesmo ao lado do botão **New Java Project**. Deve então indicar o nome da *package* no diálogo seguinte:



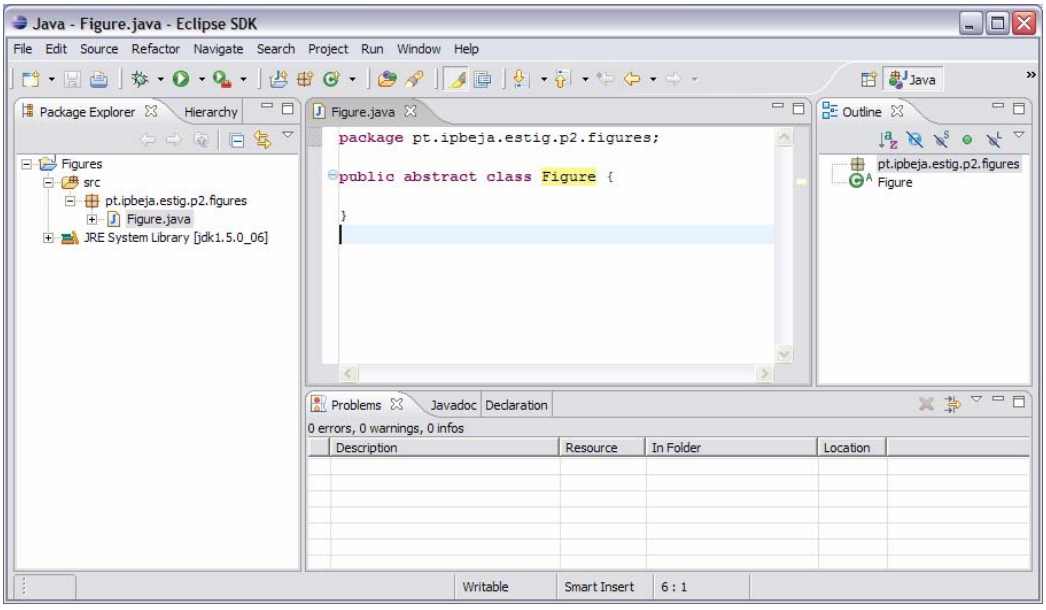
Criemos agora uma nova classe denominada Figure. Para tal seleccionamos a *package* criada no **Package Explorer** conforme ilustrado na figura seguinte:



Depois, utilizamos o botão **New Java Class**. Repare que clicando imediatamente à direita desse botão tem opções para criar um **JUnit Test Case**, **Class**, **Interface**, **Enum** ou **Annotation**. Por agora vamos ver apenas a opção **Class** que faz surgir o seguinte diálogo:

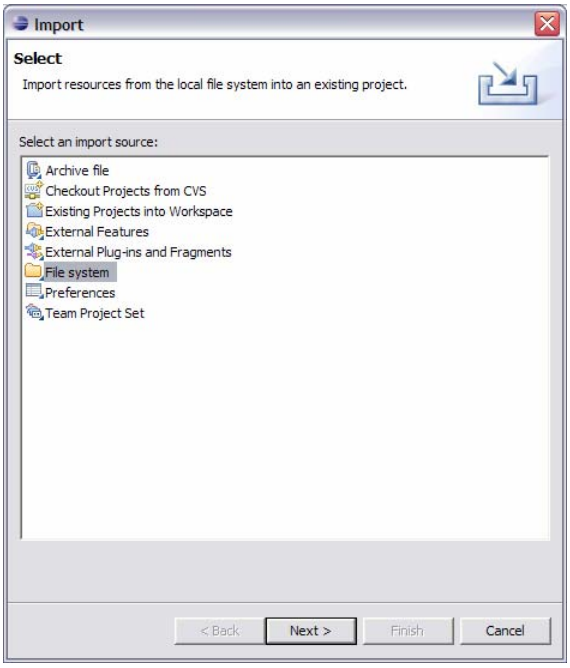


Aqui basta preencher o nome da classe (neste caso `Figure`) e dizer que se trata de uma classe abstracta. Neste momento já terá uma *workbench* um pouco mais interessante:

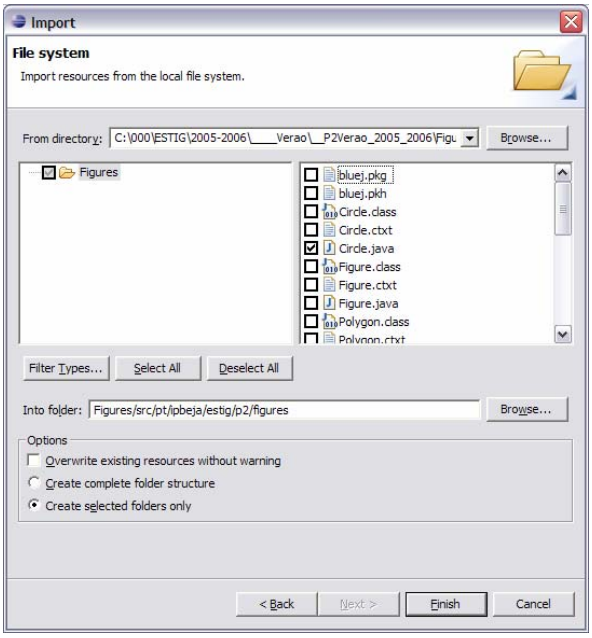


Agora vamos fazer copy/paste do restante código da classe Figure cuja resolução foi já dada. Ficará assim com uma classe completa.

Agora vamos experimentar outra forma de incluir uma classe já feita. Tal é muito útil para aproveitar código já feito: clique com o botão direito em cima da package pt.ipbeja.estig.p2.figures e seleccione a opção **Import**. Obterá o seguinte diálogo:



Clique **Next** e na janela seguinte escolha a directoria onde tem o projecto Figures:



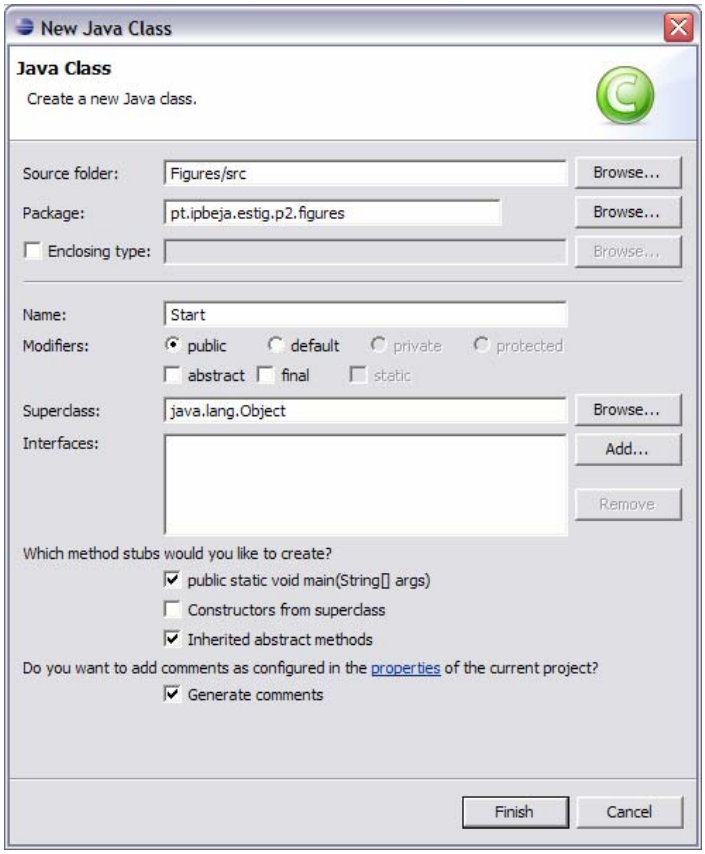
Por agora, seleccione apenas o ficheiro Circle.java, tal como ilustrado na figura anterior, e clique **Finish**.

Este ficheiro terá um erro porque não tem a indicações que está na *package* pt.ipbeja.pestig.p2.figures. Corrija esse facto acrescentando a linha

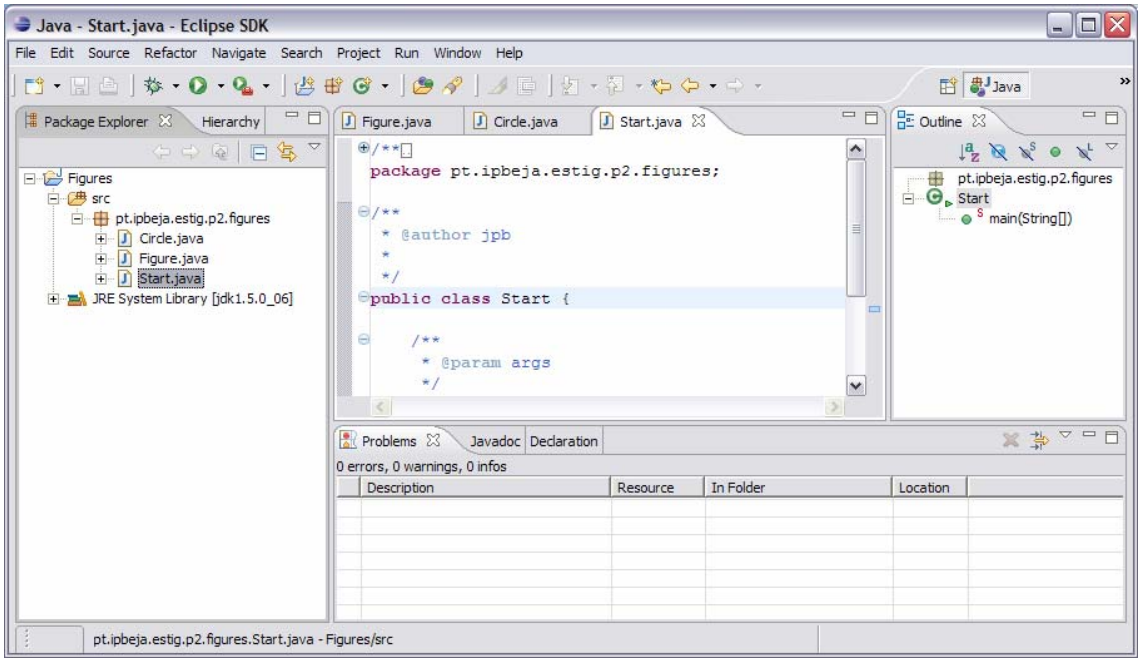
```
package pt.ipbeja.estig.p2.figures;
```

no início do ficheiro.

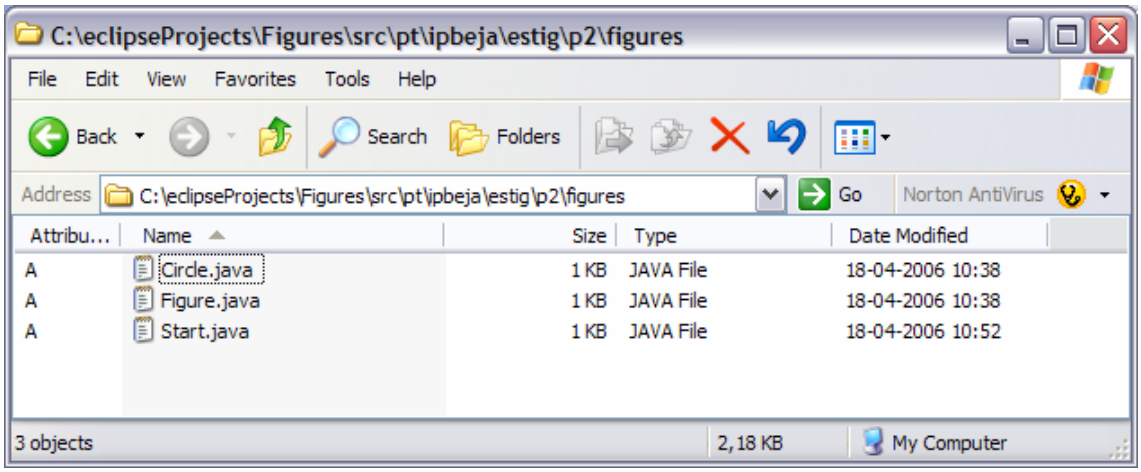
Por fim, vamos adicionar uma classe com um `main` ao nosso programa. Todas as classes podem conter um `main` mas nós vamos criar uma que só contem esse método. Agora vamos também pedir para gerar comentários:



Agora já temos três classes no nosso projecto:



Note que o código fonte das classes criadas (os ficheiros .java) estão dentro das directorias correspondentes à *package* criada. Como a *package* criada foi `pt.ipbeja.estig.p2.figures`, os ficheiros estão dentro da directoria `pt\ipbeja\estig\p2\figures`. Esta, por sua vez, está dentro da directoria `src`, e a directoria `src` está dentro da directoria correspondente ao projecto (`figures`). Finalmente, este projecto está dentro da directoria que definimos, no inicio, como o *workspace* do eclipse:

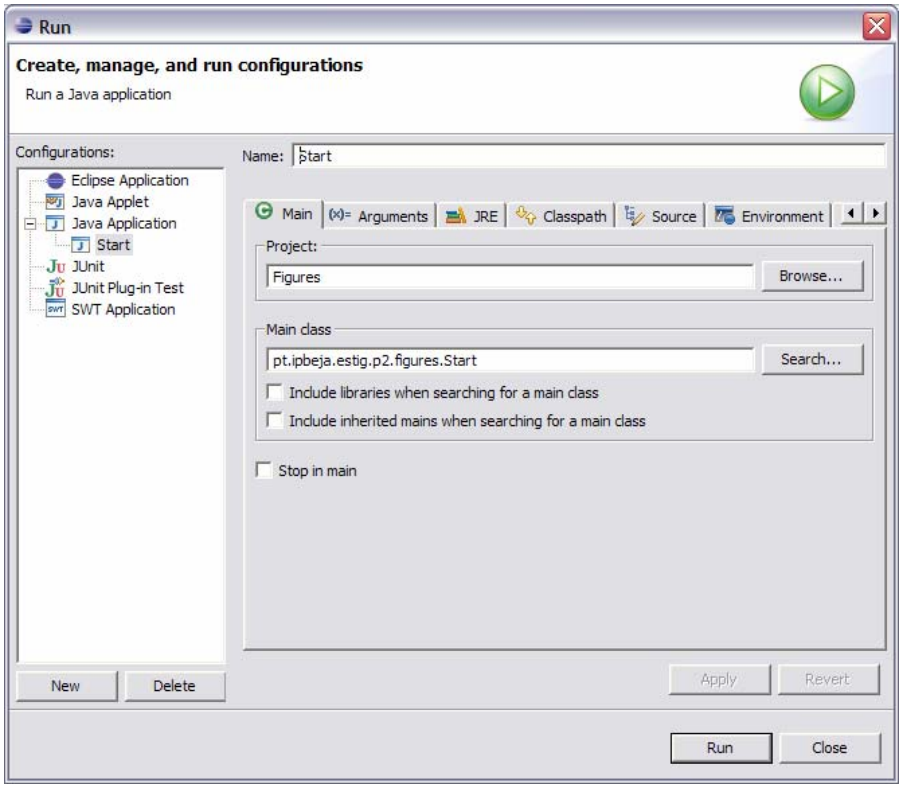


Vejamos agora o método main que foi criado: a frase // TODO Auto-generated method stub dentro do método `main` significa que esse é um sítio para fazer coisas (*to do*).

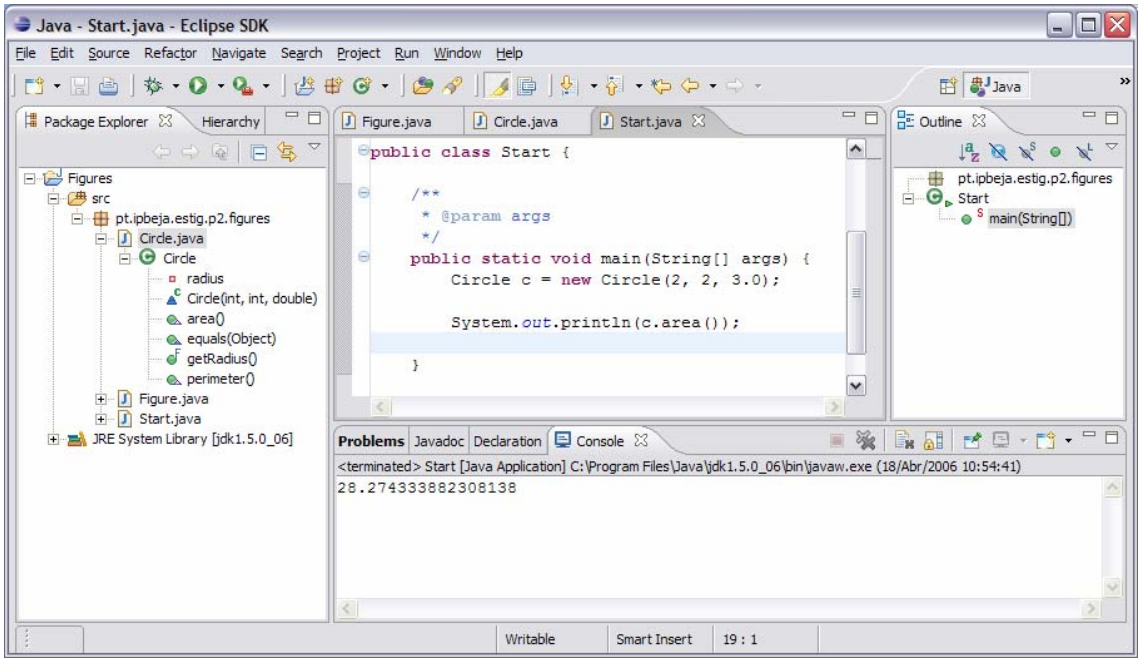
Vamos então fazer algo mínimo só para ver o programa a funcionar:

```
public static void main(String[] args)
{
        Circle c = new Circle(2, 2, 3.0);
         System.out.println(c.area());
}
```

Vamos então seleccionar o ficheiro **`Start.java`** no **Package Explorer** e clicar no botão **Run**. No diálogo que surge, escolhemos **Java Application** na lista de tipos de **Configurations** possíveis (rectângulo à esquerda) e escolher criar uma nova aplicação Java (botão **New**):



Podemos então clicar em **Run**. O resultado deve aparece na *tab* **Console** na *view* inferior da *workbench*:

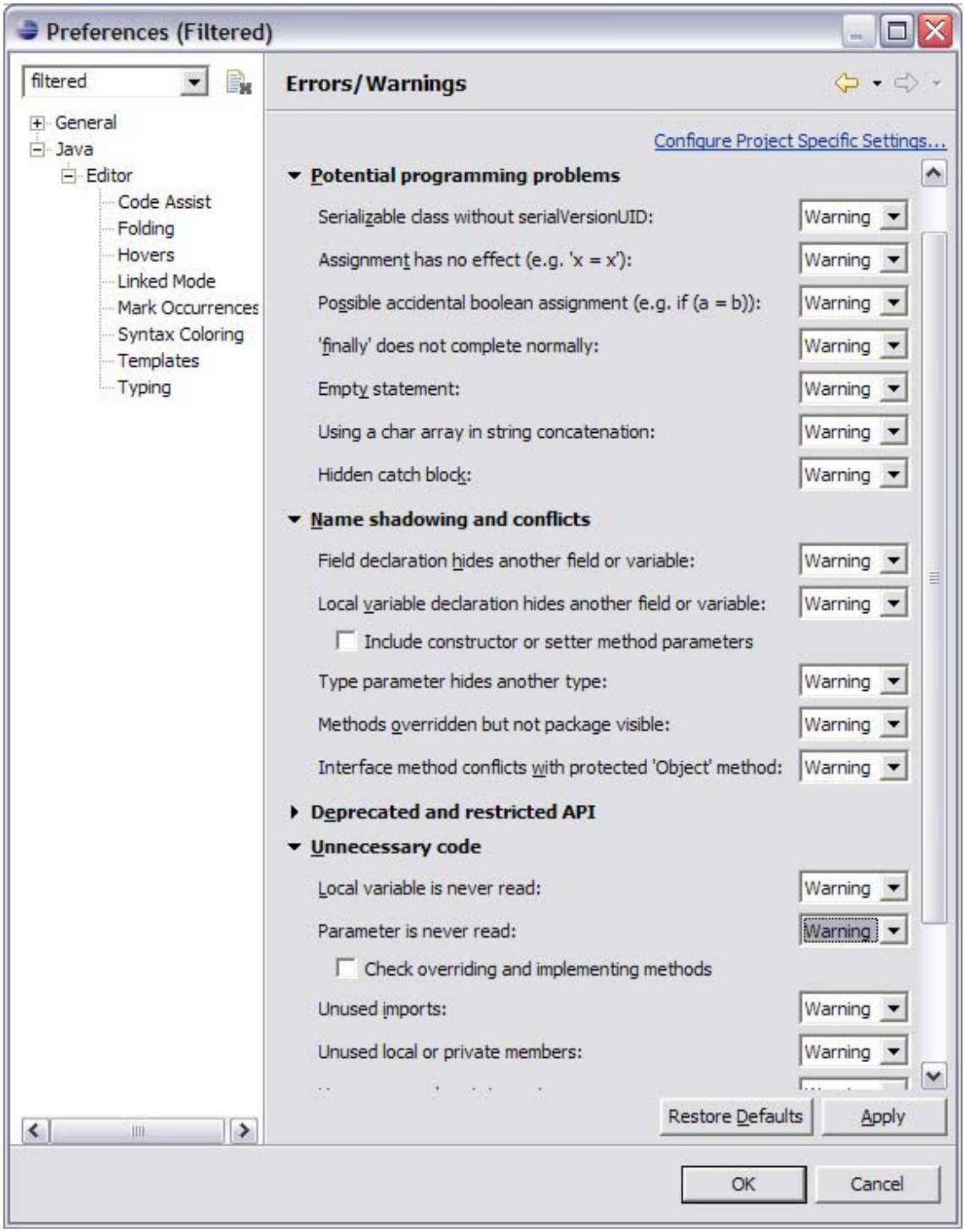


Nas próximas vezes que queiramos executar o programa, basta clicar no botão **Run** e ele irá executar a configuração de execução que está seleccionada: no nosso caso é **Run Start**.

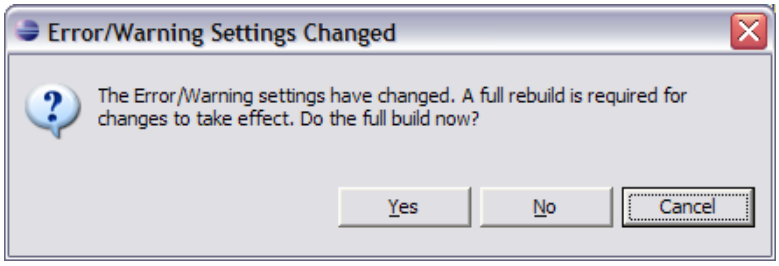
Finalmente, vamos configurar algumas coisas

4 Configuração

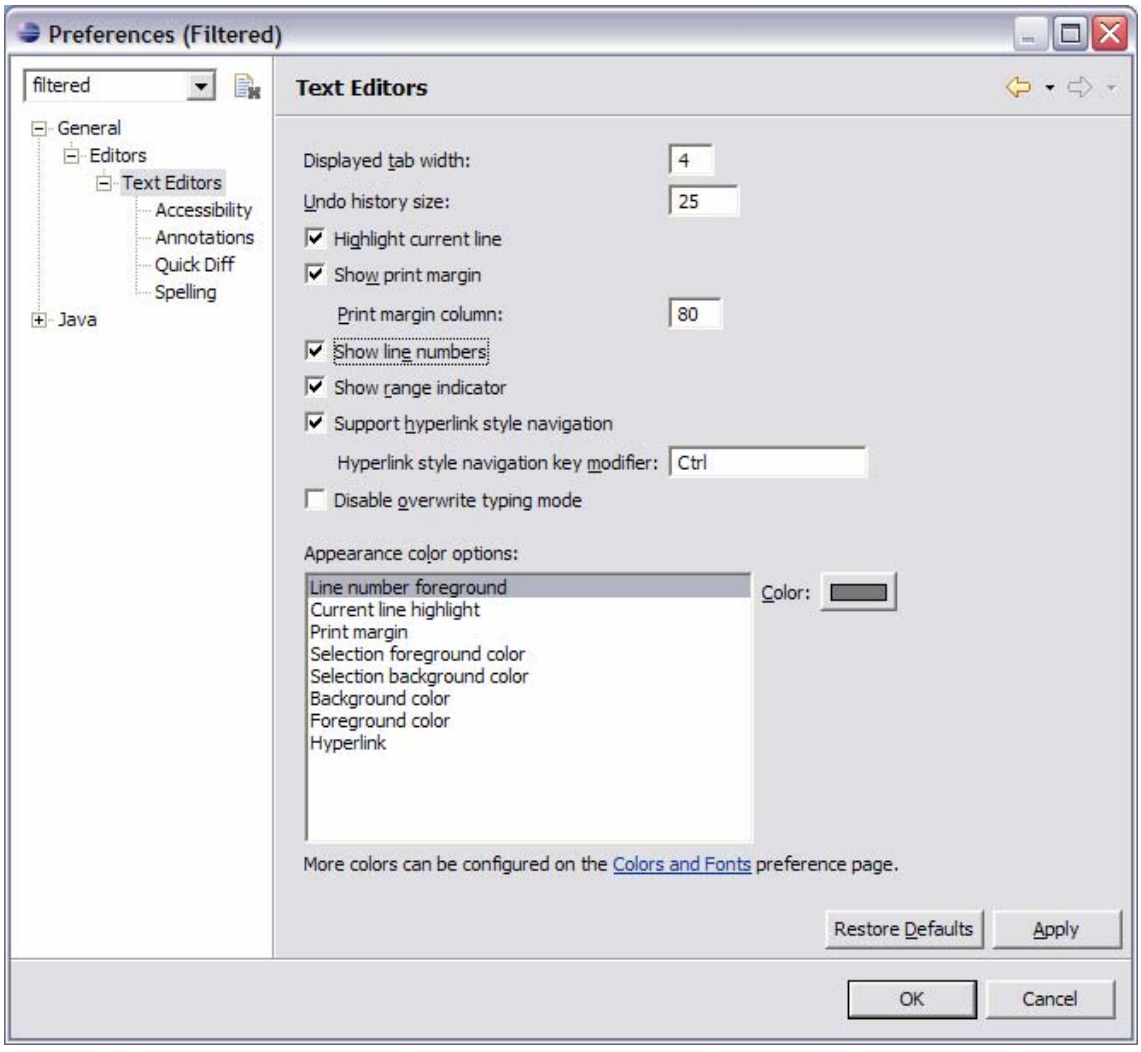
Clique com o botão direito no fundo da janela editor e escolha **Preferences** (última opção no menu de contexto). No rectângulo à esquerda, escolha **Java->Editor**. Seguidamente escolha **compiler warnings** e, no rectângulo da direita. Escolha **Warning** em todas as linhas dentro de **Potential Programming problems**, **Name shadowing and coflicts**, e **Unnecessary code**.



Clique **OK** e escolha **Yes** no diálogo seguinte:



Novamente, clique com o botão direito no fundo da janela editor e escolha **Preferences** (última opção no menu de contexto). No rectângulo à esquerda, escolha **General->Editors->Text Editors** e coloque 3 em **Displayed tab width**. Escolha também a opção **Show print margin** e **Show line numbers**:



Voltando ao editor, repare que surgem os números de linha e uma linha vertical na coluna 80.

Note e experimente ainda as seguintes funcionalidades:

- Se premir F4 quando o cursor está num identificador de classe, será aberta (ou actualizada) a **Hierarchy** view. Experimente, por exemplo, com o identificador `Circle` no método `main`.

- Se premir F3 quando o cursor está num identificador de classe, irá para a definição dessa classe. Experimente, por exemplo, com o identificador `Circle` no método `main`.

- Se premir os sinais – ou + que surgem junto aos números de linha, os blocos (ou grupos) respectivos ficam invisíveis ou visíveis, respectivamente. Experimente com o que está ao lado do cabeçalho do `main` e com outros.

- Quando escreve o nome de um objecto e depois o ponto, note que surge a lista de métodos que compõem a interface desse objecto.

- Experimente as *tabs* **Problems**, **JavaDoc** e **Declaration** na *view* na parte inferior da *workbench*. Com a **Declaration** seleccionada, experimente clicar no identificador `Circle` no editor.

- Para activar os asserts:

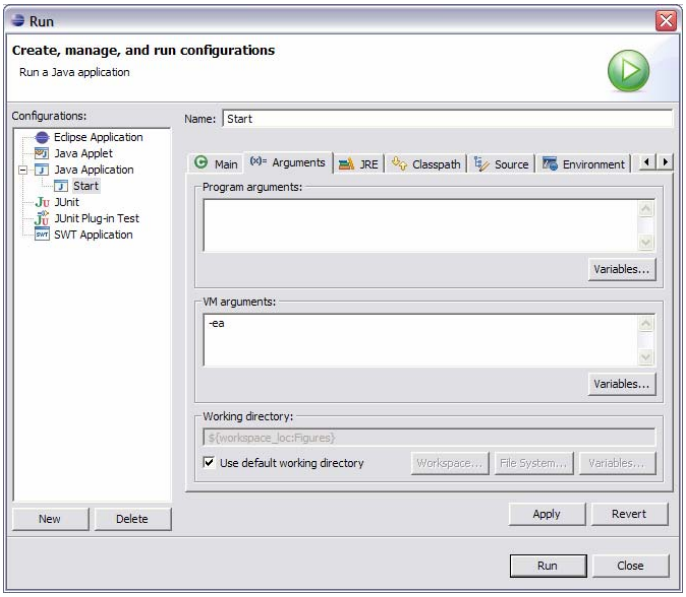
    - Para testar, adicione, por exemplo, `assert(false)` no método `main`.

    - Verifique se está seleccionada a configuração Java Application->Start.

    - Escolha a opção **Run** no menu **Run** (Run->Run).

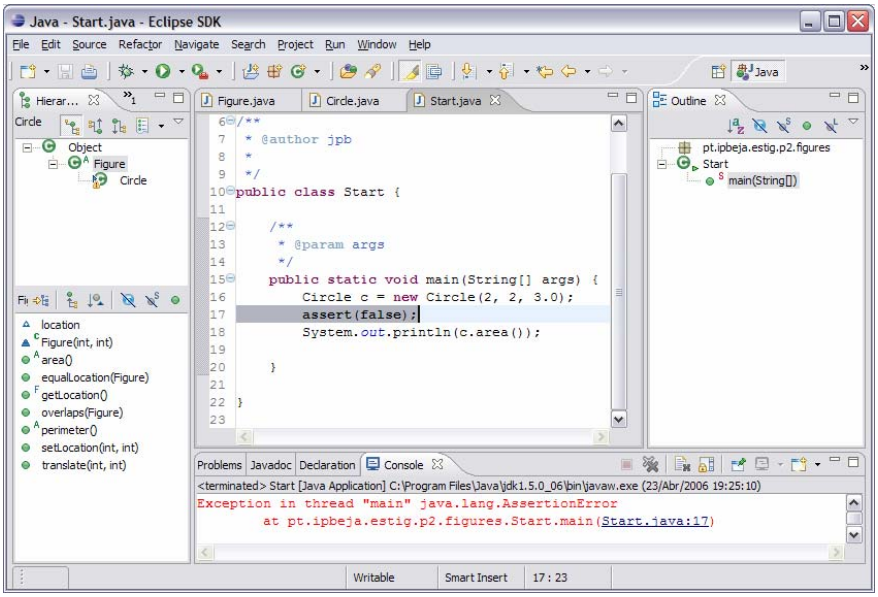
    - Seleccione a tab **Arguments**.

    - No campo de texto **VM arguments** (argumentos para a máquina virtual Java) adicione o texto "**`-ea`**".



- Clique no botão **Run**.

- Deverá obter um erro na tab **Console**. Clicando no nome do ficheiro e número de linha no final do ficheiro o editor irá para esse local:



Leitura complementar

O seguinte *link* talvez seja o melhor para uma primeira leitura sobre o que é o eclipse:

http://wiki.eclipse.org/index.php/Eclipse_FAQs

Existe imensa documentação sobre o eclipse e os seus vários constituintes. O ponto de entrada principal é o seguinte:

http://help.eclipse.org/help31/index.jsp

Tem comentários a fazer a este texto? O que é que gostou mais? E menos? E do que é que não gostou mesmo nada? Os seus eventuais comentários podem ser muito úteis para ajudar a melhorar futuras edições deste texto. Por si e também pelos seus presentes e futuros colegas, não se acanhe! Diga coisas para jpb AT estig.ipbeja.pt.

Primeira versão: Beja, 18 de Abril de 2006. Última revisão em 23 de Abril de 2006 às 19h:38m.
Autor: João Paulo Barros
Página pessoal: http://www.estig.ipbeja.pt/~jpb