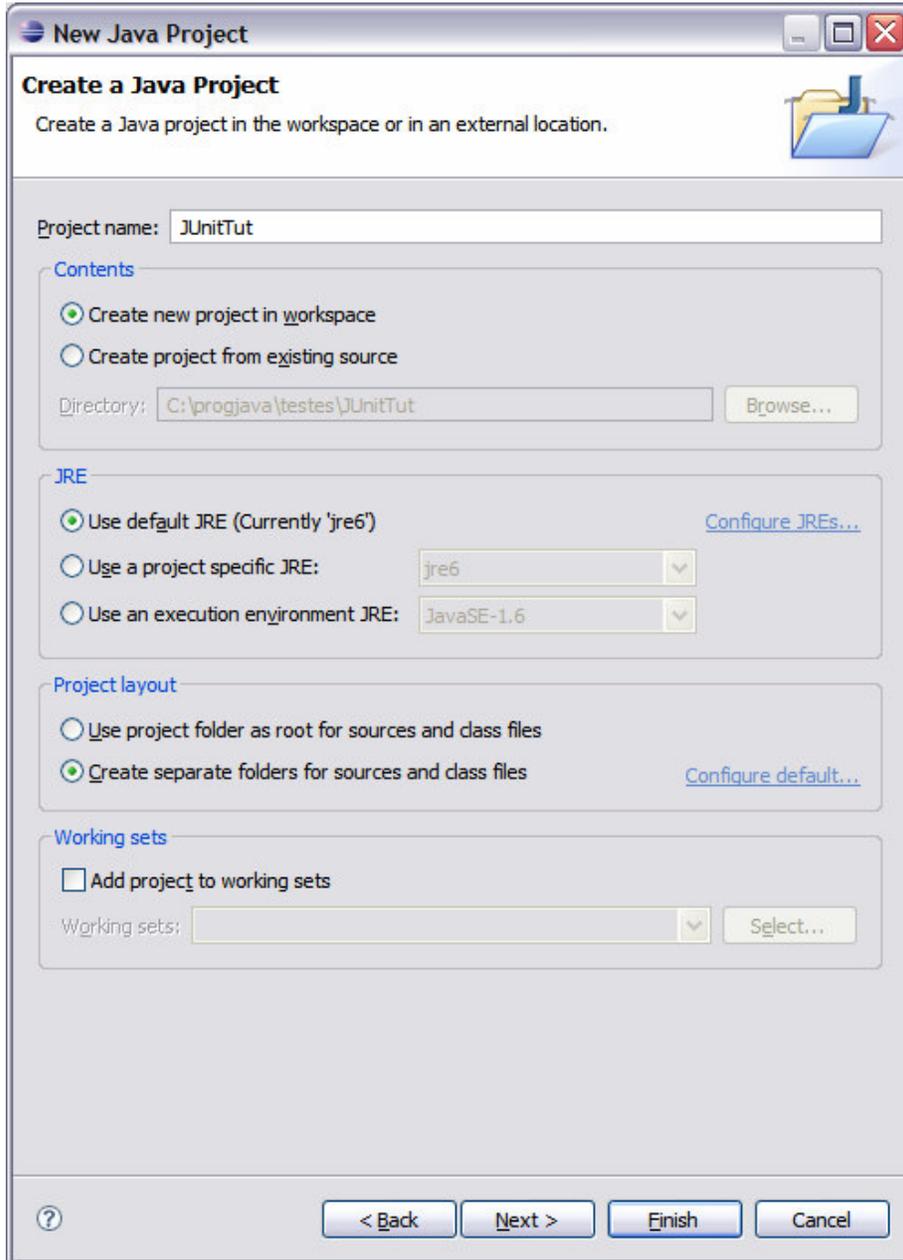


1) Para uma introdução a conceitos do JUnit consulte <http://junit.sourceforge.net/doc/cookbook/cookbook.htm>

2) Criar um projeto no Eclipse. Vá a File-> New->Project. Selecione na árvore Java->Java Project e pressione Next. Dê ao projeto o nome de JUnitTut (ver tela abaixo). Pressione Finish. Responda não à questão que se apresenta em seguida.



3) Criar a seguinte classe no Eclipse:

```
public class Maior {  
    public static int maior(int[] list) {  
        int index, max=Integer.MAX_VALUE;  
        for (index = 0; index < list.length-1; index++){  
            if (list[index] > max){  
                max = list[index];  
            }  
        }  
        return max;  
    }  
}
```

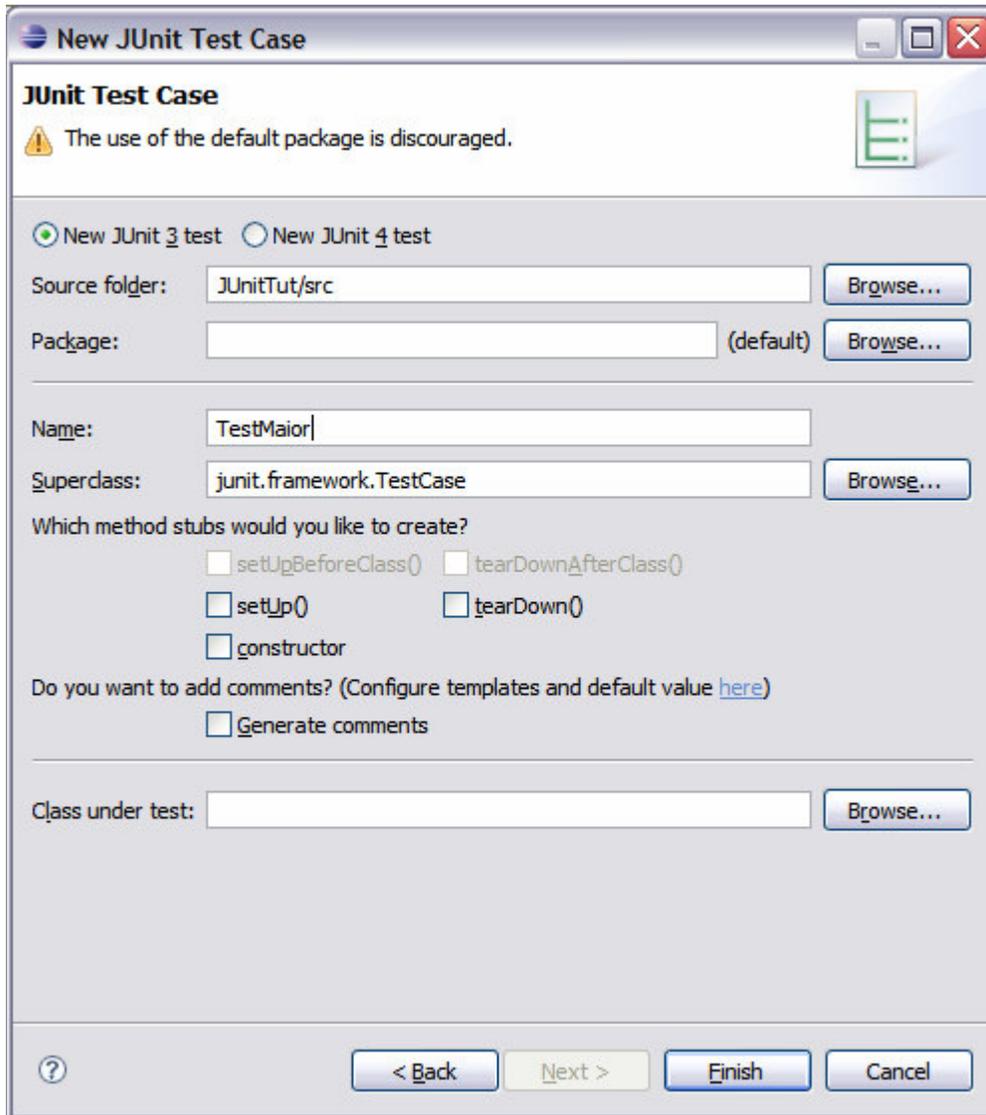
Para criar uma classe vá para o painel do lado esquerdo, clique com o botão direito em JUnitTut->src e selecione new->class. Preencha o formulário conforme abaixo e pressione next. Depois cole na classe gerada o método maior exibido acima.

The screenshot shows the 'New Java Class' dialog box in Eclipse. The title bar reads 'New Java Class'. Below the title bar, there is a warning icon and the text 'The use of the default package is discouraged.' followed by the Eclipse logo. The dialog is divided into several sections:

- Source folder:** JUnitTut/src (with a 'Browse...' button)
- Package:** (empty) (default) (with a 'Browse...' button)
- Enclosing type:** (empty) (with a 'Browse...' button)
- Name:** Maior
- Modifiers:** public, default, private, protected, abstract, final, static
- Superclass:** java.lang.Object (with a 'Browse...' button)
- Interfaces:** (empty list) (with 'Add...' and 'Remove' buttons)
- Which method stubs would you like to create?**
 - public static void main(String[] args)
 - Constructors from superclass
 - Inherited abstract methods
- Do you want to add comments? (Configure templates and default value [here](#))**
 - Generate comments

At the bottom of the dialog, there are 'Finish' and 'Cancel' buttons, and a help icon (?) on the left.

3) Vá a File->New->Other. Depois ramo da árvore Java->JUnit->JUnit Test Case. Pressione Next. Nomeie o JUnit Test Case como TestMaior. Pressione Finish. Pressione Ok na próxima janela.



4) À classe TestMaior que você acabou de criar, adicione o método abaixo:

```
public void testSimples() {  
    assertEquals(9, Maior.maior(new int[] {7,8,9}));  
}
```

5) Execute o caso de teste pressionando  e selecionando a opção Junit test unit. Você verá na janela no canto inferior esquerdo a seguinte mensagem:

```
Failure Trace
junit.framework.AssertionFailedError: expected: <9> but was: <2147483647>
at TestMaior.testSimple(TestMaior.java:6)
```

6) Procure identificar a origem do erro acima e faça a correção.

7) Vamos verificar agora, se o programa funciona quando alteramos a ordem dos elementos da lista. Acrescente à classe TestMaior o seguinte método:

```
public void testOrdem() {
    assertEquals(9, Maior.maior(new int[] {9, 8, 7}));
    assertEquals(9, Maior.maior(new int[] {7, 9, 8}));
    assertEquals(9, Maior.maior(new int[] {7, 8, 9}));
}
```

8) Testar duplicados. Acrescente mais este método:

```
public void testDups() {
    assertEquals(9, Maior.maior(new int[] {9, 7, 9, 8}));
}
```

9) Testar lista com um elemento:

```
public void testUm() {
    assertEquals(1, Maior.maior(new int[] {1}));
}
```

10) Testar lista com negativo:

```
public void testNegativo() {
    int [] negList = new int[] {-9, -8, -7};
    assertEquals(-7, Maior.maior(negList));
}
```

Provavelmente ocorreu um erro. Corrija a origem do erro.

11) Vamos testar listas vazias agora. Acrescente o seguinte código à classe Maior, após a inicialização da variável max:

```
if (list.length == 0) { throw new RuntimeException("Empty list");
}
```

12) Agora acrescente o seguinte método à classe TestMaior:

```
public void testVazio() {
    try {
        Maior.maior(new int[] {});
        fail("Deveria ter emitido exceção");
    } catch (RuntimeException e)
    { assertTrue(true); }
}
```

12) Crie agora uma classe chamada Menor e implemente sua lógica (semelhante à de maior).

13) Crie uma JUnit test case denominada TestMenor. Acrescente a ela os mesmos testes feitos para a classe Maior substituindo Maior por Menor.

14) Clique com o botão direito sobre JUnitTut-> source, depois java->JUnit-> JUnit Test Suíte. Dê o nome TestaMaiorMenor à Test Suíte. Assegure-se de selecionar as classes Maior e Menor. Clique finish.

15) Execute agora a suíte de testes. Corrija todos os erros gerados.

Exercício

Crie uma classe denominada Triangulo. Crie uma classe TestaTri com um método denominado TestaTriangulo1 que recebe o triângulo como parâmetro e retorna uma string com a classificação do mesmo (equilátero, isósceles, escaleno ou não é triângulo). Crie uma segunda classe TestaTri2 que possui um método denominado TestaTriangulo2 que verifica se um triângulo é retângulo. Crie uma suíte de testes no JUnit para testar suas classes.