

Eclipse IDE

Mestre Amilcar Borrás González

2020

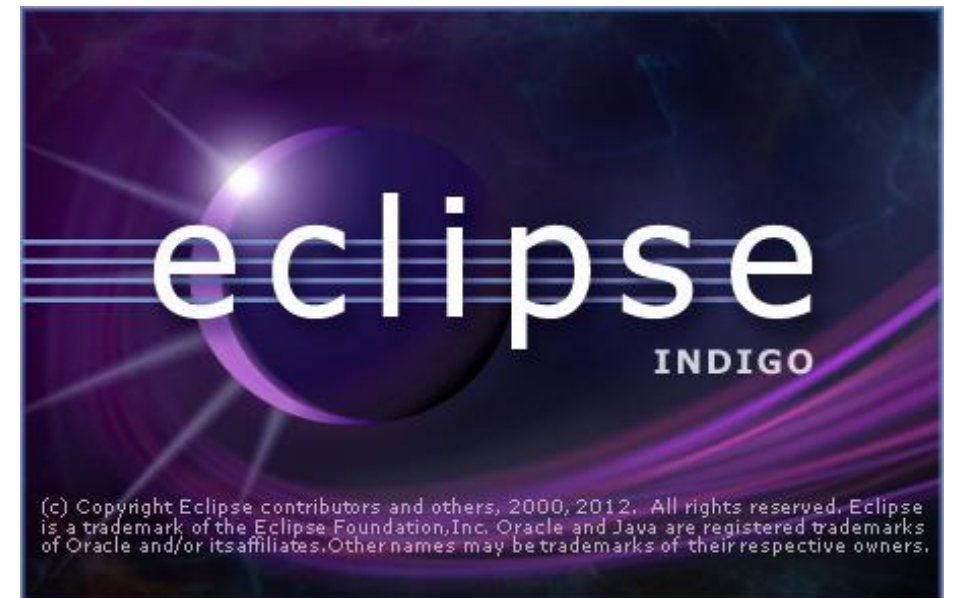
Eclipse

O Eclipse é um **IDE** (Ambiente de Desenvolvimento Integrado) que serve para desenvolver aplicativos.

Permite a realização de múltiplas tarefas de desenvolvimento no mesmo ambiente.

Existem diferentes versões de Eclipse, cada uma de elas para o desenvolvimento de aplicativos numa linguagem de programação diferente.

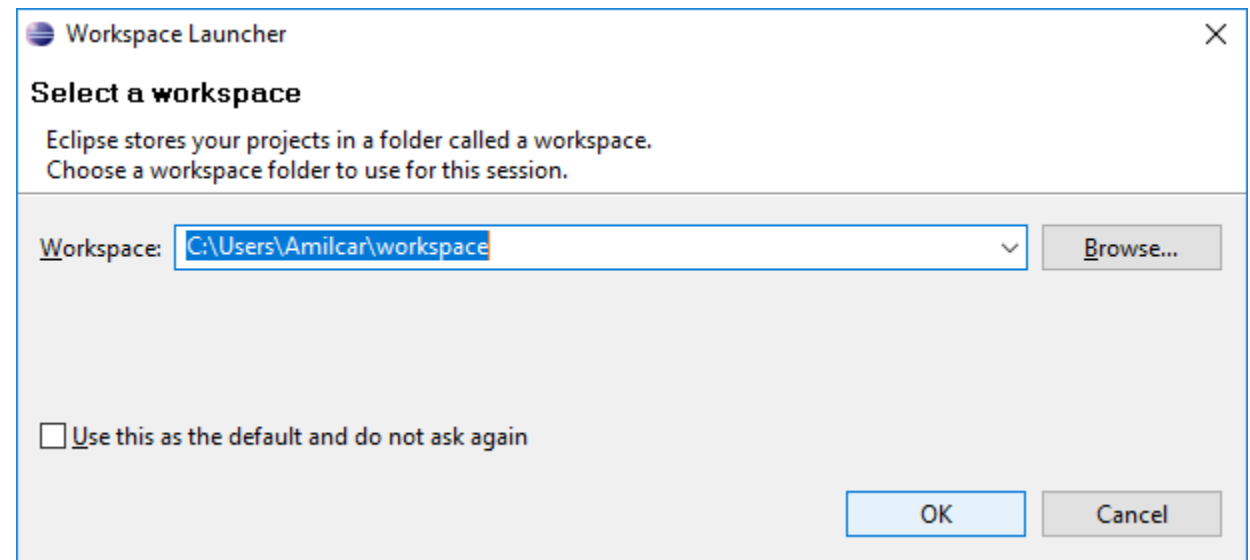
Desenvolvido sobre Java, precisamos ter Java instalado.



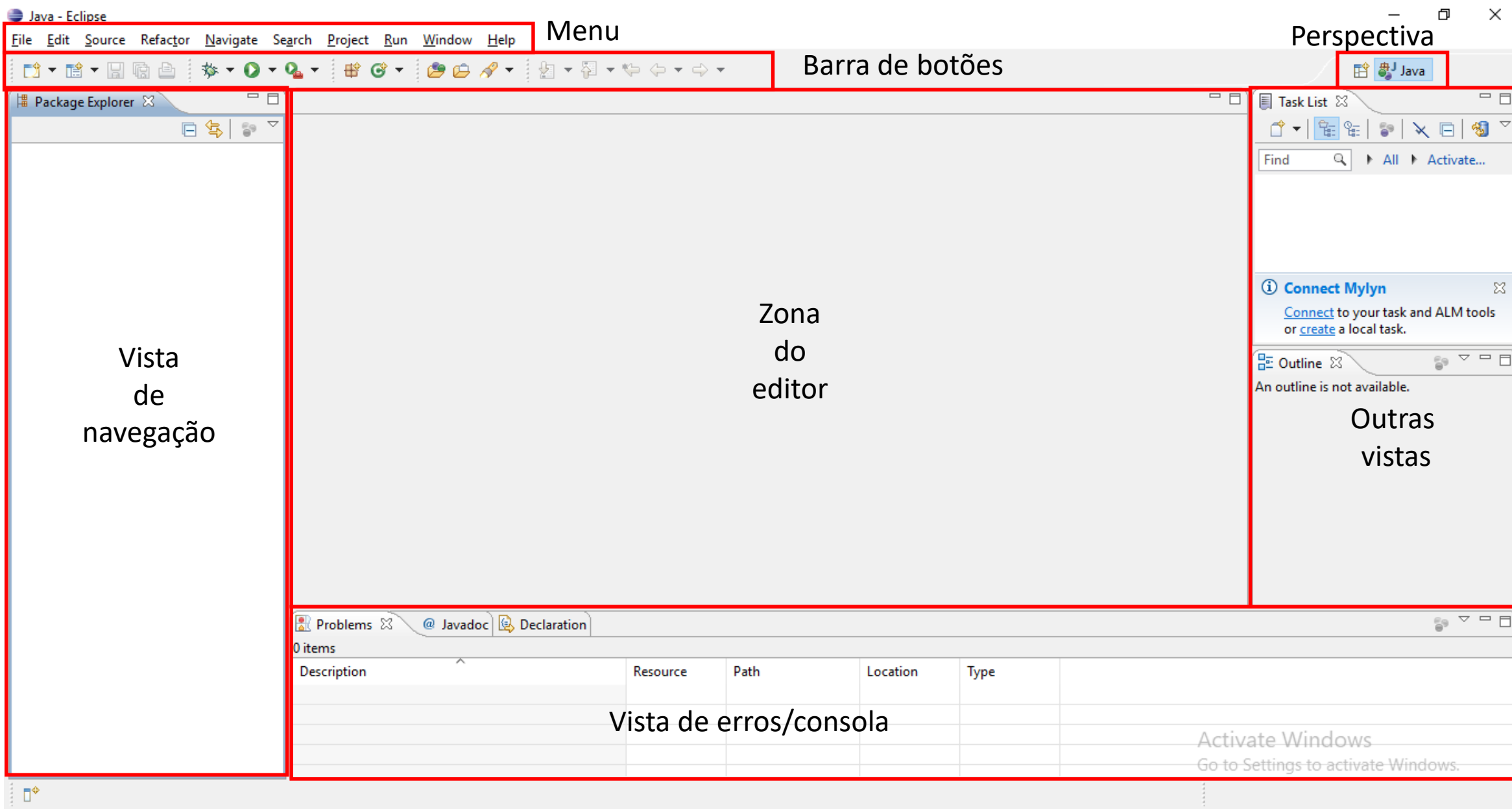
Eclipse workspace

Workspace (espaço de trabalho)

- É o espaço físico onde você está trabalhando, o espaço em disco onde tudo que fizer será armazenado.
- É o repositório/local dos projetos construídos pelo desenvolvedor.
- Podemos ter vários workspaces.

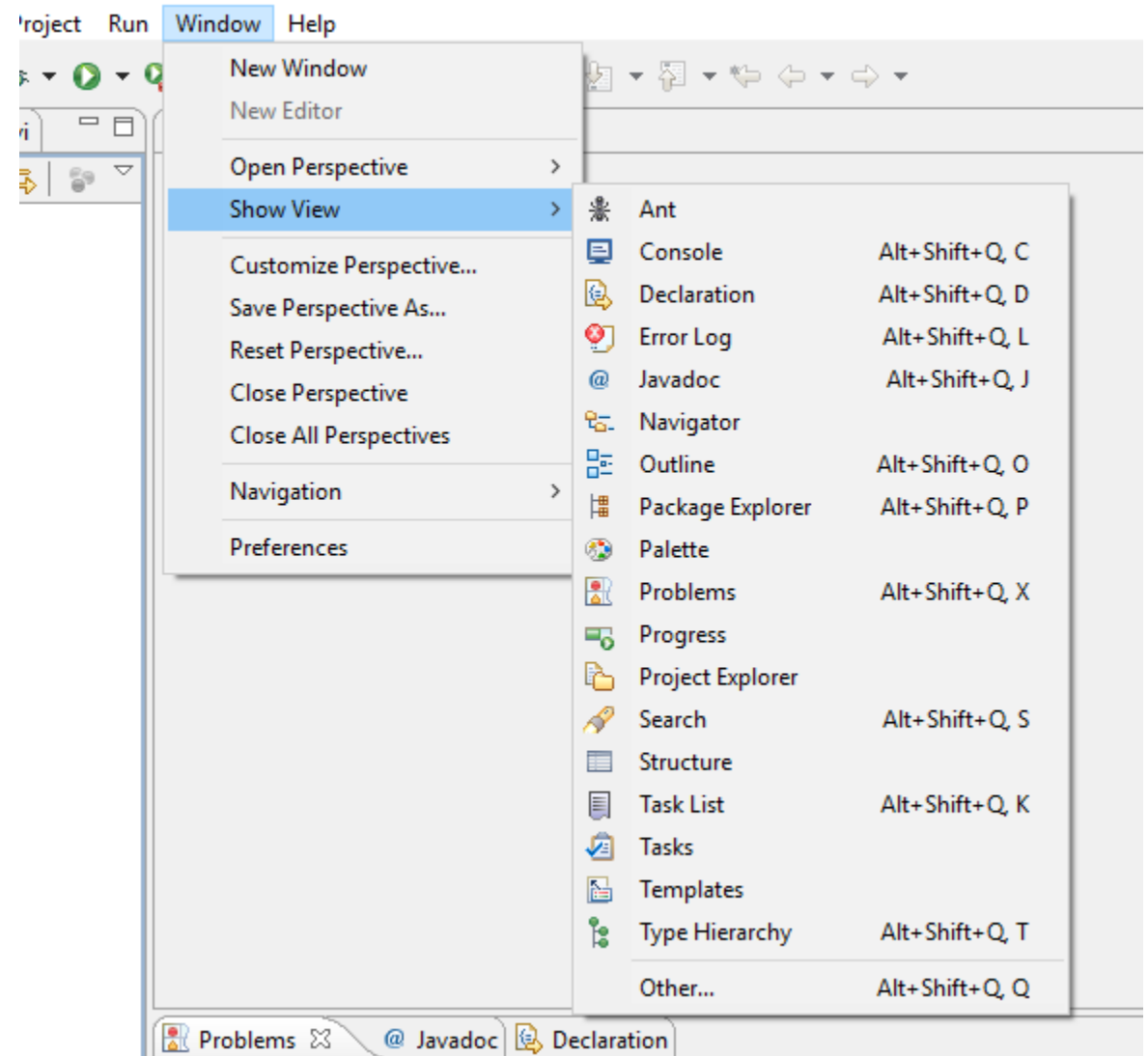


Eclipse IDE



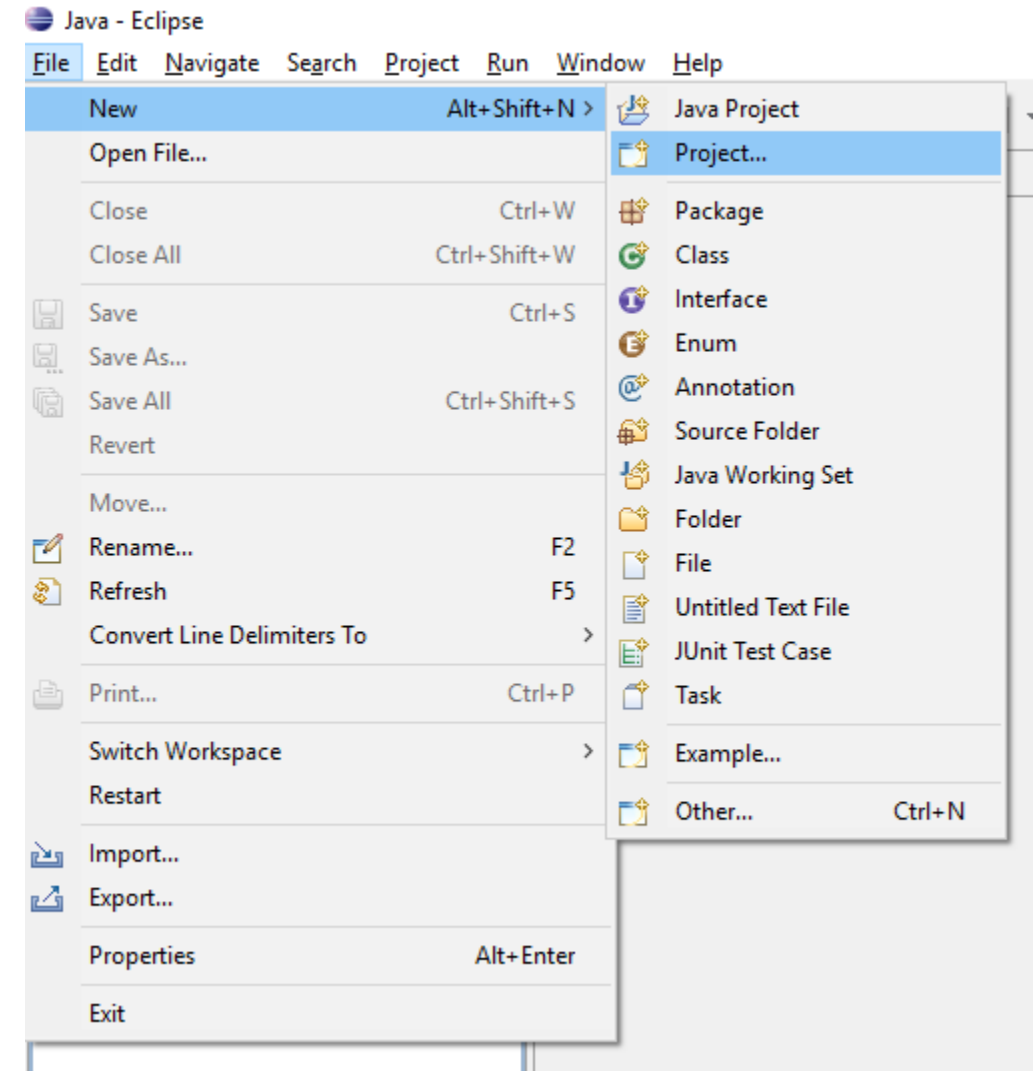
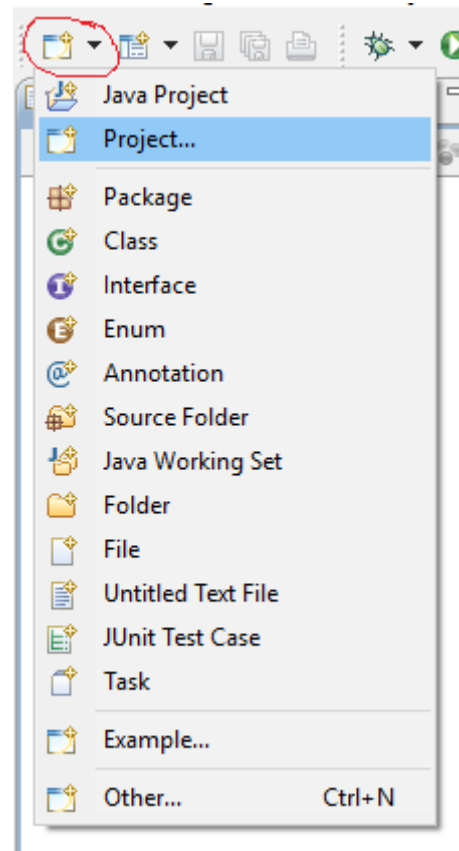
Vistas/Views de Eclipse

Podemos ter diversas views no nosso entorno de trabalho. Basta escolher as que queremos que apareçam ou as que queremos ocultar.

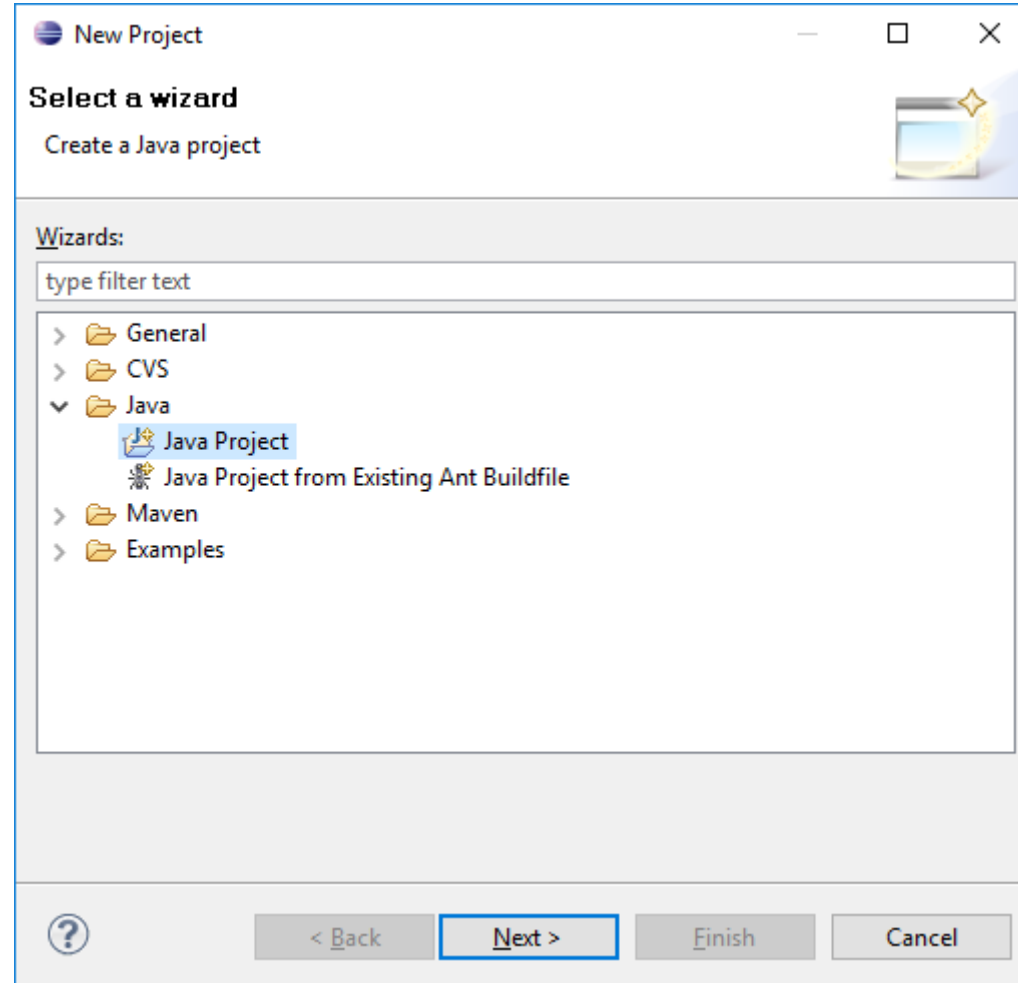


Projectos em Eclipse

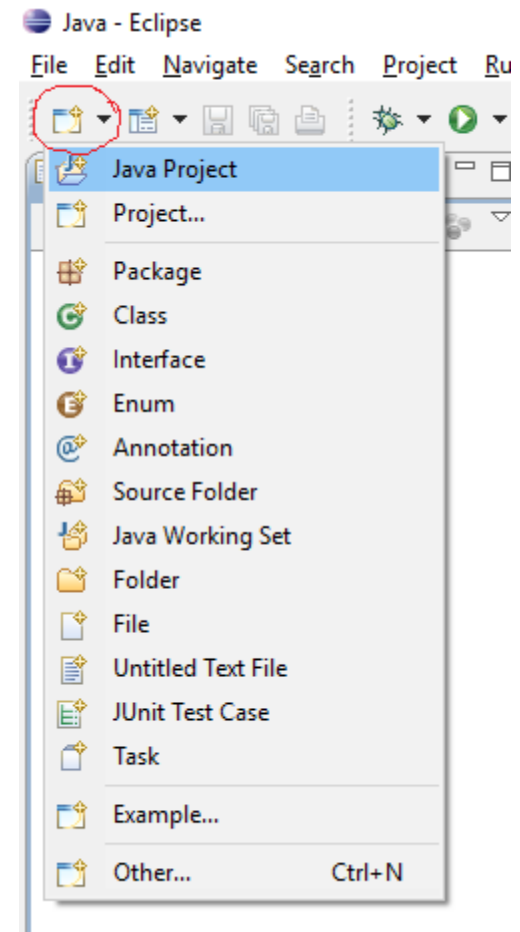
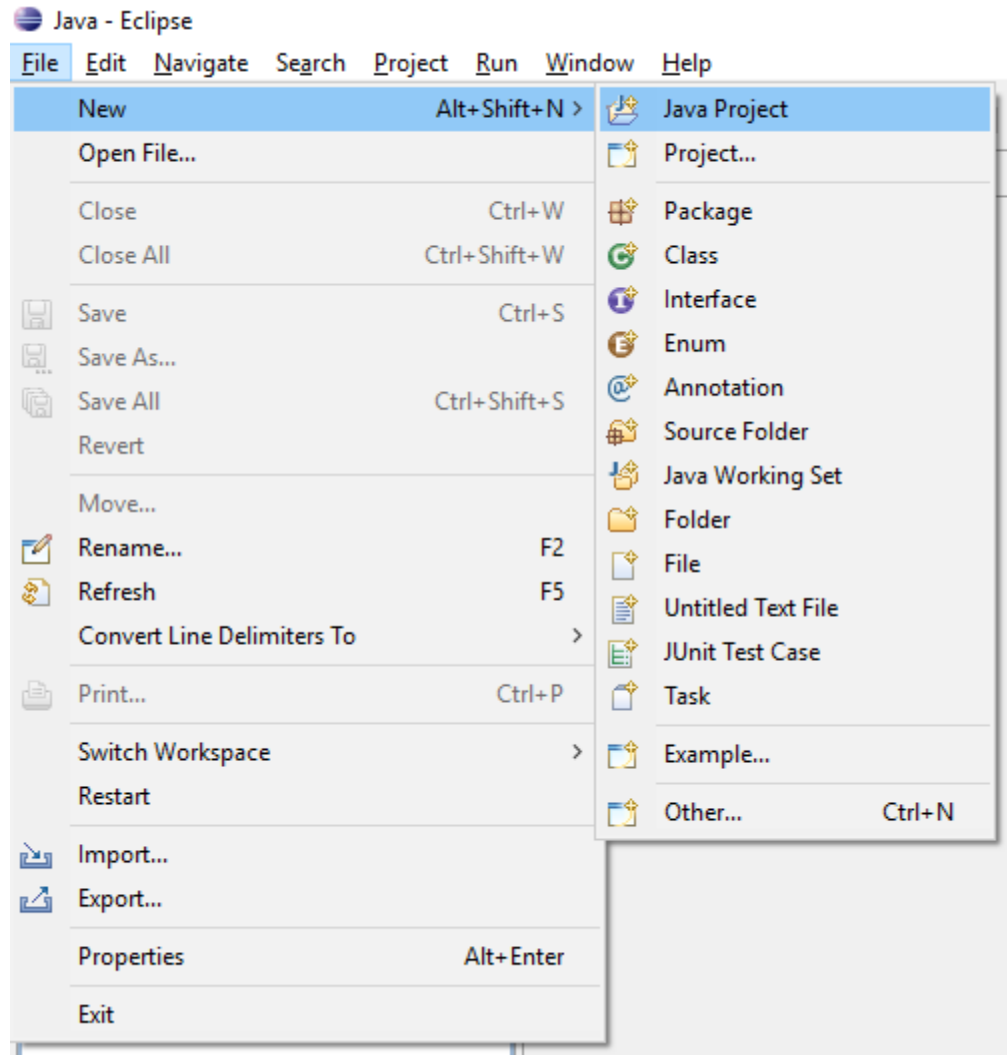
- Podemos criar projectos de 2 formas:



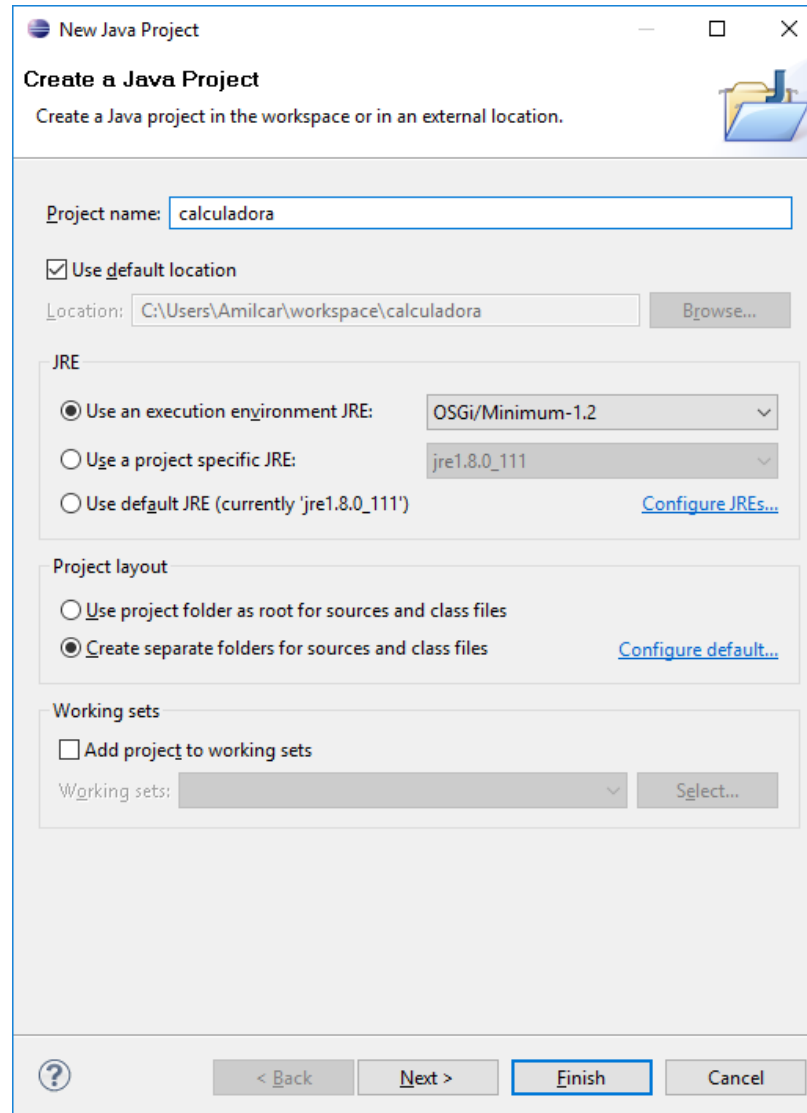
Projectos em Eclipse (cont.)



Projectos em Eclipse (cont.)



Projectos em Eclipse (cont.)



The screenshot shows the 'New Java Project' dialog box in Eclipse. The title bar says 'New Java Project'. Below the title bar, it says 'Create a Java Project' and 'Create a Java project in the workspace or in an external location.' There is a folder icon on the right. The 'Project name' field contains 'calculadora'. The 'Use default location' checkbox is checked. The 'Location' field shows 'C:\Users\Amilcar\workspace\calculadora' with a 'Browse...' button. The 'JRE' section has three radio buttons: 'Use an execution environment JRE:' (selected), 'Use a project specific JRE:', and 'Use default JRE (currently 'jre1.8.0_111')'. The first option has a dropdown menu showing 'OSGi/Minimum-1.2'. The second option has a dropdown menu showing 'jre1.8.0_111'. There is a 'Configure JREs...' link. The 'Project layout' section has two radio buttons: 'Use project folder as root for sources and class files' and 'Create separate folders for sources and class files' (selected). There is a 'Configure default...' link. The 'Working sets' section has a checkbox 'Add project to working sets' which is unchecked. Below it is a 'Working sets:' dropdown menu and a 'Select...' button. At the bottom, there are buttons for '< Back', 'Next >', 'Finish' (highlighted), and 'Cancel'.

New Java Project

Create a Java Project

Create a Java project in the workspace or in an external location.

Project name: calculadora

☒ Use default location

Location: C:\Users\Amilcar\workspace\calculadora [Browse...](#)

JRE

☒ Use an execution environment JRE: OSGi/Minimum-1.2

☐ Use a project specific JRE: jre1.8.0_111

☐ Use default JRE (currently 'jre1.8.0_111') [Configure JREs...](#)

Project layout

☐ Use project folder as root for sources and class files

☒ Create separate folders for sources and class files [Configure default...](#)

Working sets

☐ Add project to working sets

Working sets: [Select...](#)

[?](#) < Back Next > Finish Cancel

The screenshot shows the Eclipse IDE interface with a Java project named "calculadora". The left sidebar (Project Explorer) shows the project structure: "calculadora" (expanded) contains a "src" folder and a "JRE System Library [jre1.8.0_111]". The main editor area is empty. The right sidebar contains two views: "Task List" and "Outline". The "Task List" view shows a search bar and a "Connect Mylyn" button. The "Outline" view shows the message "An outline is not available." The bottom status bar shows the "Problems" view with "0 items" and a table with columns: Description, Resource, Path, Location, Type. A watermark "Activate Windows" is visible in the bottom right corner.

Proj Pack Navi

calculadora

- src
- JRE System Library [jre1.8.0_111]

Task List

Find

Connect Mylyn

Connect to your task and ALM tools or create a local task.

Outline

An outline is not available.

Problems

0 items

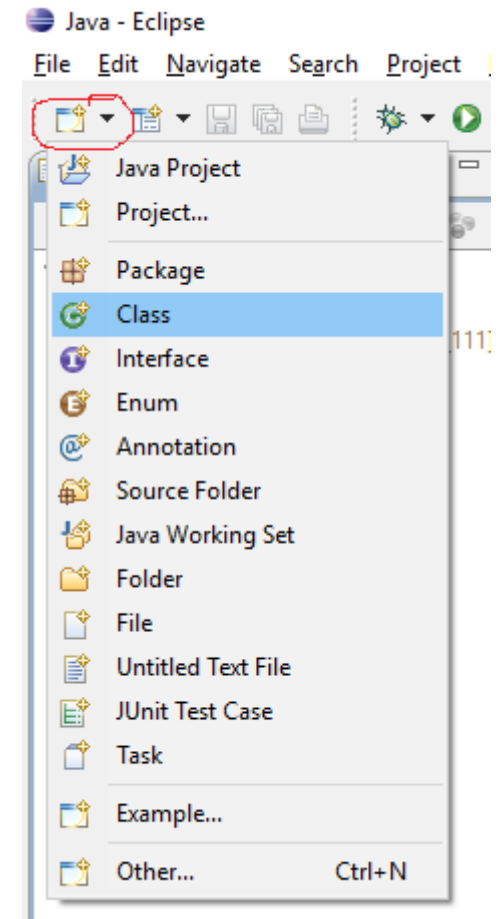
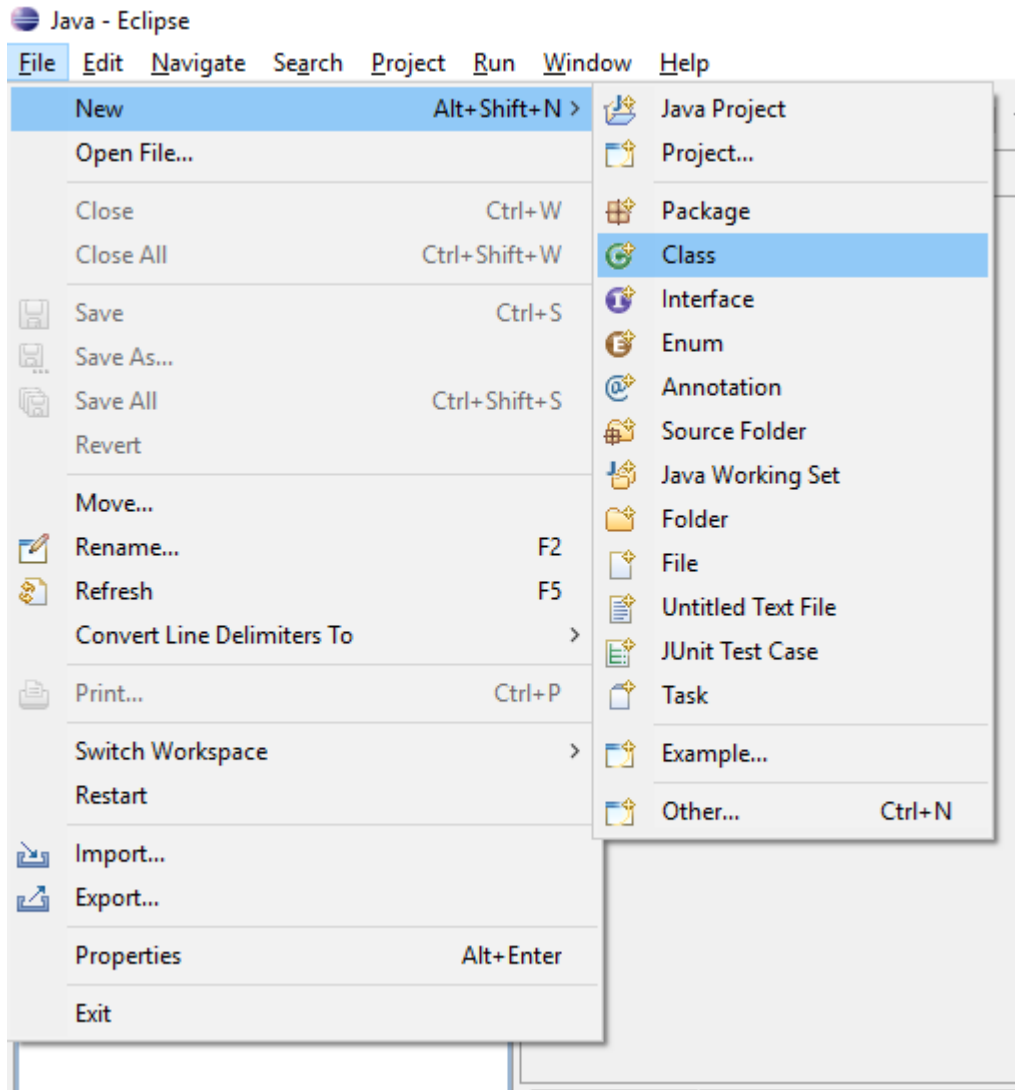
Description	Resource	Path	Location	Type

calculadora

Activate Windows

Go to Settings to activate Windows.

Criar uma classe



Criar uma classe (cont.)

New Java Class

Java Class

Create a new Java class.

Source folder: calculadora/src Browse...

Package: (default) Browse...

☐ Enclosing type: Browse...

Name:

Modifiers: ☒ public ☐ default ☐ private ☐ protected
☐ abstract ☐ final ☐ static

Superclass: java.lang.Object Browse...

Interfaces: Add... Remove

Which method stub 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

Finish Cancel

main é o ponto de entrada para
execução do código em Java

New Java Class

Java Class

The use of the default package is discouraged.

Source folder: calculadora/src Browse...

Package: (default) Browse...

☐ Enclosing type: Browse...

Name: teste

Modifiers: ☒ public ☐ default ☐ private ☐ protected
☐ abstract ☐ final ☐ static

Superclass: java.lang.Object Browse...

Interfaces: Add... Remove

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

Finish Cancel



Proj Pa teste.java

calculadora

src

(default

teste

JRE System

```
public class teste {
```

```
    /**
```

```
     * @param args
```

```
     */
```

```
    public static void main(String[] args) {
```

```
        // TODO Auto-generated method stub
```

```
    }
```

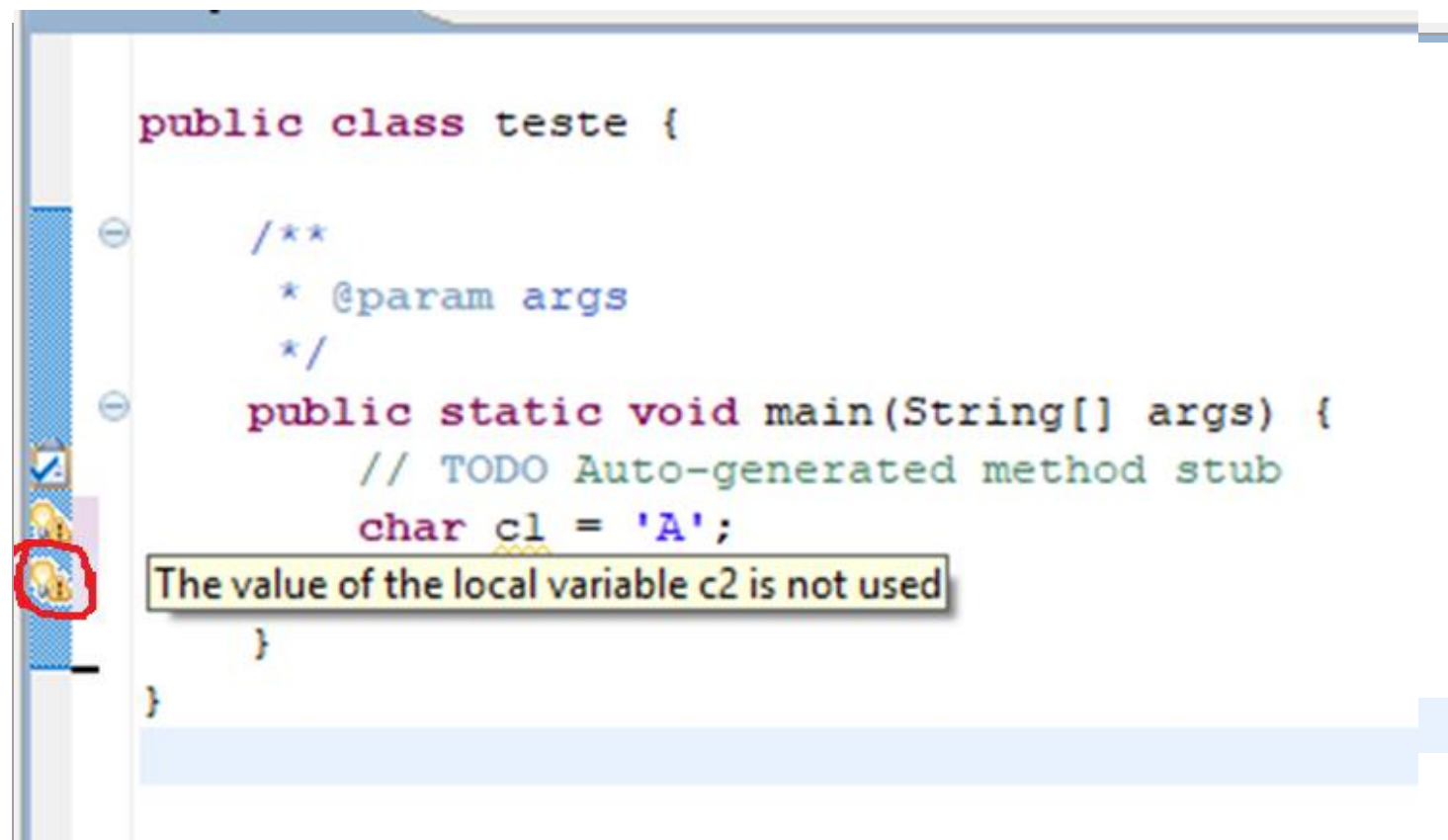
```
}
```

M tools

id

Declarar/definir variaveis

- char c1 = 'A';
- char c2 = 65;



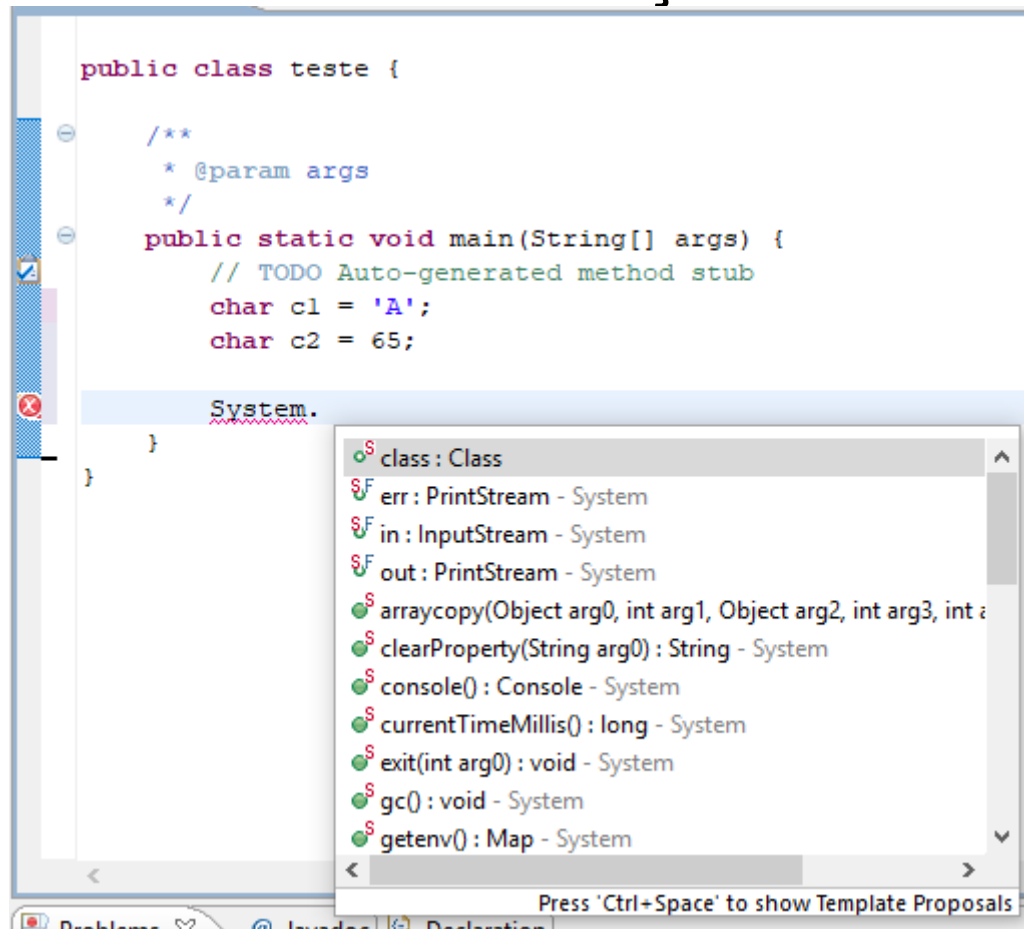
The screenshot shows a code editor with the following Java code:

```
public class teste {  
  
    /**  
     * @param args  
     */  
    public static void main(String[] args) {  
        // TODO Auto-generated method stub  
        char c1 = 'A';  
        char c2 = 65;  
    }  
}
```

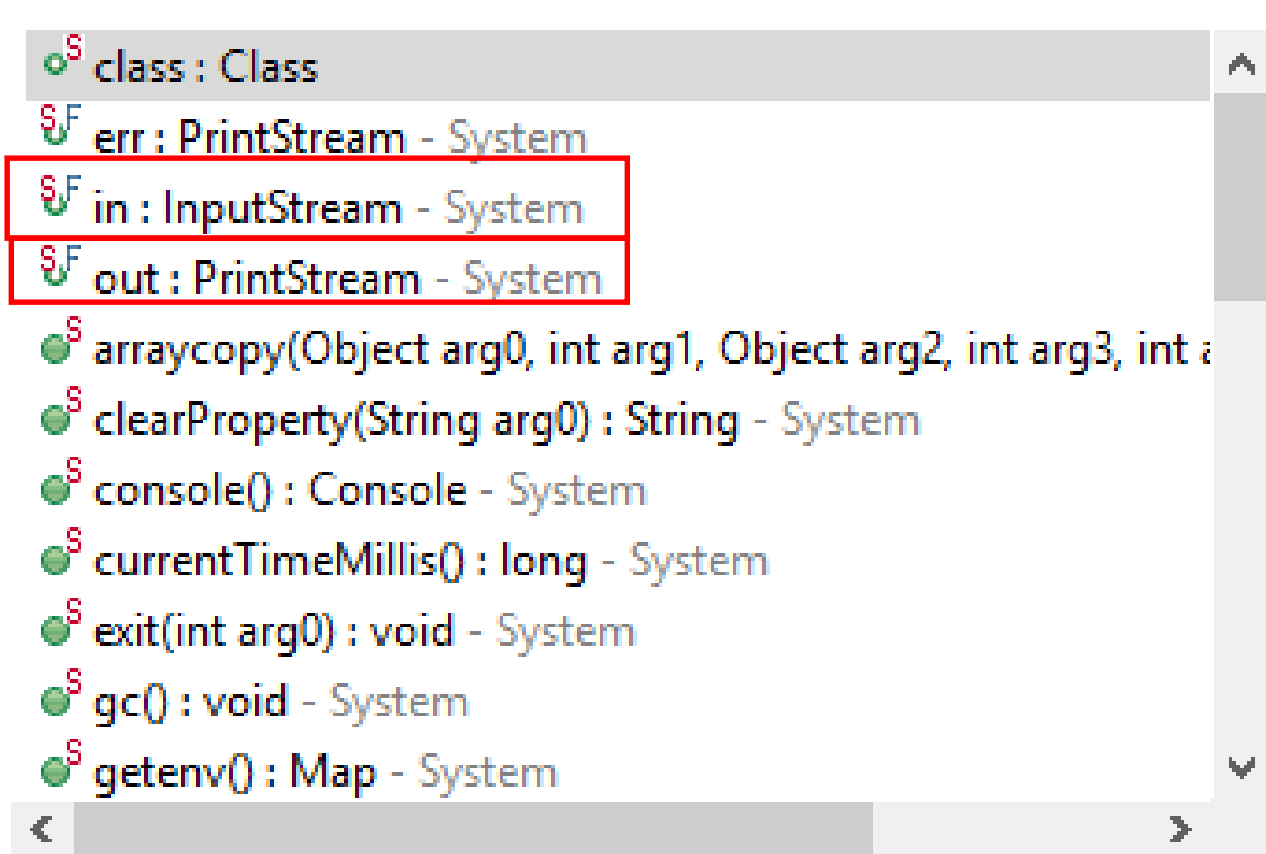
A warning icon (a yellow lightbulb with a red exclamation mark) is visible in the left margin, pointing to the line `char c2 = 65;`. A tooltip box displays the message: "The value of the local variable c2 is not used".

Mostrar o valor armazenado nas variaveis

- Para mostrar informação na tela: `System.out.println`



Mostrar o valor armazenado nas variaveis



```
class : Class
err : PrintStream - System
in : InputStream - System
out : PrintStream - System
arraycopy(Object arg0, int arg1, Object arg2, int arg3, int a
clearProperty(String arg0) : String - System
console() : Console - System
currentTimeMillis() : long - System
exit(int arg0) : void - System
gc() : void - System
getenv() : Map - System
```

Para ler informação do teclado
Para mostrar informação na tela

Mostar informação na tela

- `println() : void - PrintStream`
- `println(boolean arg0) : void - PrintStream`
- `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`
- `println(String arg0) : void - PrintStream`

Temos a mesma
função ou utilidade
para diferentes tipos
de dados

Mostar informação na tela (cont.)

```
public class teste {
```

```
    /**
```

```
     * @param args
```

```
     */
```

```
    public static void main(String[] args) {
```

```
        // TODO Auto-generated method stub
```

```
        char c1 = 'A';
```

```
        char c2 = 65;
```

```
    }
```

```
}
```

```
id main(String[] args) {
```

```
-generated method stub
```

```
;
```

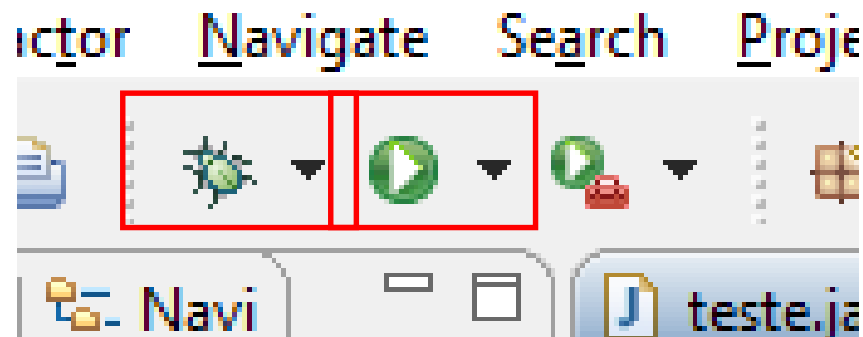
```
;
```

```
rintln(c1);
```

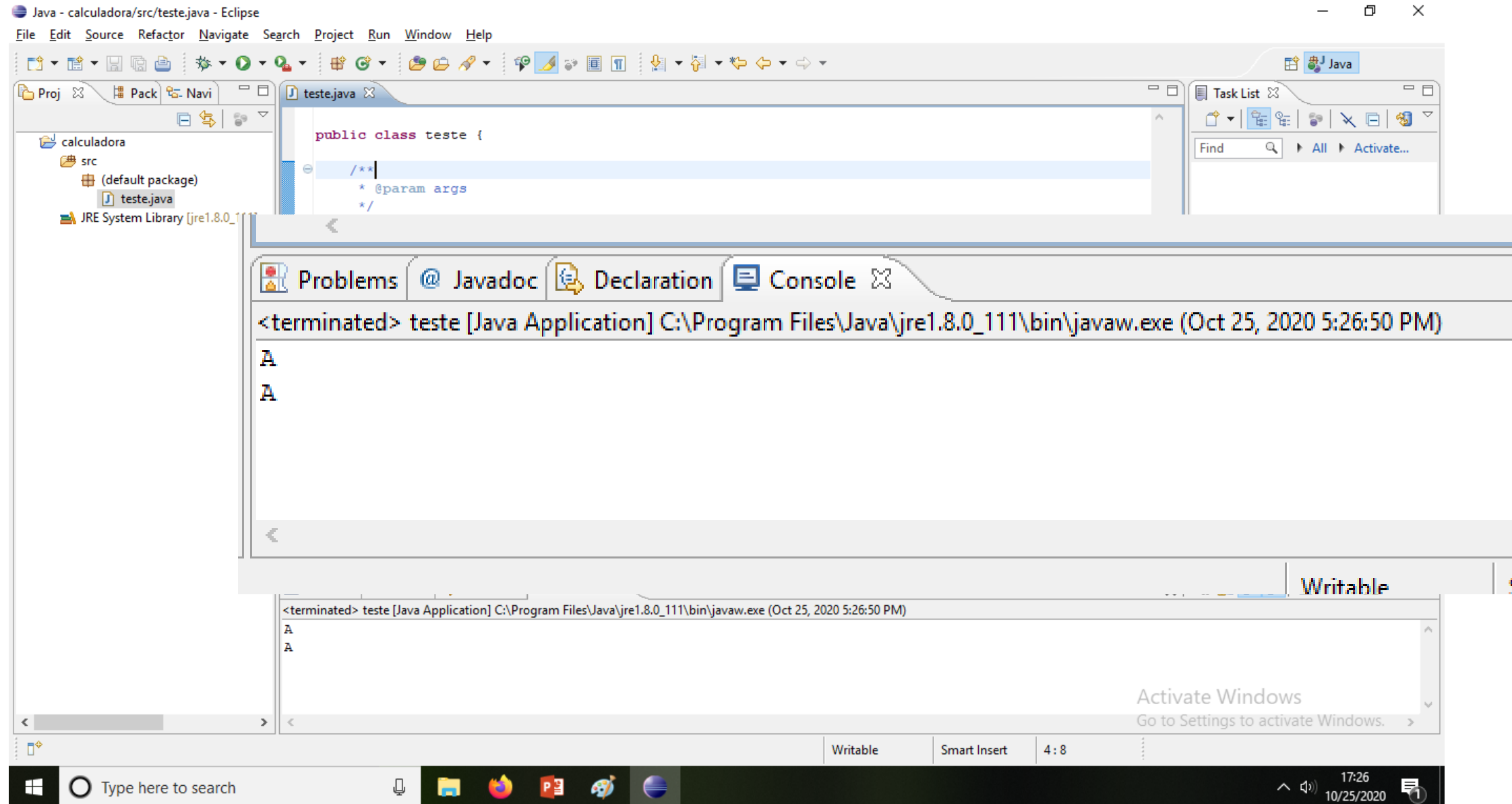
```
rintln(c2);
```

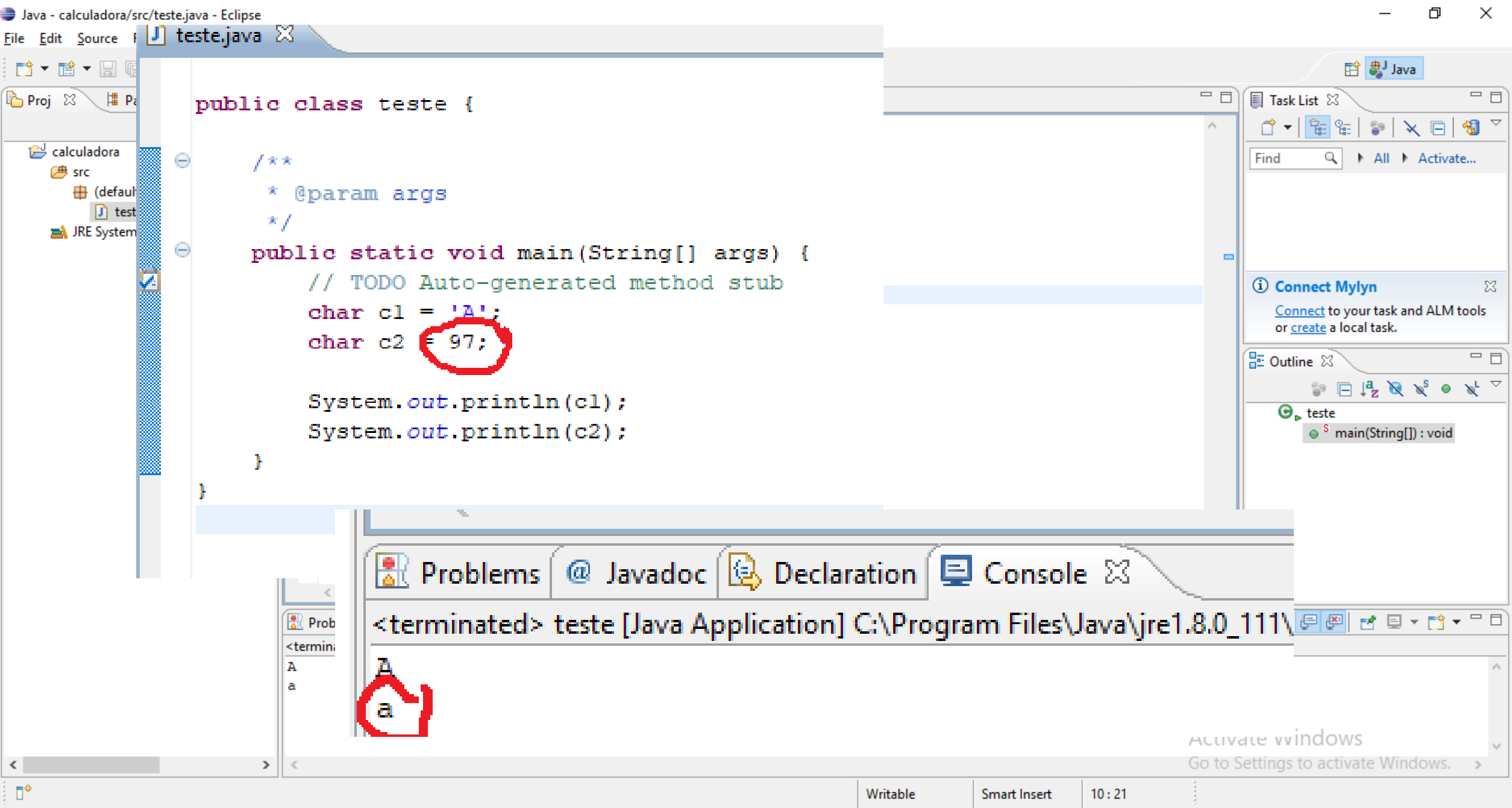
Como executar o nosso programa/código

- Para executar/rodar o nosso programa/código usamos a seguinte ferramenta.
- Para depurar o nosso programa/código em busca de possíveis erros usamos a ferramenta de depuração.



Resultado





```
public class teste {  
  
    /**  
     * @param args  
     */  
    public static void main(String[] args) {  
        // TODO Auto-generated method stub  
        char c1 = 'A';  
        char c2 = 97;  
  
        System.out.println(c1);  
        System.out.println(c2);  
    }  
}
```

<terminated> teste [Java Application] C:\Program Files\Java\jre1.8.0_111\

Prob
<termini
A
a

A
a

Activate windows
Go to Settings to activate Windows.

Pergunta

- O que acontece se mudamos o tipo de dados de uma das variaveis?
- Por exemplo:

```
public class teste {  
  
    /**  
     * @param args  
     */  
    public static void main(String[] args) {  
        // TODO Auto-generated method stub  
        char c1 = 'A';  
        byte c2 = 97;  
  
        System.out.println(c1);  
        System.out.println(c2);  
    }  
}
```

Resultados

The screenshot displays the Eclipse IDE interface. The main editor shows a Java class named `teste` with a `main` method. The code is as follows:

```
public class teste {  
    /**  
     * @param args  
     */  
    public static void main(String[] args) {  
        // TODO Auto-generated method stub  
        char c1 = 'A';  
        byte c2 = 97;  
  
        System.out.println(c1);  
        System.out.println(c2);  
    }  
}
```

The `byte` keyword and the value `97` are circled in red. The left sidebar shows a project named `calculadora` with a `src` folder containing `teste.java`. The right sidebar contains the `Task List` and `Outline` views. The `Outline` view shows the `teste` class with a `main(String[]) : void` method. The bottom of the IDE shows the `Problems`, `Javadoc`, `Declaration`, and `Console` views. The `Console` view displays the output of the program:

```
<terminated> teste [Java Application] C:\Program Files\Java\jre1.8.0_111\bin\java.exe  
A  
97
```

The output `97` is circled in red. The status bar at the bottom indicates the file is `Writable`, `Smart Insert` is enabled, and the cursor is at line `16 : 1`.

Conversão

```
public class teste {  
  
    /**  
     * @param args  
     */  
    public static void main(String[] args) {  
        // TODO Auto-generated method stub  
        byte b = 12;  
        short s = 1234;  
        int i = 12345;  
        long l = 123456;  
  
        s = b;  
    }  
}
```

Conversão implícita (automática, feita pela linguagem)

Conversão

```
public class teste {  
  
    /**  
     * @param args  
     */  
    public static void main(String[] args) {  
        // TODO Auto-generated method stub  
        byte b = 12;  
        short s = 1234;  
        int i = 12345;  
        long l = 123456;  
  
        s = b;  
        i = (int)s;  
    }  
}
```

Conversão explícita

Notificações no código

Java - calculadora/src/teste.java - Eclipse

File Edit Source Refactor Navigate Search Project Run Window Help

Proj Pack Navi

calculadora
src
(default package)
teste.java
JRE System Library [jre1.8.0_111]

```
public class teste {  
    /**  
     * @param args  
     */  
    public static void main(String[] args) {  
        // TODO Auto-generated method stub  
        byte b = 12;  
        short s = 1234;  
        int i = 12345;  
        long l = 123456;  
  
        s = b;  
        i = (int)s;  
    }  
}
```

Task List

Find All Activate...

Connect Mylyn
Connect to your task and ALM tools or create a local task.

Outline

teste
main(String[]) : void

Problems @ Javadoc Declaration Console

0 errors, 2 warnings, 0 others

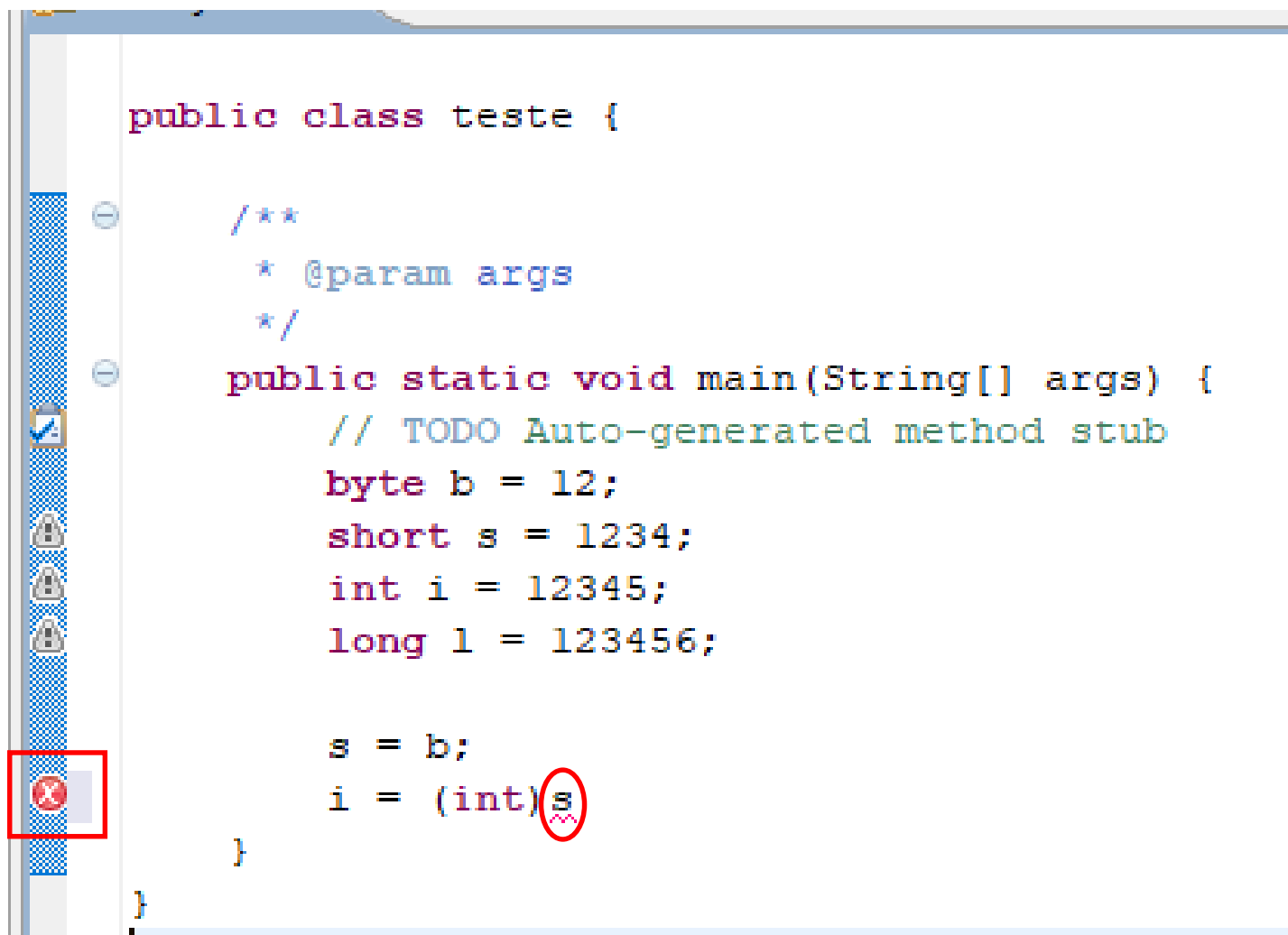
Description	Resource	Path	Location	Type
Warnings (2 items)				
The value of the local variable i is not used	teste.java	/calculadora/src	line 11	Java Problem
The value of the local variable l is not used	teste.java	/calculadora/src	line 12	Java Problem

Activate Windows
Go to Settings to activate Windows.

Writable Smart Insert 18 : 1

Erros no código

```
public class teste {  
  
    /**  
     * @param args  
     */  
  
    public static void main(String[] args) {  
        // TODO Auto-generated method stub  
        byte b = 12;  
        short s = 1234;  
        int i = 12345;  
        long l = 123456;  
  
        s = b;  
        i = (int)s;  
    }  
}
```



Erros no código

```
public class teste {
```

```
    /**
```

```
     * @param args
```

```
     */
```

```
    public static void main(String[] args) {
```

```
        // TODO Auto-generated method stub
```

```
        byte b = 12;
```

```
        short s = 1234;
```

```
        int i = 12345;
```

```
        long l = 123456;
```

```
        s = b;
```

```
        i = (int) s
```

```
    }
```

```
}
```

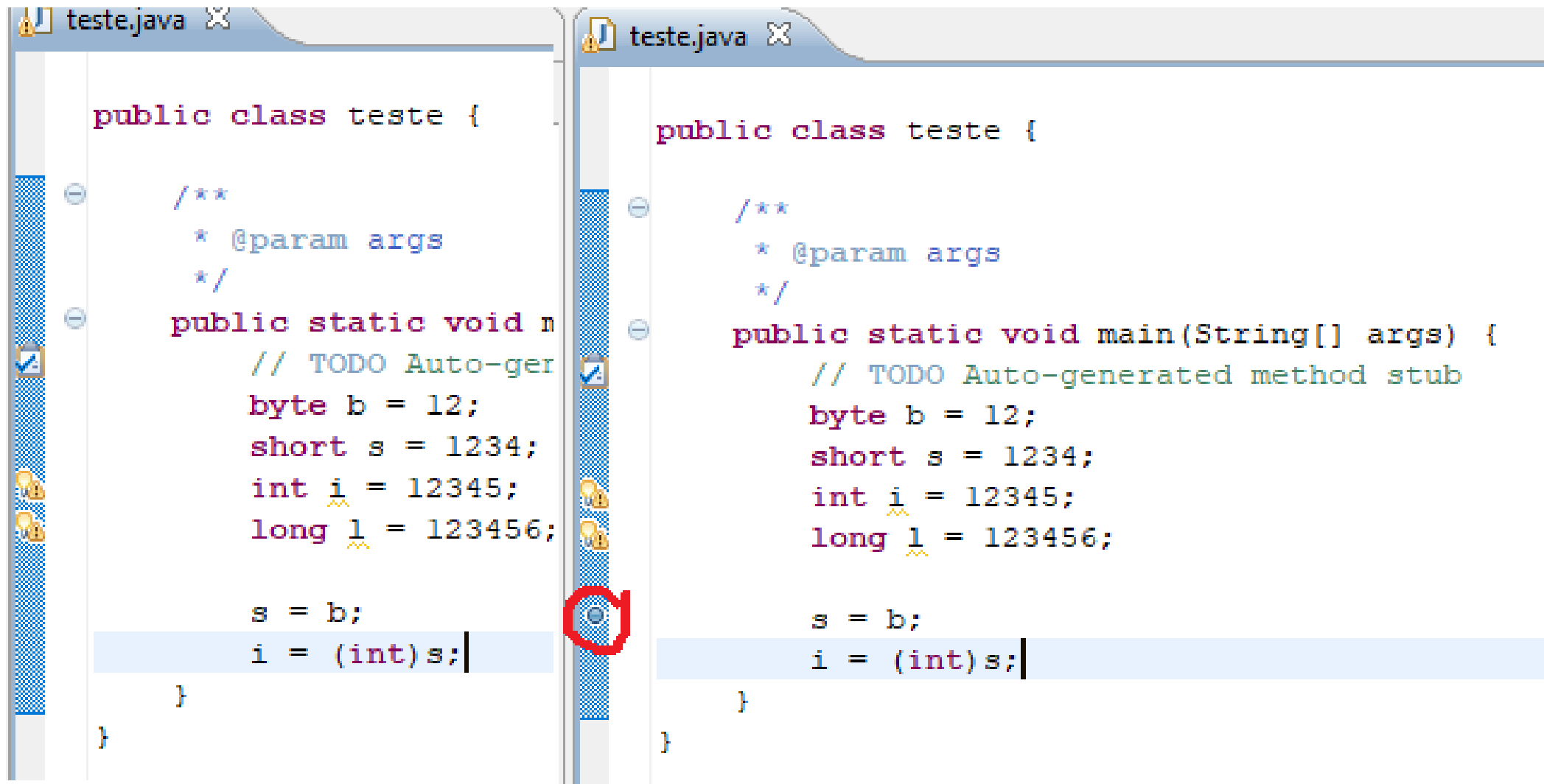
✖ Syntax error, insert ";" to complete Statement

Press 'F2' for focus

Depuração do código

- A depuração do código é um processo que serve para detectar e corrigir erros “lógicos” no nosso código.
- Para iniciar a depuração é preciso estabelecer um ponto de ruptura na execução do código (***break-point***)

Depuração do código (cont.)



The image displays two side-by-side screenshots of a Java IDE window titled 'teste.java'. Both screenshots show the same Java code:

```
public class teste {  
  
    /**  
     * @param args  
     */  
    public static void main(String[] args) {  
        // TODO Auto-generated method stub  
        byte b = 12;  
        short s = 1234;  
        int i = 12345;  
        long l = 123456;  
  
        s = b;  
        i = (int) s;  
    }  
}
```

In the left screenshot, a red circle highlights a breakpoint (a small circle with a dot) placed on the line `i = (int) s;`. The right screenshot shows the same code without the breakpoint.

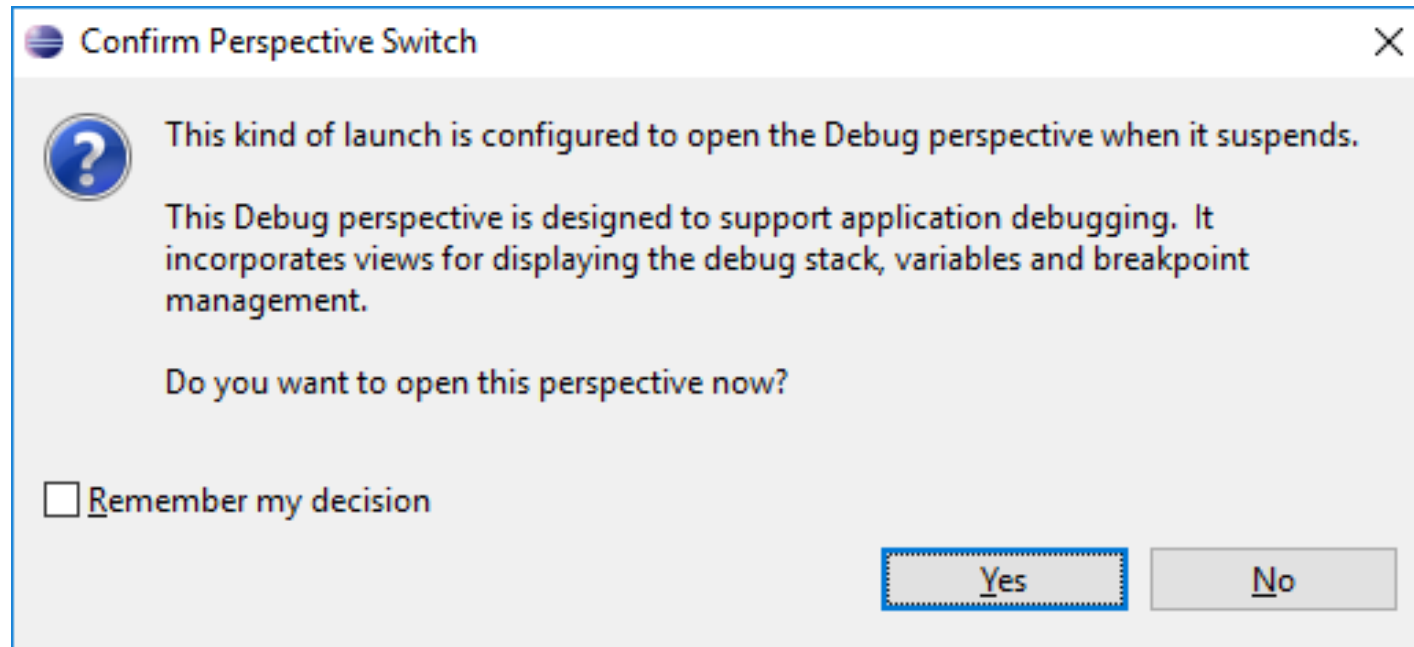
Depuração do código

- A depuração do código é um processo que serve para detectar e corrigir erros “lógicos” no nosso código.
- Para iniciar a depuração é preciso estabelecer um ponto de ruptura na execução do código (***break-point***)
- Iniciamos a depuração ao clicar no botão



Depuração do código (cont.)

- Ao clicar no botão somos questionados se queremos trocar para a perspectiva de depuração



Depuração do código (cont.)

- Esta perspectiva permite ter acesso de forma automática a um conjunto de funcionalidades que irão fazer com que a análise do código seja mais fácil

Debug Console: teste [Java Application]
teste at localhost:7718
Thread [main] (Suspended (breakpoint at line 14 in teste))
teste.main(String[]) line: 14
C:\Program Files\Java\jre1.8.0_111\bin\javaw.exe (Oct 25, 2020 7:28:30 PM)

Variables Table:

Name	Value
args	String[0] (id=16)
b	12
s	1234
i	12345
l	123456

Source Editor (teste.java):

```
* @param args
*/
public static void main(String[] args) {
    // TODO Auto-generated method stub
    byte b = 12;
    short s = 1234;
    int i = 12345;
    long l = 123456;
    s = b;
    i = (int)s;
}
```

Outline:

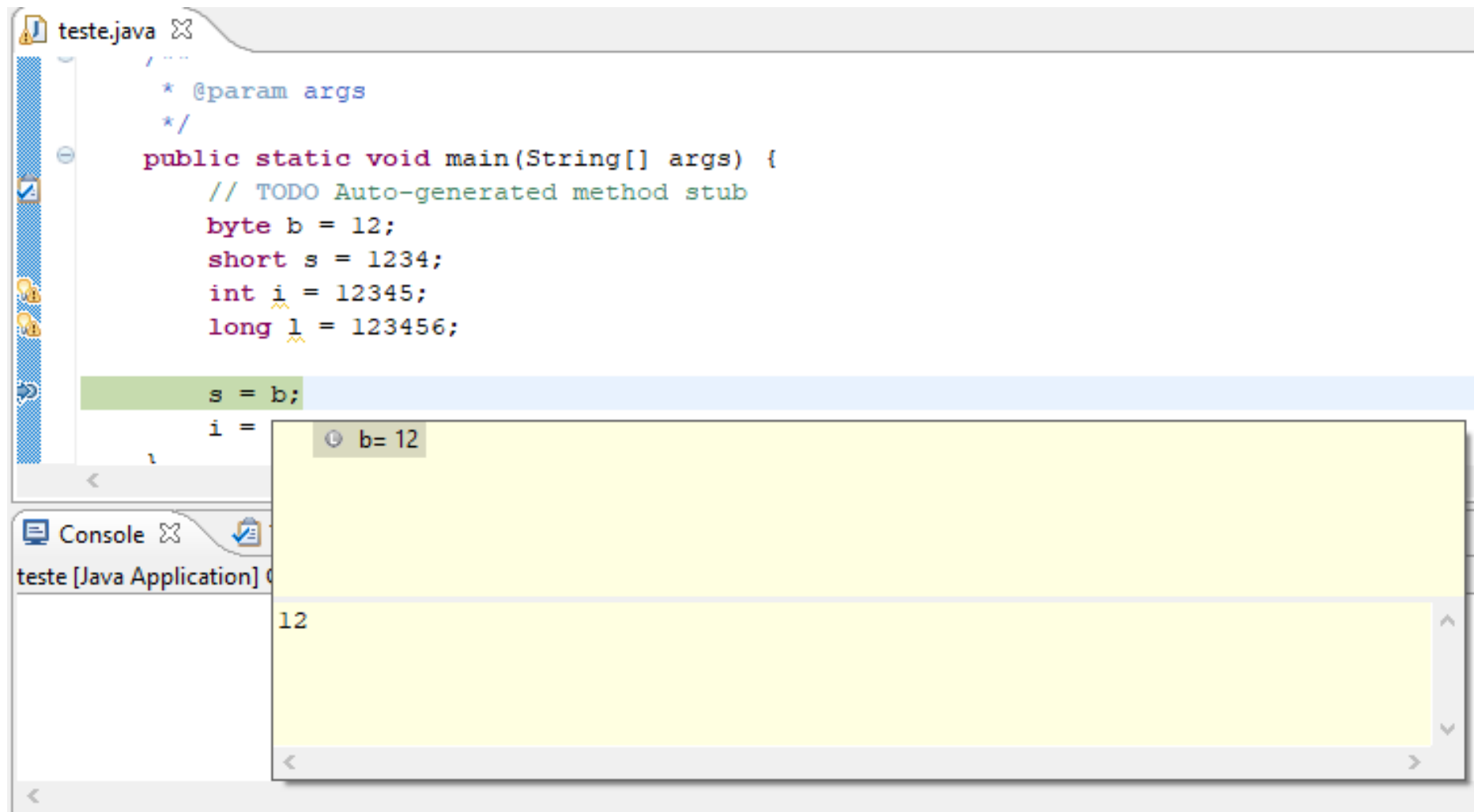
- teste
 - main(String[]) : void

Console:

```
teste [Java Application] C:\Program Files\Java\jre1.8.0_111\bin\javaw.exe (Oct 25, 2020 7:28:30 PM)
```

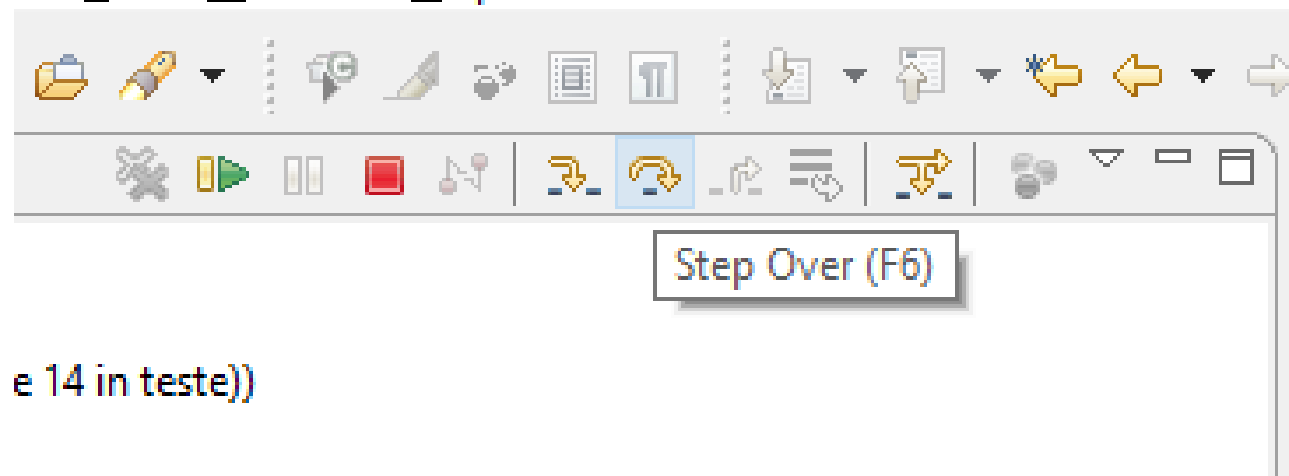
Footer: Writable Smart Insert 14 : 1

Depuração do código (cont.)

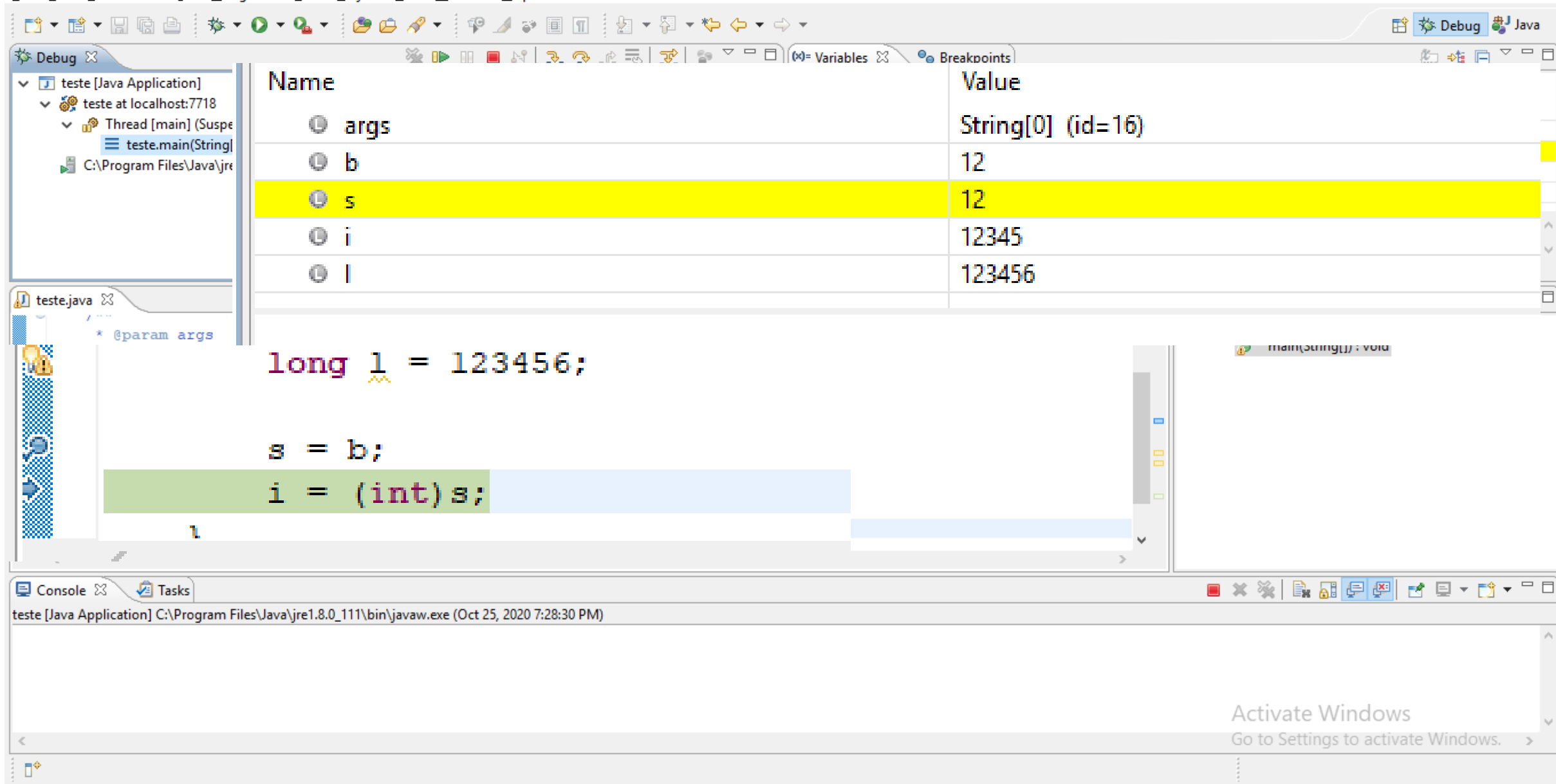


Depuração do código (cont.)

File Edit Run Window Help



e 14 in teste))



The screenshot displays the Eclipse IDE interface during a debug session. The top toolbar includes icons for file operations, running, and debugging. The left sidebar shows the project structure with 'teste [Java Application]' and 'teste at localhost:7718'. The main editor area shows the source code of 'teste.java' with the following code:

```
@param args  
  
long l = 123456;  
  
s = b;  
i = (int)s;
```

The right sidebar shows the 'Variables' view, which lists the current state of variables:

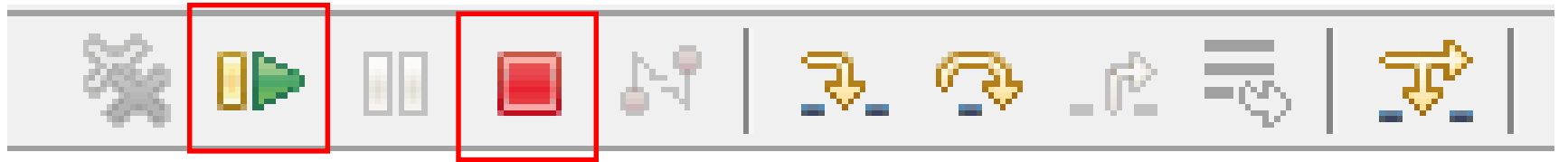
Name	Value
args	String[0] (id=16)
b	12
s	12
i	12345
l	123456

The bottom of the IDE shows the 'Console' view, which displays the output of the application:

```
teste [Java Application] C:\Program Files\Java\jre1.8.0_111\bin\javaw.exe (Oct 25, 2020 7:28:30 PM)
```

An 'Activate Windows' watermark is visible in the bottom right corner of the IDE.

Depuração do código (cont.)



Executar o código até o final

Terminar/Cancelar a depuração

FIM