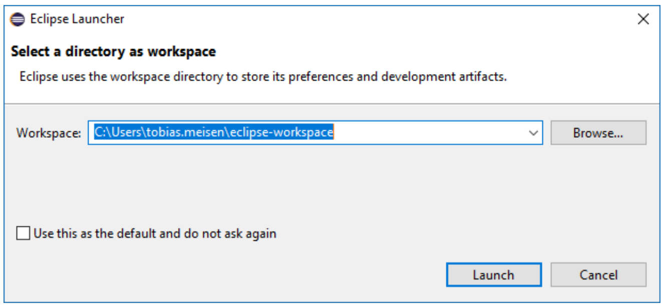
OPMS 2018 – Exercise 1

Task 1 (Hello World)

1. Open your Eclipse IDE and select a directory as workspace (most likely this should be your home directory):



Eclipse Launcher

Select a directory as workspace

Eclipse uses the workspace directory to store its preferences and development artifacts.

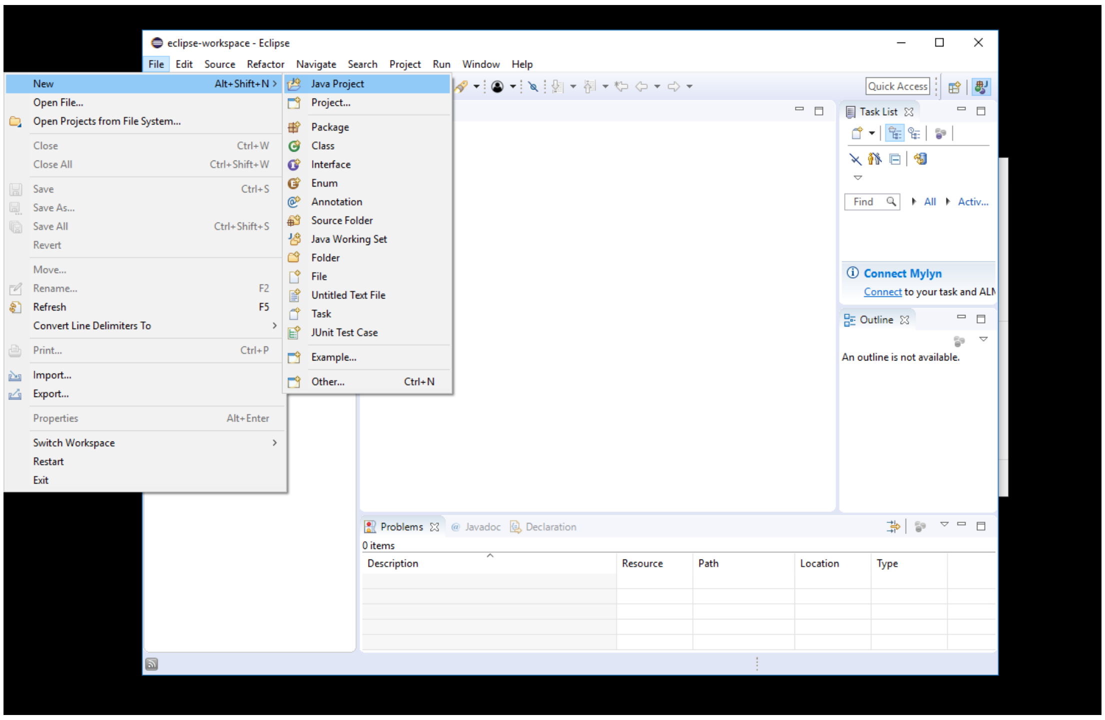
Workspace: C:\Users\tobias.meisen\eclipse-workspace   Browse...

☐ Use this as the default and do not ask again

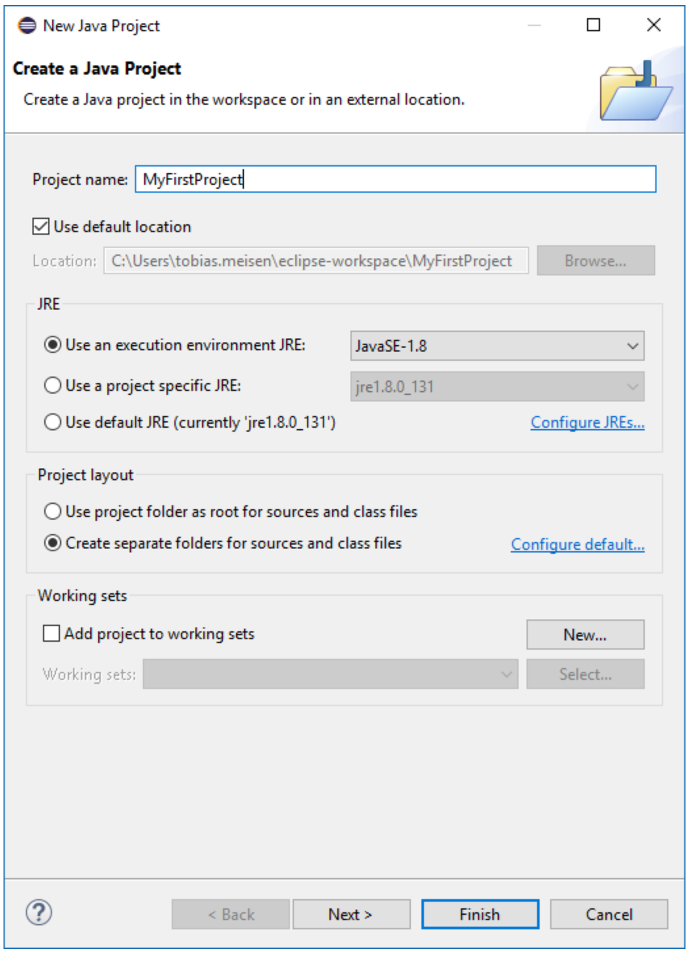
Launch   Cancel

Picture 1: Workspace selection screen after starting the Eclipse IDE

2. After the Eclipse IDE is launched, close the welcome tab and create a new first project – see the pictures below for a How-To description.



Picture 2: Creating a new project


New Java Project

Create a Java Project

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

Project name: MyFirstProject

[✓] Use default location

Location: C:\Users\tobias.meisen\eclipse-workspace\MyFirstProject

JRE

[●] Use an execution environment JRE: JavaSE-1.8

[○] Use a project specific JRE: jre1.8.0_131

[○] Use default JRE (currently 'jre1.8.0_131')

Project layout

[○] Use project folder as root for sources and class files

[●] Create separate folders for sources and class files

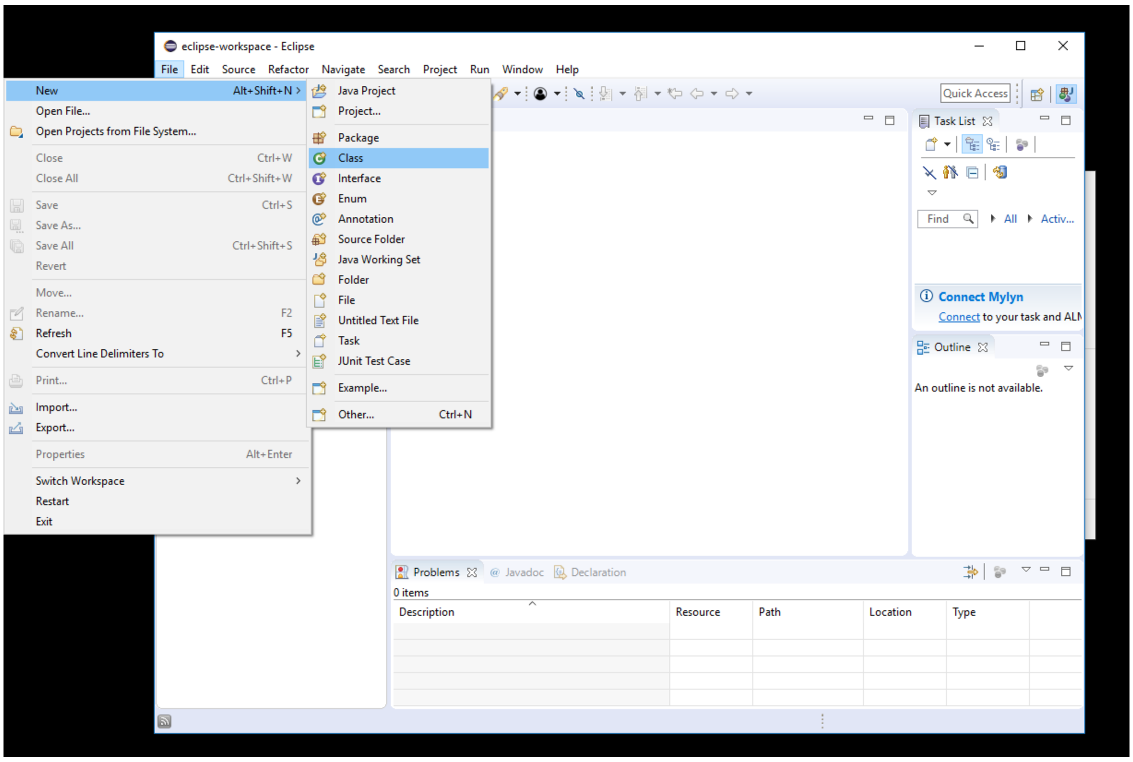
Working sets

[ ] Add project to working sets

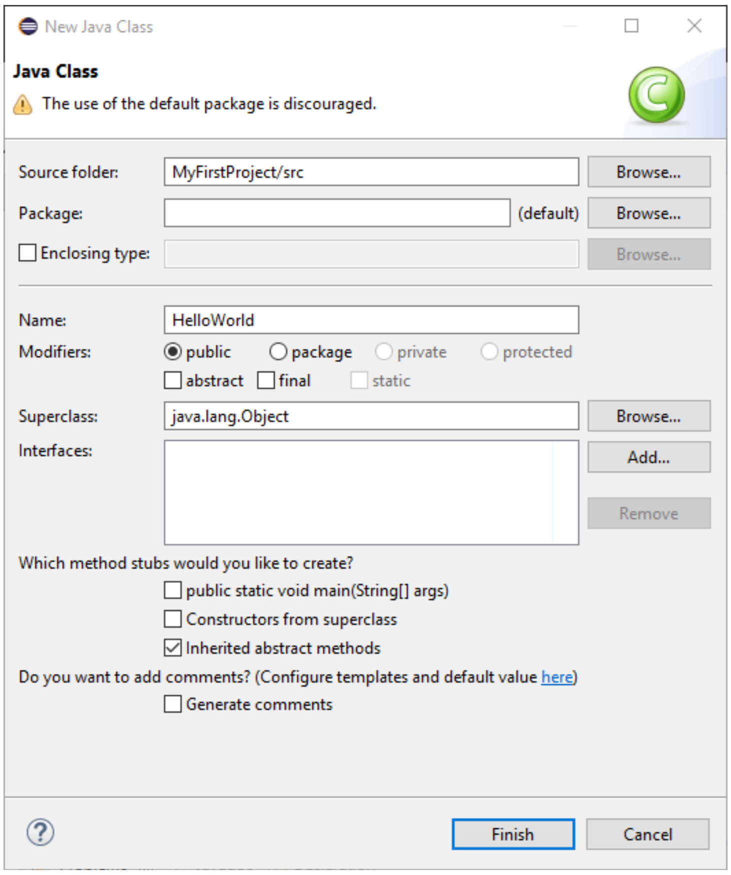
Working sets:

Picture 3: Wizard to create a new project – after entering an appropriate name, click finish

3. Create a new class (HelloWorld) – see the pictures below for more information.



Picture 4: Creating a new class in your project



The use of the default package is discouraged.

Source folder: MyFirstProject/src

Package: (default)

☐ Enclosing type:

Name: HelloWorld

Modifiers: [✓ public] package private protected
☐ abstract ☐ final ☐ static

Superclass: java.lang.Object

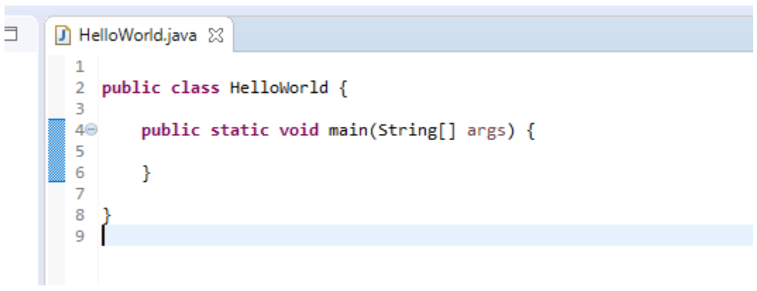
Interfaces:

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

Picture 5: Enter the class name “HelloWorld” and close the dialog with finish – ignore the warning

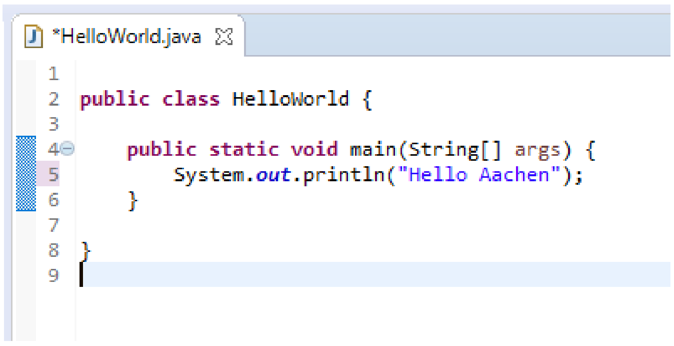
4. Add the main()-method, which is the entry point for each Java application. **Hint:** The first main()-method can be found on slide 18 of the lecture.



```

public class HelloWorld {

    public static void main(String[] args) {

    }

}

```

Picture 6: Adding the main()-method

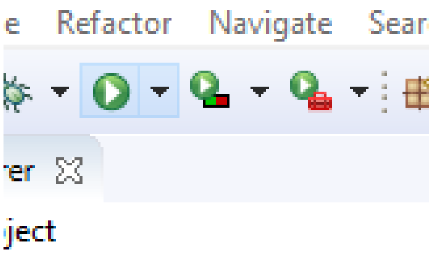
5. Extend your main()-method by a statement that prints “Hello Aachen” on the console when the application is executed.



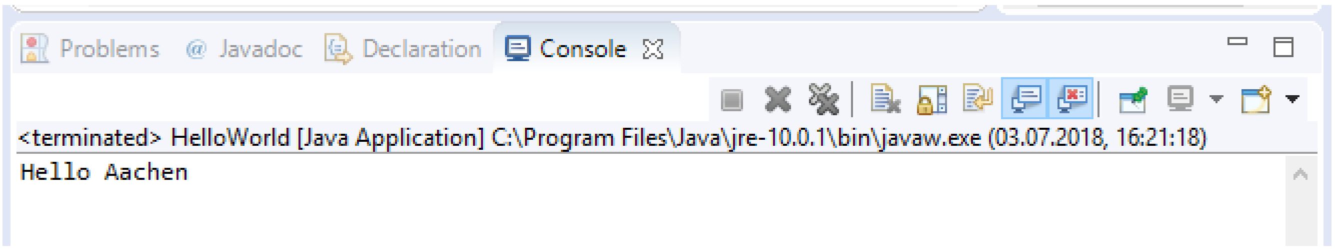
```

public class HelloWorld {

    public static void main(String[] args) {
        System.out.println("Hello Aachen");
    }

}

```

Picture 7: Printing “Hello Aachen” to the console

6. Execute your application and look for the right output (see Picture 9).



Picture 8: Executing the application by pressing the simple green circle with white arrow



```
<terminated> HelloWorld [Java Application] C:\Program Files\Java\jre-10.0.1\bin\javaw.exe (03.07.2018, 16:21:18)
Hello Aachen
```

Picture 9: Result if your application compiled correctly and ran as expected

Task 2 (First Steps)

1. Create a new project called “Exercise1”.
2. Create a new class “Task2” (use the automatic method stub creation available, to create the main()-method automatically).
3. Implement the following behavior in your main()-method and execute it.
   a. Create a new integer variable x and assign the value 0.
   b. Create a new integer variable y and assign the value 5.
   c. Now create another integer variable z and assign the sum of x and y.
   d. Print the value of z to the console, so that the following output is generated “The sum of x and y is 5”.
   e. Change the value of x and y and test if your application works as intended.

   Extension: Extend your code, so that the value of x and y is printed in round brackets: “The sum of x (0) and y (5) is 5”.

Task 3 (Type Casting)

In the following task, you have to answer the given questions. You can note your answers on a piece of paper or in a text document. **You are not allowed to use Eclipse during exercises a) – d) of this task!**

1. Give one example each for the correct definition of the following data types.
   - String
   - Boolean
   - Double

2. Give an example for an explicit type cast!

3. Define a constant with the name E and assign in the value 2.71828.

4. What is printed by the following statements?

   a. `System.out.println(9.0 / 2);`

   b. `System.out.println(3 / 9 * 9);`

   c. `System.out.println(9 * 3 / 9);`

   d. `System.out.println(5 * (int) 11 / 4.0);`

   e. `System.out.println(5 * (int) (11 / 4.0));`

   f. `System.out.println((float) 5);`

Note: For exercises e) - you are allowed to use Eclipse!

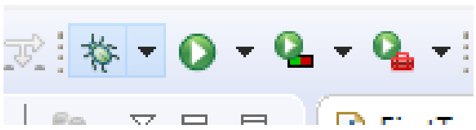
5. Create a new class "Task3" inside your project called "Exercise One" (use the automatic method stub creation available, to create the main()-method automatically).

6. Implement the expressions from d) in the main()-method.

7. Compare the results from your implementation in f) with your results from d). Did the code behave as you expected it to? If not, try to figure out why!

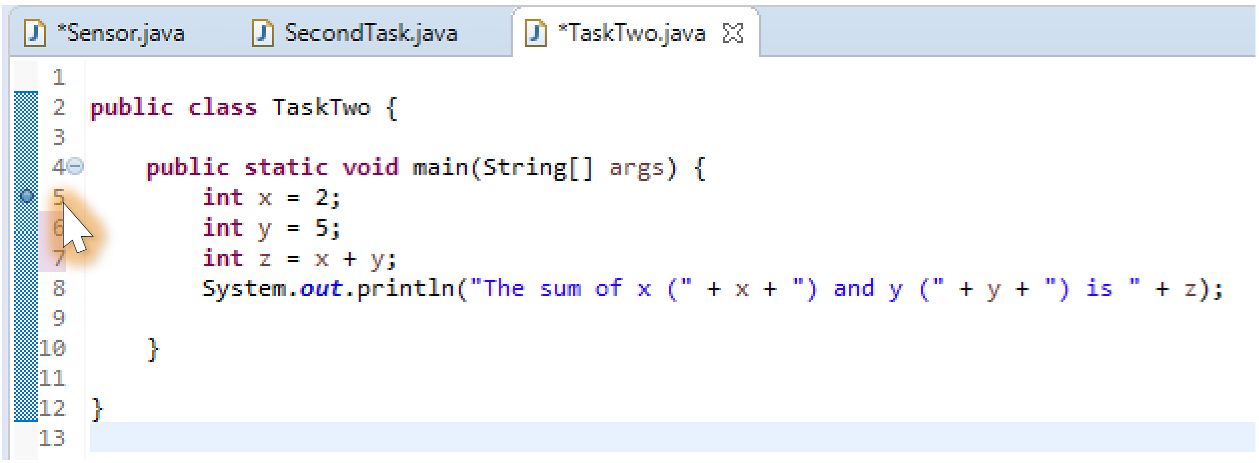
Task 4 (Debugger)

1. Open the class “Task2” which you implemented in Task 2.
2. Launch the class in debug mode by pressing the green bug right next to the run button. Your program will run until the end and exit as before.



Picture 10: Executing the application in debug mode by pressing the green bug

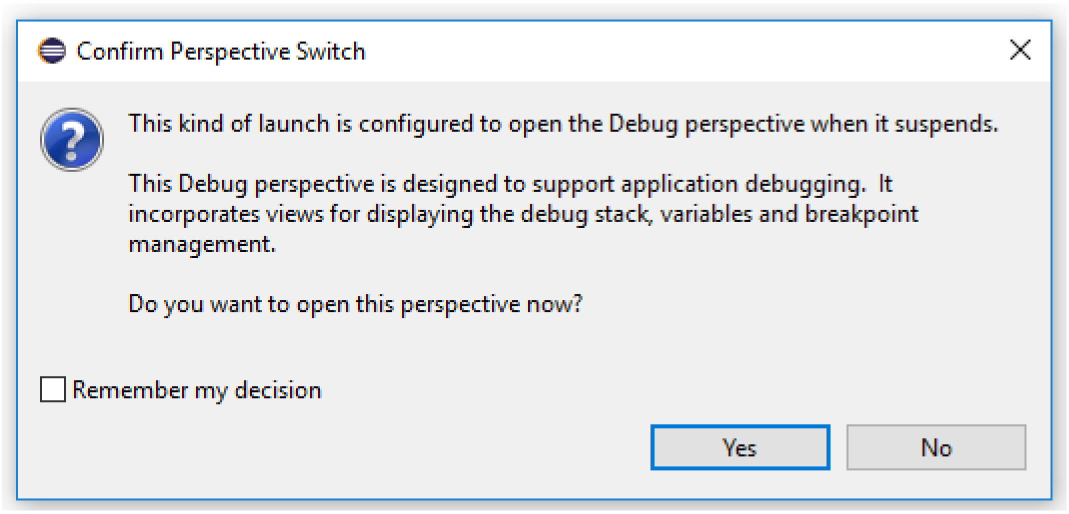
3. To pause execution place a breakpoint in the code by double clicking on the line number of the first line inside your main()-method. You should see a green circle left to the line number.



```java

public class TaskTwo {

    public static void main(String[] args) {
        int x = 2;
        int y = 5;
        int z = x + y;
        System.out.println("The sum of x (" + x + ") and y (" + y + ") is " + z);

    }

}

```

Picture 11: Executing the application in debug mode by pressing the green bug

4. Launch the debug mode again. You might be asked, if you want to open the Debug perspective. Press “Yes”.


Confirm Perspective Switch

This kind of launch is configured to open the Debug perspective when it suspends.

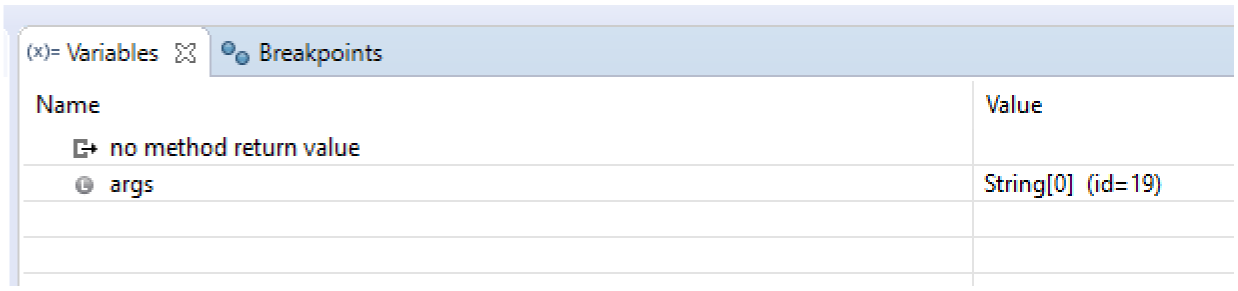
This Debug perspective is designed to support application debugging. It incorporates views for displaying the debug stack, variables and breakpoint management.

Do you want to open this perspective now?

☐ Remember my decision

Picture 12: Executing the application in debug mode by pressing the green bug

5. In the upper right window, click on the tab called “Variables”. Now you should see a list of variables, which have been set by your program. Since the program is still at the very beginning, the list is still empty (except for the parameter *args* of the main()-method). This will change in the next step.



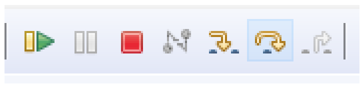
(x)= Variables

Breakpoints

Name	Value
no method return value	
args	String[0] (id=19)

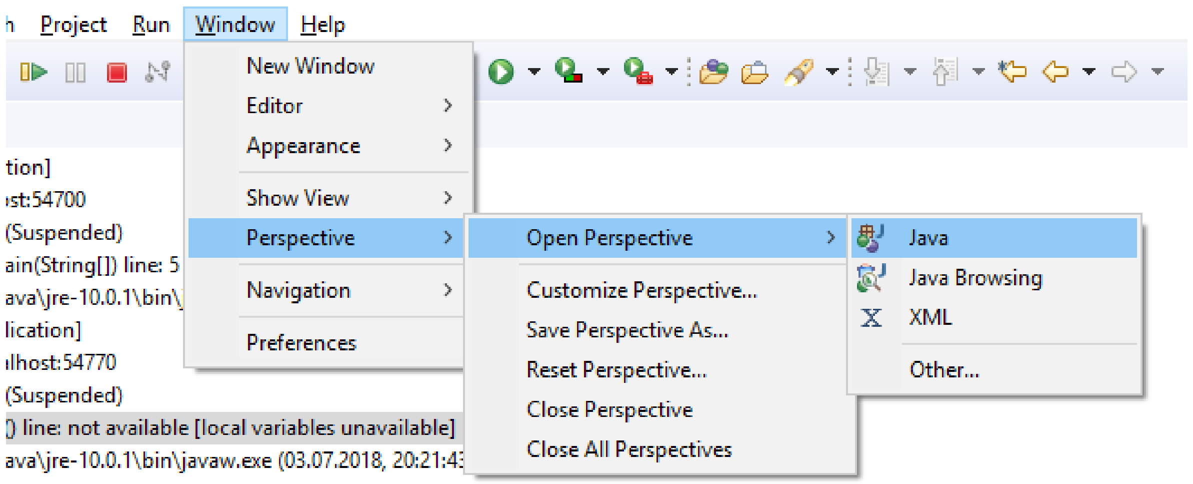
Picture 13: The Variables tab offers a view into the current values behind the variables

6. Click on the “Step Over” function to execute the next line of the code. Watch how the variable list changes.



Picture 14: These buttons are used to control the execution of a program. This way, a programmer can inspect how a program behaves and is able to find mistakes in the code. This process is called debugging.

8. Familiarize yourself with the debugger. Set different breakpoints, resume the program (green arrow next to the “Step Over” button).
9. If you want to switch out of the debugger perspective, you can do so via Window→Perspective→Open Perspective→Java



```
tion]
st:54700
(Suspended)
ain(String[]) line: 5
ava\jre-10.0.1\bin\
lication]
lhost:54770
(Suspended)
) line: not available [local variables unavailable]
ava\jre-10.0.1\bin\javaw.exe (03.07.2018, 20:21:43
```

Picture 15: Change between different perspectives via Window→Perspective→Open Perspective→Java