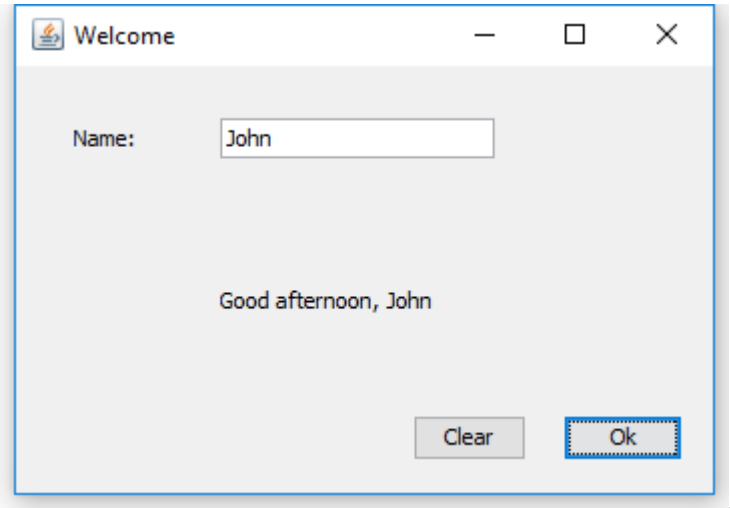


Assignment 1

Content Application Development Day 1
Lecture
The lecture provides an introduction to programming, the concept of classes and objects in Java and the Eclipse development environment.
Tutorial
Eclipse is installed during the tutorial session. You are introduced to Java and create your first program.

At the end of day 1 you will be able to
- Describe where you will encounter computers and software as an engineer.
- Describe the key elements of a computer
- Describe how programming works
- Develop and run a simple program
- Create a basic user interface
- Have your program display a message on the screen

Self-study
Book:
Head First Java (K. Sierra & B. Bates): chapter 1, 2.
Aan de slag met Java *[Getting Started with Java]* (Gertjan Laan): chapter 1 - 3.5.
Online:
codecademy.com: 1. Introduction to Java
javatpoint.com: first part "Java Tutorial"

Practice
Study questions for these chapters to increase your understanding of the subject matter. Try programming one or more of the examples in Eclipse.

Appendices
Appendix 1 explains how to use **examples** (from the books) **in Eclipse**.
Appendix 2 explains a number of **common problems**.
Appendix 3 explains how to **install Window Builder** in Eclipse.

Check and run assignments

You may complete assignments with 2 people. You don't have to, so you can also do them on your own. It is not allowed to complete assignments with 3 or more people and without any further checks this will automatically result in an unsatisfactory mark.

Have your assignments checked by the lecturer or assistant no later than at the next lecture (that is the deadline!). The purpose of the check is that you demonstrate that you understand the subject matter. The assignment is graded as a pass (1) or a fail (0).
Assignments 2 through 7 count towards the grade (details have been provided in the lecture). Please have assignment 1 also checked, to 'practice' checking and to make sure you know what is expected of you!

Introduction assignment 1

Through this assignment you become familiar with the Eclipse development environment. It is a programming environment that is also used by professional software developers and offers many possibilities. Therefore, it looks complicated at first, but you will find that it is indeed user-friendly. By doing the first assignments you will learn to find your way with it. You are going to write an Application (= simple program).

In this first assignment you will make a Java application that displays a text on the screen after pressing a button. It is not an earth-shattering event, but it provides insight into a number of actions that you will keep using. The purpose of this assignment is to become familiar with the principles of the use of Eclipse.

Tutorial steps

The assignment consists of the following steps:

1. Start Eclipse and create a new project.
2. Create a user interface.
3. Add components to the user interface.
4. Have buttons do something.

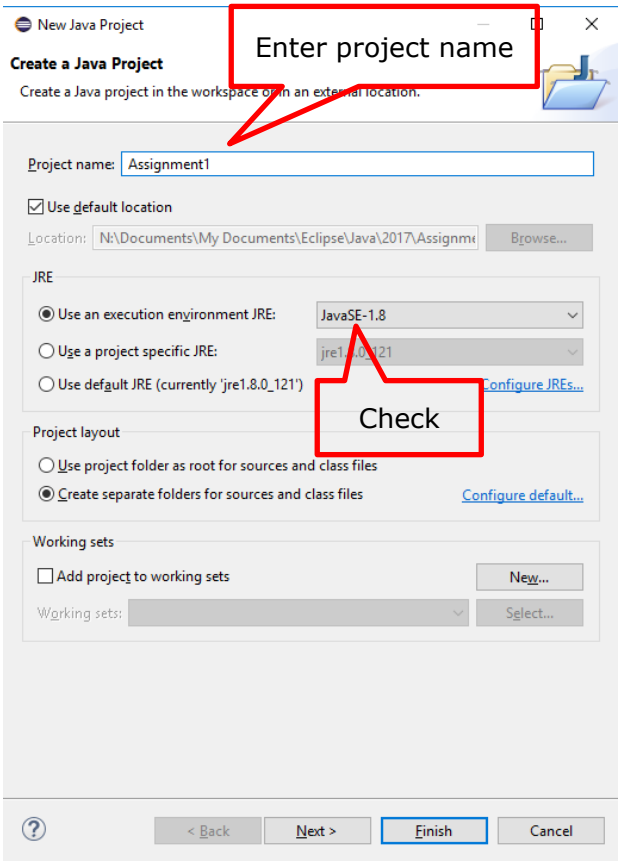
These steps are explained in detail below.

1. Start Eclipse and create a new project

Start Eclipse using the icon on the Desktop*. A dialog "Select a directory as workspace" will appear. You may accept the default directory shown there or choose another folder. Check the option "Use this as the default…" and press Ok.

Create new project
You are now first going to create a new project. A project is a folder for a number of related program components like source (code) files. It is usually located under the *Workspace* folder within your *Documents* folder.

To start a new project, select *File* > *New* > *Java Project* from the menu. A *Wizard* will appear, as shown in the figure below. Here you can enter the name at Project Name: "Assignment1".
Check at "Use an execution environment JRE" whether "JavaSE-1.8" has been selected.
Create the project by clicking on *Finish*.



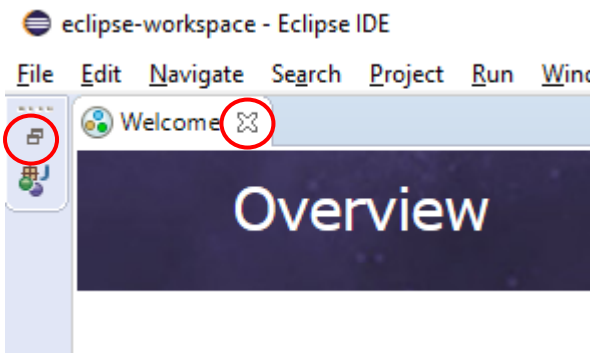
Tip: In the description of the assignments we assume that you create a new project for each day of the course.

*** No icon on the Desktop?**

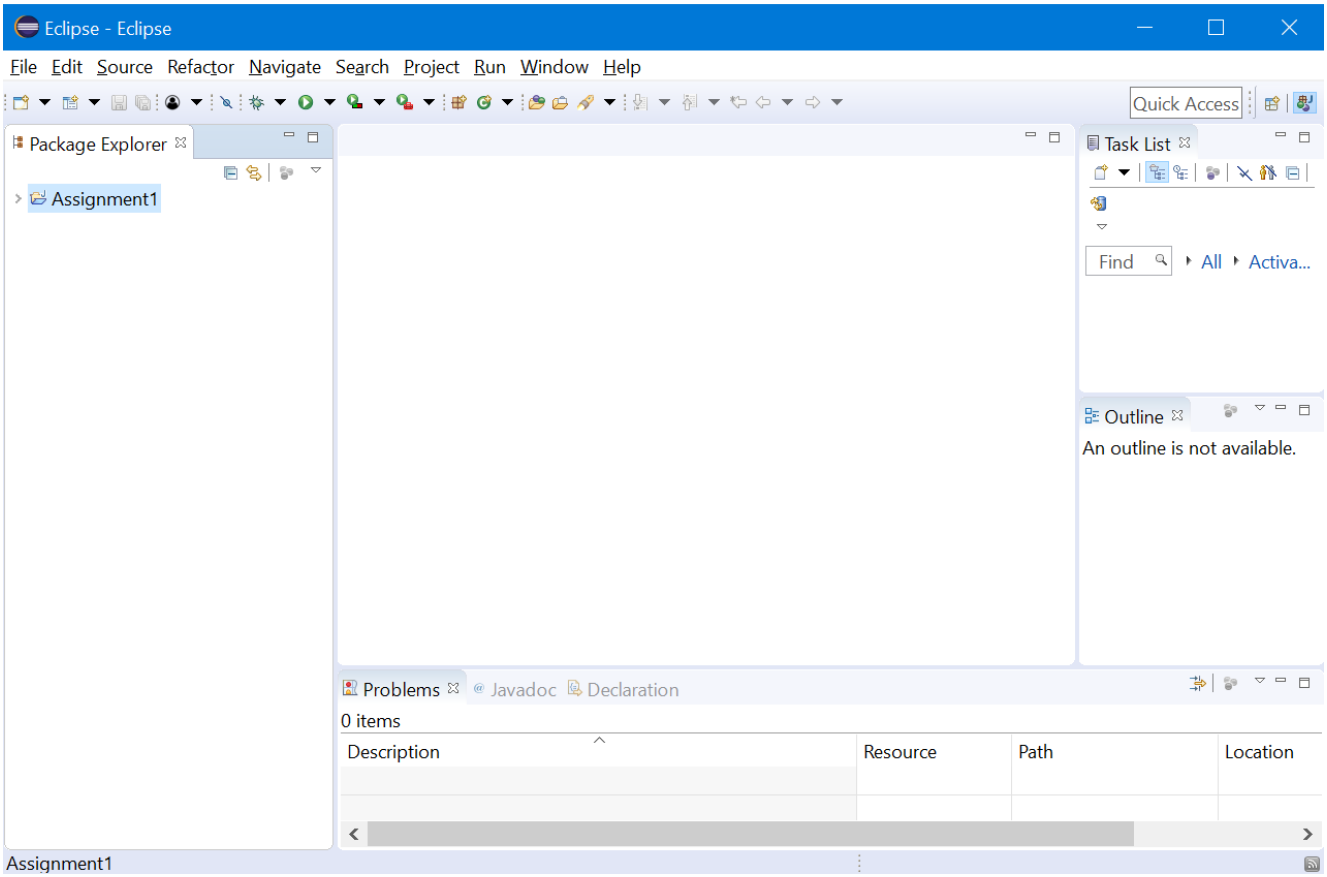
Using Windows Explorer, browse to the folder where you extracted the ZIP file using Eclipse. If you used the installation script, you can find Eclipse in the folder *C:\Users\<username>\Eclipse*

Right-klik *Eclipse.exe* and choose "Send to, Desktop" to add it to the Desktop.

If the *Welcome* screen is still visible, remove the checkmark "Always show Welcome at start up" in the lower right corner, and close the *Welcome* screen:
You might have to click the Restore icon to view the project:



The Eclipse screen should look like below:

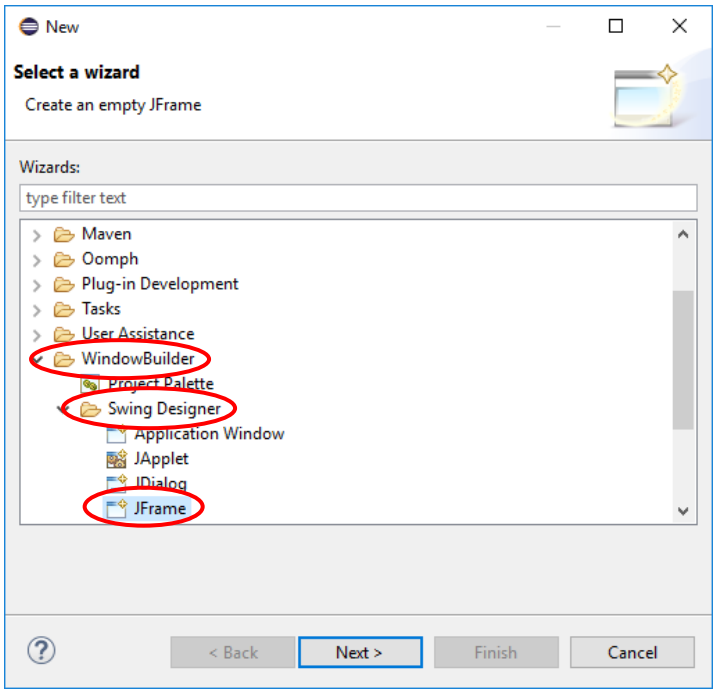


You now have a new project, but it is still empty. In the next steps we are going to add a user interface. A user interface is also called graphical user interface (GUI).

The screen you see now is the default screen layout of Eclipse. If you ever want to return to the default layout, for example if you accidentally clicked away some items, you can always return to this layout via *Window* > *Perspective* > *Reset Perspective…*

2. Create a user interface

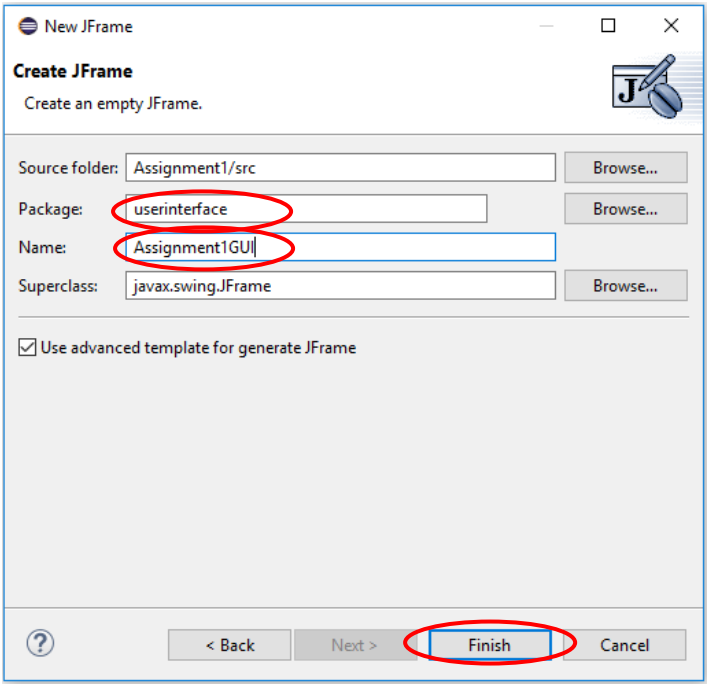
We will use a *JFrame Form* as a basis for the user interface. If you already have several projects, make sure *Assignment1* project has been selected in the Package Explorer. Select *File* > *New* > *Other* from the menu. Next, browse to *WindowBuilder*, *Swing Designer* and select *JFrame*:



No "WindowBuilder"?

If the folder WindowBuilder is missing, you might have a version of Eclipse that does not have the WindowBuilder installed. Read **Appendix 3** on how to install it.

In the next screen, type at *Name* the name "Assignment1GUI" and at *Package* type "userinterface" and click on *Finish*:

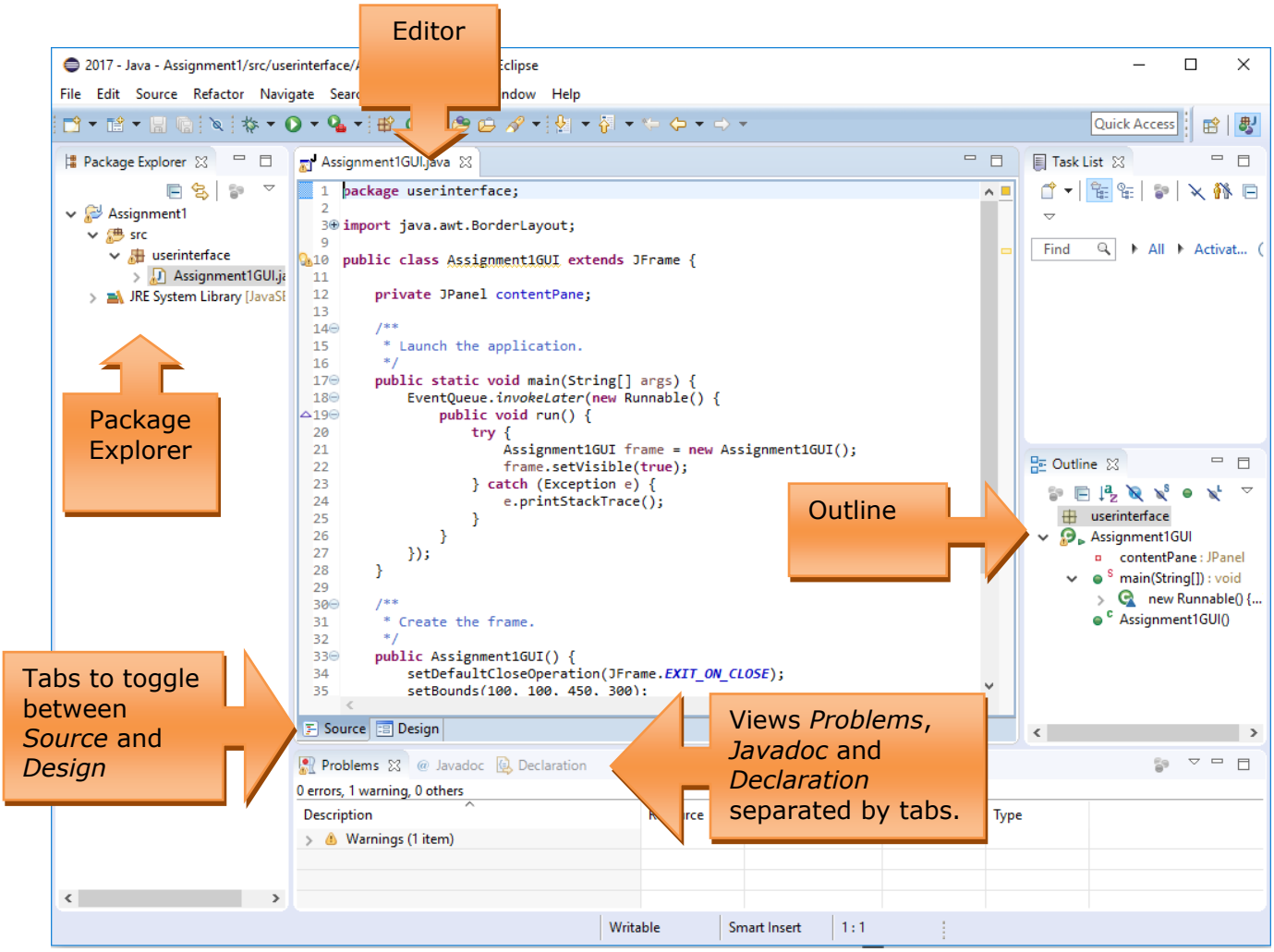


Eclipse Views

Below is an explanation of the main views of Eclipse. The complete set of views is called the *Workbench*. Within the *Workbench* there are components called *Views*. For example, the *Package Explorer* is a *View*, where you can view all the parts of a project.

Your screen should look something like the figure below. The most important view is the Editor, where you are going to write Java code. At the bottom of this *View* there are two tabs, which allow you to toggle between the source code (*Source*) and the design of the Graphical User Interface (*Design*).

In a development environment like Eclipse, user interfaces can be designed, without writing its code yourself. You do this with the visual Window Builder, which is accessible via the *Design* tab in the *Editor*.
If you want to view the source code, click on the *Source* tab.



Code in the Editor

In the Editor, you can view the code of the current open file, Assignment1GUI.java. There is already some code, which was automatically generated to create an empty JFrame form, which we will use as a base for the user interface.

Scroll all the way down in the Editor. While scrolling you see two methods: 'main' and 'Assignment1GUI'. Method 'main' is the start of the program. It creates a new frame and makes it visible (can you find these two lines of code?). The second method 'Assignment1GUI' is a special method which has the same name as the class, we call this the *constructor*. It constructs (builds) all parts of the class. For example, it creates a new JPanel (can you find that line of code?).

Add text output to Console
We can send text to the Console with this piece of code:

```
System.out.println("My first Java code");
```

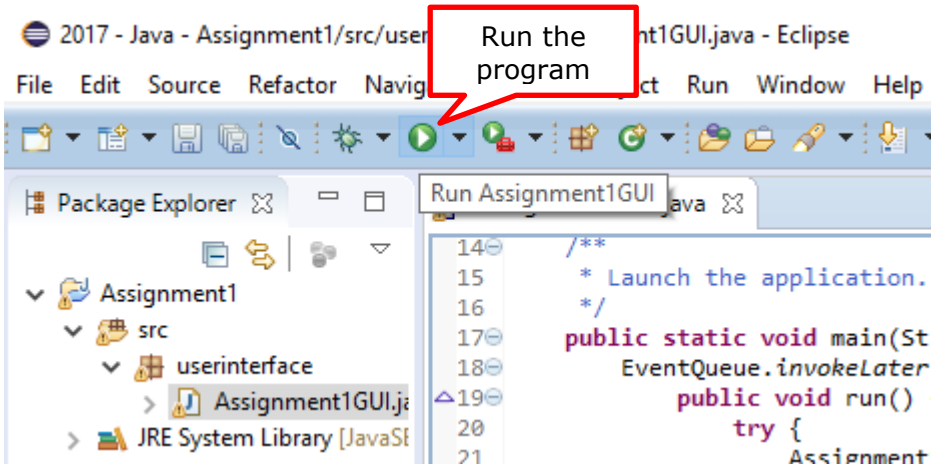
Add this piece of code on a new line in *Assignment1GUI()* after the line indicated by the arrow. Beware that you enter the code before the end of the method, before the curly bracket '}'.

```
public Assignment1GUI() {
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setBounds(100, 100, 450, 300);
    contentPane = new JPanel();
    contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
    contentPane.setLayout(new BorderLayout(0, 0));
    setContentPane(contentPane);
}
```

Click here and press *Enter* to insert an empty line

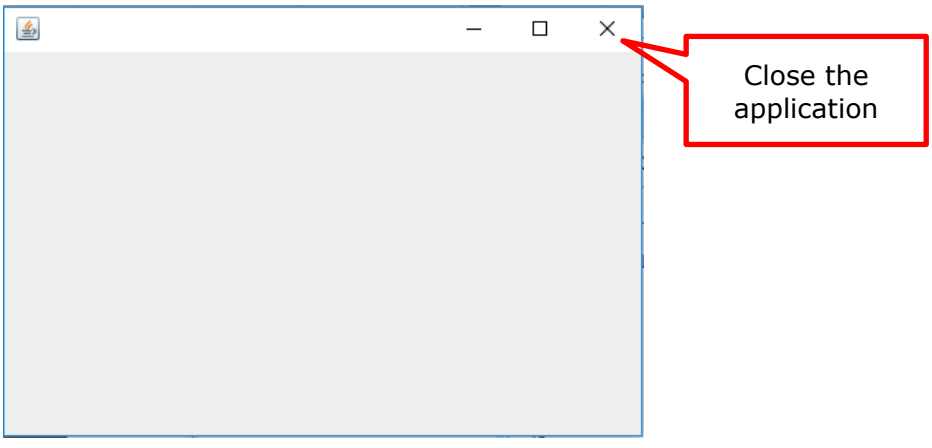
Run the program

To see the result, we can run the program: click on the green arrow in the menu bar.

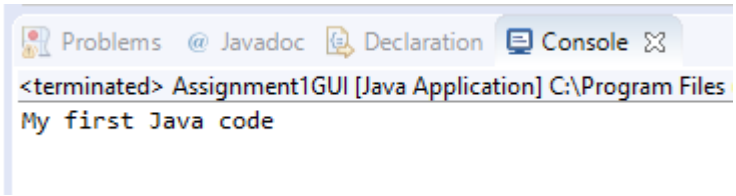


If you get a warning that the file was not saved, click *Ok*.

If you have not made any mistakes during programming, an empty Window appears that should look like the one below:



Also, a Console view will appear at the bottom in the Eclipse Window with your text message:



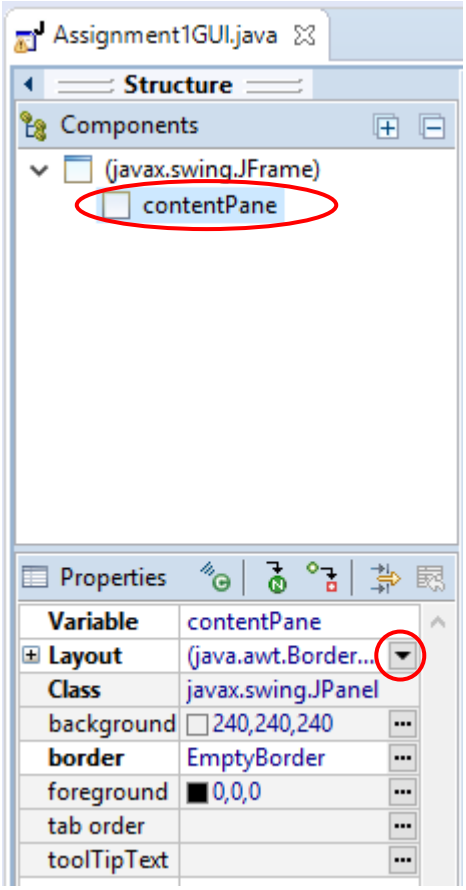
You may close your application.

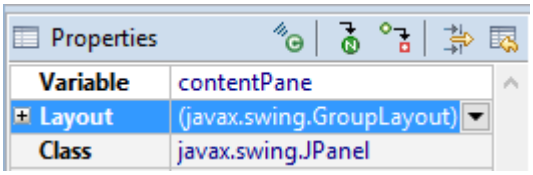
Now click on the *Design* tab in the *Editor*:

The screen that appears is also known as the WindowBuilder.
You can use this to create a user interface in a visual way (without writing code).
Again, you see a number Views, such as the *Structure*, which includes the user interface components and *Properties*, and the *Palette*.

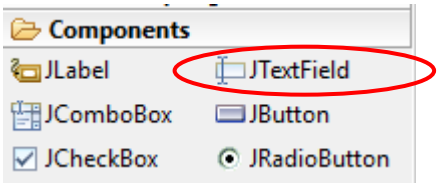
3. Add components to the user interface

Before we can add user interface components, we must select a layout. In the list of *Components* select the *contentPane* and in the *Properties* view, behind **Layout** select: "GroupLayout".

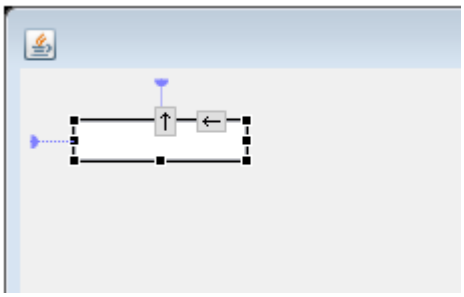




You are now going to insert a text field that is required to enter the name of the user.
Find the *JTextField* in the *Palette* (located under the heading *Components*) and select it by clicking on it with the mouse:



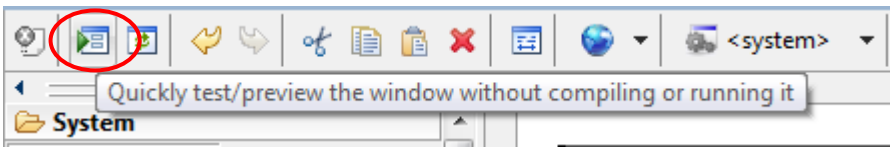
Now you can insert the text field in the WindowBuilder by clicking with the left mouse button where you want to insert the text field. The text field will appear at the specified location:



You can resize the text field by dragging the dots. You can also move the field by dragging it. The two blue dots and lines indicate that the side and the top of the text field are attached to the edge of the Window. This means that when the size of the window is changed, the text field changes accordingly. By clicking on the arrows, you can change the way it is attached.

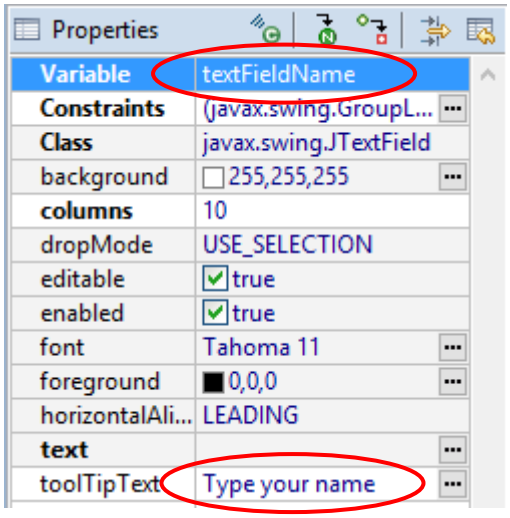
As long as the text field is selected you can adjust properties such as content (*Text*) and the variable name (*Variable*) at *Properties*.

You can get a quick preview of the user interface you have created, at the icons at the top of the editor by using the Preview button: 



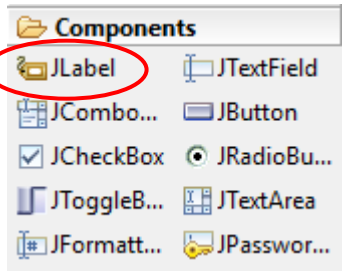
Tip: It is important that you immediately assign the right variable name to a component. Later on, you will notice that this greatly increases the readability of your code. So, give the text field a

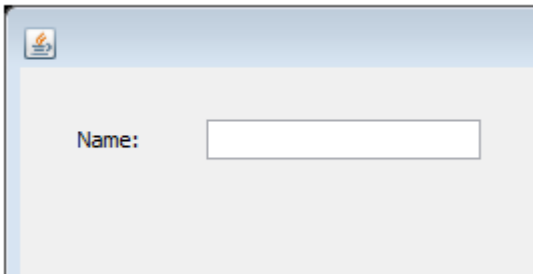
suitable name. We incorporate the type and the function in the name of the component. Then it becomes clear what it can do (type) and what it's for (function). The function of the text field is to output text. A good name would be "textFieldName". Change the **Variable** at *Properties*:



It may be useful to include a hint at the *toolTipText* to reflect the purpose of the text field.

To make the purpose of the text field even clearer, we place a Label in front of it:

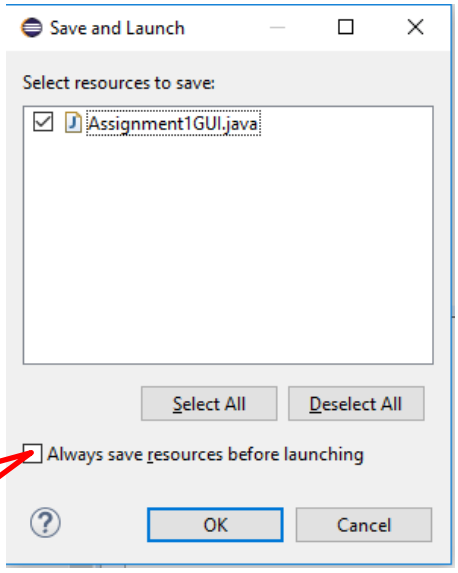




Run the program

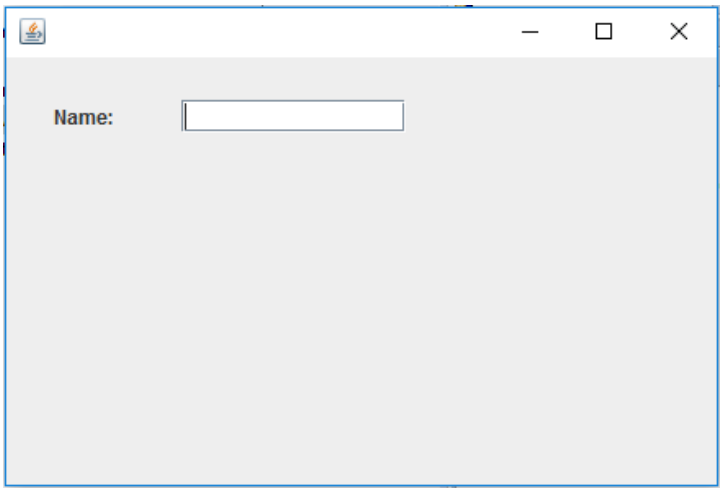
To see what the program looks like when you launch it, click on the green arrow in the menu bar.

You will probably see a warning similar to the one on the right because you have not saved the Java file. Select the option "Always save …" and click on OK.



Select this option if you do not want this message all the time

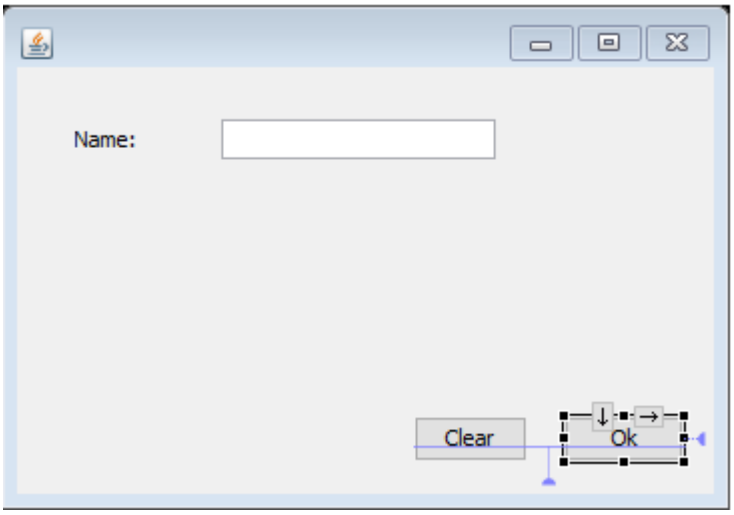
If you have not made any mistakes during programming, a screen appears that should look like this:

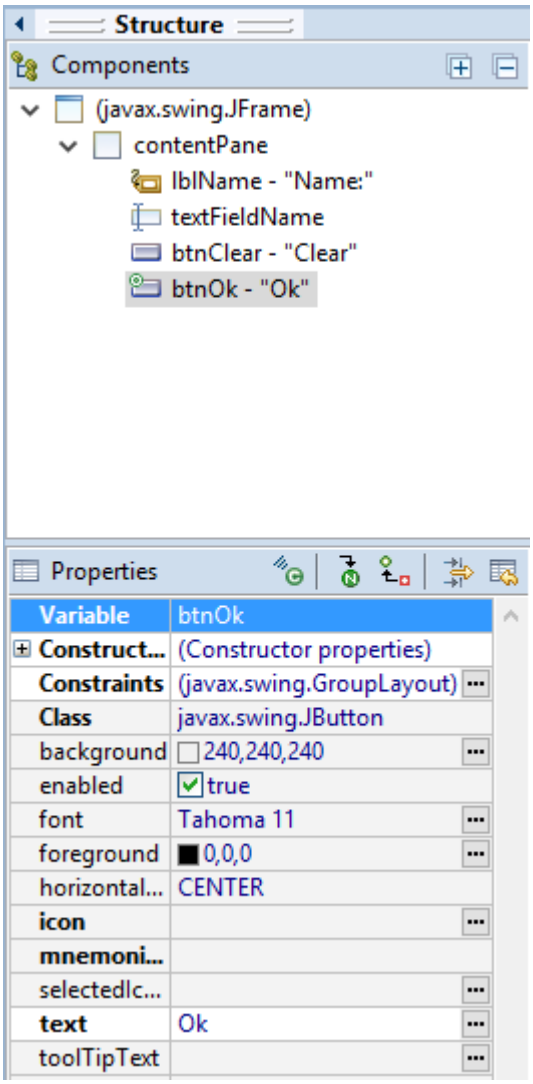


By clicking on the cross, you exit the program again and return to where you left off in Eclipse.

You may have noticed that there are one or several warnings visible in the *Problems* View. You may ignore these for now. Should there be any error messages, you will have to (try to) solve these.

Now place two buttons (JButton) in the user interface in the same way you did with the text field. Customize **text** and **Variable** of both. It will look like this, for example:





Notice the Properties of the selected button:

Did you enter proper **Variable** names for both buttons?

4. Have buttons do something

You have used the WindowBuilder to design the user interface of your program. This is an easy way to quickly create a user interface because Eclipse generates the Java code for you. In the second part of this assignment you must make sure that the user interface actually does something by adding Java code.

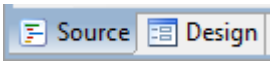
The Java code to be run can be specified in an Event Handler. An Event Handler 'handles' an Event. In this case, a click on the button. You can have Eclipse create such an Event Handler by double-clicking on a button. Do the following: Double-click the "Ok" button in the WindowBuilder.

Once you have done this, a piece of Java code is added, which is used to create the Event Handler:

```
JButton btnOk = new JButton("Ok");
btnOk.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent arg0) {
    }
});
```

Click here and press *Enter* to add a new line of code.

At the moment you click the button, Eclipse switches to the source code Editor, where you can view and edit the Java code of your application. You can switch between the source code Editor and the WindowBuilder (Design) by using the tabs at the bottom.



Now we are going to add a line of Java code to the Event Handler. Click on the place indicated above in the editor behind the curly bracket { and press *Enter*.

An empty line appears where you can type the Java code. We are going to use this text field to enter the name of a person. It is used as an input field.
In order to save the name of the person, we use a variable of type String. We declare this as follows:

```
JButton btnOk = new JButton("Ok");
btnOk.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent arg0) {
        String name
    }
});
```

Next, type a '=' sign, followed by the variable name of the text field (if necessary, check the correct variable in the code above), followed by a dot and then the text "getText":

```
JButton btnOk = new JButton("Ok");
btnOk.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent arg0) {
        String name = textFieldName.getT
    }
});
```

Enable Subtypes Completion?
getText() : String - JTextComponent
getText(int arg0, int arg1) : String - JText
getToolTipText() : String - JComponent

Click once to view info, double-click to accept that suggestion.

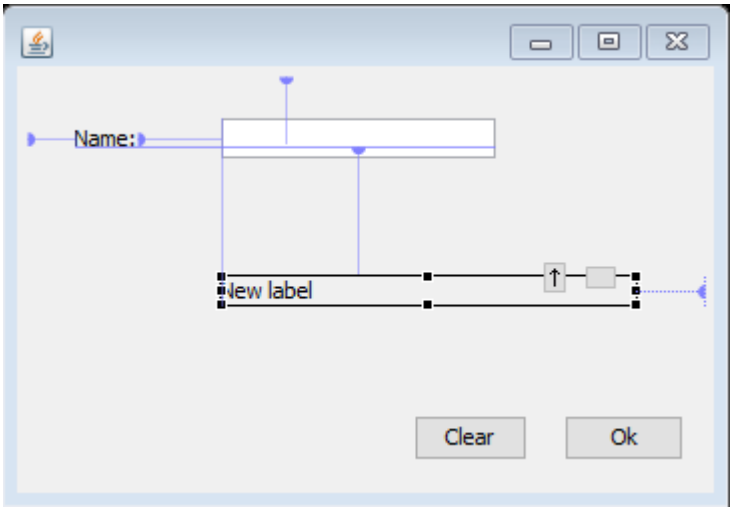
As you type it, a popup with suggestions from the Editor will appear. The Editor recognizes the name of the variable and will make corresponding suggestions. In this case, the Editor recognizes that you have used the name of a text field and will prompt you to use the properties and

methods of that field. Documentation of those properties and methods also appears. We cover this in more detail as part of the next assignments.

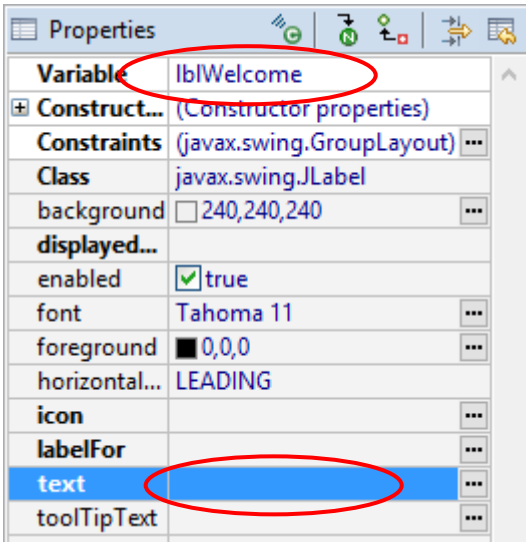
For now, you may finish the Java code as follows:

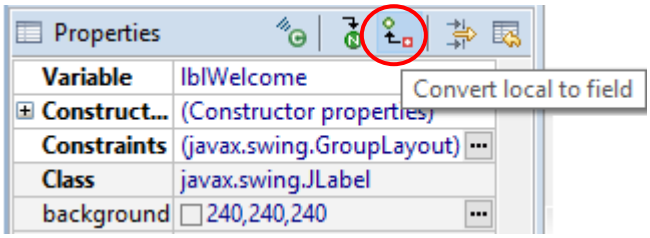
```
JButton btnOk = new JButton("Ok");
btnOk.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent arg0) {
        String naam = textFieldName.getText();
    }
});
```

We are going to show the name entered in a label in the user interface. To do this, add a label (switch to *Design* tab!):



Make sure the label is empty (it does not contain any text) by deleting the text 'New Label' at ***text*** in *Properties*. Give the label a meaningful name:





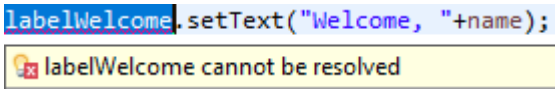
Also click on the "Convert to local field" icon. This converts the label into a field that we can use as output in the entire class (more about this later).

Now double-click again on the OK button to go to the code of the Event Handler and insert the second line of code:

```
JButton btnOk = new JButton("Ok");
btnOk.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent arg0) {
        String name = textFieldName.getText();
        lblWelcome.setText("Welcome, "+name);
    }
});
```

Please note: if you have given the label a different name, you must use that name here!

- If you get an error (*labelWelcome cannot be resolved*), go to the previous page: you might have missed to convert the label to a local field. Or your label might have a different name.

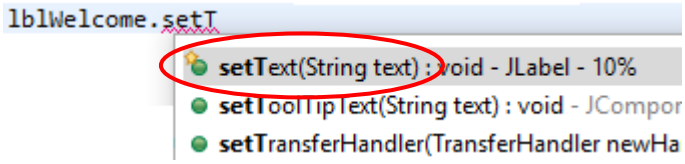


Now run your program using the Run button and see what happens when you click on the *Ok* button.

Do you understand the two lines of code you have added to the Event Handler? If not, check the lecture notes.

Now add an Event Handler for the Clear button in the same way. Double-click on this button in the WindowBuilder and type the Java code. When this button is clicked, the contents of the text field and the label for the welcome-message should both be cleared. We use the method *setText()* of the label, with an empty string between the parentheses to accomplish this:

First type the name of the text field, followed by a dot and then double-click on the method you need:



If the setText method does not appear as a hint, keep typing until it appears.

```
JButton btnClear = new JButton("Clear");
btnClear.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        lblWelcome.setText(""); // clear the welcome-label
    }
});
```

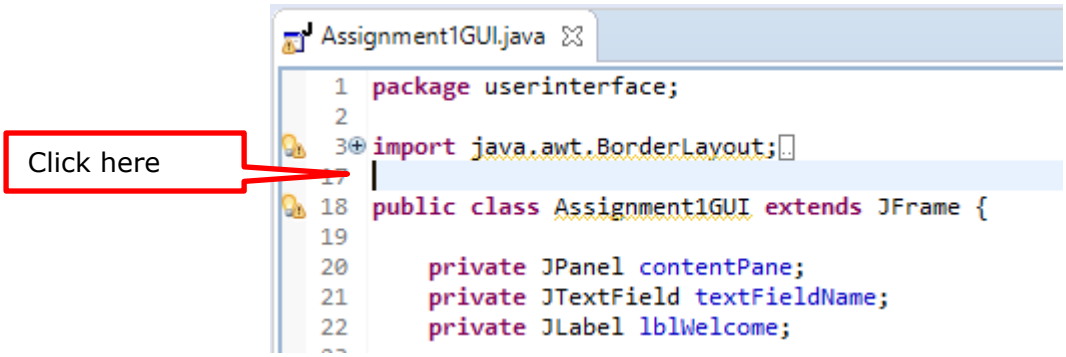
Please notice the comment which is added to the end of the line.

Do the same for the textfield also.

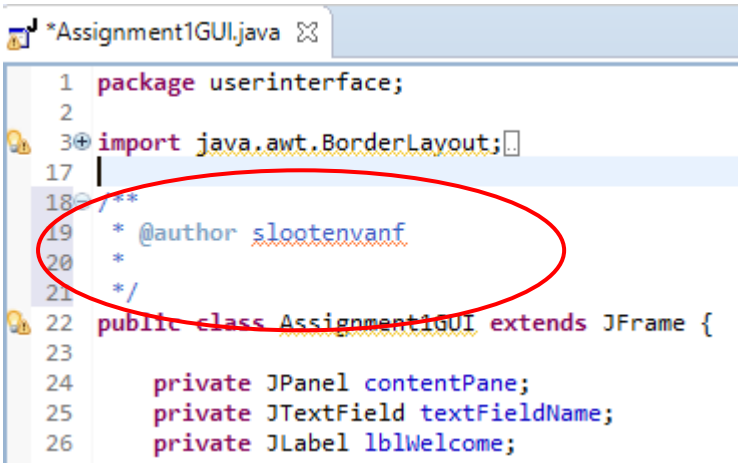
Run the program and check if the text field and the label are cleared when clicking on the Clear button.

Document the application

Comments are important means to document a program. We will add a comment at the top of the Java file, which we use to add author information and a general description of the program. Click at the top of the Editor on the empty line above the definition of the class:

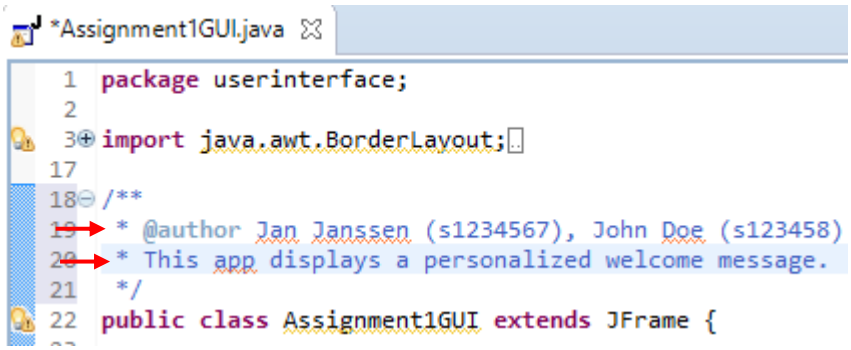


Select *Source* > *Generate Element Comment* from the menu. A Javadoc-styled comment will appear above the class definition:



To enter Javadoc comments, you may also type **/**** and press Enter.

Change the comment at the specified location so that your name(s) and student number(s) are placed there (behind `@author`). Also type a brief description of the program on the next line. It will look similar to:



Customize the app

You may now further customize your program, for instance with different text that appears in the text box and a different title*. Adjust the layout as well: e.g. different size, font or colors of components.

* The title of the application is a property of the JFrame. Go to the WindowBuilder (*Design*) and select the JFrame (javax.swing.JFrame) at *Components*. You can then look up and edit the relevant property in *Properties.*

You can customize the background color of the application by changing the *background* of the contentPane.

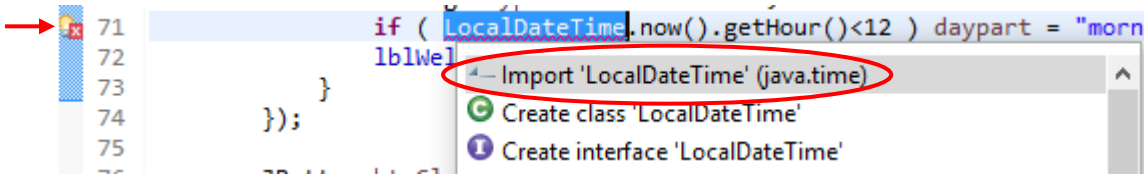
Extra challenge

If you have some time left, you may add this if you like.

Time-based welcome
Suppose, instead of using "Welcome," you want to welcome the user with a custom message according to the time of day. For example, "Good morning" if it is morning. In that case, customize the event handler of the Ok button as follows:

```
String name = textFieldName.getText();
String daypart = "afternoon";
if ( LocalDateTime.now().getHour()<12 ) daypart = "morning";
lblWelcome.setText("Good "+daypart+", "+name);
```

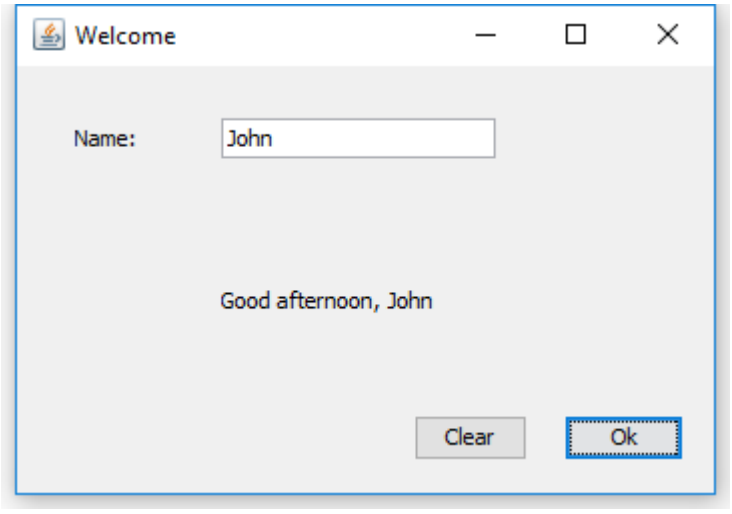
By calling the method `LocalDateTime.now().getHour()`, we can retrieve the current time, after which we determine if it is morning by using an if-statement.
If you get a warning-bubble in the side-line, click it to import the appropriate library:



Activate Eventhandler of Ok-button when user hits ENTER
You might have noticed that hitting the ENTER key on the keyboard does nothing. It is custom that hitting ENTER will activate the default button (the Ok-button).
To realize this, add the following code at the end of the userinterface code (method Assignment1GUI()):

```
// make btnOk the default button when ENTER is pressed:
getRootPane().setDefaultButton(btnOk);
```

Result:



Summary

On this first day you have familiarized yourself with Eclipse and written your first piece of code. In addition, you have learned the following.

- Create a project.
- Write a Java program in a project.
- Use the WindowBuilder to create a user interface.
- Run a program to check if it works the way you intended.
- Format a user interface.
- Written your first lines of Java code.
- Realized method calls.
- Added author details and documentation.

For the following assignments, it is important that you have practised with Eclipse and the examples, for instance from one of the books. Therefore, as extra practice do the exercise in the appendix on the next page, which explains how to use an example from the book in Eclipse.

Appendix 1: Using examples in Eclipse.

Both books ("Head first java" and "Aan de slag met Java") come with a lot of examples. Examples from both books are included in the zip-file that comes with this course.

Getting an example from a book in Eclipse involves the following four general steps. The steps will be explained in detail below.

1. Create an (empty) Java Project
2. Locate and copy .java file(s)
3. Paste .java files in the Eclipse project' **src** folder
4. Run the program

Close any other projects that are still open via *File* > *Close All*.

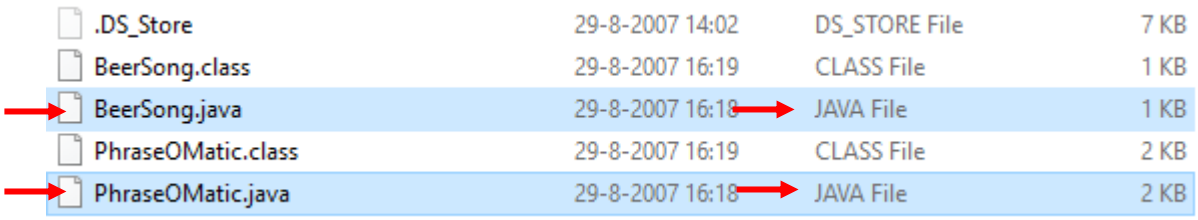
Step 1: Create an (empty) Java Project

Select *File* > *New* > *Java Project* from the menu. Give the project a name, for example "Examples".

Step 2: Locate and copy .java file(s)

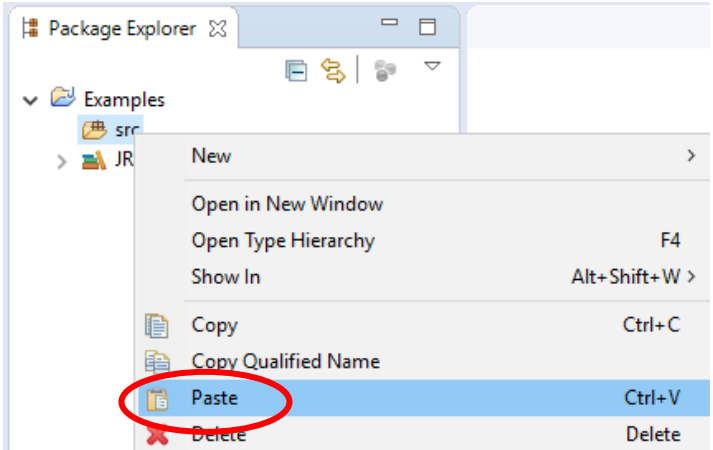
Use Windows Explorer to browse to the folder containing the examples. In the ZIP file that comes with this course, they are located in the folder "book"**.** Next, find a sub-folder which contains the actual code of the examples.
For instance, **book\Head First Java\chap01** contains the examples of chapter 1 of "Head first java".
And **book\Aan de slag met Java\Voorbeelden Broncode\H02\Vb0204\src** contains example 0204 of chapter 2 of "Aan de slag met Java".

Select only the java files, then select *Copy (CTRL+C)*:

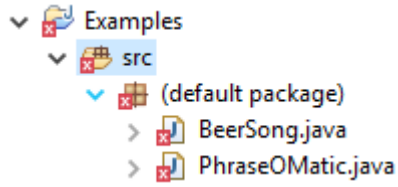


Step 3: Paste .java files in the Eclipse project' src folder

Switch to Eclipse. Paste the files in the ***src*** folder by clicking it with the right-mouse button and select Paste (or CTRL+V):

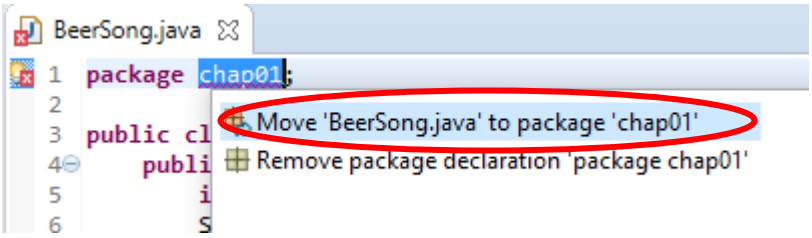


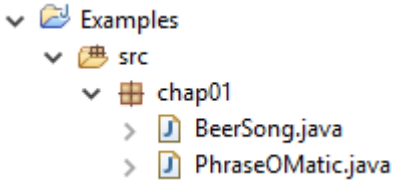
If you expand the *src* folder, you see that both files appear under *default package.* It is possible that errors occur:



> The first line of a .java file may contain a package-definition. It defines the name of the package that file is supposed to be in. If no name is given "default package" is used.
>
> In this case: BeerSong.java is part of package "chap01".

To fix the errors, open the first Java file. Click the light-bulb with the error and select (double click) the first option to move it to the right package:





Now the file should be Ok.
Repeat this for the other .java files (with errors).

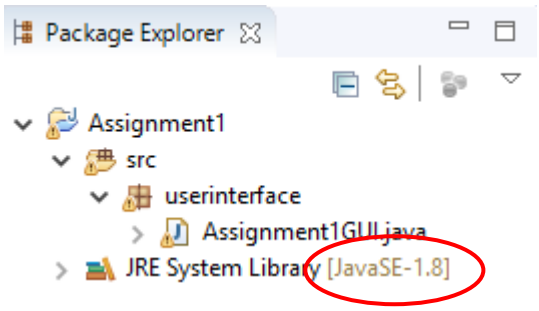
Step 4: Run the program

Hitting the Run button will start the program of which the .java file is currently open.

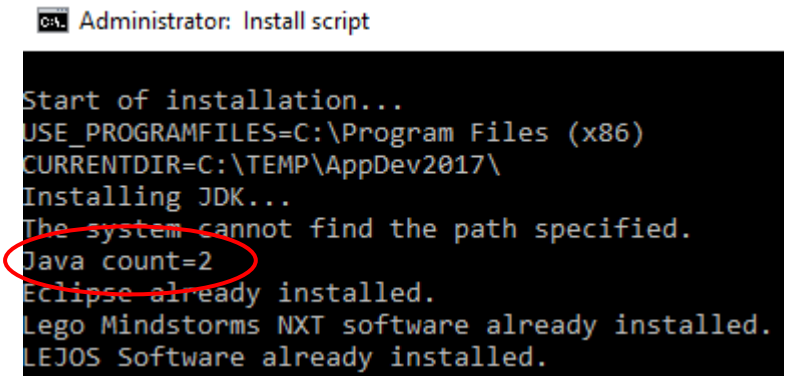
However, if an example consists of multiple .java files, you will have to find the .java file which contains the main-method (and Run that).

Appendix 2 Common problems

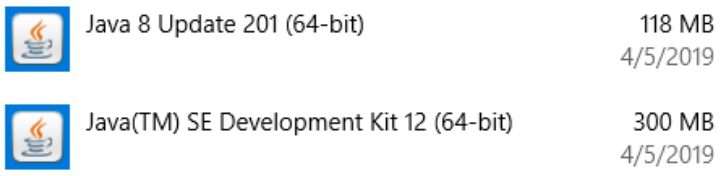
In the first step of the assignment (create project) check whether the correct Java version is used. We use JavaSE version 1.8. You can see this in the package explorer:



For example, if the WindowBuilder is causing problems in step 3 of this assignment, then make sure you do not have multiple installations of Java on your computer. This is checked by the installation script (install.bat). In the text-output of the script, the "Java count" number should be 2:



For Eclipse to work properly you should have installed the JDK and the Java Runtime Environment (both 64-bit). In the list of installed programs (*) you will see 2 lines like this:



(* Type 'programs' in the search field of the taskbar, then choose 'Programs and Features')

So, no other or additional versions, and update 201 (or higher).

If you do have other, more or older versions, remove these, leaving only the above version. Then restart your computer.

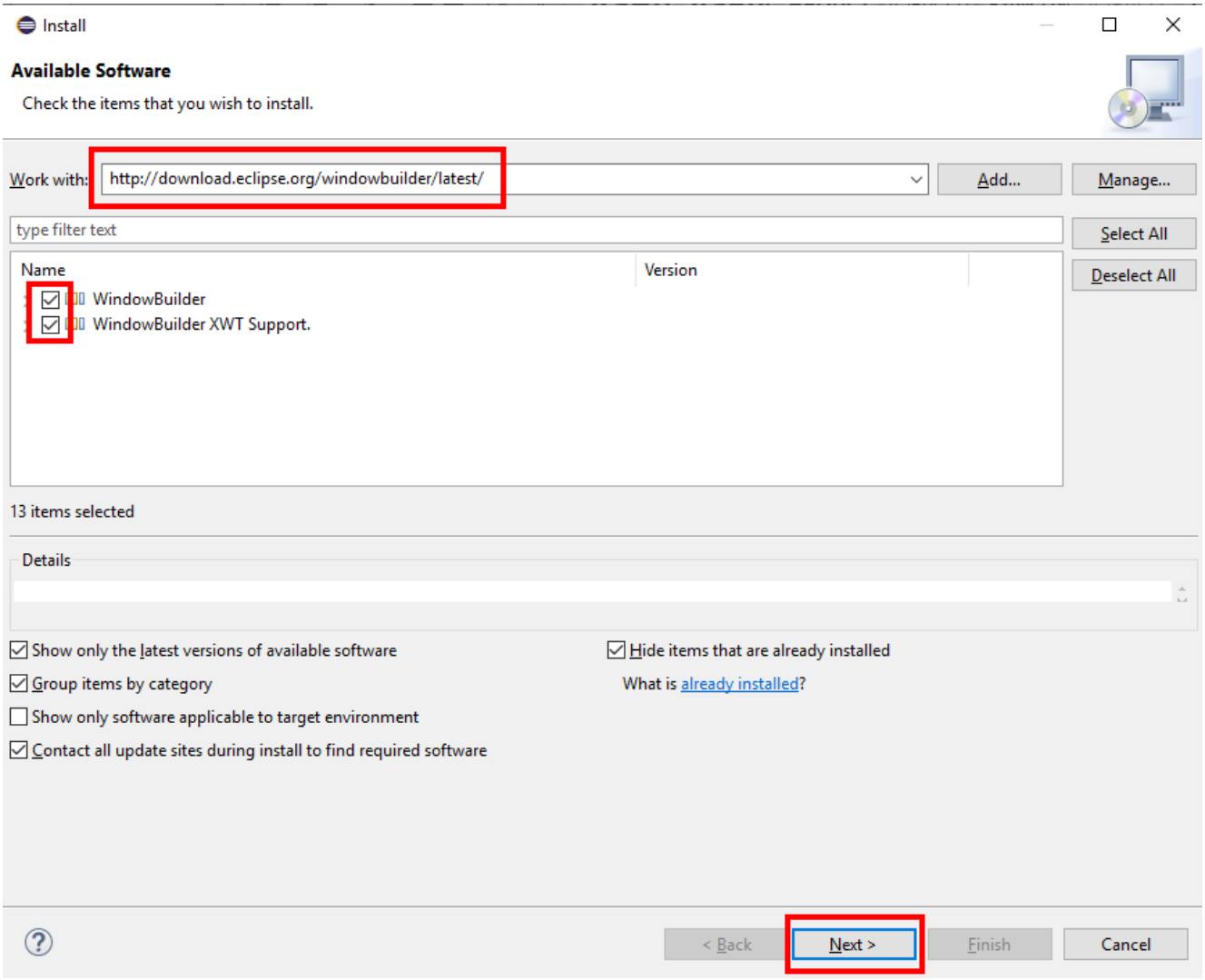
If after the above check, the WindowBuilder still causes problems, such as when switching to the Design tab, try the following: Restart your computer.

Appendix 3 Install Window Builder

The tool to create graphical user interfaces is not part of all editions of Eclipse, and might have to be added. Choose *Help > Install New Software...* In the field *Work with*, enter this address:

http://download.eclipse.org/windowbuilder/latest/

(press Enter), and select *WindowBuilder* and *WindowBuilder XWT Support*:



To start the installation press *Next* two times. Accept the license, then *Finish*. The installation will continue in the background. Wait for it to finish. Answer *Yes* if asked to restart Eclipse.