Name: John Good afternoon, John
Good afternoon, John
Good afternoon, John
Clear Ok

Content Application Development Day 1

<u>Lecture</u>

The lecture provides an introduction to programming, the concept of classes and objects in Java and the Eclipse development environment.

<u>Tutorial</u>

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. <u>It is not allowed to complete assignments with 3 or more people</u> and without any further checks this will automatically result in an unsatisfactory mark.

Have your assignments checked by the lecturer or assistant <u>no later than at the next lecture</u> (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.

Enter project name

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.

New Java Project

Create a Java Project

?

< <u>B</u>ack

Create a Java project in the work

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.

Project under the Workspace folder uments folder. Project, select File > New > Java e menu. A Wizard will appear, as pure below. Here you can enter the

.

Project layout	Ch	eck	angare mean
O Use project folder as root O Create separate folders for	or sources and class files	Conf	igure default
Norking sets			
Add projec <u>t</u> to working s	ets		Ne <u>w</u>
W <u>o</u> rking sets:		~	S <u>e</u> lect
wighting sea,			<u>Selectin</u>

<u>N</u>ext >

<u>F</u>inish

Cancel

* 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:

eclipse-workspace - Eclipse IDE



The Eclipse screen should look like below:

🖨 Eclipse - Eclipse			— C) X
<u>File Edit Source Refactor Navigate Se</u>	<u>a</u> rch <u>P</u> roject <u>R</u> un <u>W</u> indow <u>H</u> elp			
📑 ▼ 🖆 ▼ 🔚 🕼 🗿 ▼ 🔌 🕸 ▼ 🗿 ▼ 🤇	▙ ▼ ٩▙ ▼ # 6 ♥ @ @ # @ @ # ▼ ₪ ▼ 初 ▼ や ↔ → →		Quick Acce	ess 🗄 😫 🕷
Image: Image	} ▼ 0, ▼ # 3 3 ▼ ≫ ⊘ 		Quick Acce ■ Task List ×	ess i fi i i i i i i i i i i i i i i i i
0	Problems গ্র @ Javadoc 🔒 Declaration items	Decourse		p ⊽ □ □
	Jeschpuon	Nesource Pat		LOCATION
	c			>
Assignment1				2

Assignment i

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*:

New	—		ĸ
Select a wizard		\rightarrow	
Create an empty JFrame			
Wizards:			
type filter text			
> 🗁 Maven			^
> 🔁 Oomph			
> > Plug-in Development			
> > User Assistance			
🤄 🗁 WindowBuilder			
S Project Palette			
Swing Designer			
i Applet			
IDialog			
JFrame			~
? < Back Next > Finish		Cancel	

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*:

New JFrame	_	
Create JFrame Create an empty JFrame.		J
Source folder: Assignment1/src		Browse
Package: userinterface		Browse
Name: Assignment1GU		
Superclass: javax.swing.JFrame		Browse
☑ Use advanced template for generate JFrame		
? < Back Next > Finish	\supset	Cancel

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*.

Editor 2017 - Java - Assignment1/src/userinterface/ clipse X File Edit Source Refactor Navigate Sear ndow Help Quick Access 🔡 🛃 📑 • 📑 • 🔚 👘 🔌 🚸 • 💽 • 🎭 • 🔐 🦉 🕒 👝 🛷 + i 🖞 + 🖓 + 🏷 🔶 -🚦 Package Explorer 🔀 📃 🗖 🗐 Task List 🛛 - -🚽 Assignment1GUI. ava 🖄 E 😫 😨 ∇ 1 package userinterface; ^ -👕 🕶 🞏 📽 🛛 🐄 🙀 📼 ✓ 🔊 Assignment1 ~ 3⊕ import iava.awt.BorderLavout: V 🕮 src 🔍 🕨 All 🕨 Activat... (Find ✓ → userinterface №10 public class Assignment1GUI extends JFrame { > J) Assignment1GUI.ja 11 12 private JPanel contentPane; > ➡ JRE System Library [JavaSE 13 * Launch the application. 15 16 public static void main(String[] args) {
 EventQueue.invokeLater(new Runnable() { 170 18⊝ Package 19⊝ public void run() { 20 try { Explorer Assignment1GUI frame = new Assignment1GUI(); 21 - 8 22 frame.setVisible(true); 📴 Outline 🔀 } catch (Exception e) 23 🝃 📄 📴 😿 🐋 👄 ∇ × 24 25 26 27 28 e.printStackTrace(); Outline 🖶 userinterface } ₽ Assignment1GUI } }); contentPane : JPanel } S main(String[]): void 29 > 😋 new Runnable() {... 300 C Assignment1GUI() * Create the frame. 31 32 330 public Assignment1GUI() { 34 setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); Tabs to toggle setBounds(100, 100, 450, 300); 35 between Views Problems, F Source 📃 Design < > Source and Javadoc and ~ - 8 🖳 Problems 💥 🏾 avadoc 🛛 🔃 Declaration Design Declaration 0 errors, 1 warning, 0 others separated by tabs. Description Type > 💧 Warnings (1 item) < > Writable 1:1 Smart Insert

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 `}'.



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*:

🚮 🖌	signment1GUI.java 😒	
140) /**	A 🗖
15	* Launch the application.	_
16	*/	
179	<pre>public static void main(String[] args) {</pre>	_
189	EventQueue.invokeLater(new Runnable() {	
△19⊝	public void run() {	
20	try {	
21	Assignment1GUI frame = new Assignment1GUI();	
22	<pre>frame.setVisible(true);</pre>	
23	<pre>} catch (Exception e) {</pre>	
24	e.printStackTrace();	
25	}	
26	}	
27	});	
28	}	
_29		
300) /**	
31	* Create the frame.	
32	*/	
336	<pre>public Assignment1GUI() {</pre>	
34	<pre>setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);</pre>	
35	setBounds(100, 100, 450, 300);	
36	contentPane = new JPanel();	
37	<pre>contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));</pre>	
38	<pre>contentPane.setLayout(new BorderLayout(0, 0));</pre>	
39	setContentPane(contentPane);	
40	System.out.println("My first Java code");	
41	}	
42		
43	}	×
		>
📻 So	purce 😑 Design	

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".

Properties	🍬 🔓 💿	
Variable	contentPane 🔨	
■ Layout	(javax.swing.GroupLayout) 💌	
Class	javax.swing.JPanel	

You are now going to insert a text field that is required to enter the name of the user.

Find the J*TextField* in the *Palette* (located under the heading *Components*) and select it by clicking on it with the mouse:

Components	-	
🔄 JLabel 🧲	[☐ JTextField	>
📳 JComboBox	JButton	
JCheckBox	 JRadioButton 	

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:

4	
•	E-

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: \boxtimes

9 🗊 🖉 💝 🖓 🗈 🖻	🗙 📰 🌚 🕶 🐺 <system> 💌</system>			
 Quickly test/preview the window without compiling or running it 				
🗁 System				

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 © University of Twente 9

Assignment	:1GUI.java 🔀	
Structure ===		
🍾 Componen	ts	Ŧ.
v [javax.s	wing JFrame itentPane	>
Properties	*• 7 °	┣ ♣ 國
 Properties Variable 	ିତ 🗟 ° contentPane	
 Properties Variable Layout 	G ontentPane (java.awt.Bo	2 2 2 2 2 3 2 3 3 3 3 3 3 3 3 3 3 3 3 3
 Properties Variable Layout Class 	ContentPane (java.awt.Bo javax.swing.	'금│ ╬ e rder▼
 Properties Variable Layout Class background 	ContentPane (java.awt.Bo javax.swing. 240,240,240,240	Pal > E
 Properties Variable Layout Class background border 	ContentPane (java.awt.Bo javax.swing 240,240,24 EmptyBorde	Parel IPanel IPanel IP
Properties Variable Layout Class background border foreground	Image: Weight of the second s	2 → R rder IPanel 10 17
Properties Variable Layout Class background border foreground tab order	Image: Weight of the second secon	Parel IP anel 40 Ir

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*:

Properties	🍖 🚡 °a 🛱 🦝
Variable 🔵	textFieldName
Constraints	(Javax.swing.GroupL •••
Class	javax.swing.JTextField
background	255,255,255
columns	10
dropMode	USE_SELECTION
editable	✓ true
enabled	✓ true
font	Tahoma 11 🛛 💀
foreground	■ 0,0,0
horizontalAli	LEADING
text	
toolTipText	Type your name 🔵 🚥

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:

	<mark> Componen</mark>				
\langle	🔁 JLabel	🛅 JTextField	£		
	JCombo	JButton			
	✓ JCheckBox	⊙ JRadioBu		Name:	
	📗 JToggleB	JTextArea			
	Ĩ∰ JFormatt	🤤 JPasswor			

Run the program

To see what the program looks like on the green arrow 🖸 in the menu

To see what the program looks like	Save and Launch			\times	
on the green arrow 🖤 in the meni	u bar.	Select resources to save:			
You will probably see a warning sin because you have not saved the Ja "Always save" and click on OK.	milar to the one on the right ava file. Select the option	Assignment1GUI.jav	/a		
		Select All		Deselect /	AII
	Select this option if you	Always save <u>r</u> esources be	fore lau	nching	
	do not want this message all the time	? ОК		Cance	:1

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

<u></u>		-	×
Name:			

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 ^(a) 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:

<u>\$</u>	
Name:	
	Clear Ok

Notice the Properties of the selected button:

Did you enter proper Variable names for both buttons?

 Structure 	ture 🚃						
🎦 Componen	ts	H					
 (javax.swing.JFrame) contentPane IblName - "Name:" textFieldName btnClear - "Clear" btnOk - "Ok" 							
E Deservation	↓ ∓	● →⊾ ===					
Properties	%⊖ ₹	2. [추 國					
 Properties Variable Construct 	[%] ⊖ ∎ btnOk	, £a 持 छ					
 Properties Variable Construct Constraints 	btnOk (Constructor proj	perties)					
 Properties Variable Construct Constraints Class 	btnOk (Constructor prop (javax.swing.Grou javax.swing.Butt	perties)					
 Properties Variable Construct Constraints Class background 	[®] ⊙ btnOk (Constructor prop (javax.swing.Grou javax.swing.JButto □240.240.240	perties) upLayout) on					
 Properties Variable Construct Constraints Class background enabled 	♥ ● ● btnOk (Constructor prop (javax.swing.Grou javax.swing.JButto □ 240,240,240 ♥ true	perties) upLayout) on					
 Properties Variable Construct Constraints Class background enabled font 	Image: Second structure Image: Second structure btnOk (Constructor prop (javax.swing.Grou javax.swing.Grou javax.swing.JButte 240,240,240 Image: Second structure 240,240,240 Image: Second structure Tahoma 11	perties) upLayout) on					
 Properties Variable Construct Constraints Class background enabled font foreground 	© btnOk (Constructor prop (javax.swing.Grou javax.swing.JButto ⊇240,240,240 V true Tahoma 11 ■ 0,0,0	perties) upLayout) ···· on ····					
 Properties Variable Construct Constraints Class background enabled font foreground horizontal 	Image: Weight of the system btnOk (Constructor prop (javax.swing.Group javax.swing.JButto 240,240,240 Image: Weight of the system 240,240,240 Image: Weight of the system Tahoma 11 Image: O,0,0 CENTER	perties) µpLayout) on 					
 Properties Variable Construct Constraints Class background enabled font foreground horizontal icon 	Image: Weight of the second structure Image: Weight of the second structure btnOk (Constructor proposed structure (javax.swing.Group (javax.swing.JButter javax.swing.JButter 240,240,240 Image: Weight of the second structure (javax.swing.JButter 1 1 Image: Weight of the second structure (javax.swing.JButter 1 1 Image: Weight of the second structure (javax.swing.JButter 1 1 Image: Weight of the second structure (javax.swing.JButter 1 1 Image: Weight of the second structure (javax.swing.JButter	perties) upLayout) on 					
 Properties Variable Construct Constraints Class background enabled font foreground horizontal icon mnemoni 	© btnOk (Constructor prop (javax.swing.Grou javax.swing.JButto ⊇240,240,240 ✓ true Tahoma 11 ■ 0,0,0 CENTER	perties) upLayout) on 					
Properties Variable Construct Constraints Class background enabled font foreground horizontal icon mnemoni selectedlc	Image: Weight of the second system btnOk (Constructor prop (javax.swing.Grouting.Grouting.Grouting.JButtoni	perties) µDayout) ···· on ···· ····					
 Properties Variable Construct Constraints Class background enabled font foreground horizontal icon mnemoni selectedlc text 	Image: Weight of the system btnOk (Constructor prop (javax.swing.Group javax.swing.JButto 240,240,240 Image: Weight of the system 240,240,240 Image: Weight of the system Tahoma 11 Image: Weight of the system 0,0,0 CENTER Ok	perties) µLayout) ···· on ···· ···					

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 btn0k = new JButton("0k");
btn0k.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
    }
});
});
Click once to view info,
    double-click to accept
    e getText(int arg0, int arg1): String - JText
    e getToolTipText(): String - JText
```

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:

Properties	4 ₀	ð 2.	静 🔖
Variable	lblWelcome	\supset	~
Construct	(Constructor	propertie	s)
Constraints	(javax.swing.(GroupLay	out) 😶
Class	javax.swing.JL	.abel	
background	240,240,240)	•••
displayed			
enabled	✓ true		
font	Tahoma 11		•••
foreground	0,0,0		•••
horizontal	LEADING		
icon			•••
labelFor			•••
text 🤇		\rightarrow	•••
toolTipText			•••

Properties	[≉] ⊝ ₹(€				
Variable	lblWelcome	Convert local to field			
E Construct	(Constructor properties)				
Constraints	(javax.swing.GroupLa	ayout) 🚥			
Class	javax.swing.JLabel				
background	240,240,240	•••			

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.

labelWelcome.setText("Welcome, "+name);

labelWelcome cannot be resolved

Now run your program using the Run button \bigcirc 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.

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 <u>Javadoc</u>-styled comment will appear above the class definition:



To enter <u>Javadoc</u>					
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:

```
*Assignment1GULjava ☆
1 package userinterface;
2
3⊕ import java.awt.BorderLavout;
17
18⊖ /**
19 * @author Jan Janssen (s1234567), John Doe (s123458)
20 * This app displays a personalized welcome message.
21 */
22 public class Assignment1GUI extends JFrame {
23
```

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

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

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

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:

	2000000				
-	23	71		<pre>if (LocalDateTime.now().getHour()<12) daypart =</pre>	"morn
		72		lblwel	
		73	}	Import LocalDateTime (Java.time)	<u>^</u>
		74	});	Create class 'LocalDateTime'	
		75		Oreate interface 'LocalDateTime'	
		7.0	70.11	11.61	

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:

실 Welcome		_		×
Name:	John			
	Good afternoon, John			
		Clear	Ok	

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".

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

.DS_Store	29-8-2007 14:02	DS_STORE File	7 KB
BeerSong.class	29-8-2007 16:19	CLASS File	1 KB
📕 BeerSong.java	29-8-2007 16:18	JAVA File	1 KB
PhraseOMatic.class	29-8-2007 16:19	CLASS File	2 KB
 PhraseOMatic.java	29-8-2007 16:18	JAVA File	2 KB

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):

🚦 Package Explore	er 🛛 🗖 🗖	
✓ ≥ Examples	Ē <mark>\$</mark> ē ⊽	
> 🛋 JR	New	>
	Open in New Window Open Type Hierarchy Show In	F4 Alt+Shift+W >
	Copy <u>Copy Q</u> ualified Name	Ctrl+C
Ē	Paste	Ctrl+V
7	Beiere	Delete

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

~	P	Exa	mp	les	
	\sim	₽	src		
		~	×	(de	fault package)
			>	Ð	BeerSong.java
			>	Ð	PhraseOMatic.java

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".

🚺 PhraseOMatic.java

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:

🛃 BeerSong.java 🙁	
1 package chap01; 2 Move 'BeerSong java' to package 'chap01'	
3 public c1 4⊖ publi	🗸 😂 Examples
5 i 6 S	✓ (巻 src ✓ ⊕ chap01
Now the file should be Ok	> 🚺 BeerSong.iava

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

Step 4: Run the program

Hitting the Run \bigcirc 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:

Administrator: Install script

```
Start of installation...
USE_PROGRAMFILES=C:\Program Files (x86)
CURRENTDIR=C:\TEMP\AppDev2017\
Installing JDK...
The system cannot find the path specified.
Java count=2
Eclipse already installed.
Lego Mindstorms NXT software already installed.
LEJOS Software already installed.
```

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:



Java 8 Update 201 (64-bit) 118 MB 4/5/2019 Java(TM) SE Development Kit 12 (64-bit) 300 MB 4/5/2019

(* 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:

Install		_	
Available Software Check the items that you wish to install.			
Work with: http://download.eclipse.org/windowbuilder/latest/	~	<u>A</u> dd	<u>M</u> anage
type filter text			<u>S</u> elect All
Name	Version		Deselect All
13 items selected			
Details			Â
Show only the latest versions of available software	✓ Hide items that are already installed		
Group items by category	What is <u>already installed</u> ?		
Show only software applicable to target environment			
☑ <u>C</u> ontact all update sites during install to find required software			
?	< <u>B</u> ack <u>N</u> ext >	<u>F</u> inish	Cancel

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.