LAUREA
UNIVERSITY OF APPLIED SCIENCES
Together we are stronger

Programming Graphical User Interfaces (GUI) with Java

1/2

Antonius Camara

GUI Libraries or toolkits

- Ready set of Classes representing graphical elements and their behavior
  - Screens, Windows, Panels, Dialogs
  - UI components (widgets) : Text boxes, Labels, Check boxes, Command buttons, Tables, Scroll boxes, etc…
- There are many GUI libraries available in Java
  - Abstract Window Toolkit (**AWT**): early library
  - **Swing**: Improvements over AWT, most commonly used
  - Standard Window Toolkit (**SWT**): IBM version of AWT
  - **JavaFX**: newer, targeting rich Internet apps and mobile apps. Not yet succeeding in replacing Swing. Competing with HTML/CSS frameworks in the web/mobile scene
  - Other toolkits: Specific purposes, 3rd parties, commercial

Swing Containers and Controls

- Containers and Controls are the building blocks to create the user interfaces
- Containers are used to hold components
  - A ”Canvas” where you can place the UI components
  - JFrame, JPanel, JWindow, JDialog, JApplet
- Components
  - Basic GUI entities with a specific purpose
  - Command buttons, text boxes, labels, check boxes, etc…

Swing components (1/2)

CLASS	USED AS	LOOKS LIKE
JButton	`JComponent component = new JButton("BUTTONS!");`	
JLabel	`JComponent component = new JLabel("A label");`	
JList	`JComponent component = new JList<String>( new String[]{ "---1---", "---2---" });`	
JProgressBar	`JProgressBar component = new JProgressBar(0, 100); component.setValue(20);`	
JScrollBar	`JComponent component = new JScrollBar( JScrollBar.HORIZONTAL, 50, 20, 1, 500);`	
JSlider	`JComponent component = new JSlider(0, 100, 33);`	
JSpinner	`JComponent component = new JSpinner();`	

Picture from: Beginning Java Programming: The Object-Oriented Approach, Baesens Bart, Wiley, 2015

Swing components (2/2)

CLASS	USED AS	LOOKS LIKE
JTextField	`JComponent component = new JTextField("Text field");`	Text field
JTextArea	`JComponent component = new JTextArea("Text area");`	Text area
JComboBox	`JComponent component = new JComboBox<String>( new String[]{ "---1---", "---2---" });`	---1---
JCheckBox	`JComponent component = new JCheckBox("Check boxes");`	Check boxes
JRadioButton	`JComponent component = new JRadioButton( "And radio buttons");`	And radio buttons

Picture from: Beginning Java Programming: The Object-Oriented Approach, Baesens Bart, Wiley, 2015

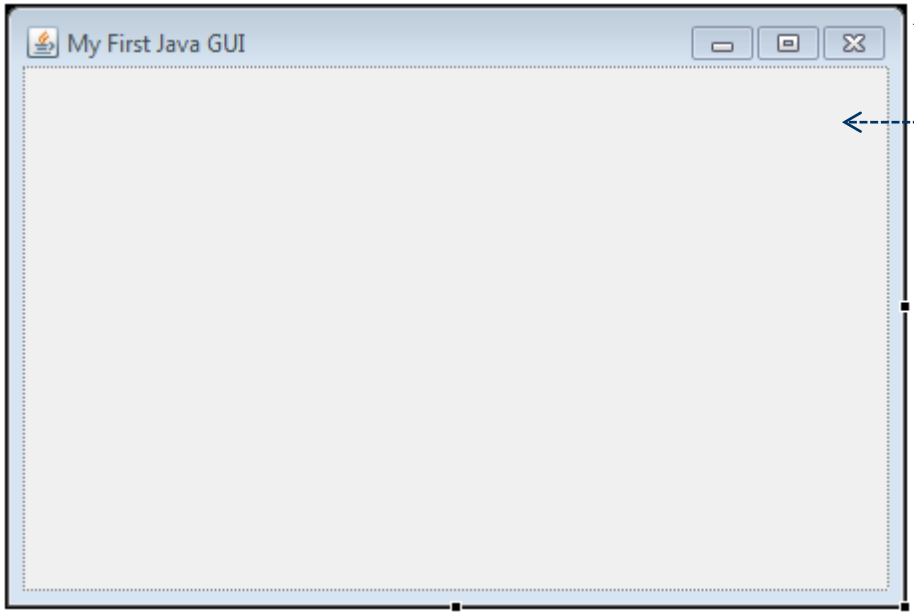
Layout Managers

- Classes that help us control how UI components are laid out in a container
- Layout managers automatically position and resize components to match different screen sizes and resolutions. Responsive UIs.
- When you create a Container, you specify a Layout Manager for it
- Swing Layout Managers:
  - Absolute position (no Layout Manager. “NULL”)
  - FlowLayout, BorderLayout, GridLayout, GridBagLayout, CardLayout, BoxLayout, GroupLayout, SpringLayout
- The effectiveness and ease of use of these layout mangers may vary a lot. Depending of the needs of your application you will need to use different layout managers
- You may also need to create hierarchies of Containers (containers inside containers) with different Layout Managers to achieve your needs
- Experienced programmers can also create their own Layout Managers
- Designing fully functional user interfaces with Layout Managers is *outside* the scope of this course. In this course it is enough you understand the key concepts related to Layout Managers.

First steps in building a GUI

1. Create a JFrame
   1. The JFrame contains a default container: “Root pane”
   2. Set some properties for the JFrame and its root container
2. Add elements to the root container
   1. Sub-containers
   2. UI Components
   3. Set properties for these elements
3. Add components to Sub-containers

JFrame and its root Container



JFrame properties

Properties

Class	javax.swing.JFrame
alwaysOnTop	false
autoRequestFocus	true
background	240,240,240
defaultCloseOperation	HIDE_ON_CLOSE
enabled	true
font	
foreground	
iconImage	
modalExclusionType	NO_EXCLUDE
opacity	1.0
resizable	true
tab order	
title	My First Java GUI
type	NORMAL

Container properties

Properties

Layout	(absolute)
Class	java.awt.Container
background	240,240,240
enabled	true
font	Tahoma 11
foreground	0,0,0
tab order	

Layout manager

Swing also uses Classes from AWT. Container is an AWT Class

Placing components in a Container

JLabel properties

Properties

Variable	lblSignIn
Constructor	(Constructor properties)
Bounds	(20, 11, 200, 27)
Class	javax.swing.JLabel
background	240,240,240
displayedMnemonic(ch...	
enabled	true
font	Tahoma 16
foreground	0,0,0
horizontalAlignment	LEADING
icon	
labelFor	
text	Welcome to this App
toolTipText	
verticalAlignment	CENTER

JTextField properties

Properties

Variable	textUsername
Bounds	(87, 50, 133, 20)
Class	javax.swing.JTextField
background	255,255,255
columns	10
dropMode	USE_SELECTION
editable	true
enabled	true
font	Tahoma 11
foreground	0,0,0
horizontalAlignment	LEADING
text	
toolTipText	

JButton properties

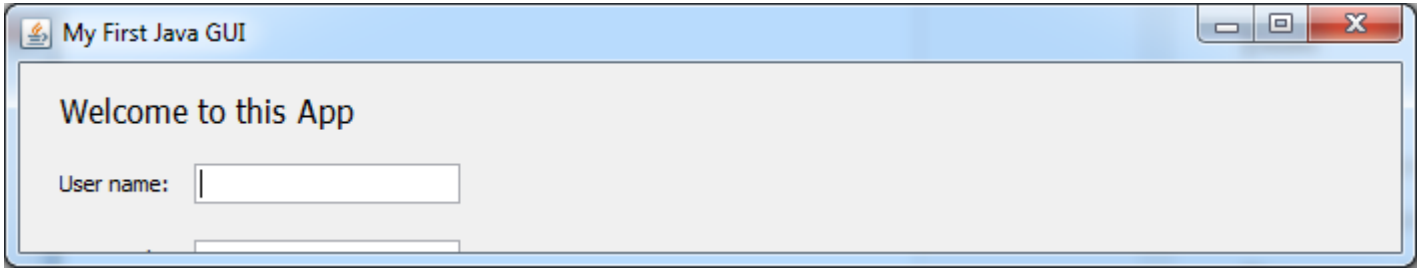
Properties

Variable	btnLogIn
Constructor	(Constructor properties)
Bounds	(20, 132, 89, 23)
Class	javax.swing.JButton
background	240,240,240
enabled	true
font	Tahoma 11
foreground	0,0,0
horizontalAlignment	CENTER
icon	
mnemonic(char)	
selectedIcon	
text	Log in
toolTipText	
verticalAlignment	CENTER

Container with no Layout Manager (Absolute Positioning)

Original dialog as designed by the programmer. The programmer manually coded the position of the UI elements

Resized screen: The UI elements remain in their original positions

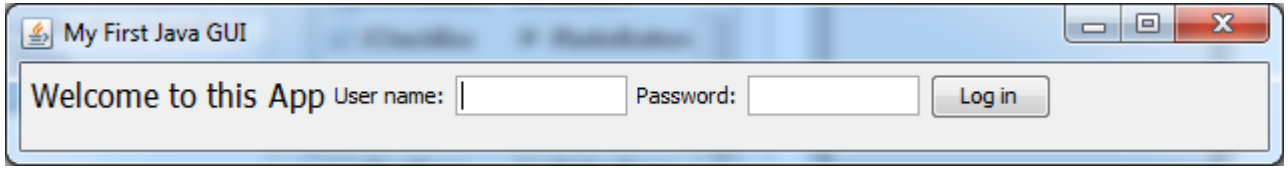


Container with a Layout Manager (Flow Layout)

Container with FlowLayout. In this example, elements are Left aligned. The Flow Layout tries to position all elements in a single row if the container size allows so.

Layout	(java.awt.FlowLayout)
Class	java.awt.FlowLayout
Constructor	(Constructor properties)
alignOnBaseline	false
alignment	LEFT
hgap	5
vgap	5
Class	java.awt.Container
background	240,240,240
enabled	true
font	Tahoma 11
foreground	0,0,0
tab order	

Resized screen: The UI elements are automatically repositioned by the Layout Manager



Note: As you see, the resizing does not have an ideal outcome. You will need to further design the UI so it works better. For example, the Username and Password field could start in a second row. To be able to optimally design the layout of your interface you may need to use sub-containers under the root container and assign different layout managers for different containers. Properly designing a user interface with Layout Managers is *outside* the scope of this course. In this course it is enough you understand the purpose of Layout Managers.

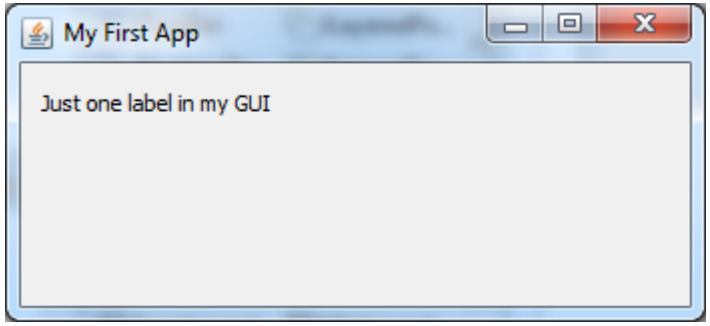
Coding the User Interface

LAUREA
UNIVERSITY OF APPLIED SCIENCES
Together we are stronger

- Using the methods available by the Container Classes and Component Classes
- In Eclipse, you can use the WindowBuilder plugin to help you designing (i.e “drawing”) the layout
  - Drag-and-Drop, WYSIWYG designer
  - WindowBuilder automatically generates the code
  - You will still need to “tweak” the code so it fits well to the overall architecture and standards of your code. Ex:
    - Renaming components’ variable names
    - Re-defining the scope of the variables generated by WindowBuilder
      - Local variables vs. Instance Variables vs. Class variables

Code example: MyFirstGUI.java

```java
import javax.swing.JFrame;
import javax.swing.JLabel; // After this line, WindowBuilder will add other imports related to other components you add there

// Change the name of this class to the name of your program
public class MyFirstGUI extends JFrame {
    // Here you can define class or instance variables needed in your application

    // This is the Class Constructor method. It has the same name as the Class name
    public MyFirstGUI (){
        super("My First App"); // Calling the constructor method of the superclass JFrame to create a JFrame with the given title

        setDefaultCloseOperation(EXIT_ON_CLOSE);
        getContentPane().setLayout(null); // Sets the JFrame's container with a null Layout Manager (Absolute positioning)
        setBounds(0,0,350,159); // Sets the size of the JFrame
        setLocationRelativeTo(null);  // Positioning the JFrame (Window) at the center of the computer's screen

        // Adding and positioning UI components in the container
        JLabel label1 = new JLabel ("Just one label in my GUI");
        label1.setBounds(10,10,150,20);
        getContentPane().add(label1); // After this line, WindowBuilder will add code of other components you add via WindowBuilder

    }

    // This is the entry point for your application
    public static void main(String[] args) {
        MyFirstGUI frame = new MyFirstGUI();
        frame.setVisible(true);
    }
}
```



Using Eclipse with WindowBuilder

LAUREA
UNIVERSITY OF APPLIED SCIENCES
Together we are stronger

Java project

Default package

Main Class of your app

antonius_eclipse - Java - MyFirstGUI/src/MyFirstGUI.java - Eclipse

File Edit Source Refactor Navigate Search Project Run Window Help

Quick Access <PHP>

Package Explorer

- Antonius-JavaOOP-App
- Antonius-JavaOOP-CarExamples
- Class examples
- Formative test
- Fundamentals of programming
- GUI test
- MultipleScreens
- MyFirstEvent
- MyFirstGUI
  - src
    - (default package)
      - MyFirstGUI.java
  - JRE System Library [JavaSE-1.8]
- ParkingApp
- php_examples
- school
- SessionApp
- SimplestGUI
- Tests
- Working with methods
- Working with methods - One class

MyFirstGUI.java

Search

```
1  import javax.swing.JFrame;
2  import javax.swing.JLabel; // After this line, WindowBuilder will add other imports related to other
3
4  // Change the name of this class to the name of your program
5  public class MyFirstGUI extends JFrame {
6      // Here you can define class or instance variables needed in your application
7
8      // This is the Class Constructor method. It has the same name as the Class name
9      public MyFirstGUI (){
10         super("My First App"); // Calling the constructor method of the superclass JFrame to create
11
12         setDefaultCloseOperation(EXIT_ON_CLOSE);
13         getContentPane().setLayout(null); // Sets the JFrame's container with a null Layout Manager
14         setBounds(0,0,350,159); // Sets the size of the JFrame
15         setLocationRelativeTo(null);  // Positioning the JFrame (Window) at the center of the comput
16
17         // Adding and positioning UI components in the container
18         JLabel label1 = new JLabel ("Just one label in my GUI");
19         label1.setBounds(10,10,150,20);
20         getContentPane().add(label1); // After this line, WindowBuilder will add code of other compo
21
22     }
23
24     // This is the entry point for your application
25     public static void main(String[] args) {
26         MyFirstGUI frame = new MyFirstGUI();
27         frame.setVisible(true);
28     }
29 }
30
```

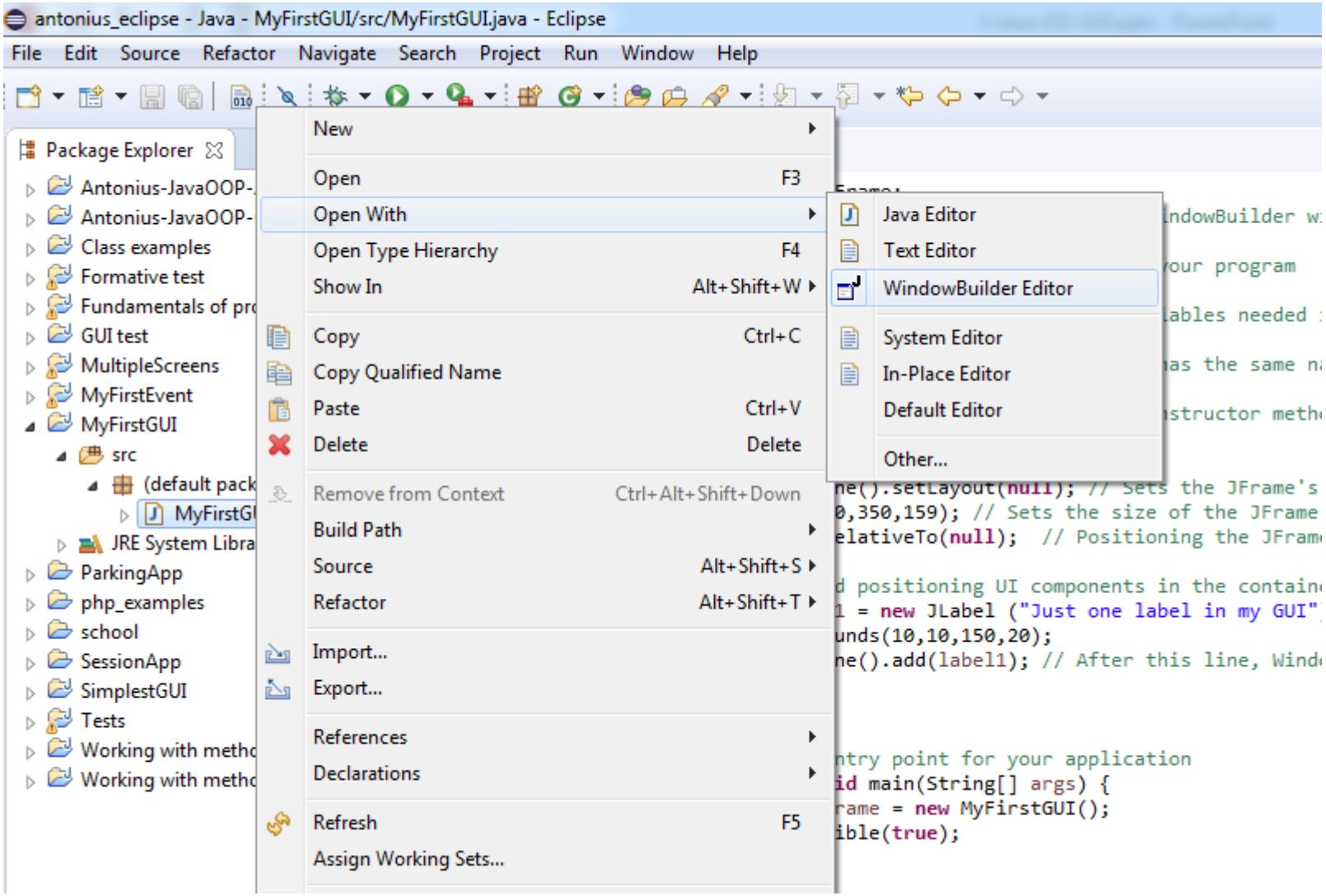
Source Design

Problems @ Javadoc Declaration Console

No consoles to display at this time.

Writable Smart Insert 30 : 1

Open your java file (MyFirstGUI.java) with WindowBuilder editor



Switch between source and design view

LAUREA
UNIVERSITY OF APPLIED SCIENCES
Together we are stronger

antonius_eclipse - Java - MyFirstGUI/src/MyFirstGUI.java - Eclipse

File Edit Source Refactor Navigate Search Project Run Window Help

Quick Access <PHP>

P... MyFirstGUI.java

- Antonius-Java(
- Antonius-Java(
- Class examples
- Formative test
- Fundamentals
- GUI test
- MultipleScreen
- MyFirstEvent
- MyFirstGUI
  - src
    - (default
      - MyF
  - JRE System
- ParkingApp
- php_examples
- school
- SessionApp
- SimplestGUI
- Tests
- Working with r
- Working with r

```
1  import javax.swing.JFrame;
2  import javax.swing.JLabel; // After this line, WindowBuilder will add other imports related to other components you
3
4  // Change the name of this class to the name of your program
5  public class MyFirstGUI extends JFrame {
6      // Here you can define class or instance variables needed in your application
7
8      // This is the Class Constructor method. It has the same name as the Class name
9      public MyFirstGUI (){
10         super("My First App"); // Calling the constructor method of the superclass JFrame to create a JFrame with th
11
12         setDefaultCloseOperation(EXIT_ON_CLOSE);
13         getContentPane().setLayout(null); // Sets the JFrame's container with a null Layout Manager (Absolute posit
14         setBounds(0,0,350,159); // Sets the size of the JFrame
15         setLocationRelativeTo(null);  // Positioning the JFrame (Window) at the center of the computer's screen
16
17         // Adding and positioning UI components in the container
18         JLabel label1 = new JLabel ("Just one label in my GUI");
19         label1.setBounds(10,10,150,20);
20         getContentPane().add(label1); // After this line, WindowBuilder will add code of other components you add v
21
22     }
23
24     // This is the entry point for your application
25     public static void main(String[] args) {
26         MyFirstGUI frame = new MyFirstGUI();
27         frame.setVisible(true);
28     }
29 }
30
```

Source Design

Problems @ Javadoc Declaration Console

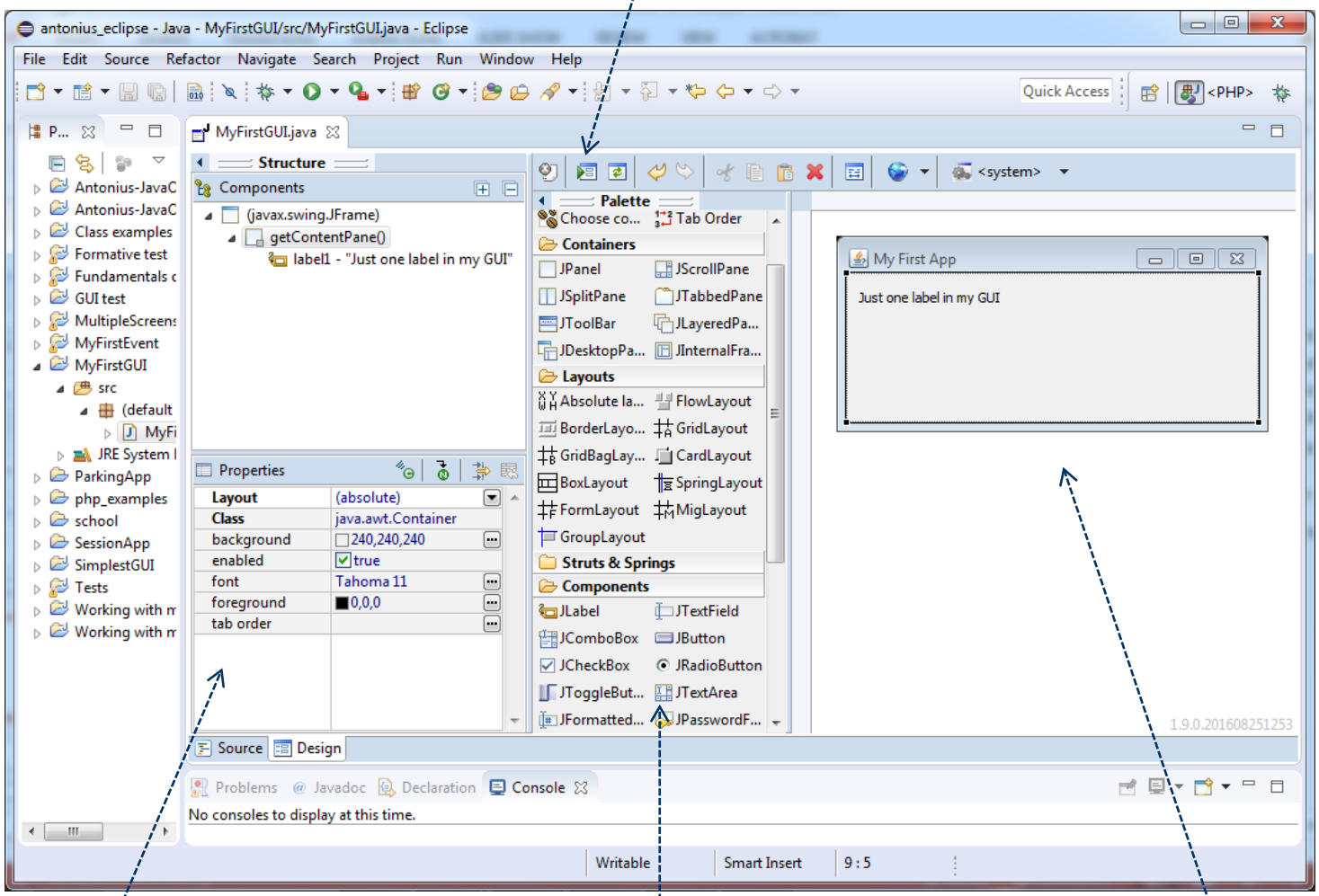
No consoles to display at this time.

MyFirstGUI.java - MyFirstGUI/src

Source View **Design View**

Design view

Test/Preview the window by clicking on this icon



Element Properties

Pick elements from these sections

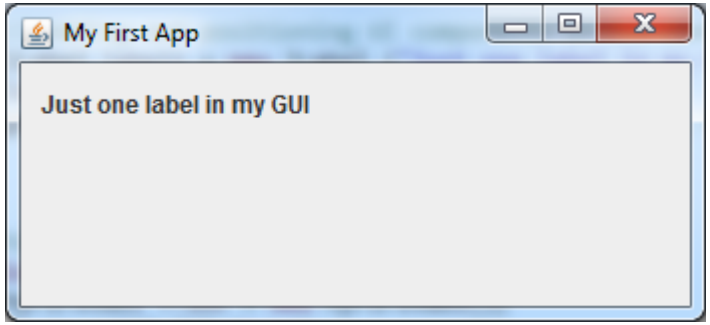
Design canvas, drag-and-drop elements here

Exercise - MyFirstGUI

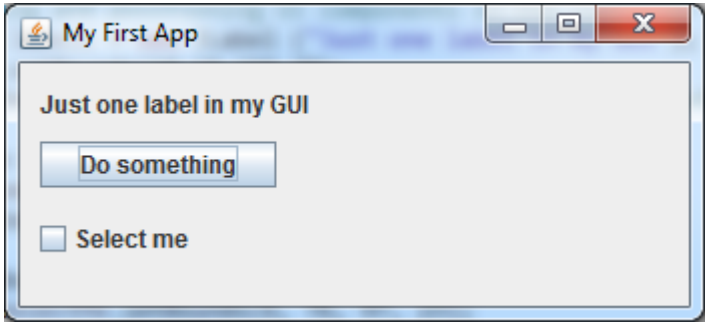
- Download MyFirstGUI.java from GitHub
- Create a project in Eclipse with the source code from MyFirstGUI.java
- Inspect the code and the user interface using WindowBuilder
- Via WindowBuilder's Design tool, add:
  - One command button with label "Do something"
  - One check box with label "Select me"
- Inspect the code to see the lines that were automatically added by WindowBuilder
- Run MyFirstGUI.java as a java application to see how the window looks and behave

Nothing happens when you click on "Do something", why?

Exercise - MyFirstGUI



Before



After

Event handling

- Event: An action taken by a user when interacting with the UI.
  - Clicking on a command button
  - Entering text in a text box
  - Moving the mouse over a UI component
- Applications interact with the user by reacting to these events
  - An application should contain methods called “Event Listeners” or “Event Handlers” to react to events and process them
  - Some event listeners are readily coded (built in) in a Swing component. Ex: Typing text in a text box
  - For some other events you will need to code the event listener. Ex: Do some action when the user clicks a command button
- To learn how to code an event listener you will need to refer to a book that explains the coding process in detail, with examples. Ex:
  - Beginning Java Programming: The Object-Oriented Approach, Baesens Bart, Wiley, 2015
  - Beginning Java 8 APIs, Extensions and Libraries, Kishori Sharan, Apress 2014

Example: Reacting to a command button (MyFirstEvent.java)

```
9  public class MyFirstEvent extends JFrame {
10     /* I can not declare the following components as local variables in the constructor MyFirstEvent()
11      * because I will need to access them in the event handling method of MyEventHandler Class
12      */
13     JLabel lblSay = new JLabel("");
14     JButton btnYes = new JButton ("Say \"Yes\"");
15     JButton btnSaySomething = new JButton("Say something...");
16
17     // This is the Class Constructor method. It has the same name as the Class name
18     public MyFirstEvent (){..
47
48     // This class will handle the Action events generated by both buttons in the GUI
49     private class MyEventHandler implements ActionListener
50     {
51         // In this method we will process the Action events generated by both buttons in the GUI
52         public void actionPerformed (ActionEvent myEvent)
53         {
54             if (myEvent.getSource() == btnYes){
55                 lblSay.setText("Yes");
56             }
57             else if (myEvent.getSource() == btnSaySomething){
58                 lblSay.setText("Java is fun!");
59             }
60             else {
61                 lblSay.setText("Some strange event was captured...");
62             }
63         }
64     }
65
66     // This is the entry point for the application
67     public static void main(String[] args) {
68         MyFirstEvent frame = new MyFirstEvent();
```

I coded this Class to handle the "ActionEvent" generated by the command button

The class should contain the method "actionPeformed" to process the ActionEvent

Example: Reacting to a command button (MyFirstEvent.java)

```
public class MyFirstEvent extends JFrame {
    /* I can not declare the following components as local variables in the constructor MyFirstEvent()
     * because I will need to access them in the event handling method of MyEventHandler Class
     */
    JLabel lblSay = new JLabel("");
    JButton btnYes = new JButton ("Say \"Yes\"");
    JButton btnSaySomething = new JButton("Say something...");

    // This is the Class Constructor method. It has the same name as the Class name
    public MyFirstEvent (){

    // This class will handle the Action events generated by both buttons in the GUI
    private class MyEventHandler implements ActionListener
    {
        // In this method we will process
        public void actionPerformed (Actio
        {
            if (myEvent.getSource() == btn
                lblSay.setText("Yes");
            }
            else if (myEvent.getSource() =
                lblSay.setText("Java is fu
            }
            else {
                lblSay.setText("Some stran
            }
        }
    }

    // This is the entry point for the app
    public static void main(String[] args)
        MyFirstEvent frame = new MyFirstEv
```

```
public MyFirstEvent (){
    super("My First Event"); /* Calling the constructor method of
    the superclass JFrame to create a JFrame with the given title */

    setDefaultCloseOperation(EXIT_ON_CLOSE);
    getContentPane().setLayout(null); // Sets the JFrame's container with a null La
    setBounds(0,0,436,200); // Sets the size of the JFrame
    setLocationRelativeTo(null);  // Positioning the JFrame (Window) at the center

    // Adding and positioning UI components in the container
    btnYes.setBounds(10,10,135,22);
    getContentPane().add(btnYes);

    btnSaySomething.setBounds(166, 9, 135, 23);
    getContentPane().add(btnSaySomething);

    JLabel lblYouAskedMe = new JLabel("You asked me to say:");
    lblYouAskedMe.setBounds(10, 62, 135, 14);
    getContentPane().add(lblYouAskedMe);

    lblSay.setFont(new Font("Tahoma", Font.PLAIN, 20));
    lblSay.setBounds(10, 87, 370, 78);
    getContentPane().add(lblSay);

    // Creating a MyEventHandler to handle events generated by the command buttons
    // Registering the event handler (MyEventHandler) for both buttons
    MyEventHandler commandHandler = new MyEventHandler();
    btnYes.addActionListener(commandHandler);
    btnSaySomething.addActionListener(commandHandler);
}
```

Here, you create a MyEventHandler object (commandHandler) and register it as an ActionListener for each button you want to react to

Explaining with a programming jargon

The action type **ActionEvent** is fired when you click on a JButton.
Your application should implement a Class of type **ActionListener** to handle the event.
In this Class you should define a method called **actionPerformed** to process the event.
The component that generates the event has a method called **addActionListener** that you should call to register your event handler for that object

LISTENER (INTERFACE): METHOD TO ADD	CAN BE USED WITH: PURPOSE	METHOD(S) TO IMPLEMENT
Action Listener (ActionListener) - addActionListener	Fires ActionEvent when user performs primary action on component. Can be used with JButton, JCheckBox, JComboBox, JRadioButton, and JTextField.	actionPerformed: Code that reacts to the action.

```
btnYes.addActionListener(commandHandler);
```

Some other event types in Swing

LISTENER (INTERFACE): METHOD TO ADD	CAN BE USED WITH: PURPOSE	METHOD(S) TO IMPLEMENT
Item Listener (ItemListener) - addItemListener	Fires ItemEvent whenever a state change occurs in the component's items. JCheckBox, JComboBox, and other components keep a list of items that accept this listener.	itemStateChanged: Code that reacts when state of items changes.
List Selection Listener (ListSelectionListener) - addListSelectionListener	Fires ListSelectionEvent when selection changes in the component's items. Only JList and JTable accept this listener.	valueChanged: Code that reacts when another item is selected from the component's items.
Mouse Motion Listener (MouseMotionListener) - addMouseMotionListener	Fires MouseEvent when the mouse pointer is dragged or moved over a component. All components accept this listener.	mouseDragged: Code that reacts when a user moves the mouse while holding a mouse button down. mouseMoved: Code that reacts when a user moves the mouse.

Source: Beginning Java Programming: The Object-Oriented Approach, Baesens Bart, Wiley, 2015

For a complete list check:
Beginning Java Programming: The Object-Oriented Approach, Baesens Bart, Wiley, 2015
Chapter: Understanding Eevents
Section: Event Listeners

Exercise: MyFirstEvent

- Download MyFirstEvent.java from GitHub
- Create a project in Eclipse with the source code from MyFirstEvent.java
- Inspect the code and the user interface using WindowBuilder
- Edit the code to add a phrase you would like the application to display when you click on “Say something”
- Run MyFirstEvent.java as a java application to check the results