TND002 Object-oriented programming

Lecture 11: Graphical User Interfaces 1

Mark Eric Dieckmann,
MIT division,
ITN, Linköping University.

February 20, 2023

Outline of the lecture

- I discuss pop-up windows with or without the possibility to input text.
- I discuss constructors in the class with the method *main(..)*.
- The class **JFrame** is discussed: its instance is a GUI.
- We discuss how to set the layout of a GUI.
- The classes **JLabel** and **JPanel** are presented.

The Swing library

Swing allows you to have pop-up windows.

```
import javax.swing.JOptionPane; // library for Swing classes.
public class MainClass {
public static void main(String[] args) {
JOptionPane.showMessageDialog(null, "Hello");
String input = JOptionPane.showInputDialog("User input:");
System.out.println(input);}}
```

The message dialog (left) puts text into a window.

The input dialog (right) asks you a question and waits for the answer. The code writes the answer to the console.

Constructor and *main()*

We will integrate the code, which specifies the GUI, into the class with *main()*. How does that work?

```
public class MainClass {
public MainClass(String s) { System.out.println(s);}
public static void main(String[] args) {
System.out.println("Hello");
MainClass mc = new MainClass("world");}}
```

The console output is "Hello" followed by "world".

The JVM jumps first into *main(arg)*. The constructor is called when we create the instance *mc* of **MainClass**.

We can place code that defines the GUI into the constructor.

The class JFrame

JFrame allows us to define Graphical user interfaces (GUIs).

```
import javax.swing.JFrame;
public class GUI extends JFrame {
public GUI() { setSize(400,200); setVisible(true);
setTitle("My first GUI"); System.out.println(getTitle());
setDefaultCloseOperation(EXIT_ON_CLOSE);}
public static void main(String[] args) {GUI mc = new GUI();}}
```

EXIT_ON_CLOSE closes the window and stops the code when you click on the X.

setVisible(true) makes the GUI visible (the default is *false*).

The Abstract Window Toolkit (AWT) and color

We import the complete AWT library (many required classes).

We will repeatedly change the color of the GUI.

Java has a class **Color** in the library *java.awt*

The RGB model sets color by mixing red, green, and blue.

```
import java.awt.*;
public class MainClass {
public static void main(String[] args) {
Color myColor = new Color(255,0,0); myColor = Color.RED;}}
```

Initially, we take a user-defined color and give it maximum red (255), no green (0) and blue (0).

Color also has class constants, which are predefined colors.

Changing the appearance of the GUI

We can access content and properties of a GUI through the method *getContentPane()* (inherited from **JFrame**).

It returns a reference to that instance of **Container** (a list), which stores the AWT components of our GUI.

```
import javax.swing.JFrame; import java.awt.*;
public class GUI extends JFrame {
public GUI() { setSize(400,200); setVisible(true);
setTitle("GUI");
Container c = getContentPane(); c.setBackground(Color.RED);
setDefaultCloseOperation(EXIT_ON_CLOSE);}
public static void main(String[] args) {GUI mc = new GUI();}}
```

Classes starting with "J" (**JFrame**) come from the Swing library. They are often based on AWT classes (**Frame**).

The class JLabel

Instances of this class can display text or images.

```
JLabel myLabel1 = new JLabel("Hello");
JLabel myLabel2 = new JLabel("Hello", int arg);
```

arg sets the position of the text on the label. Values are *JLabel.LEFT* (default), *JLabel.CENTER* or *JLabel.RIGHT*.

We can set and get the text with the instance methods *setText(String arg)* and *getText()*.

The default font is small. We can change the text size and appearance by changing the font.

```
myLabel1.setFont(Font arg);
```

The AWT class Font

A font is defined by its name, which can be "Serif", "SansSerif", "Dialog", "DialogInput" or "MonoSpaced".

A font can also have one of the styles *Font.PLAIN*, *Font.BOLD*, or *Font.ITALIC*.

Its size is set by an integer.

```
JLabel myLabel1 = new JLabel("Hello", JLabel.CENTER);
JLabel myLabel2 = new JLabel("World");
myLabel1.setFont(new Font("SansSerif", Font.BOLD, 22));
Font myFont = new Font("Serif", Font.PLAIN, 25);
myLabel2.setFont(myFont);
```

Setting properties of the JLabel

We can change the text color by calling the instance method *setForeground(Color arg)* of **JLabel**.

The background of a JLabel is transparent by default and its color is not visible. We make it visible with *setOpaque(true);*

```
public GUI() {setSize(400,200); setTitle("The GUI");
Container c = getContentPane(); jl = new JLabel("Hello");
// jl.setOpaque(true);
jl.setFont(new Font("SansSerif",Font.PLAIN,25));
jl.setForeground(Color.BLUE); jl.setBackground(Color.MAGENTA);
c.setBackground(Color.YELLOW); c.add(jl);
setVisible(true); setDefaultCloseOperation(EXIT_ON_CLOSE);}
```

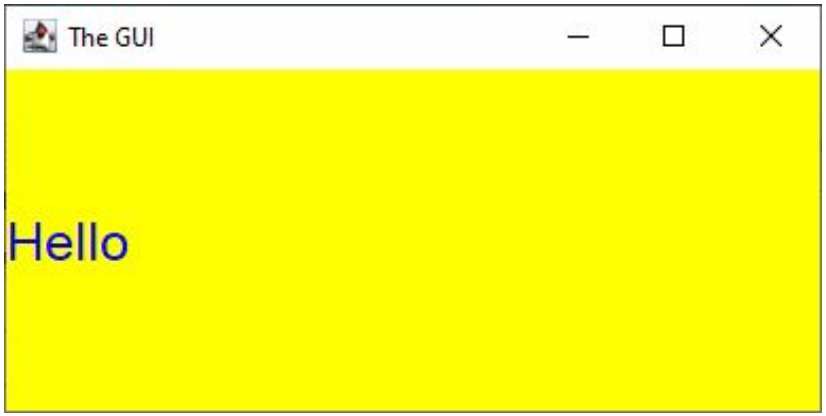
(Results on next slide)

Background of JLabel and JFrame

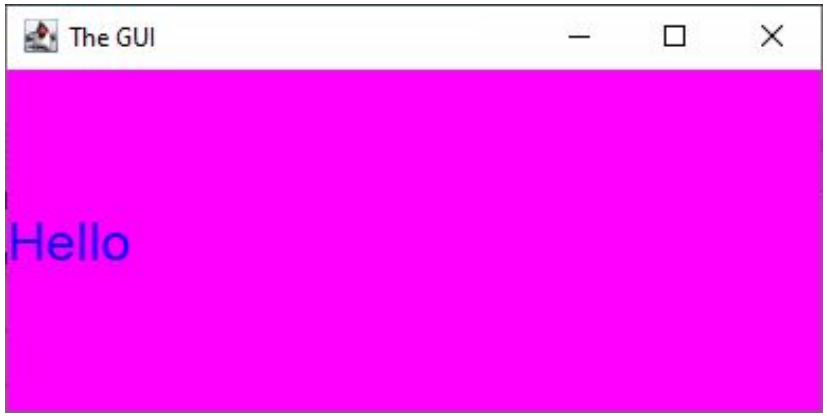
c.setBackground(Color.YELLOW): The background color of the GUI is yellow.

jl.setBackground(Color.MAGENTA): The background color of the JLabel is yellow.

With *jl.setOpaque(false);*



With *jl.setOpaque(true);*



Layout managers

Our present implementation can deal with only one AWT component. The code below puts "World" on our GUI.

```
Container c = getContentPane();
c.add(new JLabel("Hello")); c.add(new JLabel("World"));
```

Layout managers allow you to place more than one AWT component on your GUI. We use mostly a grid layout.

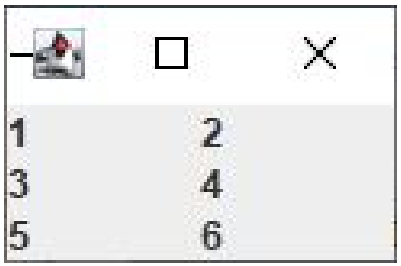
```
Container c = getContentPane();
c.setLayout(new GridLayout(3,2));
```

subdivides the GUI into 3 rows and 2 columns.

We can place 6 elements on the GUI.

Using the grid layout

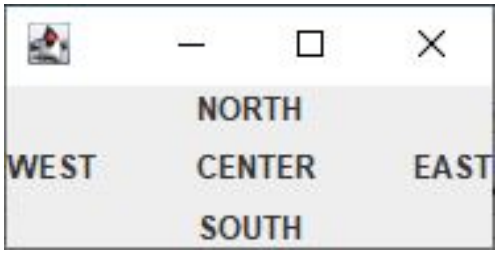
```
import javax.swing.*; import java.awt.*;
public class GUI extends JFrame {
private JLabel jl1, jl2, jl3, jl4, jl5, jl6;
public GUI() {
jl1 = new JLabel("1"); jl2 = new JLabel("2"); jl3 = new JLabel("3");
jl4 = new JLabel("4"); jl5 = new JLabel("5"); jl6 = new JLabel("6");
Container c = getContentPane(); c.setLayout(new GridLayout(3,2));
c.add(jl1); c.add(jl2); c.add(jl3); c.add(jl4); c.add(jl5); c.add(jl6);
setVisible(true); pack();
setDefaultCloseOperation(EXIT_ON_CLOSE); }
public static void main(String[] args) {GUI myGUI = new GUI();} }
```



Two columns and three rows.

pack() sets the size of the GUI such that all components just fit.

Using the border layout



We select the location of the text label with the help of constants of **BorderLayout**.

Class constants (used as text for the GUI labels)

BorderLayout.CENTER, *BorderLayout.NORTH*, *BorderLayout.EAST*, *BorderLayout.SOUTH*, and *BorderLayout.WEST*.

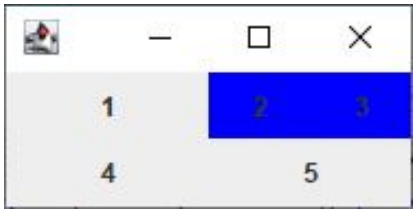
Using the border layout (code)

```
public class GUI extends JFrame {
private JLabel jl1, jl2, jl3, jl4, jl5;
public GUI() { setSize(200,100);
jl1 = new JLabel("CENTER",JLabel.CENTER);
jl2 = new JLabel("NORTH",JLabel.CENTER);
jl3 = new JLabel("EAST",JLabel.CENTER);
jl4 = new JLabel("SOUTH",JLabel.CENTER);
jl5 = new JLabel("WEST",JLabel.CENTER);
Container c = getContentPane(); c.setLayout(new BorderLayout());
c.add(jl1,BorderLayout.CENTER); c.add(jl2,BorderLayout.NORTH);
c.add(jl3,BorderLayout.EAST); c.add(jl4,BorderLayout.SOUTH);
c.add(jl5,BorderLayout.WEST);
setVisible(true); setDefaultCloseOperation(EXIT_ON_CLOSE);}
public static void main(String[] args) { GUI myGUI = new GUI();}}
```

The class JPanel

A grid layout allows us to subdivide the GUI into sectors.

We can subdivide a sector with an instance of **JPanel**.



I give the JPanel a grid layout with 2 columns and set its background color to blue.

```
JLabel jl1 = new JLabel("1",JLabel.CENTER);
JLabel jl2 = new JLabel("2",JLabel.CENTER);
JLabel jl3 = new JLabel("3",JLabel.CENTER);
JLabel jl4 = new JLabel("4",JLabel.CENTER);
JLabel jl5 = new JLabel("5",JLabel.CENTER);
JPanel pn = new JPanel(); pn.setBackground(Color.BLUE);
pn.setLayout(new GridLayout(1,2)); pn.add(jl2); pn.add(jl3);
Container c = getContentPane(); c.setLayout(new GridLayout(2,2));
c.add(jl1); c.add(pn); c.add(jl3); c.add(jl4);
```

Summary

We have discussed

- pop-up windows with or without the possibility to input text.
- constructors in the class with the method *main(..)*.
- the class **JFrame**: its instance is a GUI.
- how to set the layout of a GUI.
- the classes **JLabel** and **JPanel**.