Object Oriented Programming

PART V

Swing, Introduction

Beniamin Bogosel

Aurel Vlaicu University, Arad

Swing: graphical interfaces

⋆ Swing is a java library allowing to construct graphical interfaces
⋆ Its behavior is independent of the Operating System
JFrame: the equivalent of windows. They have

- title
- dimension
- aspect of graphical elements

A first test

```
package org.ldv.slam;
import javax.swing.*;
public class TestSwing
{
    public static void main(String args[]) {
        JFrame f=new JFrame("Hello World!!");
        f.setVisible(true);
    }
}
```

Write this code and test it.
The window is very small (default size is 0x0). Resizing-it by hand will make the title visible.

- By default a JFrame is invisible (created but not shown)
- That is why we need to add `f.setVisible(true)` which makes it visible
- When closing the window you can see that the Java program continues to run
- When closing the window with the mouse Swing does not close the program running
- in the second version we give a default size of the window and we quit the program when we close the window

Update

```
package org.ldv.slam;
import javax.swing.*;
public class TestSwing
{
    public static void main(String args[]) {
        JFrame f=new JFrame("Hello World!!");
        f.setSize(new Dimension(500,300));
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        f.setVisible(true);
    }
}
```

We added two lines:

- `setSize` allows to specify the size of the window
- `setDefaultCloseOperation` defines what should be done when closing the window
- `JFrame.EXIT_ON_CLOSE` specifies that we quit the program

Layouts

- To add objects in a `JFrame` we need `JPanel`
- A `JPanel` is a box in which we can place elements of the graphical interface; it stores objects
- Here is a list of most used components:
  - `JLabel label = new JLabel("a text") // for some text`
  - `JTextField text= new JTextField() // editable text field`
  - `JButton bouton=new JButton(''Quitter'') // a button`
  - etc.

Update our program

- New JPanel to store elements: `JPanel panel=new JPanel();`
- The text will be a JLabel: `JLabel label=new JLabel("Hello Everybody");`
- Add the JLabel to the panel: `panel.add(label);`
- Define a LayoutManager: to handle positioning:
  `panel.setLayout(new FlowLayout(FlowLayout.CENTER));`
- Set a FlowLayout for the window: `f.setLayout(new FlowLayout());`
- add the panel to the window: `f.add(panel);`

Improvements...

This technique is not ideal: the main function
should not be the handler of the graphics for the application...
⋆ Create a class which specializes the JFrame class and contains the needed components

New code

```
package test;

import javax.swing.*;
import java.awt.*;

public class testSwing2 extends JFrame {

    public testSwing2() {
        // Instead of JFrame frame= new JFrame("test");
        super("test"); // calls constructor from JFrame
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.setSize(500,400);
        JPanel panel=new JPanel();
        JLabel label=new JLabel("Bonjour tout le monde");
        panel.add(label);
        panel.setLayout(new FlowLayout(FlowLayout.CENTER));
        this.setLayout(new FlowLayout());
        this.add(panel);
    }

    public static void main(String[] args) {
        JFrame frame=new testSwing2();
        frame.setVisible(true);
    }

}
```

- The keyword `extends` means we inherit the class JFrame (all objects, methods, etc are available)
- the keyword `super` indicates that we call a constructor of the parent class (in this case JFrame). It is not needed to create a new object with
  `JFrame frame= new JFrame("test");`
  but simply write instead `super("test")`
- The keyword `this` makes reference to the current object. In this case it replaces the keyword `frame`.

Swing: positioning

⋆ need to import package `java.awt.*`

FlowLayout: most used and simple to manipulate;

⋆ it tries to put as many components on a line before goint to the next one

⋆ for each line you can specify the alignment:

- FlowLayout.LEFT, FlowLayout.CENTER, FlowLayout.RIGHT

BorderLayout

⋆ handles the space inside a container: five zones

- BorderLayout.NORTH
- BorderLayout.SOUTH
- BorderLayout.WEST
- BorderLayout.EAST
- BorderLayout.CENTER

In each area you can place a unique component

GridLayout

⋆ placing strategy using lines and columns

- `panelLeft.setLayout(new GridLayout(4,1)); // 4 lines 1 column`
- `panelLeft.setPreferredSize(new Dimension(180,120)); // fixes the dimension`

Event handling

⋆ to attribute an action to a button we need to add a **listener**: `ActionListener`
⋆ to add a listener to the current class, add `implements ActionListener`

```
public class TestButton extends JFrame implements ActionListener
```

⋆ Must import `ActionListener` and `ActionEvent`

```
import java.awt.event.*;
```

```
public void actionPerformed(ActionEvent e) {
    // something here
}
button1.addActionListener(this); // Add to the window as button
   listener
```

To see something we must add an action in the listener for the button. We do this in the function `actionPerformed`.

Defining the action

⋆ see from where the event comes from `getSource()`

```
public void actionPerformed(ActionEvent e) {
    if(e.getSource() == bouton1){
        //if the action comes from button 1
        number++;
        //increase the number with 1
        text.setText("You clicked " + number + " times on the button");
    }
}
```

When doing this if you click, the welcome text will vanish and will contain the number of clicks.

Asking text from user

- ⋆ show text on screen OK
- ⋆ buttons to create events OK
- ⋆ next stage: ask user for text
  - `JTextField`: basic – enter text on one line
  - `JTextArea`: complete text on multiple lines
  - `JEditorPane`: very complete, many options