

G51PRG:

Introduction to Programming

Second semester

Lecture 15: Graphical User Interfaces in Java

Natasha Alechina

School of Computer Science & IT

nza@cs.nott.ac.uk

Plan of today's lecture

- GUI packages in Java
- Comparison of `java.awt` and `javax.swing`
- Containers and components
- Creating new components
- Adding components to a container (layout managers)
 - Flow Layout
 - Grid Layout
 - Border Layout
- Containment hierarchies

After Easter...

- Event handling
- Basics of graphics
- Thread issues
- Applets

Lecture 15: GUI

3

Java GUI Packages

- **java.awt**: Abstract Window Toolkit (AWT)
Used in Java 1.0 and 1.1
- **javax.swing**: Swing is part of the Java Foundation Classes. Can be used in Java 1.1 (as an extension), Java 1.2 and above (where it is a core package)
- use one or the other—*never mix AWT and Swing components.*
- Sun recommends using Swing components.

Lecture 15: GUI

4

Comparison of AWT and Swing

- AWT: simpler, more robust but less flexible than Swing.
Uses native code to display GUI components (hence the look and feel depends on the platform, e.g, Motif/CDE, Windows, MacOS).
- Swing: lots of additional features (pluggable look and feel, assistive technologies, drag and drop). Does not use native code. More verbose. Not thread-safe.

Lecture 15: GUI

5

Overview of libraries

- Components & Containers (windows, buttons, menus etc.)
- use the ones from **javax.swing**
- Layout managers (arranging components relative to each other and the size of screen) –from **java.awt**
- Event handlers respond to user input (mouse clicks, text input etc.) –from **java.awt**
- Classes dealing with graphics (colours, shapes, fonts) and images; basic utilities are mostly in **java.awt**
(**java.awt.Graphics**, **java.awt.Color** etc.)

Lecture 15: GUI

6

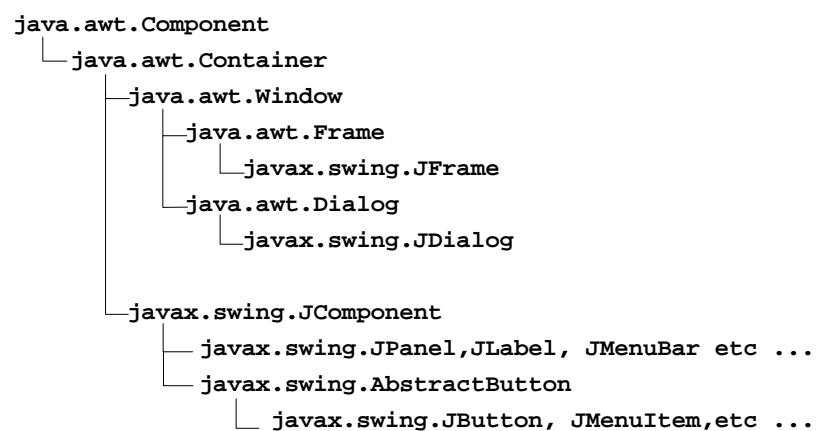
Components and Containers

- *Components* are things like windows, menus, buttons, text labels, text fields etc.
- Some components are also *Containers*: they can contain other components and are used for layout
- Some Components are *atomic components*, e.g., buttons, text labels and text fields

Lecture 15: GUI

7

Overview of the Swing Class Hierarchy

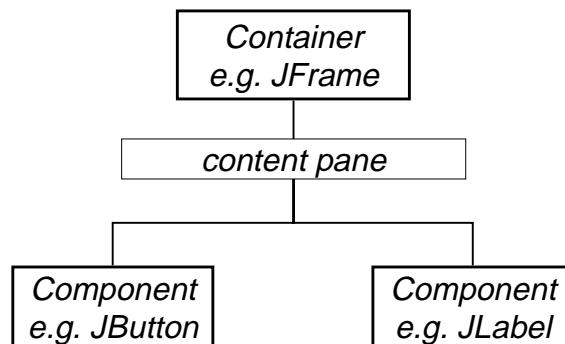


Lecture 15: GUI

8

A Simple Swing Application

The simplest Swing application consists of a ***top-level Container*** and one or more Components

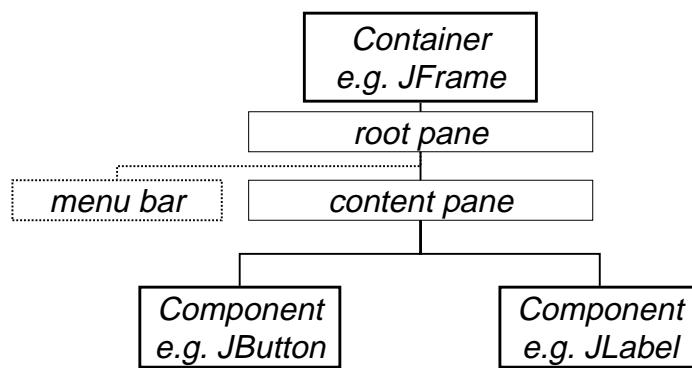


Lecture 15: GUI

9

A Simple Swing Application

The simplest Swing application consists of a ***top-level Container*** and one or more Components



Lecture 15: GUI

10

Creating a Single Window

There are two ways to make a window:

- in the `main()`, create an instance of a `JFrame` and make it visible; or
- extend `JFrame`, and in the `main()` create and make visible an instance of your new class.

Creating Components Inside a Window

Swing provides many classes for GUI components including buttons, labels and text fields, e.g.:

- to create a button with text *OK*, use

```
JButton okButton = new JButton("OK");
```

- to create a text label with text *Hi!*, use

```
JLabel label = new JLabel("Hi!");
```

Positioning Components in a Container

In Swing *relative placement* is used rather than *absolute placement*:

- components are placed and sized relative to other components
- how this is done depends on the choice of *layout manager*

Lecture 15: GUI

13

Layout Managers

All layout managers implement one of two interfaces:

- **LayoutManager**: interface for classes that know how to lay out containers, e.g., **FlowLayout**, **GridLayout** etc.
- **LayoutManager2** (extends **LayoutManager**): interface for classes that know how to layout containers based on constraint objects, e.g., **BorderLayout**, **BoxLayout**, **GridBagLayout**, etc.

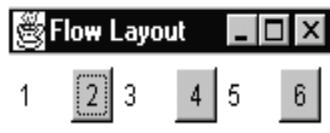
Lecture 15: GUI

14

Flow Layout

FlowLayout is the simplest way to fill in a window with components:

- start at the left upper corner and add the components one by one in a row
- when the right hand side of the container is reached start a new row



- default layout manager for **JPanels**

Flow Layout Example

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

public class FlowExample extends JFrame {
    public FlowExample(){
        getContentPane().setLayout(new FlowLayout());
        getContentPane().add(new JLabel("1"));
        getContentPane().add(new JButton("2"));
        getContentPane().add(new JLabel("3"));
        getContentPane().add(new JButton("4"));
        getContentPane().add(new JLabel("5"));
        getContentPane().add(new JButton("6"));
    }
}
```

Flow Layout Example continued

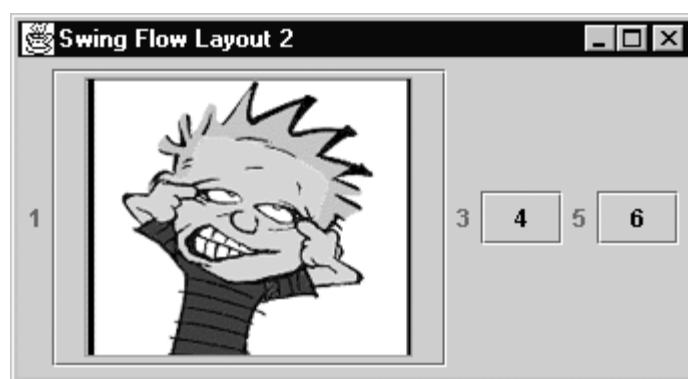
```
public static void main(String[] args) {  
    FlowExample window = new FlowExample();  
    window.setTitle("Flow Layout");  
    window.pack();  
    window.setVisible(true);  
}  
}
```

Lecture 15: GUI

17

Digression: Buttons with Icons

Adding a picture to the first button:



Lecture 15: GUI

18

Buttons with Icons Example

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

public class ButtonExample extends JFrame {

    public ButtonExample(){
        ImageIcon icon = new ImageIcon("calvin.gif");
        JButton calvin = new JButton(icon);
        getContentPane().setLayout(new FlowLayout());
        getContentPane().add(new JLabel("1"));
        getContentPane().add(calvin);
        ...
    }
}
```

Lecture 15: GUI

19

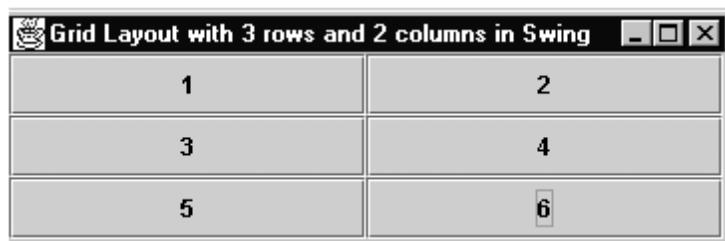
Grid Layout

- The **GridLayout** class is a layout manager that lays out a container's components in a rectangular grid.
- If the **add()** method is used without indicating row and column, the rows are filled from left to right and from top to bottom.

Lecture 15: GUI

20

Grid Layout Example



Lecture 15: GUI

21

Grid Layout Example

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

public class GridExample extends JFrame {
    public GridExample(){
        getContentPane().setLayout(new GridLayout(3,2));
        getContentPane().add(new JButton("1"));
        getContentPane().add(new JButton("2"));
        getContentPane().add(new JButton("3"));
        getContentPane().add(new JButton("4"));
        getContentPane().add(new JButton("5"));
        getContentPane().add(new JButton("6"));
    }
}
```

Lecture 15: GUI

22

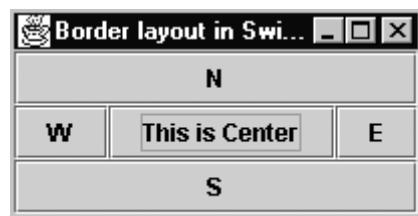
Border Layout

- The **BorderLayout** class is the default layout manager for **JFrame**, **JDialog** and **JApplet**
- In border layout, the container is divided into:
 - the CENTER area, which expands to take all available space
 - four borders: NORTH (top), SOUTH (bottom), WEST (left) and EAST (right)
- Often a container uses only one or two areas of the border layout, e.g., CENTER and SOUTH

Lecture 15: GUI

23

Border Layout Example



Lecture 15: GUI

24

Border Layout Example

```
import javax.swing.*;
import java.awt.*;
public class BorderExample extends JFrame {
    public BorderExample(){
        // getContentPane().setLayout(new BorderLayout());
        getContentPane().add(new JButton("This is Center"),
                            BorderLayout.CENTER );
        getContentPane().add(new JButton("N"),
                            BorderLayout.NORTH);
        getContentPane().add(new JButton("S"),
                            BorderLayout.SOUTH);
        getContentPane().add(new JButton("E"),
                            BorderLayout.EAST);
        getContentPane().add(new JButton("W"),
                            BorderLayout.WEST);
    }
}
```

Lecture 15: GUI

25

Containment Hierarchies

- For more control over layout, components are often grouped together in an *intermediate container*, which is then added to another container, and so on.
- A **JPanel** is a container used for grouping components within a larger container, e.g., two buttons can be grouped on a **JPanel** to keep them together.

Lecture 15: GUI

26

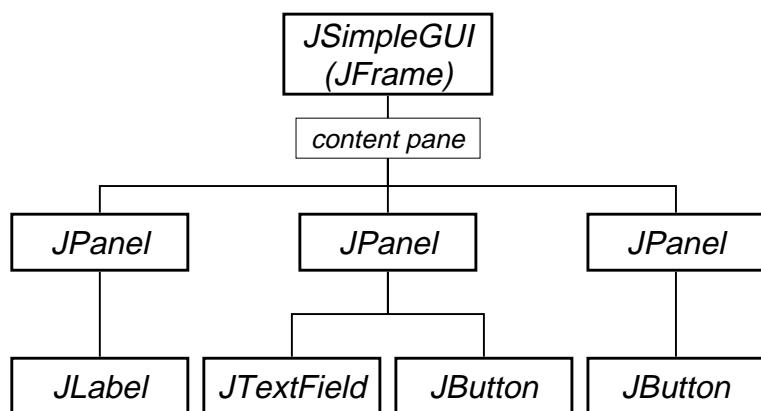
Intermediate Container Example



Lecture 15: GUI

27

Intermediate Container Example



Lecture 15: GUI

28

JTextAreas and JTextFields

A **JTextField** object is a text component that allows for the editing of a single line of text

- can be created with a specified width in columns and/or with a predefined line of text.

A **JTextArea** object is a multi-line region that displays text.

- can be editable or to be read-only
- can be created with scrollbars or without, with a given text, with a specified number of rows and columns, etc.

Intermediate Container Example

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

class JSimpleGUI1 extends JFrame {
    private JLabel label;
    private JTextField tf;
    private JButton okB, exitB;

    public JSimpleGUI1() {
        // Create the atomic components
        label = new JLabel("Enter your name");
        tf = new JTextField(20);
        okB = new JButton("Ok");
        exitB = new JButton("Exit");
    }
}
```

Intermediate Container Example

```
// Create the intermediate containers
JPanel panel = new JPanel(); // default flow layout
panel.add(label);

JPanel pane2 = new JPanel(); // default flow layout
pane2.add(tf);
pane2.add(okB);

JPanel pane3 = new JPanel(new
                           FlowLayout(FlowLayout.RIGHT));
pane3.add(exitB);

this.getContentPane().setLayout(new GridLayout(0, 1));
this.getContentPane().add(panel);
this.getContentPane().add(pane2);
this.getContentPane().add(pane3);}
```

Lecture 15: GUI

31

Intermediate Container Example

```
public static void main(String args) {
    JSimpleGUI1 window = new JSimpleGUI1();
    window.setTitle("Simple GUI");
    window.pack();
    window.setVisible(true);
}
```

Lecture 15: GUI

32

Summary

- Classes from **java.awt** and **javax.swing** define various components of a graphical user interface
- Components are grouped together in containers
- Components are laid out relative to each other using layout managers
- For more control over layout, intermediate containers can be used

<http://java.sun.com/docs/books/tutorial/uiswing/index.html>