1495 UNIVERSITY OF ABERDEEN

JC2002 Java Programming

Day 7: Graphical user interfaces (AI, CS)

Wednesday, 8 November / Thursday, 9 November

1495 UNIVERSITY OF ABERDEEN

JC2002 Java Programming

Day 7, Session 1: Introduction to graphical user interfaces

References and learning objectives

- Today's sessions are mostly based on:
  - Sierra et al., ***Head First Java*** (2nd Ed), O'Reilly, Chapters 12, 13 (available in the library)
  - https://docs.oracle.com/javase/tutorial/uiswing/components/menu.html
  - ***Introduction to Programming with Java***, Chapter 2.14 https://runestone.academy/runestone/books/published/csjava/index.html
- After today's session, you should be able to:
  - Explain the main concepts and terms of graphical user interfaces and
  - Implement simple user interfaces using standard Swing components
  - Select appropriate GUI components for different purposes

Graphical user interface (GUI)

- A graphical user interface (GUI) presents a user-friendly mechanism for interacting with an app
  - GUI (pronounced “GOO-ee”) gives an app a distinctive *look-and-feel*
  - GUI provides apps with consistent, intuitive user interface components giving users a sense of familiarity even with a new app
- GUIs are built from *GUI components*, also called *controls* or *widgets* (short for window gadgets)
  - A GUI components is an object with which the user interacts via the mouse, the keyboard or another form of input, e.g., voice recognition

Java GUI on different platforms

- Java code is *platform independent*: Java GUI uses the GUI components provided by the underlying platform
  - Different platforms give different look-and-feels

Mac OS

Linux OS

Solaris (Unix) OS

Windows OS

Java GUI libraries

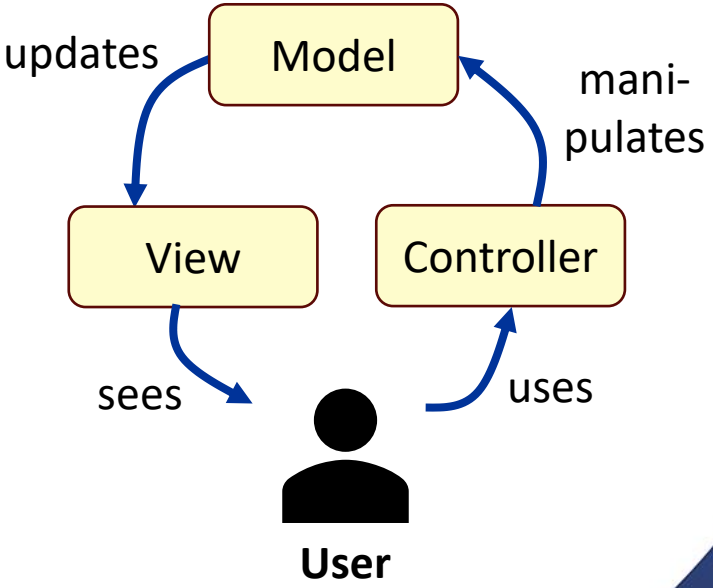
- There are different GUI libraries for Java:
  - **Abstract Window Toolkit (AWT)** was Java's original GUI library (the oldest of all Java GUIs)
    - AWT is heavyweight and platform dependent
  - **Swing** was added to the platform in Java SE 1.2
    - Until recently, Swing was the primary Java GUI technology
    - Lighter, platform independent, purely for desktop
  - **JavaFX** was announced in 2007 and released in 2008 as a competitor to Adobe Flash and Microsoft Silverlight
    - Smaller number of components, better integration to modern devices

Java Swing library

- In this course, we use primarily Swing for GUIs
  - Swing is still widely used and focused
  - Swing does the “heavy lifting” in desktop applications
  - Swing uses the common model-view-controller (MVC) design pattern
  - Swing is cross-platform (as Java in general) with suitable look-and-feel
  - Swing is extensive and what is learned with Swing is easy to move to JavaFX in the future

Model-view-controller (MVC)

- In general, a visual component is a composite of three distinct aspects:
  - The way that the component looks when rendered on the screen (*view*)
  - The way the component reacts to the user (*controller*)
  - The state information associated with the component (*model*)
- Over the years, MVC architecture has proven itself to be exceptionally effective

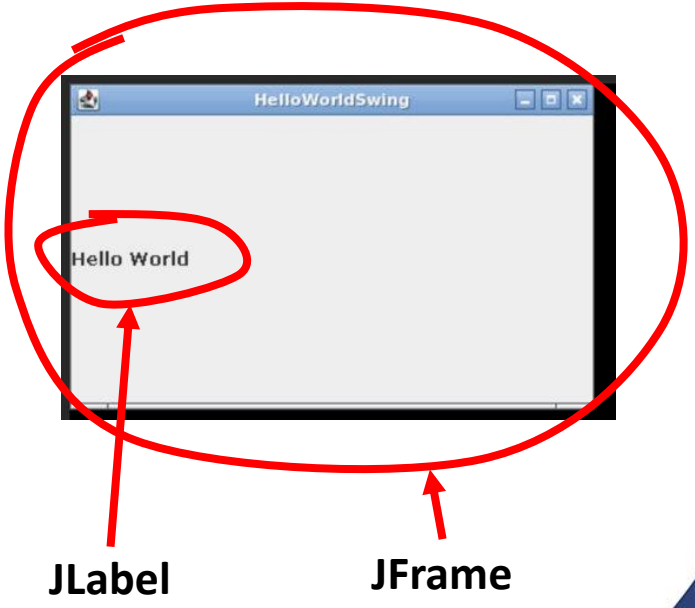


Java foundation classes (JFC) and Swing

- JFC encompass a group of features for building GUIs and adding rich graphics functionality and interactivity to Java applications
- JFC contains the features:
    - Swing GUI Components
    - Pluggable look-and-feel Support
    - Accessibility API
    - Java 2D API
    - Internationalisation

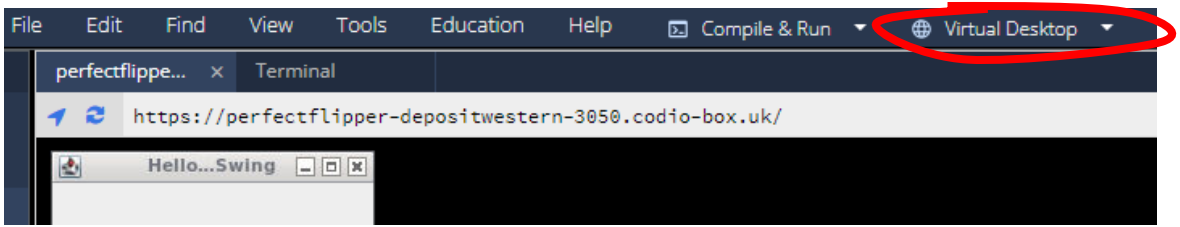
Swing components

- In Swing, the GUI is composed of graphical components
  - The graphical components are classes, and to use them, you must declare objects of them
  - Usually, GUI is build on **`JFrame`** component that is a *window* or *container* for GUI
  - Other commonly used components include e.g., **`JLabel`** that can include static text or an image, and **`JButton`** that implements a button that can be pressed



Running Swing in Codio

- To run Java with GUI, the underlying platform must support GUI
- In Codio, you can run GUI by using *virtual desktop*
  - To install, follow the instructions in: https://docs.codio.com/common/develop/ide/boxes/installsw/gui.html
  - Note that you need to relaunch Codio before virtual desktop can be used
  - You should be then able to open Virtual Desktop tab in Codio



Example of using JFrame (1)

```java
import javax.swing.*;
public class FrameTest {
  private static void createAndShowGUI() {

    JFrame frame = new JFrame("HelloWorldSwing");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    JLabel label = new JLabel("Hello World");
    frame.getContentPane().add(label);

    frame.pack();
    frame.setVisible(true);
  }
  public static void main(String[] args) {
    javax.swing.SwingUtilities.invokeLater(new Runnable() {
      public void run() {
        createAndShowGUI();
      }
    });
  }
}
```

Example of using JFrame (2)

```java
import javax.swing.*;
public class FrameTest {
  private static void createAndShowGUI() {

    JFrame frame = new JFrame("HelloWorldSwing");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    JLabel label = new JLabel("Hello World");
    frame.getContentPane().add(label);

    frame.pack();
    frame.setVisible(true);
  }
  public static void main(String[] args) {
    javax.swing.SwingUtilities.invokeLater(new Runnable() {
      public void run() {
        createAndShowGUI();
      }
    });
  }
}
```

Import Swing classes

Create `JFrame` component

Create `JLabel` component

Standard code to run Swing GUI apps

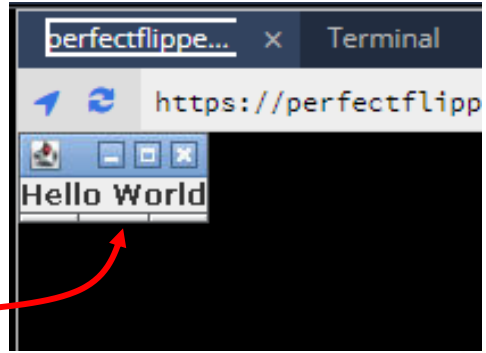
Example of using JFrame (3)

```java
import javax.swing.*;
public class FrameTest {
  private static void createAndShowGUI() {

    JFrame frame = new JFrame("HelloWorldSwing");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    JLabel label = new JLabel("Hello World");
    frame.getContentPane().add(label);

    frame.pack();
    frame.setVisible(true);
  }
  public static void main(String[] args) {
    javax.swing.SwingUtilities.invokeLater(new Runnable() {
      public void run() {
        createAndShowGUI();
      }
    });
  }
}
```

Console:

```
$ javac FrameTest.java
$ java FrameTest
```

Virtual Desktop:



You can use mouse to resize the app

Setting frame size

- You need to determine how big you want the frame to be
  - You can delegate decision to the app with `frame.pack()`
  - The pack method sizes the frame so that all its contents are at or above their preferred sizes
  - An alternative to `pack()` is to establish a frame size explicitly by calling `setSize()` or `setBounds()` (which also sets the frame location)
  - In general, using `pack()` is preferable to calling `setSize()`, since `pack()` layout managers are good at adjusting to platform dependencies and other factors that affect component size

Example of using setSize

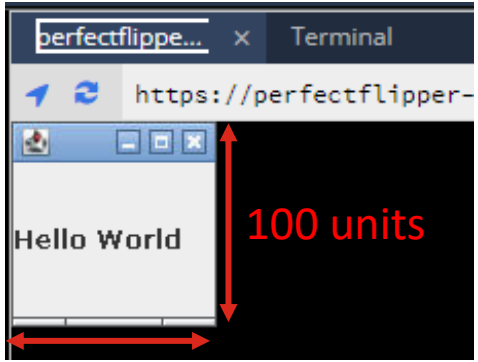
```java
import javax.swing.*;
public class SetSizeTest {
  private static void createAndS

    JFrame frame = new JFrame("H
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    JLabel label = new JLabel("Hello World");
    frame.getContentPane().add(label);

    frame.setSize(100,100);
    frame.setVisible(true);
  }
  public static void main(String[] args) {
    javax.swing.SwingUtilities.invokeLater(new Runnable() {
      public void run() {
        createAndShowGUI();
      }
    });
  }
}
```

Using `setSize()` to set the frame size

Console:

```
$ javac SetSizeTest.java
$ java SetSizeTest
```

Virtual Desktop:



Questions, comments?

1495 UNIVERSITY OF ABERDEEN

JC2002 Java Programming

Day 7, Session 2: Using top-level containers in Swing

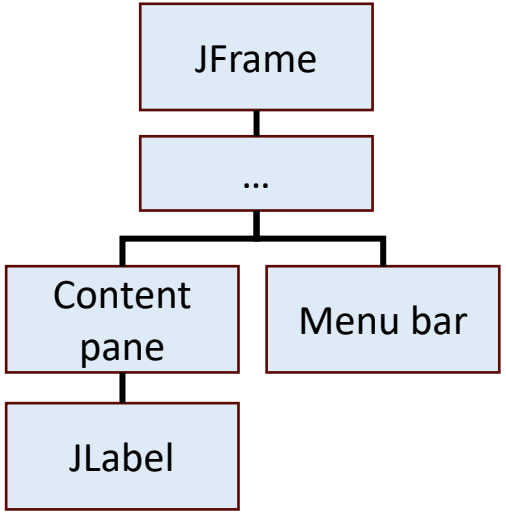
Top-level container classes

- Swing provides two generally useful top-level container classes: `JFrame` and `JDialog`
  - Each program that uses Swing components has at least one top-level container that is the root of a *containment hierarchy*
  - To appear onscreen, every GUI component must be part of a containment hierarchy
  - Each GUI component can be contained only once; if a component is already in a container and you try to add it to another container, the component will be removed from the first container and then added to the second

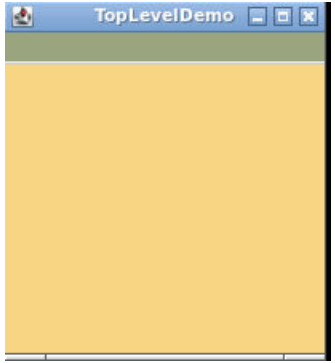
Content pane of top-level container classes

- Each top-level container has a *content pane* that contains (directly or indirectly) the visible components in that top-level container's GUI
- You can optionally add a *menu bar* to a top-level container
  - The menu bar is by convention positioned within the top-level container, but outside the content pane
  - Some look-and-feels, such as the Mac OS, give you the option of placing the menu bar in another place more appropriate for the look-and-feel, such as at the top of the screen

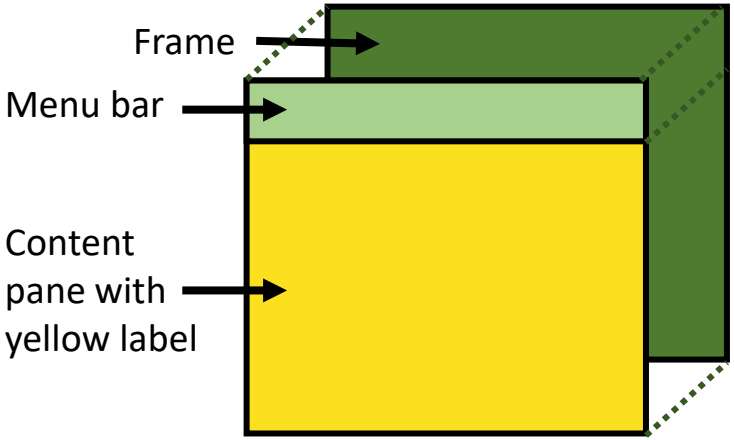
JFrame top-level container



Containment hierarchy



Window on screen



Container structure

Example of top-level container (1)

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

public class TopLevelDemo {
  private static void createAndShowGUI() {
    JFrame frame = new JFrame("TopLevelDemo");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    JMenuBar greenMenuBar = new JMenuBar();
    greenMenuBar.setOpaque(true);
    greenMenuBar.setBackground(new Color(154, 165, 127));
    greenMenuBar.setPreferredSize(new Dimension(200, 20));
    JLabel yellowLabel = new JLabel();
    yellowLabel.setOpaque(true);
    yellowLabel.setBackground(new Color(248, 213, 131));
    yellowLabel.setPreferredSize(new Dimension(200, 180));
    frame.setJMenuBar(greenMenuBar);
    frame.getContentPane().add(yellowLabel, BorderLayout.CENTER);
    frame.pack();
    frame.setVisible(true);
  }
```

```java
  public static void main(String[] args) {
    javax.swing.SwingUtilities.invokeLater(new Runnable() {
      public void run() { createAndShowGUI(); }
    });
  }
}
```

This part is standard, and we will not show it in the following slides

Example of top-level container (2)

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

public class TopLevelDemo {
  private static void createAndShowGUI() {
    JFrame frame = new JFrame("TopLevelDemo");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    JMenuBar greenMenuBar = new JMenuBar();
    greenMenuBar.setOpaque(true);
    greenMenuBar.setBackground(new Color(154, 165, 127));
    greenMenuBar.setPreferredSize(new Dimension(200, 20));
    JLabel yellowLabel = new JLabel();
    yellowLabel.setOpaque(true);
    yellowLabel.setBackground(new Color(248, 213, 131));
    yellowLabel.setPreferredSize(new Dimension(200, 180));
    frame.setJMenuBar(greenMenuBar);
    frame.getContentPane().add(yellowLabel, BorderLayout.CENTER);
    frame.pack();
    frame.setVisible(true);
  }
```

Essential imports

Create and set up the window (JFrame)

Example of top-level container (3)

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

public class TopLevelDemo {
  private static void createAndShowGUI() {
    JFrame frame = new JFrame("TopLevelDemo");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    JMenuBar greenMenuBar = new JMenuBar();
    greenMenuBar.setOpaque(true);
    greenMenuBar.setBackground(new Color(154, 165, 127));
    greenMenuBar.setPreferredSize(new Dimension(200, 20));
    JLabel yellowLabel = new JLabel();
    yellowLabel.setOpaque(true);
    yellowLabel.setBackground(new Color(248, 213, 131));
    yellowLabel.setPreferredSize(new Dimension(200, 180));
    frame.setJMenuBar(greenMenuBar);
    frame.getContentPane().add(yellowLabel, BorderLayout.CENTER);
    frame.pack();
    frame.setVisible(true);
  }
```

Create a menu bar with size (200,20) and green background

Example of top-level container (4)

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

public class TopLevelDemo {
  private static void createAndShowGUI() {
    JFrame frame = new JFrame("TopLevelDemo");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    JMenuBar greenMenuBar = new JMenuBar();
    greenMenuBar.setOpaque(true);
    greenMenuBar.setBackground(new Color(154, 165, 127));
    greenMenuBar.setPreferredSize(new Dimension(200, 20));
    JLabel yellowLabel = new JLabel();
    yellowLabel.setOpaque(true);
    yellowLabel.setBackground(new Color(248, 213, 131));
    yellowLabel.setPreferredSize(new Dimension(200, 180));
    frame.setJMenuBar(greenMenuBar);
    frame.getContentPane().add(yellowLabel, BorderLayout.CENTER);
    frame.pack();
    frame.setVisible(true);
  }
```

Create a label with yellow background

Example of top-level container (5)

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

public class TopLevelDemo {
  private static void createAndShowGUI() {
    JFrame frame = new JFrame("TopLevelDemo");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    JMenuBar greenMenuBar = new JMenuBar();
    greenMenuBar.setOpaque(true);
    greenMenuBar.setBackground(new Color(154, 165, 127));
    greenMenuBar.setPreferredSize(new Dimension(200, 20));
    JLabel yellowLabel = new JLabel();
    yellowLabel.setOpaque(true);
    yellowLabel.setBackground(new Color(248, 213, 131));
    yellowLabel.setPreferredSize(new Dimension(200, 180));
    frame.setJMenuBar(greenMenuBar);
    frame.getContentPane().add(yellowLabel, BorderLayout.CENTER);
    frame.pack();
    frame.setVisible(true);
  }
```

Add the menu bar and the label to the frame

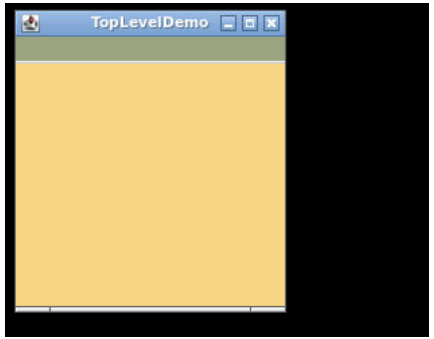
Example of top-level container (6)

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

public class TopLevelDemo {
  private static void createAndShowGUI() {
    JFrame frame = new JFrame("TopLevelDemo");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    JMenuBar greenMenuBar = new JMenuBar();
    greenMenuBar.setOpaque(true);
    greenMenuBar.setBackground(new Color(154, 165, 127));
    greenMenuBar.setPreferredSize(new Dimension(200, 20));
    JLabel yellowLabel = new JLabel();
    yellowLabel.setOpaque(true);
    yellowLabel.setBackground(new Color(248, 213, 131));
    yellowLabel.setPreferredSize(new Dimension(200, 180));
    frame.setJMenuBar(greenMenuBar);
    frame.getContentPane().add(yellowLabel, BorderLayout.CENTER);
    frame.pack();
    frame.setVisible(true);
  }
```

Console:

```
$ java TopLevelDemo
```

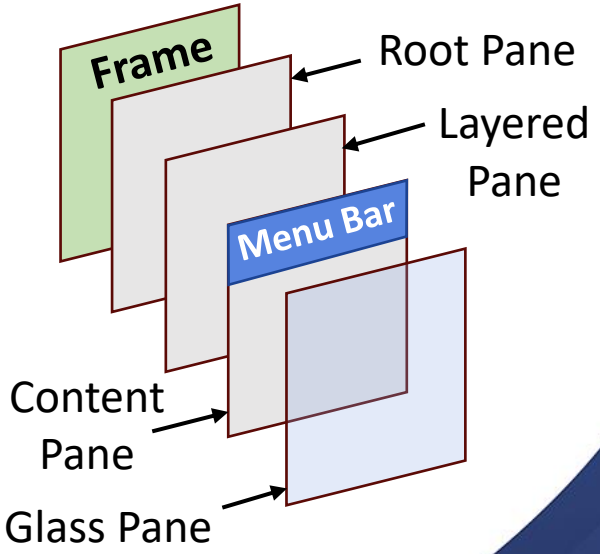
Virtual Desktop:



The root pane

- Each top-level container relies on a reclusive intermediate container called the *root pane*
  - The root pane manages the content pane and the menu bar, along with a couple of other containers, such as content and glass pane
  - The layered pane contains the menu bar and content pane
  - The glass pane is often used to intercept input events over the top-level container, and it can also be used to paint over other components

A list of the components that a root pane provides to a frame



Dialogs

- A *dialog* is an independent sub window (or a *pop-up window*) carrying notifications apart from the main application window
  - Most dialogs present an error message or warning to a user, but dialogs can also present images, directory trees, etc.
  - A dialog is *modal* if it blocks user input to all the other windows in the program until it is closed
- In Swing, class **`JDialog`** is used to instantiate top-level containers for dialogs
  - Class **`JOptionPane`** provides simple standard modal dialog boxes, but to create a *non-modal* dialog, you must use `JDialog` class directly

Simple example dialog via JOptionPane

```
1   import javax.swing.*;
2   public class OptionPaneExample {
3     JFrame f;
4     OptionPaneExample(){
5       f=new JFrame();
6       JOptionPane.showMessageDialog(f,"Hello, Welcome to JC2002.");
7     }
8     public static void main(String[] args) {
9       new OptionPaneExample();
10    }
11  }
```

Non-modal example dialog via JDialog (1)

```
1   import javax.swing.*;
2   import java.awt.*;
3   import java.awt.event.*;
4   public class DialogExample {
5     private static JDialog d;
6     DialogExample() {
7       JFrame f= new JFrame();
8       d = new JDialog(f, "Dialog Example", true);
9       d.setLayout(new FlowLayout());
10      JButton b = new JButton ("OK");
11      b.addActionListener (new ActionListener() {
12        public void actionPerformed(ActionEvent e) {
13          DialogExample.d.setVisible(false);
14        }
15      });
16      d.add(new JLabel ("Click button to continue."));
17      d.add(b);
18      d.setSize(300,300);
19      d.setVisible(true);
20    }
21    public static void main(String args[]) {
22      new DialogExample();
23    }
24  }
```

Non-modal example dialog via JDialog (2)

```java
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class DialogExample {
  private static JDialog d;
  DialogExample() {
    JFrame f= new JFrame();
    d = new JDialog(f, "Dialog Example", true);
    d.setLayout(new FlowLayout());
    JButton b = new JButton ("OK");
    b.addActionListener (new ActionListener() {
      public void actionPerformed(ActionEvent e) {
        DialogExample.d.setVisible(false);
      }
    });
    d.add(new JLabel ("Click button to continue."));
    d.add(b);
    d.setSize(300,300);
    d.setVisible(true);
  }
```

Create JFrame object to contain the dialog

Create JDialog object

Create JButton object

```java
  public static void main(String args[]) {
    new DialogExample();
  }
}
```

Non-modal example dialog via JDialog (3)

```java
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class DialogExample {
  private static JDialog d;
  DialogExample() {
    JFrame f= new JFrame();
    d = new JDialog(f, "Dialog Example", true);
    d.setLayout(new FlowLayout());
    JButton b = new JButton ("OK");
    b.addActionListener (new ActionListener() {
      public void actionPerformed(ActionEvent e) {
        DialogExample.d.setVisible(false);
      }
    });
    d.add(new JLabel ("Click button to continue."));
    d.add(b);
    d.setSize(300,300);
    d.setVisible(true);
  }
```

We will discuss about layouts and action listeners later

```java
  public static void main(String args[]) {
    new DialogExample();
  }
}
```

UNIVERSITY OF ABERDEEN

Non-modal example dialog via JDialog (4)

```
1   import javax.swing.*;
2   import java.awt.*;
3   import java.awt.event.*;
4   public class DialogExample {
5     private static JDialog d;
6     DialogExample() {
7       JFrame f= new JFrame();
8       d = new JDialog(f, "Dialog Example", true);
9       d.setLayout(new FlowLayout());
10      JButton b = new JButton ("OK");
11      b.addActionListener (new ActionListener() {
12        public void actionPerformed(ActionEvent e) {
13          DialogExample.d.setVisible(false);
14        }
15      });
16      d.add(new JLabel ("Click button to continue."));
17      d.add(b);
18      d.setSize(300,300);
19      d.setVisible(true);
20    }
21    public static void main(String args[]) {
22      new DialogExample();
23    }
24  }
```

Add a label, the button, and make the dialog visible

Non-modal example dialog via Jdialog (5)

```java
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class DialogExample {
  private static JDialog d;
  DialogExample() {
    JFrame f= new JFrame();
    d = new JDialog(f, "Dialog Example", true);
    d.setLayout(new FlowLayout());
    JButton b = new JButton ("OK");
    b.addActionListener (new ActionListener() {
      public void actionPerformed(ActionEvent e) {
        DialogExample.d.setVisible(false);
      }
    });
    d.add(new JLabel ("Click button to continue."));
    d.add(b);
    d.setSize(300,300);
    d.setVisible(true);
  }
```

```
$ java DialogExample
```


```java
  public static void main(String args[]) {
    new DialogExample();
  }
}
```

Questions, comments?

1495 UNIVERSITY OF ABERDEEN

JC2002 Java Programming

Day 7, Session 3: Using layouts and buttons in Swing

Adding JComponents

- Different GUI components (buttons, images, text fields, etc.) are represented by subclasses of `JComponent`
- GUI components are added to containers by using `add()` method with the added `JComponent` subclass object as a parameter

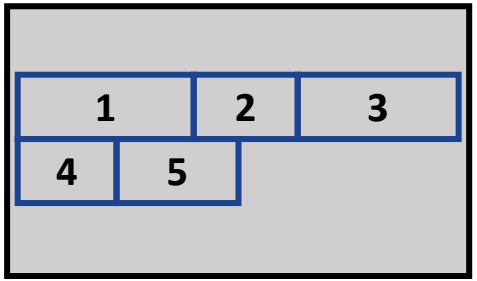
```
1   import javax.swing.*;
2   public class ButtonExample {
3     public static void main(String[] args) {
4       JFrame f=new JFrame("Button Example");
5       JButton b=new JButton("Click Here");
6       b.setBounds(50,35,200,30);
7       f.add(b);
8       f.setSize(300,100);
9       f.setLayout(null);
10      f.setVisible(true);
11    }
12  }
```

Button location and size

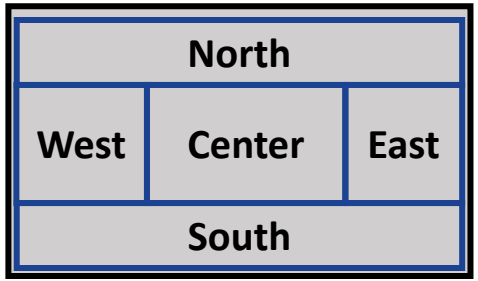
Add button

Layout managers

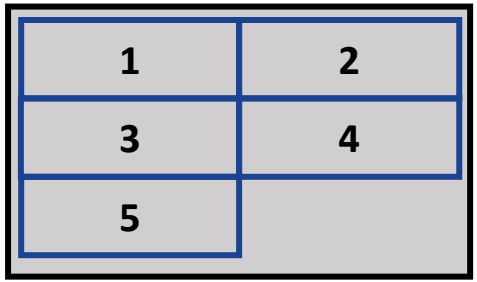
- Setting absolute positions and sizes for the components may look bad if you do not know the screen resolution of the target platform
- Several Swing and AWT classes provide layout managers for dynamic allocation of GUI components, according to different specific rules



FlowLayout (default)



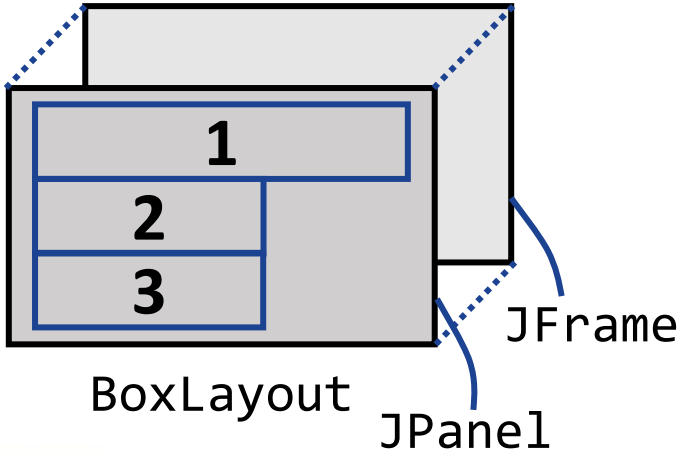
BorderLayout



GridLayout

Layouts on JPanel

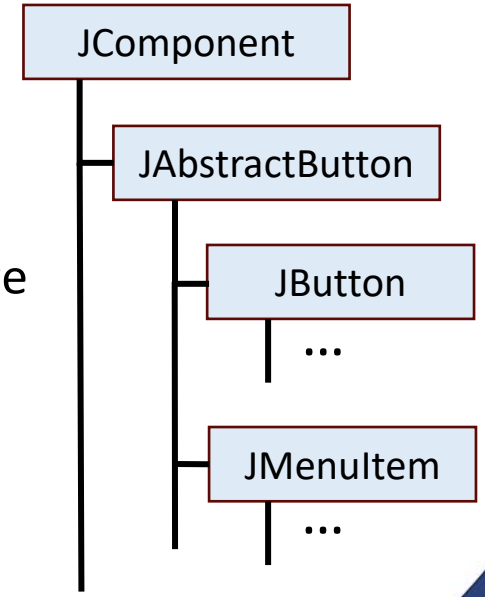
- Some layouts, like `FlowLayout`, can be used on `JFrame` directly
- Some layouts, like `GridLayout` and `BoxLayout`, require creating `JPanel` object between `JFrame` and the components




```
JFrame frame = new JFrame("Win");
JPanel panel = new JPanel();
BoxLayout boxlayout = new
BoxLayout(panel, BoxLayout.Y_AXIS);
panel.setLayout(boxlayout);
...
frame.add(panel);
```

Buttons: class JButton

- Buttons are among the most widely used GUI components
  - Several subtypes: radio buttons, menu items, check boxes, etc.
- Swing class `JButton` can display both text and image
  - The underlined letter in button's text shows the *mnemonic* (the keyboard alternative) for the button
    - Usually, user can click a button by pressing the **Alt** key and the mnemonic
  - *Tool tip* can be defined to explain the meaning of the button



Initialising JButton object (1)

- Example button with both icon and text (both are optional)

```
ImageIcon unhappyButtonIcon = new ImageIcon("unhappy.png");
b1 = new JButton("Unhappy", unhappyButtonIcon);
b1.setSize(100,100);
b1.setVerticalTextPosition(AbstractButton.BOTTOM);
b1.setHorizontalTextPosition(AbstractButton.CENTER);
b1.setMnemonic(KeyEvent.VK_U);
b1.setToolTipText("Click this if you are unhappy.");
```

Initialising JButton object (2)

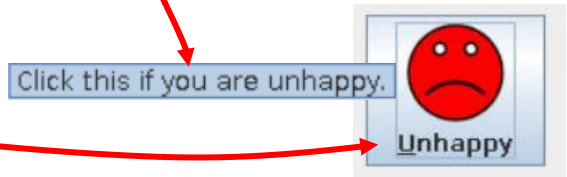
- Define icon (image file) and text

```
ImageIcon unhappyButtonIcon = new ImageIcon("unhappy.png");
b1 = new JButton("Unhappy", unhappyButtonIcon);
b1.setSize(100,100);
b1.setVerticalTextPosition(AbstractButton.BOTTOM);
b1.setHorizontalTextPosition(AbstractButton.CENTER);
b1.setMnemonic(KeyEvent.VK_U);
b1.setToolTipText("Click this if you are unhappy.");
```

Initialising JButton object (3)

- Define mnemonic and tool tip text

```
ImageIcon unhappyButtonIcon = new ImageIcon("unhappy.png");
b1 = new JButton("Unhappy", unhappyButtonIcon);
b1.setSize(100,100);
b1.setVerticalTextPosition(AbstractButton.BOTTOM);
b1.setHorizontalTextPosition(AbstractButton.CENTER);
b1.setMnemonic(KeyEvent.VK_U);
b1.setToolTipText("Click this if you are unhappy.");
```

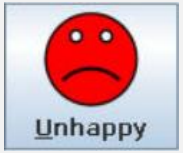


Button example: initialise

```java
import javax.swing.*;
import javax.swing.border.*;
import java.awt.BorderLayout;
import java.awt.event.*;
public class ButtonExample1  {
    public static void main(String[] args) {
        JButton b1, b2, b3;
        JLabel questionLabel, responseLabel;
        JFrame frame = new JFrame();
        questionLabel = new JLabel("Tell me how happy you are with Java!\n",
            SwingConstants.CENTER);
        responseLabel = new JLabel("No answer given",
            SwingConstants.CENTER);

        // Create button icons
        ImageIcon unhappyButtonIcon = new ImageIcon("unhappy.png");
        ImageIcon neutralButtonIcon = new ImageIcon("neutral.png");
        ImageIcon happyButtonIcon = new ImageIcon("happy.png");
```

Button example: define buttons

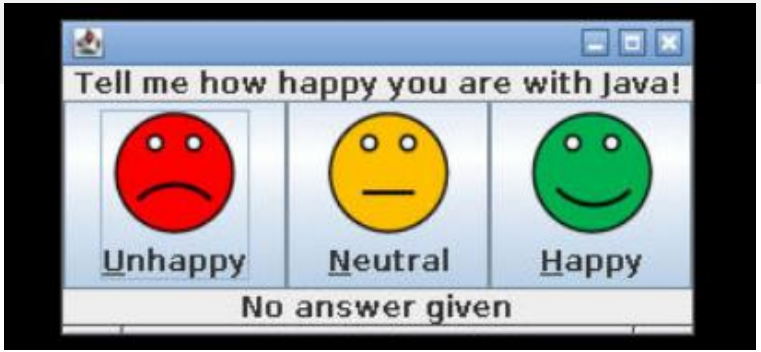
```
19          // Define unhappy button
20          b1 = new JButton("Unhappy", unhappyButtonIcon);
21          b1.setVerticalTextPosition(AbstractButton.BOTTOM);
22          b1.setHorizontalTextPosition(AbstractButton.CENTER);
23          b1.setMnemonic(KeyEvent.VK_U);
24          b1.setToolTipText("Click this if you are unhappy.");
25          // Define neutral button
26          b2 = new JButton("Neutral", neutralButtonIcon);
27          b2.setVerticalTextPosition(AbstractButton.BOTTOM);
28          b2.setHorizontalTextPosition(AbstractButton.CENTER);
29          b2.setMnemonic(KeyEvent.VK_N);
30          b2.setToolTipText("Click this if you feel neutral.");
31          // Define happy button
32          b3 = new JButton("Happy", happyButtonIcon);
33          b3.setVerticalTextPosition(AbstractButton.BOTTOM);
34          b3.setHorizontalTextPosition(AbstractButton.CENTER);
35          b3.setMnemonic(KeyEvent.VK_H);
36          b3.setToolTipText("Click this if you are happy.");
```



Button example: layout

```
37
38          // Add Components to the frame
39          frame.add(questionLabel,BorderLayout.PAGE_START);
40          frame.add(b1,BorderLayout.LINE_START);
41          frame.add(b2,BorderLayout.CENTER);
42          frame.add(b3,BorderLayout.LINE_END);
43          frame.add(responseLabel,BorderLayout.PAGE_END);
44
45          frame.pack();
46          frame.setVisible(true);
47      }
48  }
```

Using BorderLayout



Check boxes: class JCheckBox

- The `JCheckBox` class provides support for check box buttons
- For check boxes in menus use the `JCheckBoxMenuItem` class
- `JCheckbox` inherits `JAbstractButton`; therefore, it has the usual button characteristics and methods available for buttons

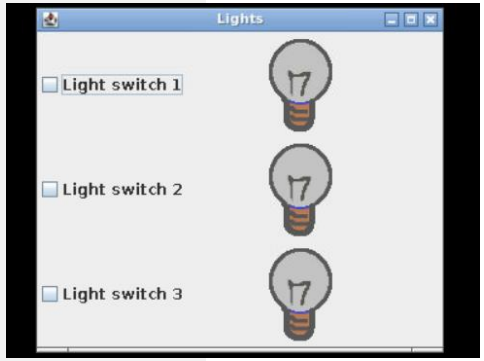
Check box example: use GridLayout

```java
import javax.swing.*;
import java.awt.*;
public class CheckBoxExample1  {
  public static void main(String[] args) {
    JFrame frame = new JFrame("Lights");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    ImageIcon lightOffIcon = new ImageIcon("light_off.png");
    ImageIcon lightOnIcon = new ImageIcon("light_on.png");
    JLabel lights[] = new JLabel[3];
    JCheckBox cb[] = new JCheckBox[3];
    JPanel panel = new JPanel();
    GridLayout gridlayout = new GridLayout(3,2);
    panel.setLayout(gridlayout);
    for(int i=0; i<3; i++) {
      lights[i] = new JLabel();
      cb[i] = new JCheckBox("Light switch " + (i+1));
      lights[i].setIcon(lightOffIcon);
      panel.add(cb[i]);
      panel.add(lights[i]);
    }
    frame.add(panel);
    frame.setSize(500,500);
    frame.setVisible(true);
  }
}
```

Note that a label can contain an image (icon) instead of text!

Check box example: define check boxes

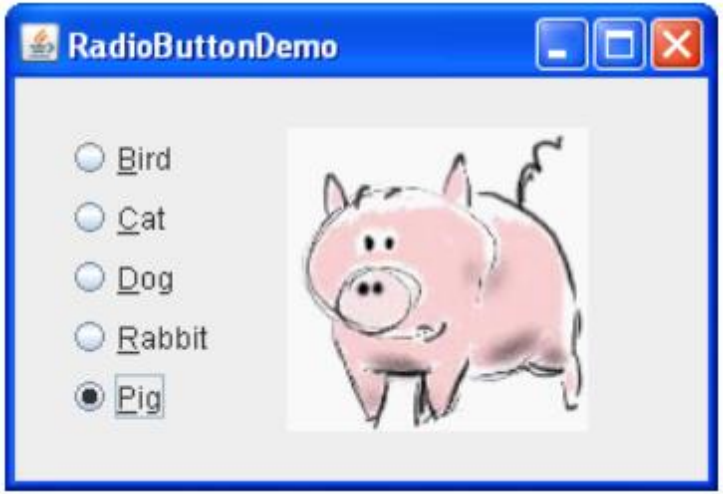
```
1   import javax.swing.*;
2   import java.awt.*;
3   public class CheckBoxExample1  {
4     public static void main(String[] args) {
5       JFrame frame = new JFrame("Lights");
6       frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
7       ImageIcon lightOffIcon = new ImageIcon("light_off.png");
8       ImageIcon lightOnIcon = new ImageIcon("light_on.png");
9       JLabel lights[] = new JLabel[3];
10      JCheckBox cb[] = new JCheckBox[3];
11      JPanel panel = new JPanel();
12      GridLayout gridlayout = new GridLayout(3,2);
13      panel.setLayout(gridlayout);
14      for(int i=0; i<3; i++) {
15        lights[i] = new JLabel();
16        cb[i] = new JCheckBox("Light switch " + (i+1));
17        lights[i].setIcon(lightOffIcon);
```




```
18        panel.add(cb[i]);
19        panel.add(lights[i]);
20      }
21      frame.add(panel);
22      frame.setSize(500,500);
23      frame.setVisible(true);
24    }
25  }
```

Radio buttons: class JRadioButton

- The **JRadioButton** class provides support for check box buttons
- For check boxes in menus use the **JRadioButtonMenuItem** class
- **JRadioButton** also inherits from **JAbstractButton**, therefore it has the usual button characteristics and methods available for buttons
- Use **ButtonGroup** to make sure only one button is checked at time



Radio button example: use BoxLayout

```java
import javax.swing.*;
public class RadioButtonExample1  {
  public static void main(String[] args) {
    JFrame frame = new JFrame("Quiz");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    JPanel panel = new JPanel();
    BoxLayout boxlayout = new BoxLayout(panel, BoxLayout.Y_AXIS);
    panel.setLayout(boxlayout);
    JLabel question = new JLabel("What is the capital of China?");
    JButton submit = new JButton("Submit your answer");
    ButtonGroup group = new ButtonGroup();
    JRadioButton rb[] = new JRadioButton[4];
    rb[0] = new JRadioButton("Shanghai");
    rb[1] = new JRadioButton("Beijing");
    rb[2] = new JRadioButton("Guangzhou");
    rb[3] = new JRadioButton("Chongqing");
    submit.setEnabled(false);
    panel.add(question);
    for(int i=0; i<4; i++) {
        group.add(rb[i]);
        panel.add(rb[i]);
    }
    panel.add(submit);
    frame.add(panel);
    frame.pack();
    frame.setVisible(true);
  }
}
```

Radio button example

```java
import javax.swing.*;
public class RadioButtonExample1  {
  public static void main(String[] args) {
    JFrame frame = new JFrame("Quiz");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    JPanel panel = new JPanel();
    BoxLayout boxlayout = new BoxLayout(panel, BoxLayout.Y_AXIS);
    panel.setLayout(boxlayout);
    JLabel question = new JLabel("What is the capital of China?");
    JButton submit = new JButton("Submit your answer");
    ButtonGroup group = new ButtonGroup();
    JRadioButton rb[] = new JRadioButton[4];
    rb[0] = new JRadioButton("Shanghai");
    rb[1] = new JRadioButton("Beijing");
    rb[2] = new JRadioButton("Guangzhou");
    rb[3] = new JRadioButton("Chongqing");
```


```java
    submit.setEnabled(false);
    panel.add(question);
    for(int i=0; i<4; i++) {
      group.add(rb[i]);
      panel.add(rb[i]);
    }
    panel.add(submit);
    frame.add(panel);
    frame.pack();
    frame.setVisible(true);
  }
}
```

Only one of the radio buttons in ButtonGroup can be pressed at the same time!

Questions, comments?

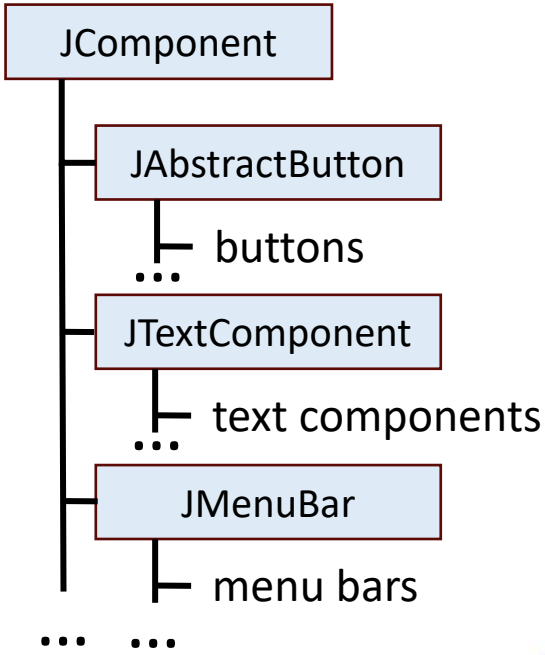
1495 UNIVERSITY OF ABERDEEN

JC2002 Java Programming

Day 7, Session 4: More JComponents

Selection of JComponents

- There is a large variety of different GUI components inherited from `JComponent` class for different purposes
  - In this session, we will only cover the most important classes
  - Different components have different methods for customising their appearance, adjusting their absolute size and position, setting and getting their internal state etc.



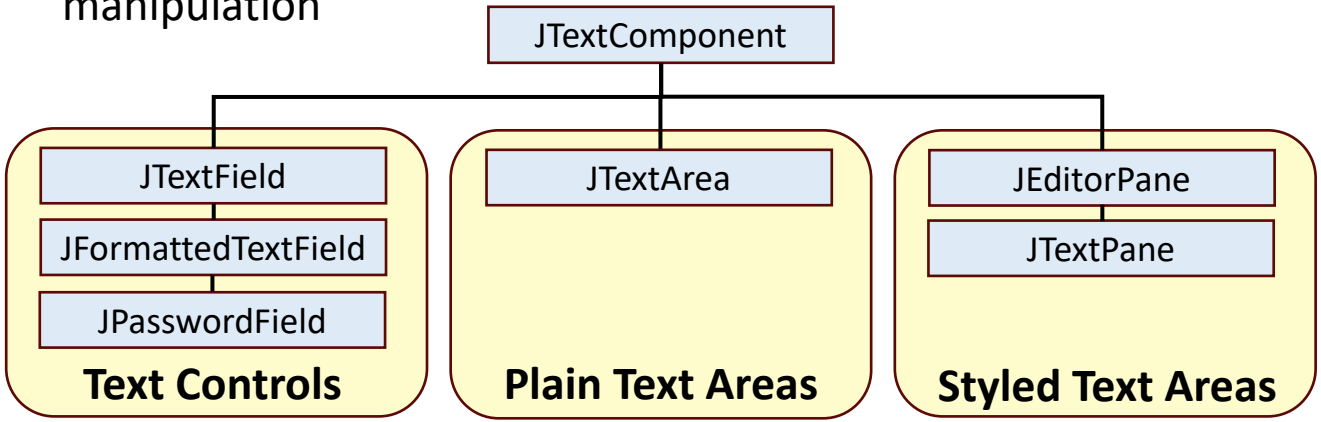
Using JLabel to display images

- `JLabel` is inherited directly from `JComponent` and it does not react to any user input
  - We have already used `JLabel` to show text, but it can be used also to display images by using its `setIcon()` method

```java
import javax.swing.*;
class LabelExample {
  public static void main(String args[]){
    JFrame f = new JFrame("Label Example");
    ImageIcon icon = new ImageIcon("image.jpg");
    JLabel label = new JLabel();
    label.setIcon(icon);
    f.add(label);
    f.setSize(200,150);
    f.setVisible(true);
  }
}
```

Text components

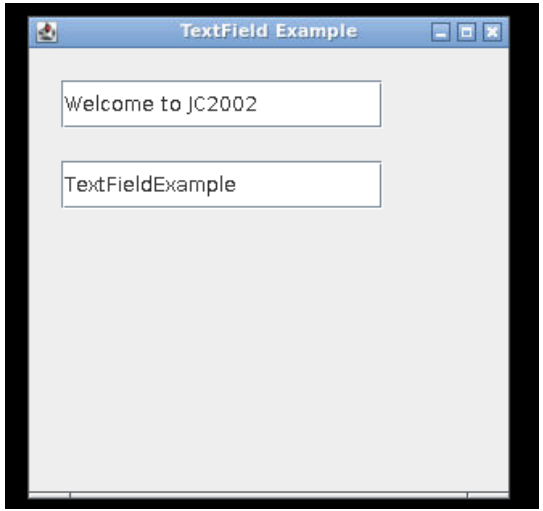
- Swing provides six text components, along with supporting classes and interfaces, for displaying and editing text
  - All the Swing text components inherit from `JTextComponent`, which provides a highly configurable and powerful foundation for text manipulation



Example of JTextField

- Use `JTextField` class to create a component that allows editing a single line of text

```
import javax.swing.*;
class TextFieldExample{
  public static void main(String args[]){
    JFrame f= new JFrame("TextField Example");
    JTextField t1,t2;
    t1 = new JTextField("Welcome to JC2002");
    t1.setBounds(20,20, 200,30);
    t2=new JTextField("TextFieldExample");
    t2.setBounds(20,70, 200,30);
    f.add(t1); f.add(t2);
    f.setSize(300,300);
    f.setLayout(null);
    f.setVisible(true);
  }
}
```



Example of JTextArea

- Use `JTextArea` class to create a component that allows editing a multiple lines of text

```
import javax.swing.*;
class TextAreaExample{
  public static void main(String args[]){
    JFrame f= new JFrame("TextField Example");
    JTextArea t1;
    t1 = new JTextField("Welcome to \nJC2002!");
    t1.setBounds(20,20, 200,100);
    f.add(t1);
    f.setSize(300,300);
    f.setLayout(null);
    f.setVisible(true);
  }
}
```

Example of JTextPane (1)

- Use `JTextPane` class for editing or displaying *styled* text (even HTML)

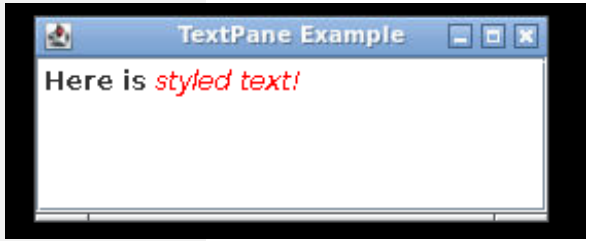
```
...
...
public class TextPaneExample {
    public static void main(String args[]) throws BadLocationException {
        JFrame frame = new JFrame("TextPane Example");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        Container cp = frame.getContentPane();
        JTextPane pane = new JTextPane();
        SimpleAttributeSet attributeSet = new SimpleAttributeSet();
        StyleConstants.setBold(attributeSet, true);
        pane.setCharacterAttributes(attributeSet, true);
        pane.setText("Here is ");
        attributeSet = new SimpleAttributeSet();
        StyleConstants.setItalic(attributeSet, true);
        StyleConstants.setForeground(attributeSet, Color.red);
        Document doc = pane.getStyledDocument();
        doc.insertString(doc.getLength(), "styled text!", attributeSet);
        JScrollPane scrollPane = new JScrollPane(pane);
        cp.add(scrollPane, BorderLayout.CENTER);
        frame.setSize(400, 300);
        frame.setVisible(true);
    }
}
```

Imports not shown here

Different styles (colours, italic, bold, etc.) can be defined for the text

Example of JTextPane (2)

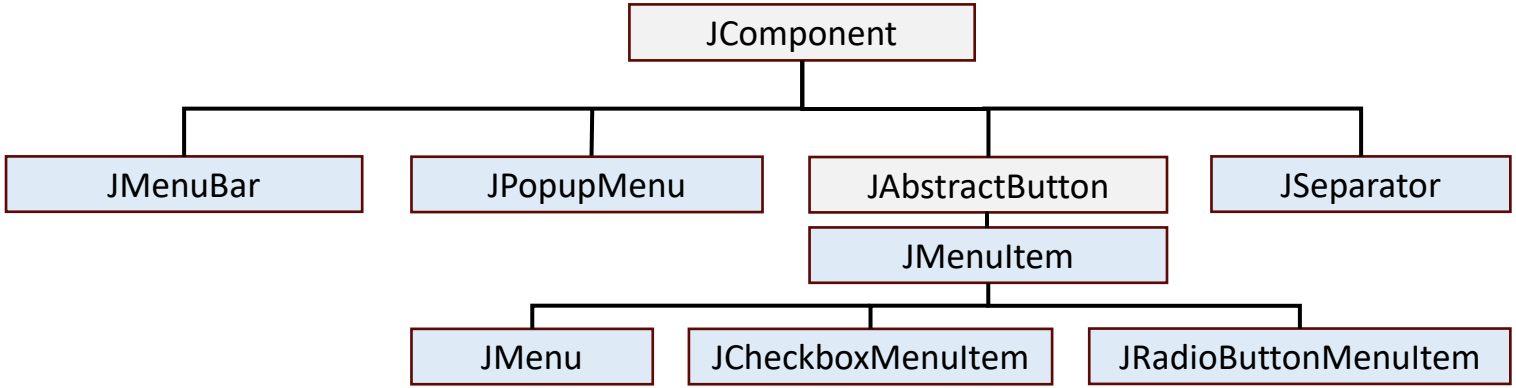
```
...
11  ...
12  public class TextPaneExample {
13      public static void main(String args[]) throws BadLocationException {
14          JFrame frame = new JFrame("TextPane Example");
15          frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
16          Container cp = frame.getContentPane();
17          JTextPane pane = new JTextPane();
18          SimpleAttributeSet attributeSet = new SimpleAttributeSet();
19          StyleConstants.setBold(attributeSet, true);
20          pane.setCharacterAttributes(attributeSet, true);
21          pane.setText("Here is ");
22          attributeSet = new SimpleAttributeSet();
23          StyleConstants.setItalic(attributeSet, true);
24          StyleConstants.setForeground(attributeSet, Color.red);
25          Document doc = pane.getStyledDocument();
26          doc.insertString(doc.getLength(), "styled text!", attributeSet);
27          JScrollPane scrollPane = new JScrollPane(pane);
28          cp.add(scrollPane, BorderLayout.CENTER);
29          frame.setSize(200, 100);
30          frame.setVisible(true);
31        }
32  }
```



- For more information, see: https://docs.oracle.com/javase/tutorial/uiswing/components/editorpane.html

Menu components

- There are several menu components providing many options for implementing menus in a small space
- Menu items (`JMenuItem`) are different kinds of special buttons showing labels



Example of using menu components (1)

- Example of creating a menu bar with a submenu

```
1   import javax.swing.*;
2   class MenuExample {
3     JMenu menu, submenu;
4     JMenuItem i1, i2, i3, i4, i5;
5     MenuExample(){
6       JFrame f = new JFrame("Menu and MenuItem Example");
7       JMenuBar mb=new JMenuBar();
8       menu=new JMenu("Menu");
9       submenu=new JMenu("Sub Menu");
10      i1=new JMenuItem("Item 1");
11      i2=new JMenuItem("Item 2");
12      i3=new JMenuItem("Item 3");
13      i4=new JMenuItem("Item 4");
14      i5=new JMenuItem("Item 5");
15      menu.add(i1);
16      menu.add(i2);
17      menu.add(i3);
18      submenu.add(i4);
19      submenu.add(i5);
20      menu.add(submenu);
21      mb.add(menu);
22      f.setJMenuBar(mb);
23      f.setSize(400,400);
24      f.setLayout(null);
25      f.setVisible(true);
26    }
27    public static void main(String args[]){
28      new MenuExample();
29    }
30  }
```

Example of using menu components (2)

- Instantiate menu bar, menu, and submenu

```
1   import javax.swing.*;
2   class MenuExample {
3     JMenu menu, submenu;
4     JMenuItem i1, i2, i3, i4, i5;
5     MenuExample(){
6       JFrame f = new JFrame("Menu and MenuItem Example");
7       JMenuBar mb=new JMenuBar();
8       menu=new JMenu("Menu");
9       submenu=new JMenu("Sub Menu");
10      i1=new JMenuItem("Item 1");
11      i2=new JMenuItem("Item 2");
12      i3=new JMenuItem("Item 3");
13      i4=new JMenuItem("Item 4");
14      i5=new JMenuItem("Item 5");
15      menu.add(i1);
16      menu.add(i2);
17      menu.add(i3);
18      submenu.add(i4);
19      submenu.add(i5);
20      menu.add(submenu);
21      mb.add(menu);
22      f.setJMenuBar(mb);
23      f.setSize(400,400);
24      f.setLayout(null);
25      f.setVisible(true);
26    }
27    public static void main(String args[]){
28      new MenuExample();
29    }
30  }
```

Example of using menu components (3)

- Create five menu item objects of class `JMenuItem`

```
1   import javax.swing.*;
2   class MenuExample {
3     JMenu menu, submenu;
4     JMenuItem i1, i2, i3, i4, i5;
5     MenuExample(){
6       JFrame f = new JFrame("Menu and MenuItem Example");
7       JMenuBar mb=new JMenuBar();
8       menu=new JMenu("Menu");
9       submenu=new JMenu("Sub Menu");
10      i1=new JMenuItem("Item 1");
11      i2=new JMenuItem("Item 2");
12      i3=new JMenuItem("Item 3");
13      i4=new JMenuItem("Item 4");
14      i5=new JMenuItem("Item 5");
15      menu.add(i1);
16      menu.add(i2);
17      menu.add(i3);
18      submenu.add(i4);
19      submenu.add(i5);
20      menu.add(submenu);
21      mb.add(menu);
22      f.setJMenuBar(mb);
23      f.setSize(400,400);
24      f.setLayout(null);
25      f.setVisible(true);
26    }
27    public static void main(String args[]){
28      new MenuExample();
29    }
30  }
```

Example of using menu components (4)

- Add the first three menu items to the main menu

```
1   import javax.swing.*;
2   class MenuExample {
3     JMenu menu, submenu;
4     JMenuItem i1, i2, i3, i4, i5;
5     MenuExample(){
6       JFrame f = new JFrame("Menu and MenuItem Example");
7       JMenuBar mb=new JMenuBar();
8       menu=new JMenu("Menu");
9       submenu=new JMenu("Sub Menu");
10      i1=new JMenuItem("Item 1");
11      i2=new JMenuItem("Item 2");
12      i3=new JMenuItem("Item 3");
13      i4=new JMenuItem("Item 4");
14      i5=new JMenuItem("Item 5");
15      menu.add(i1);
16      menu.add(i2);
17      menu.add(i3);
18      submenu.add(i4);
19      submenu.add(i5);
20      menu.add(submenu);
21      mb.add(menu);
22      f.setJMenuBar(mb);
23      f.setSize(400,400);
24      f.setLayout(null);
25      f.setVisible(true);
26    }
27    public static void main(String args[]){
28      new MenuExample();
29    }
30  }
```

Example of using menu components (5)

- Add the last two menu items to the submenu

```java
import javax.swing.*;
class MenuExample {
  JMenu menu, submenu;
  JMenuItem i1, i2, i3, i4, i5;
  MenuExample(){
    JFrame f = new JFrame("Menu and MenuItem Example");
    JMenuBar mb=new JMenuBar();
    menu=new JMenu("Menu");
    submenu=new JMenu("Sub Menu");
    i1=new JMenuItem("Item 1");
    i2=new JMenuItem("Item 2");
    i3=new JMenuItem("Item 3");
    i4=new JMenuItem("Item 4");
    i5=new JMenuItem("Item 5");
    menu.add(i1);
    menu.add(i2);
    menu.add(i3);
    submenu.add(i4);
    submenu.add(i5);
    menu.add(submenu);
    mb.add(menu);
    f.setJMenuBar(mb);
    f.setSize(400,400);
    f.setLayout(null);
    f.setVisible(true);
  }
  public static void main(String args[]){
    new MenuExample();
  }
}
```

Example of using menu components (6)

- Add submenu to the main menu, and main menu to the menu bar

```
1   import javax.swing.*;
2   class MenuExample {
3     JMenu menu, submenu;
4     JMenuItem i1, i2, i3, i4, i5;
5     MenuExample(){
6       JFrame f = new JFrame("Menu and MenuItem Example");
7       JMenuBar mb=new JMenuBar();
8       menu=new JMenu("Menu");
9       submenu=new JMenu("Sub Menu");
10      i1=new JMenuItem("Item 1");
11      i2=new JMenuItem("Item 2");
12      i3=new JMenuItem("Item 3");
13      i4=new JMenuItem("Item 4");
14      i5=new JMenuItem("Item 5");
15      menu.add(i1);
16      menu.add(i2);
17      menu.add(i3);
18      submenu.add(i4);
19      submenu.add(i5);
20      menu.add(submenu);
21      mb.add(menu);
22      f.setJMenuBar(mb);
23      f.setSize(400,400);
24      f.setLayout(null);
25      f.setVisible(true);
26    }
27    public static void main(String args[]){
28      new MenuExample();
29    }
30  }
```

Example of using menu components (7)

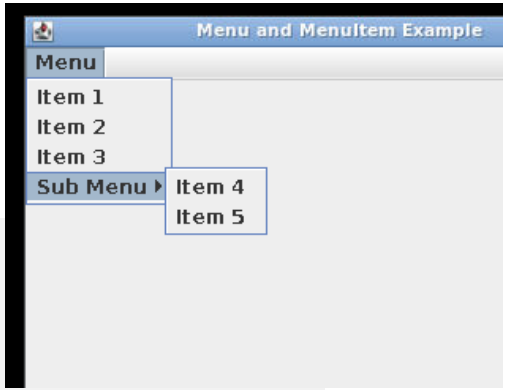
- Add menu bar to the frame using method `setJMenuBar()`

```
1   import javax.swing.*;
2   class MenuExample {
3     JMenu menu, submenu;
4     JMenuItem i1, i2, i3, i4, i5;
5     MenuExample(){
6       JFrame f = new JFrame("Menu and MenuItem Example");
7       JMenuBar mb=new JMenuBar();
8       menu=new JMenu("Menu");
9       submenu=new JMenu("Sub Menu");
10      i1=new JMenuItem("Item 1");
11      i2=new JMenuItem("Item 2");
12      i3=new JMenuItem("Item 3");
13      i4=new JMenuItem("Item 4");
14      i5=new JMenuItem("Item 5");
15      menu.add(i1);
16      menu.add(i2);
17      menu.add(i3);
18      submenu.add(i4);
19      submenu.add(i5);
20      menu.add(submenu);
21      mb.add(menu);
22      f.setJMenuBar(mb);
23      f.setSize(400,400);
24      f.setLayout(null);
25      f.setVisible(true);
26    }
27    public static void main(String args[]){
28      new MenuExample();
29    }
30  }
```

Example of using menu components (8)

- Run the program

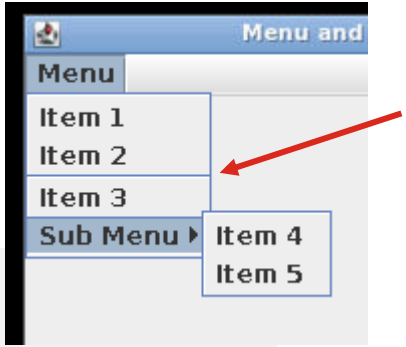
```java
import javax.swing.*;
class MenuExample {
  JMenu menu, submenu;
  JMenuItem i1, i2, i3, i4, i5;
  MenuExample(){
    JFrame f = new JFrame("Menu and MenuItem Example");
    JMenuBar mb=new JMenuBar();
    menu=new JMenu("Menu");
    submenu=new JMenu("Sub Menu");
    i1=new JMenuItem("Item 1");
    i2=new JMenuItem("Item 2");
    i3=new JMenuItem("Item 3");
    i4=new JMenuItem("Item 4");
    i5=new JMenuItem("Item 5");
    menu.add(i1);
    menu.add(i2);
    menu.add(i3);
    submenu.add(i4);
    submenu.add(i5);
    menu.add(submenu);
    mb.add(menu);
    f.setJMenuBar(mb);
    f.setSize(400,400);
    f.setLayout(null);
    f.setVisible(true);
  }
  public static void main(String args[]){
    new MenuExample();
  }
}
```



Example of using JSeparator

- Use `JSeparator` instance to separate groups of menu items

```
1   import javax.swing.*;
2   class MenuExample2 {
3     JMenu menu, submenu;
4     JMenuItem i1, i2, i3, i4, i5;
5     JSeparator s;
6     MenuExample(){
7       JFrame f = new JFrame("Menu and MenuItem Example");
8       JMenuBar mb=new JMenuBar();
9       menu=new JMenu("Menu");
10      submenu=new JMenu("Sub Menu");
11      i1=new JMenuItem("Item 1");
12      i2=new JMenuItem("Item 2");
13      i3=new JMenuItem("Item 3");
14      i4=new JMenuItem("Item 4");
15      i5=new JMenuItem("Item 5");
16      s=new JSeparator();
17      menu.add(i1);
18      menu.add(i2);
19      menu.add(s);
20      menu.add(i3);
21      submenu.add(i4);
22      submenu.add(i5);
23      menu.add(submenu);
24      mb.add(menu);
25      f.setJMenuBar(mb);
26      f.setSize(400,400);
27      f.setLayout(null);
28      f.setVisible(true);
29    }
30    public static void main(String args[]){
31      new MenuExample();
32    }
33  }
```



Summary

- A graphical user interface provides a user-friendly mechanism to interact with an app that is easy to learn
  - GUIs are usually based on similar logic, but different look-and-feel in different platforms
- In Java, Swing is a commonly used library for implementing GUI
  - In Swing, different GUI components are implemented using classes inherited from `JComponent` class
  - Swing allows different layouts for organising the GUI components
  - Some of the most typical GUI components were introduced, including buttons, text fields, and menu items

Questions, comments?