JC2002 Java Programming

Lecture 21: Using layouts and buttons in Swing

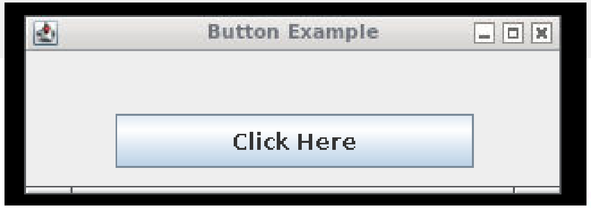
Adding JComponents

- 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

```
import javax.swing.*;
public class ButtonExample {
  public static void main(String[] args) {
    JFrame f=new JFrame("Button Example");
    JButton b=new JButton("Click Here");
    b.setBounds(50,35,200,30);
    f.add(b);
    f.setSize(300,100);
    f.setLayout(null);
    f.setVisible(true);
  }
}
```

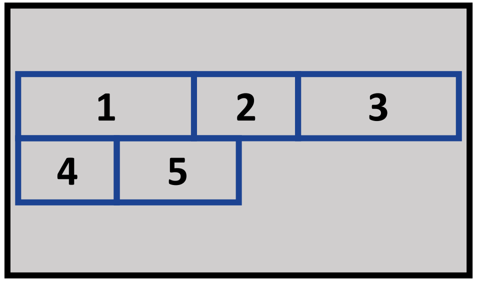
Button location and size

Add button

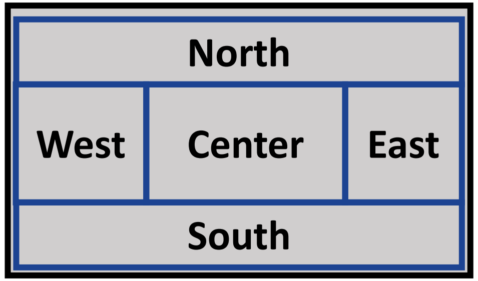


Layout managers

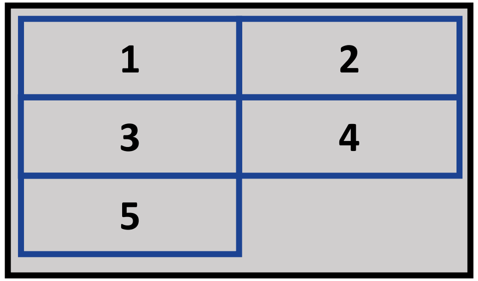
- Setting absolute positions and sizes for the components may look bad if you do not know the screen resolution of the target platform
- Several Swing and AWT classes provide layout managers for dynamic allocation of GUI components, according to different specific rules



FlowLayout (default)



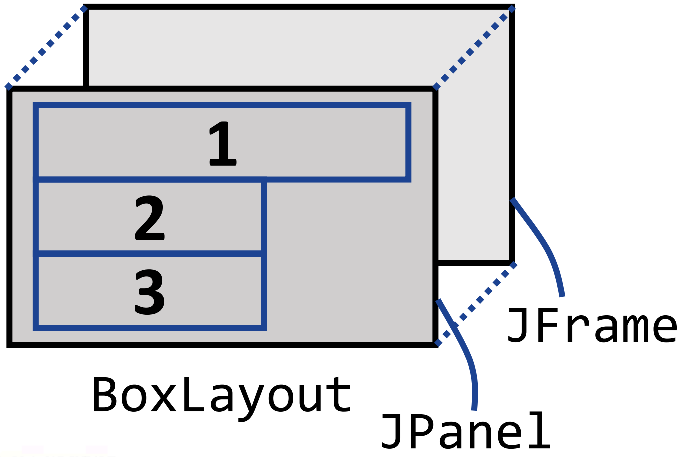
BorderLayout



GridLayout

Layouts on JPanel

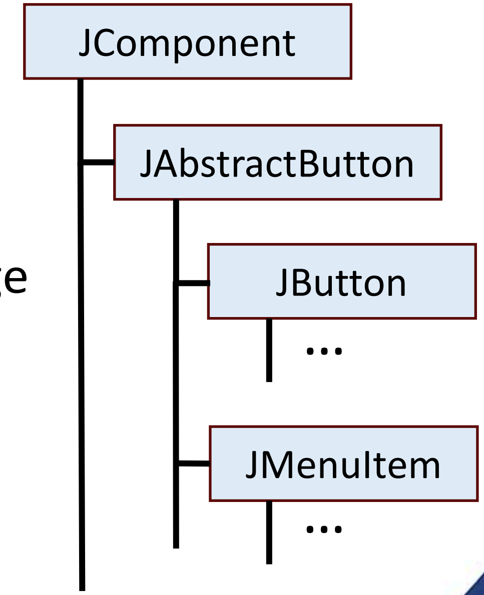
- Some layouts, like `FlowLayout`, can be used on `JFrame` directly
- Some layouts, like `GridLayout` and `BoxLayout`, require creating `JPanel` object between `JFrame` and the components



```
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)

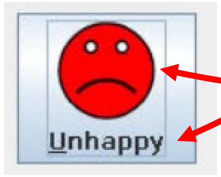
- Example button with both icon and text (both are optional)

```java
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

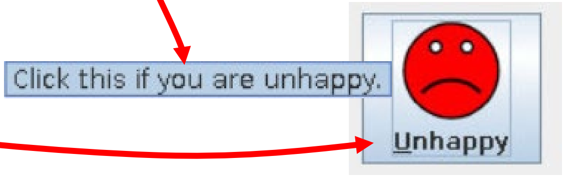
```
ImageIcon unhappyButtonIcon = new ImageIcon("unhappy.png");
b1 = new JButton("Unhappy", unhappyButtonIcon);
b1.setSize(100,100);
b1.setVerticalTextPosition(AbstractButton.BOTTOM);
b1.setHorizontalTextPosition(AbstractButton.CENTER);
b1.setMnemonic(KeyEvent.VK_U);
b1.setToolTipText("Click this if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



Initialising JButton object (3)

- Define mnemonic and tool tip text

```java
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.");
```

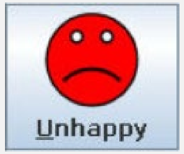


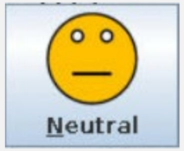
Button example: initialise

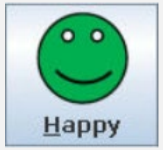
```java
import javax.swing.*;
import javax.swing.border.*;
import java.awt.BorderLayout;
import java.awt.event.*;
public class ButtonExample1  {
    public static void main(String[] args) {
        JButton b1, b2, b3;
        JLabel questionLabel, responseLabel;
        JFrame frame = new JFrame();
        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

```
        // Define unhappy button
        b1 = new JButton("Unhappy", unhappyButtonIcon);
        b1.setVerticalTextPosition(AbstractButton.BOTTOM);
        b1.setHorizontalTextPosition(AbstractButton.CENTER);
        b1.setMnemonic(KeyEvent.VK_U);
        b1.setToolTipText("Click this if you are unhappy.");
        // Define neutral button
        b2 = new JButton("Neutral", neutralButtonIcon);
        b2.setVerticalTextPosition(AbstractButton.BOTTOM);
        b2.setHorizontalTextPosition(AbstractButton.CENTER);
        b2.setMnemonic(KeyEvent.VK_N);
        b2.setToolTipText("Click this if you feel neutral.");
        // Define happy button
        b3 = new JButton("Happy", happyButtonIcon);
        b3.setVerticalTextPosition(AbstractButton.BOTTOM);
        b3.setHorizontalTextPosition(AbstractButton.CENTER);
        b3.setMnemonic(KeyEvent.VK_H);
        b3.setToolTipText("Click this if you are happy.");
```



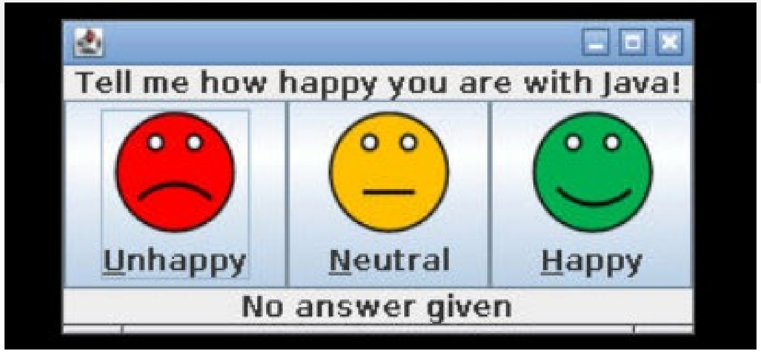




Button example: layout

```
        // Add Components to the frame
        frame.add(questionLabel,BorderLayout.PAGE_START);
        frame.add(b1,BorderLayout.LINE_START);
        frame.add(b2,BorderLayout.CENTER);
        frame.add(b3,BorderLayout.LINE_END);
        frame.add(responseLabel,BorderLayout.PAGE_END);

        frame.pack();
        frame.setVisible(true);
    }
}
```

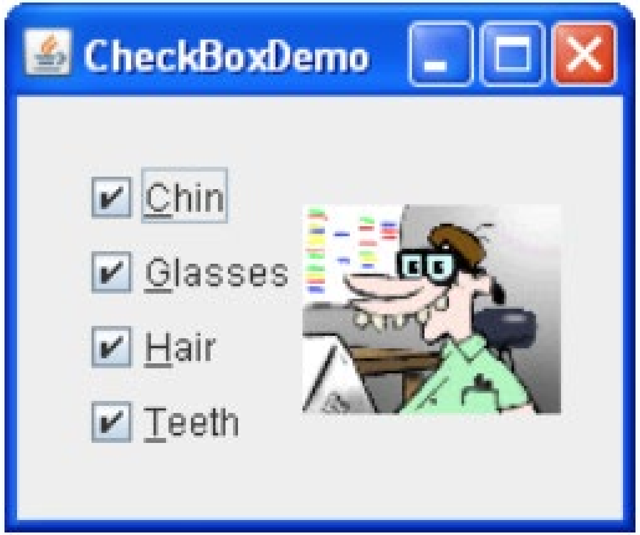
$\left.\right\}$ Using BorderLayout


Tell me how happy you are with Java!

No answer given

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



UNIVERSITY OF ABERDEEN

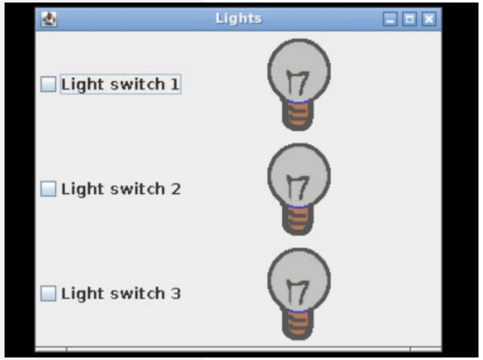
Check box example: use GridLayout

```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!

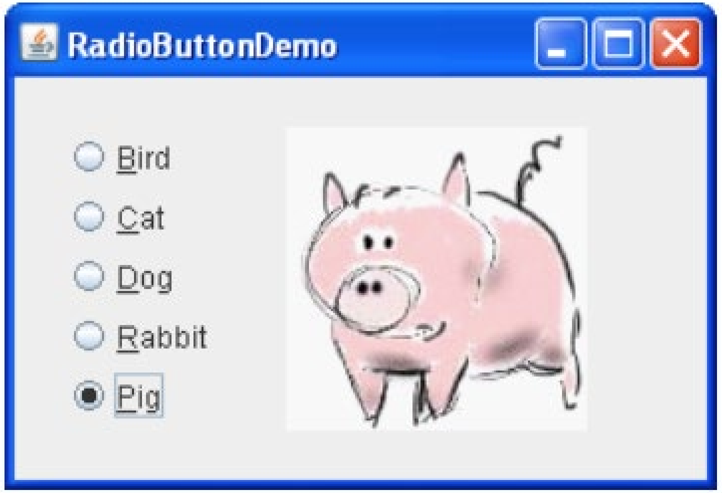
Check box example: define check boxes

```
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);
  }
}
```



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



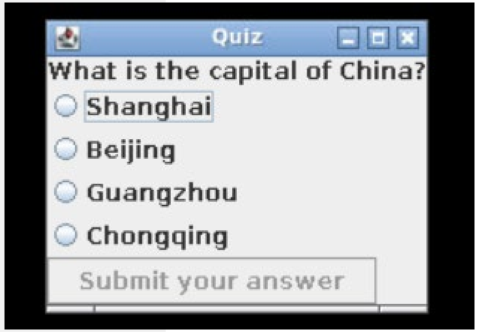
UNIVERSITY OF ABERDEEN

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

```
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);
  }
}
```



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

Questions, comments?