1495 UNIVERSITY OF ABERDEEN

# JC2002 Java Programming

## Day 8: Event-driven programming (AI, CS)

Friday, 10 November

1495 UNIVERSITY OF ABERDEEN

# JC2002 Java Programming

## Day 8, Session 1: Event listeners and event handlers

# References and learning objectives

- Today's sessions are mostly based on Oracle documentation:
  - https://docs.oracle.com/en/java/javase/11/docs/api/java.desktop/java/awt/AWTEvent.html
  - https://docs.oracle.com/en/java/javase/11/docs/api/java.desktop/java/awt/event/AWTEventListener.html
- After today's session, you should be able to:
  - Explain the concept of event-driven programming in Java
  - Use event listeners to implement interactive GUI elements
  - Select proper event listeners to different applications

# Event listeners

- JAVA API provides several *event listeners* that can be used to respond to events fired by swing components, for example:
  - **`ActionListener`:** listens to events fired by clickable components such as buttons, combo boxes, menu items, toggle buttons , etc.
  - **`ItemListener`:** listens to events fired by components that implement the `ItemSelectable` interface such as check boxes, check menu items, combo boxes, etc.
  - **`WindowListener`:** listens to events fired after some window activity such as opening or closing a window, focusing and defocusing a window, maximising the window, etc.

# Action listeners

- An action event occurs, whenever an action is performed by the user (e.g., clicking a button)
- To create an action listener object, you need to declare a class that implements the **`ActionListener`** interface *or* extends a class that implements an **`ActionListener`** interface
  - Action listener is usually assigned to a GUI component using **`addActionListener()`** method of the component; however, the implementation details depend on the component class
  - When an action event occurs, the program executes method **`actionPerformed()`** of the action listener: You can implement the desired functionality in this method

# Button example with action listener (1)

```java
import javax.swing.*;
import javax.swing.border.*;
import java.awt.BorderLayout;
import java.awt.event.*;
public class ButtonExample2 {
    protected static JButton b1, b2, b3;
    protected static JLabel questionLabel, responseLabel;
    static class ActionHandler implements ActionListener {
        public void actionPerformed(ActionEvent e) {
            if(e.getActionCommand().equals("happy")) {
                responseLabel.setText("Glad to hear you are happy!");
            } else if(e.getActionCommand().equals("neutral")) {
                responseLabel.setText("Thank you for your feedback!");
            } else if(e.getActionCommand().equals("unhappy")) {
                responseLabel.setText("Sorry to hear you are not happy.");
            }
        }
    }
```

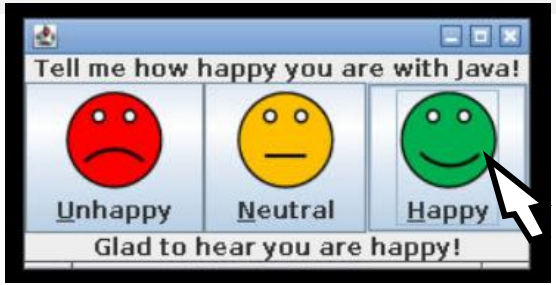
Nested class (class inside class) for handling events

Implements the required actions

# Button example with action listener (2)

```
...   ...
25    ActionHandler actionHandler = new ActionHandler();
...   ...
51    b1.setActionCommand("unhappy");
52    b1.addActionListener(actionHandler);
53    b2.setActionCommand("neutral");
54    b2.addActionListener(actionHandler);
55    b3.setActionCommand("happy");
56    b3.addActionListener(actionHandler);
...   ...
```

Set action commands and assign the action listener object to the buttons

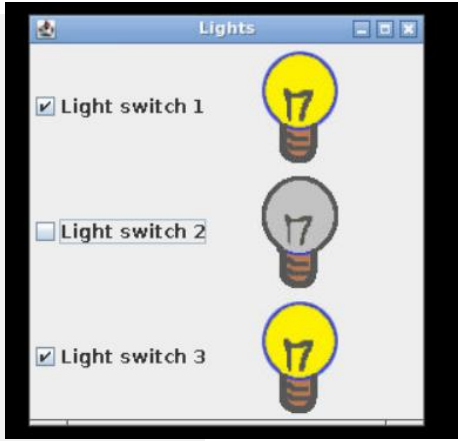


# Item listener

- An item event occurs, when a check box is clicked
- To create an item listener object, you need to declare a class that implements the **`ItemListener`** interface *or* extends a class that implements an **`ItemListener`** interface
  - When an item event occurs, the program executes method **`itemStateChanged(ItemEvent e)`**
  - You can test if the item is checked or unchecked by invoking ItemEvent method **`getStateChange()`**, that can be either ***`ItemEvent.SELECTED`*** or ***`ItemEvent.DESELECTED`***
  - You can use method **`getSource()`** to obtain the button that is the source of the event

# Check box example with item listener

```
...   ...
3     import java.awt.event.*;
4     public class CheckBoxExample2  {
...   ...
22        ItemListener itemHandler = new ItemListener() {
23          public void itemStateChanged(ItemEvent e) {
24            for(int i=0; i<3; i++) {
25              if(e.getSource()==cb[i]) {
26                if(e.getStateChange() == ItemEvent.SELECTED) {
27                  lights[i].setIcon(lightOnIcon);
28                } else {
29                  lights[i].setIcon(lightOffIcon);
30                }
31              }
32            }
33          }
34        };
35        for(int i=0; i<3; i++)
36          cb[i].addItemListener(itemHandler);
...   ...
```




Anonymous class implementing itemStateChanged() defined as item listener

# Multiple event listeners

- In the previous examples, we have used one nested class or an anonymous class as event listeners
  - Works ok when only simple functionality is required
- It is also possible to implement several event listeners for different components

Radio button event

Submit button event

# Radio button example with events (1)

```
1   import javax.swing.*;
2   import java.awt.event.*;
3
4   public class RadioButtonExample2 {
5
6     protected static JLabel question;
7     protected static JButton submit;
8     protected static JRadioButton rb[];
9     protected static JFrame frame;
10    protected static JPanel panel;
11    protected static ButtonGroup group;
12
13    protected static class RadioListener implements ActionListener {
14      public void actionPerformed(ActionEvent e) {
15        submit.setEnabled(true);
16      }
17    }
```

Components moved here so that they can be accessed by the event listeners

ActionListener for the radio buttons (just enables the submit button, in case it is not yet enabled)

# Radio button example with events (2)

```
18    protected static class SubmitListener implements ActionListener {
19      public void actionPerformed(ActionEvent e) {
20        if(rb[1].isSelected()) {
21          JOptionPane.showMessageDialog(frame,"Your answer is correct!");
22        } else {
23          JOptionPane.showMessageDialog(frame,"Your answer is incorrect!");
24        }
25        frame.dispatchEvent(new WindowEvent(frame,
26                                WindowEvent.WINDOW_CLOSING));
27      }
28    }
29    protected static SubmitListener submitListener;
30    protected static RadioListener radioListener;
```

Close the frame and quit the program after showing the message dialog

ActionListener for the submit button (checks if the answer is correct or not)

# Radio button example with events (3)

```
31      public static void main(String[] args) {
32          frame = new JFrame();
33          frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
34          panel = new JPanel();
35          BoxLayout boxlayout = new BoxLayout(panel, BoxLayout.Y_AXIS);
36          panel.setLayout(boxlayout);
37          question = new JLabel("What is the capital of China?");
38          panel.add(question);
39          submit = new JButton("Submit your answer");
40          submit.setEnabled(false);
41          submitListener = new SubmitListener();
42          submit.addActionListener(submitListener);
43          rb = new JRadioButton[4];
44          rb[0] = new JRadioButton("Shanghai");
45          rb[1] = new JRadioButton("Beijing");
46          rb[2] = new JRadioButton("Guangzhou");
47          rb[3] = new JRadioButton("Chongqing");
```

Initialize the window and the layout

# Radio button example with events (4)

```
31      public static void main(String[] args) {
32          frame = new JFrame();
33          frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
34          panel = new JPanel();
35          BoxLayout boxlayout = new BoxLayout(panel, BoxLayout.Y_AXIS);
36          panel.setLayout(boxlayout);
37          question = new JLabel("What is the capital of China?");
38          panel.add(question);
39          submit = new JButton("Submit your answer");
40          submit.setEnabled(false);
41          submitListener = new SubmitListener();
42          submit.addActionListener(submitListener);
43          rb = new JRadioButton[4];
44          rb[0] = new JRadioButton("Shanghai");
45          rb[1] = new JRadioButton("Beijing");
46          rb[2] = new JRadioButton("Guangzhou");
47          rb[3] = new JRadioButton("Chongqing");
```

Initialize submit button and assign to the action listener

# Radio button example with events (5)

```
31      public static void main(String[] args) {
32          frame = new JFrame();
33          frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
34          panel = new JPanel("Quiz");
35          BoxLayout boxlayout = new BoxLayout(panel, BoxLayout.Y_AXIS);
36          panel.setLayout(boxlayout);
37          question = new JLabel("What is the capital of China?");
38          panel.add(question);
39          submit = new JButton("Submit your answer");
40          submit.setEnabled(false);
41          submitListener = new SubmitListener();
42          submit.addActionListener(submitListener);
43          rb = new JRadioButton[4];
44          rb[0] = new JRadioButton("Shanghai");
45          rb[1] = new JRadioButton("Beijing");
46          rb[2] = new JRadioButton("Guangzhou");
47          rb[3] = new JRadioButton("Chongqing");
```

Create radio buttons

# Radio button example with events (6)

```
48      group = new ButtonGroup();
49      radioListener = new RadioListener();
50      for(int i=0; i<4; i++) {
51        group.add(rb[i]);
52        panel.add(rb[i]);
53        rb[i].addActionListener(radioListener);
54      }
55      panel.add(submit);
56      frame.add(panel);
57      frame.pack();
58      frame.setVisible(true);
59    }
60  }
```

Assign radio buttons to a button group

# Radio button example with events (7)

```
48      group = new ButtonGroup();
49      radioListener = new RadioListener();
50      for(int i=0; i<4; i++) {
51        group.add(rb[i]);
52        panel.add(rb[i]);
53        rb[i].addActionListener(radioListener);
54      }
55      panel.add(submit);
56      frame.add(panel);
57      frame.pack();
58      frame.setVisible(true);
59    }
60  }
```

Assign radio buttons to an action listener

# Radio button example with events (8)

```
48      group = new ButtonGroup();
49      radioListener = new RadioListener();
50      for(int i=0; i<4; i++) {
51        group.add(rb[i]);
52        panel.add(rb[i]);
53        rb[i].addActionListener(radioListener);
54      }
55      panel.add(submit);
56      frame.add(panel);
57      frame.pack();
58      frame.setVisible(true);
59    }
60  }
```

Add radio buttons to the panel and finalise the view

# Questions, comments?

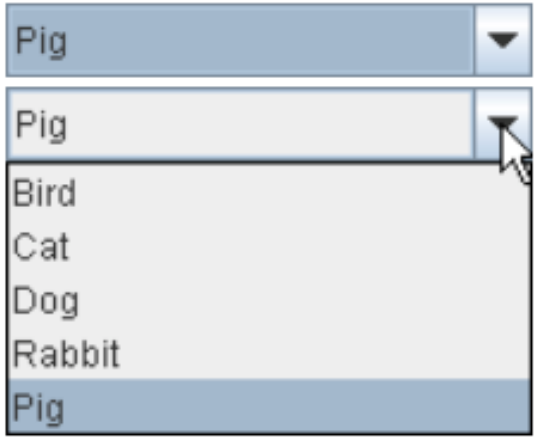
1495 UNIVERSITY OF ABERDEEN

# JC2002 Java Programming

## Day 8, Session 2: More examples of event handlers

# Combo boxes: JComboBox

- Combo boxes let the user choose one of several choices from a dropdown list
  - In Swing, combo boxes created with **`JComboBox`** component
  - Combo box events are **`ActionEvent`**s
- Two main forms of combo boxes: uneditable and editable



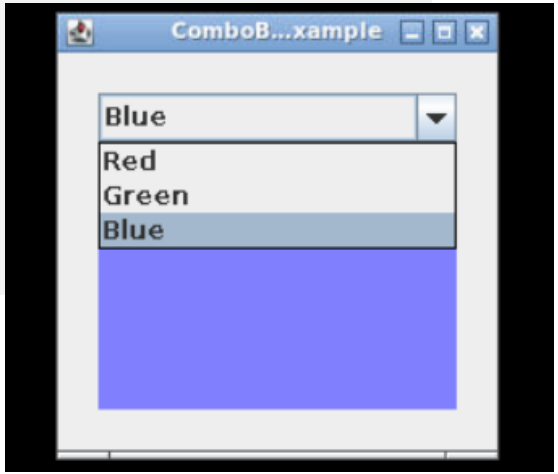

Reference:
https://docs.oracle.com/javase/tutorial/uiswing/components/combobox.html

# Combo box example: view initialisation

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class ComboBoxExample extends JPanel
                             implements ActionListener {
    JLabel picture;
    public ComboBoxExample() {
        super(new BorderLayout());
        String[] colorStrings = { "Red", "Green", "Blue" };
        JComboBox<String> colorList = new JComboBox<>(colorStrings);
        colorList.setSelectedIndex(0);
        colorList.addActionListener(this);
        picture = new JLabel();
        picture.setOpaque(true);
        picture.setBackground(new Color(255, 128, 128));
        picture.setBorder(BorderFactory.createEmptyBorder(10,0,0,0));
        picture.setPreferredSize(new Dimension(177, 122+10));
        add(colorList, BorderLayout.PAGE_START);
        add(picture, BorderLayout.PAGE_END);
        setBorder(BorderFactory.createEmptyBorder(20,20,20,20));
    }
```

# Combo box example: event listener

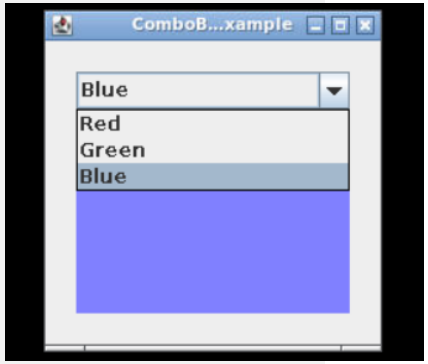
```
22
23    public void actionPerformed(ActionEvent e) {
24      JComboBox cb = (JComboBox)e.getSource();
25      String colorName = (String)cb.getSelectedItem();
26      if(colorName.equals("Red"))
27        picture.setBackground(new Color(255, 128, 128));
28      else if(colorName.equals("Green"))
29        picture.setBackground(new Color(128, 255, 128));
30      else if(colorName.equals("Blue"))
31        picture.setBackground(new Color(128, 128, 255));
32    }
```




- Listener for the combo box events
- Changes color of the label

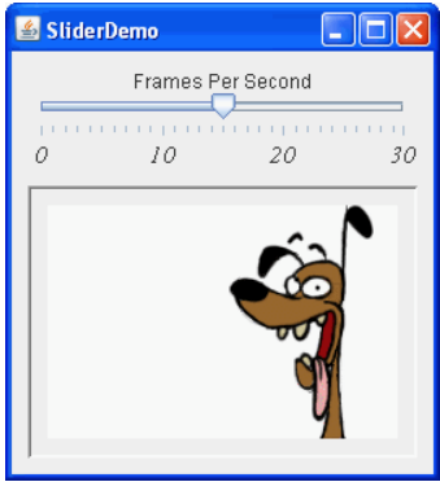
# Combo box example: entry point

```
33
34      private static void createAndShowGUI() {
35          JFrame frame = new JFrame("ComboBoxExample");
36          frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
37          JComponent newContentPane = new ComboBoxExample();
38          newContentPane.setOpaque(true);
39          frame.setContentPane(newContentPane);
40          frame.pack();
41          frame.setVisible(true);
42      }
43
44      public static void main(String[] args) {
45          javax.swing.SwingUtilities.invokeLater(new Runnable() {
46              public void run() {
47                  createAndShowGUI();
48              }
49          });
50      }
51  }
```



# Sliders: JSlider

- Sliders can be used to give numerical input bounded with minimum and maximum value
  - In Swing, sliders created using **`JSlider`** component
  - `JSlider` events are **`ChangeEvent`** objects listened by **`ChangeListener`**



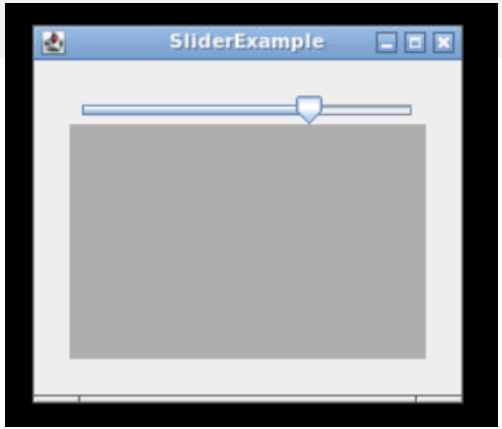

Reference:
https://docs.oracle.com/javase/tutorial/uiswing/components/slider.html

# Slider example: view initialisation

```
1   import java.awt.*;
2   import javax.swing.event.*;
3   import javax.swing.*;
4   public class SliderExample extends JPanel
5                                 implements ChangeListener {
6       JLabel picture;
7       public SliderExample() {
8           super(new BorderLayout());
9           JSlider brightness = new JSlider(JSlider.HORIZONTAL,0,255,128);
10          brightness.addChangeListener(this);
11          picture = new JLabel();
12          picture.setOpaque(true);
13          picture.setBackground(new Color(128, 128, 128));
14          picture.setBorder(BorderFactory.createEmptyBorder(10,0,0,0));
15          picture.setPreferredSize(new Dimension(177, 122+10));
16          add(brightness, BorderLayout.PAGE_START);
17          add(picture, BorderLayout.PAGE_END);
18          setBorder(BorderFactory.createEmptyBorder(20,20,20,20));
19      }
```

# Slider example: event listener

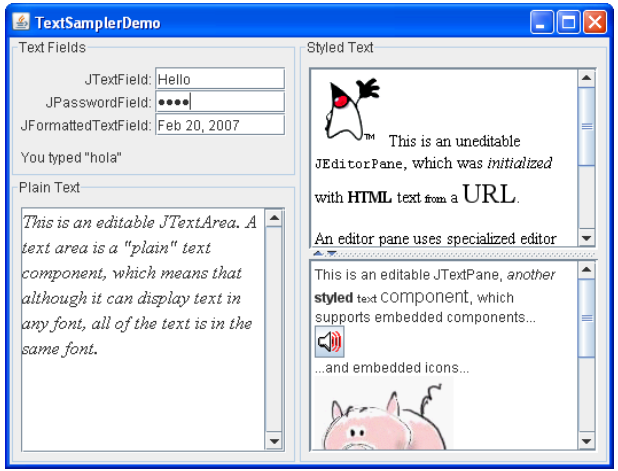
```
20
21      public void stateChanged(ChangeEvent e) {
22          JSlider slider = (JSlider)e.getSource();
23          int value = (int)slider.getValue();
24          picture.setBackground(new Color(value,value,value));
25      }
...     ...
```




- Listener for the slider events (ChangeEvent)
- Changes brightness of the label

# Text fields

- Text components, such as `JTextArea` and `JTextPane`, can be editable
  - Method **`getText()`** can be used to get the edited text when needed
  - You can add an `actionListener` to a text component, but it is not usually necessary




References:
https://docs.oracle.com/javase/tutorial/uiswing/components/textfield.html
https://docs.oracle.com/javase/tutorial/uiswing/components/editorpane.html

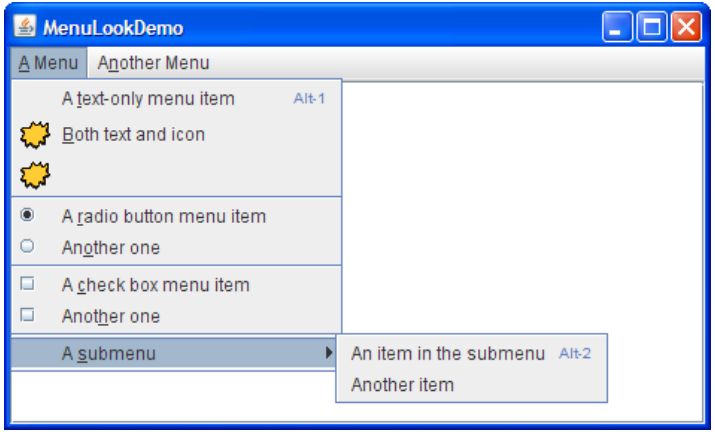
# Text field example with interaction

```
...  ...
4    public class TextFieldExample extends JPanel
5                                 implements ActionListener {
6        JTextField textField;
7        JButton submitButton;
8        JLabel output;
9        public TextFieldExample() {
10           super(new BorderLayout(10,10));
11           textField = new JTextField(30);
12           submitButton = new JButton("Submit text");
13           submitButton.addActionListener(this);
14           output = new JLabel("  ");
15           add(textField, BorderLayout.PAGE_START);
16           add(submitButton, BorderLayout.CENTER);
17           add(output, BorderLayout.PAGE_END);
18           setBorder(BorderFactory.createEmptyBorder(20,20,20,20));
19       }
20       public void actionPerformed(ActionEvent e) {
21           output.setText(textField.getText());
22       }
...      ...
```


Outputs the text written in text field

# Adding functionality to menu items

- Yesterday, we demonstrated how to add items to the menu bar
  - It is also possible to use icons, radio buttons and check boxes in menus
- We can use `ActionListener` to handle events from menu items, just like from buttons




Reference:
https://docs.oracle.com/javase/tutorial/uiswing/components/menu.html

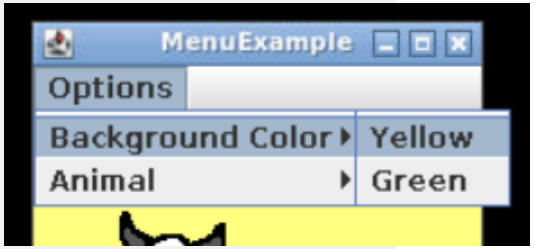
# Menu example: view initialiser

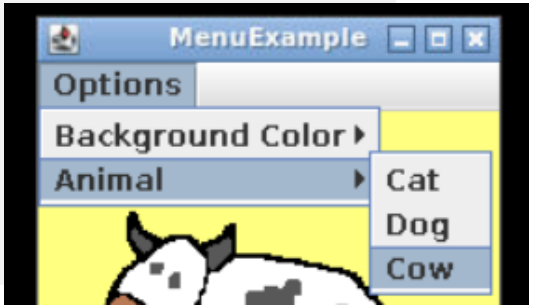
```
1   import javax.swing.*;
2   import java.awt.*;
3   import java.awt.event.*;
4   class MenuExample2 implements ActionListener {
5       JMenu menu, submenu1, submenu2;
6       JMenuItem i1, i2, i3, i4, i5;
7       JLabel label;
8       ImageIcon catIcon, dogIcon, cowIcon;
9       MenuExample2() {
10          JFrame f= new JFrame("MenuExample");
11          f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
12          JMenuBar mb=new JMenuBar();
13          menu=new JMenu("Options");
14          submenu1=new JMenu("Background Color");
15          submenu2=new JMenu("Animal");
```

Initialize menu and submenus

# Menu example: create menu items

```
16          i1=new JMenuItem("Yellow");
17          i2=new JMenuItem("Green");
18          i3=new JMenuItem("Cat");
19          i4=new JMenuItem("Dog");
20          i5=new JMenuItem("Cow");
21          submenu1.add(i1);
22          submenu1.add(i2);
23          submenu2.add(i3);
24          submenu2.add(i4);
25          submenu2.add(i5);
26          menu.add(submenu1);
27          menu.add(submenu2);
28          mb.add(menu);
29          f.setJMenuBar(mb);
30
```



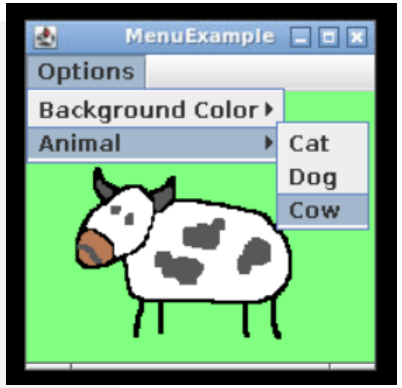
# Menu example: finalise view

```
31          label = new JLabel();
32          catIcon = new ImageIcon("cat.png");
33          dogIcon = new ImageIcon("dog.png");
34          cowIcon = new ImageIcon("cow.png");
35          label.setOpaque(true);
36          label.setBackground(new Color(255,255,255));
37          label.setPreferredSize(new Dimension(200, 200));
38          label.setIcon(catIcon);
39          f.getContentPane().add(label, BorderLayout.CENTER);
40
41          i1.addActionListener(this);
42          i2.addActionListener(this);
43          i3.addActionListener(this);
44          i4.addActionListener(this);
45          i5.addActionListener(this);
46
47          f.setSize(200,200);
48          f.setVisible(true);
49      }
```

Assign action listener to the menu items

# Menu example: action listener

```
50    public void actionPerformed(ActionEvent e) {
51      if((JMenuItem)e.getSource()==i1)
52        label.setBackground(new Color(255, 255, 128));
53      else if((JMenuItem)e.getSource()==i2)
54        label.setBackground(new Color(128, 255, 128));
55      else if((JMenuItem)e.getSource()==i3)
56        label.setIcon(catIcon);
57      else if((JMenuItem)e.getSource()==i4)
58        label.setIcon(dogIcon);
59      else if((JMenuItem)e.getSource()==i5)
60        label.setIcon(cowIcon);
61    }
62    public static void main(String args[]){
63      new MenuExample2();
64    }
65  }
```




Submenu 1: change background colour

Submenu 2: change animal icon

# Further reading

- There is an enormous number of GUI components, layouts etc. defined in Swing, and we have only covered some of the most important ones
- For more examples, see the official Java documentation:
  - Components: https://docs.oracle.com/javase/tutorial/uiswing/components/index
  - Event listeners: https://docs.oracle.com/javase/tutorial/uiswing/events/index.html

# Summary

- In Java, interactive GUI components are implemented using event listeners implementing interface `EventListener` or its subinterface
  - Event listeners can be assigned to GUI components to react to user actions, such as clicking a button or selecting a menu item
- The proper way to implement event listeners depends on the component
  - For example, buttons and menu items use `ActionListener` interface, sliders use `ChangeListener` interface
  - Refer to the documentation for more details about specific components

# Questions, comments?