1495 UNIVERSITY OF ABERDEEN

# JC2002 Java Programming

## Lecture 23: 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)

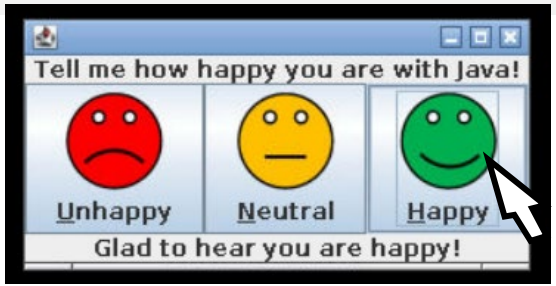
```
1  import javax.swing.*;
2  import javax.swing.border.*;
3  import java.awt.BorderLayout;
4  import java.awt.event.*;
5  public class ButtonExample2 {
6      protected static JButton b1, b2, b3;
7      protected static JLabel questionLabel, responseLabel;
8      static class ActionHandler implements ActionListener {
9          public void actionPerformed(ActionEvent e) {
10             if(e.getActionCommand().equals("happy")) {
11                 responseLabel.setText("Glad to hear you are happy!");
12             } else if(e.getActionCommand().equals("neutral")) {
13                 responseLabel.setText("Thank you for your feedback!");
14             } else if(e.getActionCommand().equals("unhappy")) {
15                 responseLabel.setText("Sorry to hear you are not happy.");
16             }
17         }
18     }
```

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);
...   ...
```



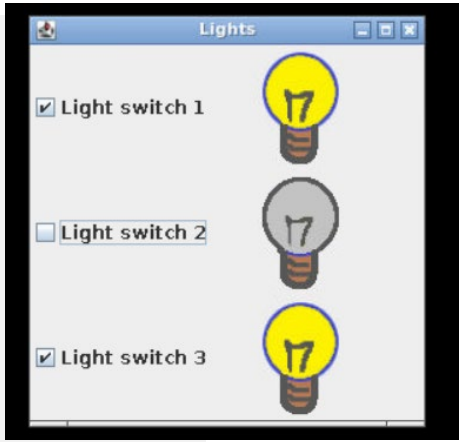

Set action commands and assign the action listener object to the buttons

# 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

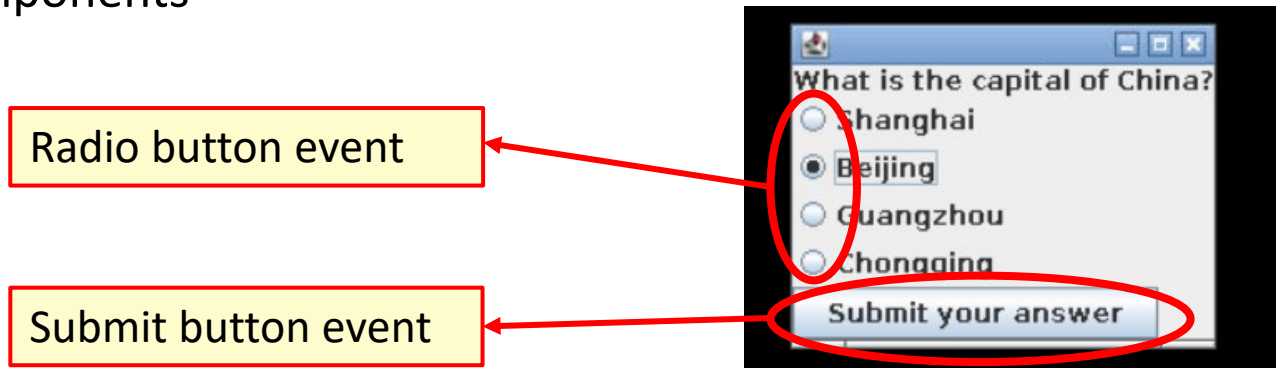
```
...  ...
3    import java.awt.event.*;
4    public class CheckBoxExample2  {
...  ...
22       ItemListener itemHandler = new ItemListener() {
23         public void itemStateChanged(ItemEvent e) {
24           for(int i=0; i<3; i++) {
25             if(e.getSource()==cb[i]) {
26               if(e.getStateChange() == 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 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)

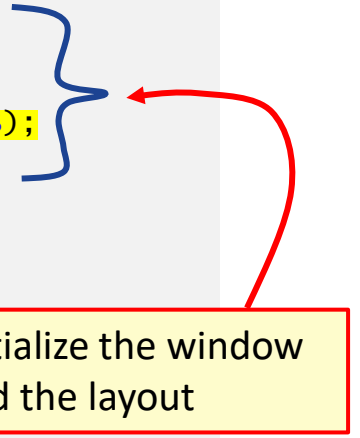
```
18   protected static class SubmitListener implements ActionListener {
19     public void actionPerformed(ActionEvent e) {
20       if(rb[1].isSelected()) {
21         JOptionPane.showMessageDialog(frame,"Your answer is correct!");
22       } else {
23         JOptionPane.showMessageDialog(frame,"Your 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");
```



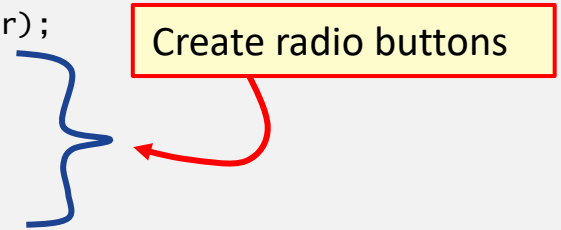
# Radio button example with events (4)

```
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");
```



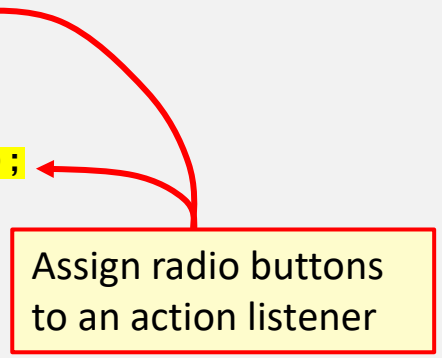
# Radio button example with events (6)

```
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  }
```



# 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?