CS 102

Object Oriented Programming

GUI Event Handling

Reyyan Yeniterzi

reyyan.yeniterzi@ozyegin.edu.tr

Sources:
http://www3.ntu.edu.sg/home/ehchua/programming/java/j4a_gui.html

The Hour of Code is here @OzU between Dec 5-6, 2016.

You learned how to code, now it is your chance to teach your friends. We need your help. To become a volunteer please fill out this form and join us.

Event Handling

- Java uses Event Driven Model for event handling.
- An event is generated when user interacts with GUI
- Some example events:
  - Clicking a button
  - Presing the ENTER key
  - Typing a key
  - Closing a frame
  - Clicking a mouse button

Event Handling

- Once an event is generated, it is passed to other objects which handle the event.
- This is also known as the Event Delegation Model.

Event Handling

- Three types of objects are involved
  - Event Source
  - Event Listener(s)
  - Event Object

Event Handling

- Three types of objects are involved
  - Event Source
    - Usually components (like button or textfield) but they can be other kind of objects (like windows) too.
    - It registers to event listener(s)
    - It generates event objects
  - Event Listener(s)
  - Event Object

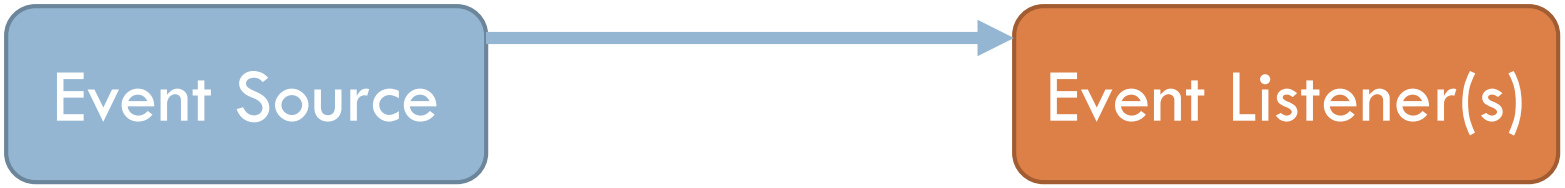
Event Handling

- Three types of objects are involved
  - Event Source
  - Event Listener(s)
    - Receives an event object from the event source and performs the implemented approppriate action
  - Event Object

Event Handling

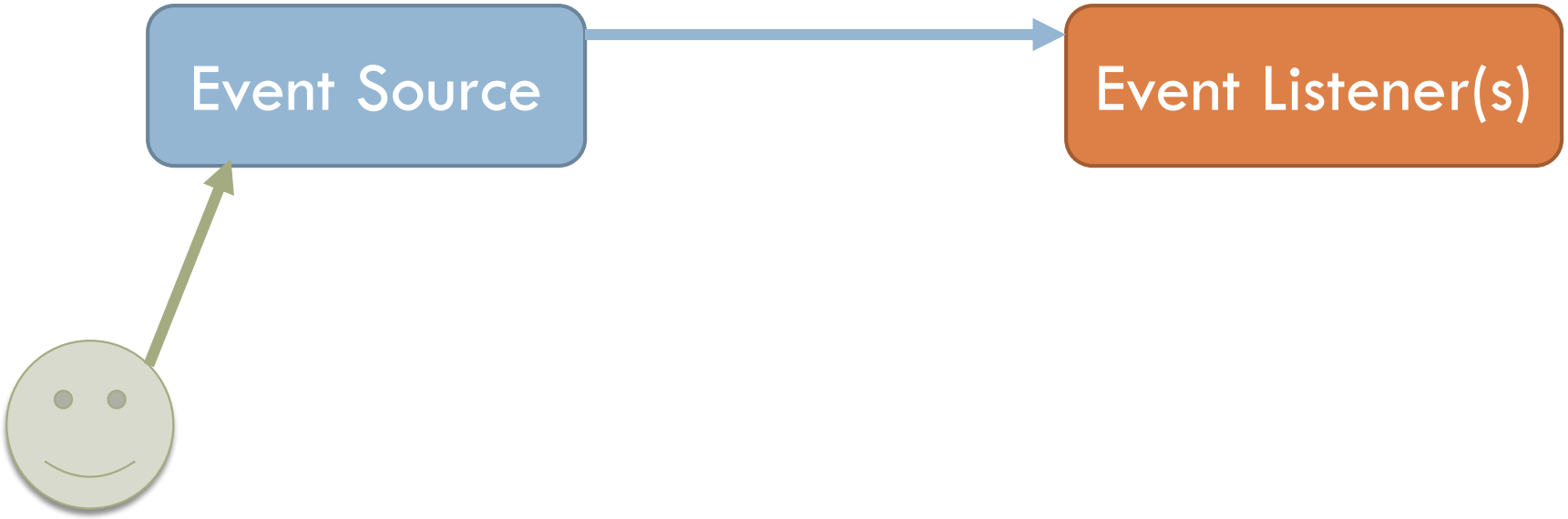
- Three types of objects are involved
  - Event Source
  - Event Listener(s)
  - Event Object
    - Created by the event source and sent to the registered event listener(s)
    - It describes the event by encapsulating the necessary information (like (x,y) coordinates in a mouse event)

Event Handling Process



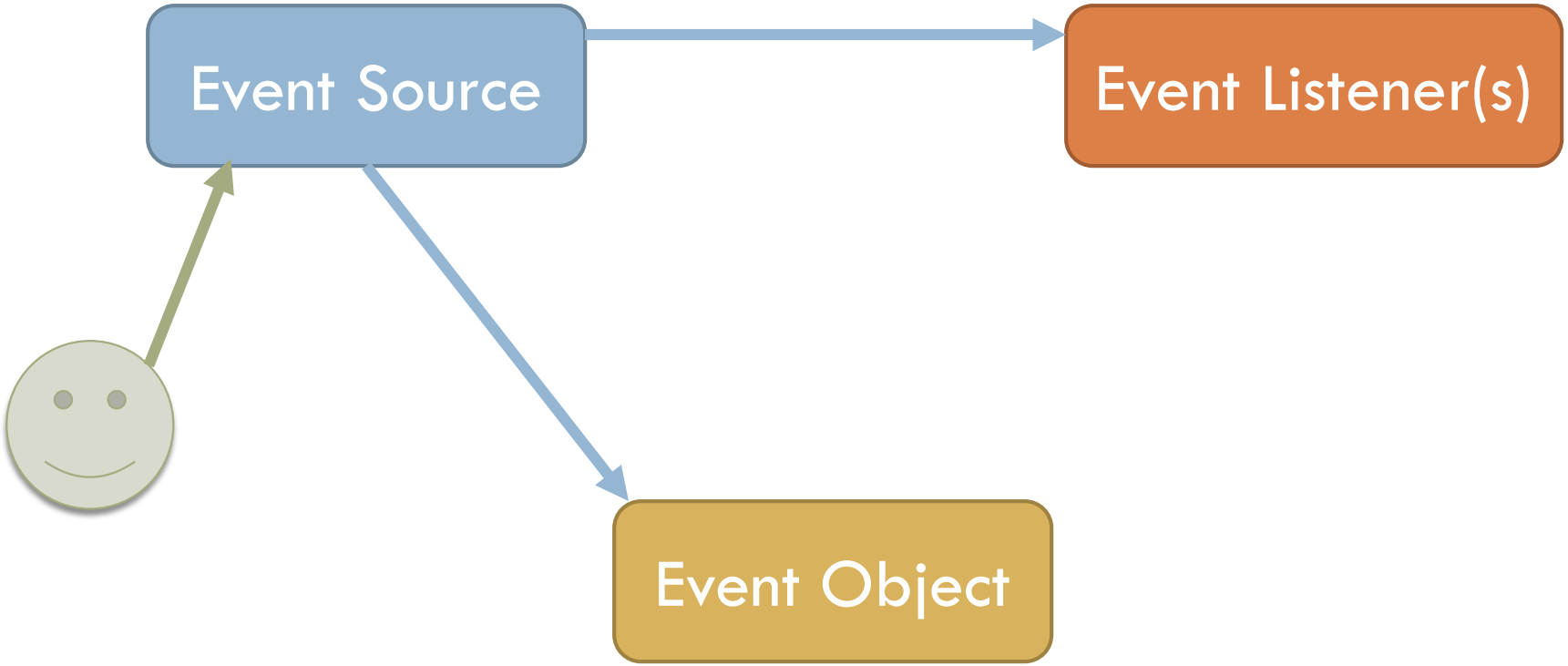
- An event source is registered with an event listener

Event Handling Process



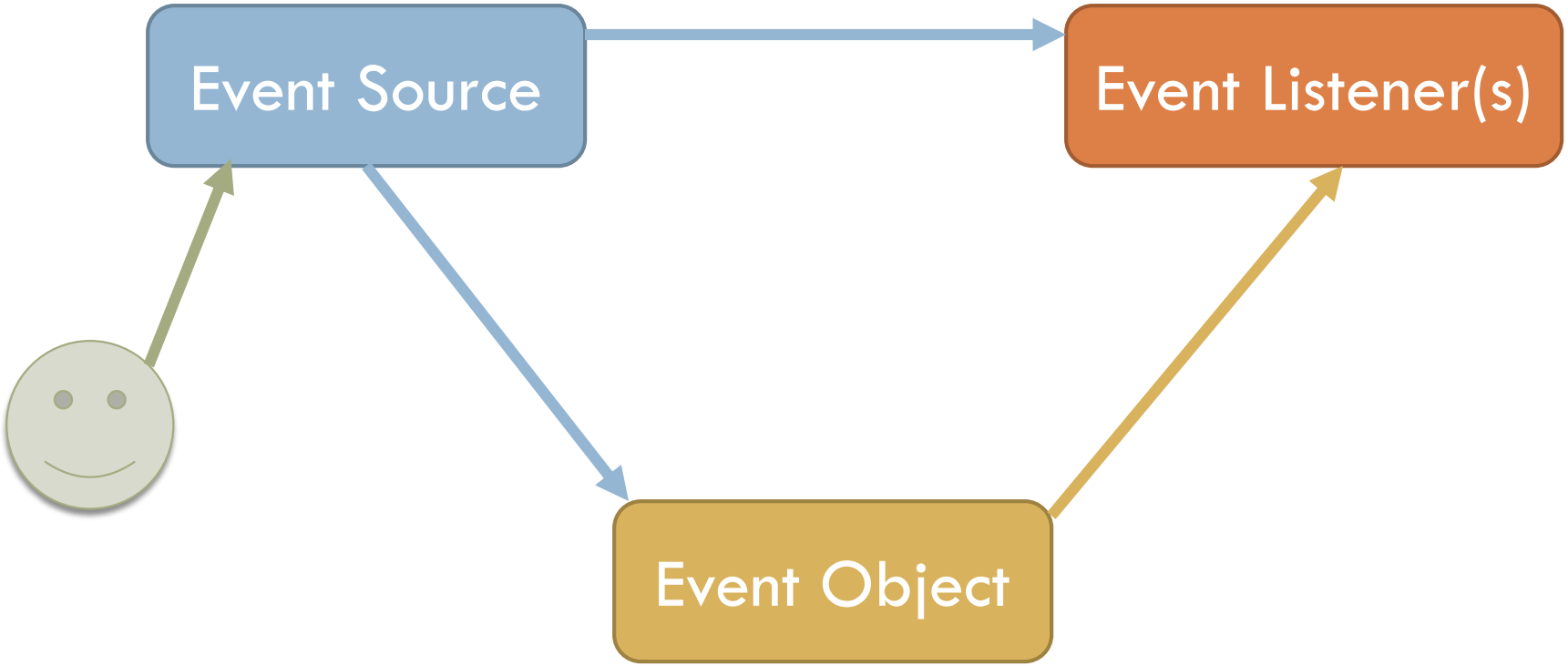
- User interacts with the event source. Triggers an event.

Event Handling Process



- An event object is created.

Event Handling Process



- Appropriate listener is invoked

Event Handling

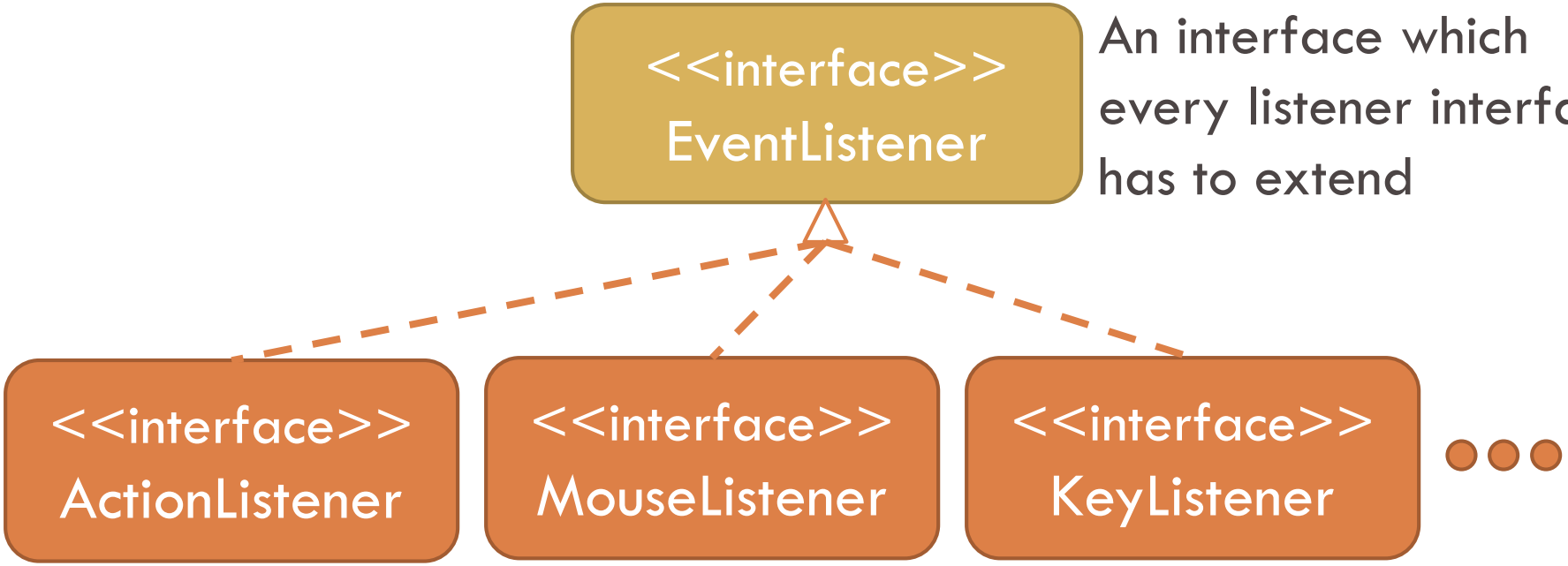
- Not procedural model.
  - Code is not executed in a sequential manner.
- Sequential Model vs. Event Model
  - Sequential model
  - Event Model (Event loop)
- All event handling code executes in a single thread
  - One event handler finishes before the next one can start

Implementing Event Handlers

- There are three steps:

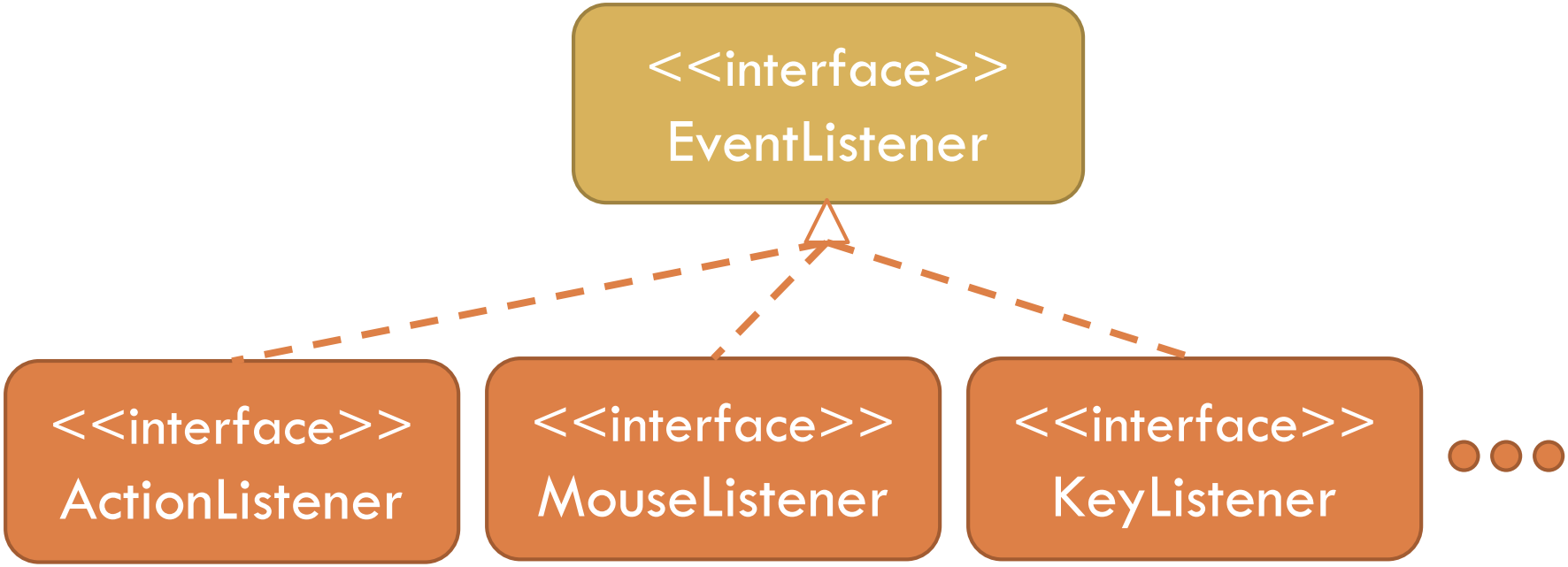
1. An event handler (event listener) class needs to be declared
2. An instance of the event listener needs to be registered to one or more components
3. The event handler class must implement the methods of the interface

Listener Interfaces



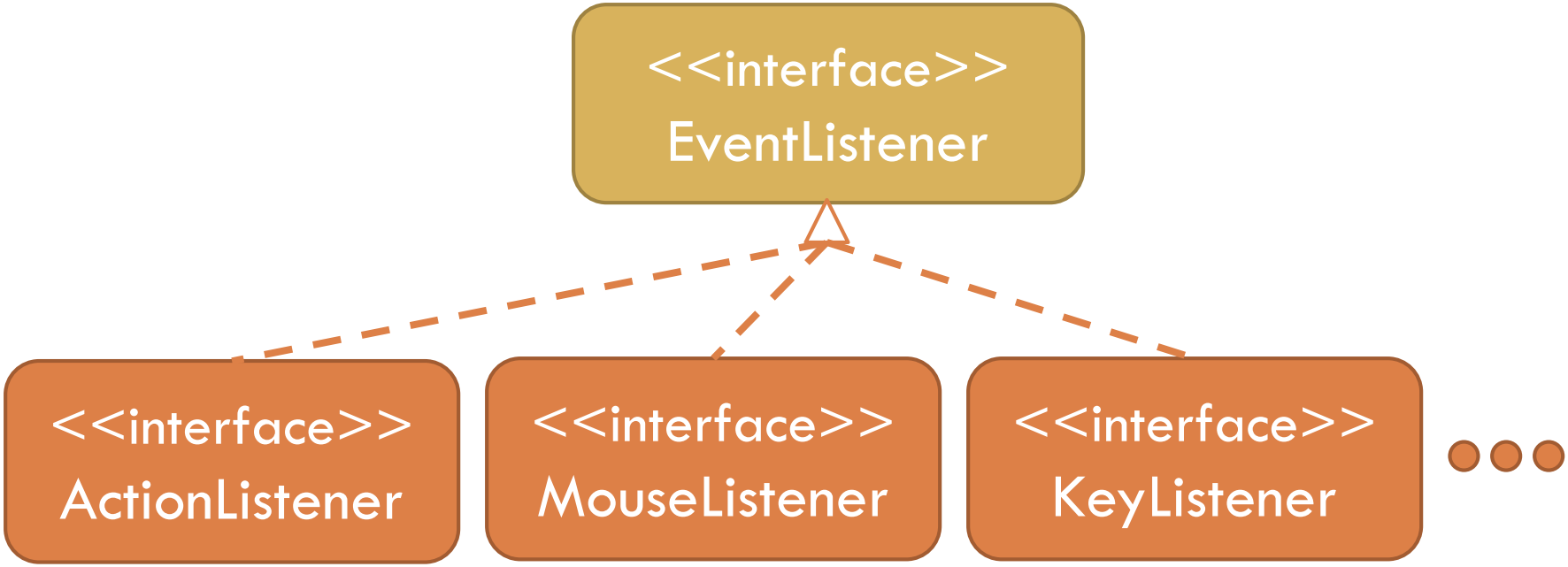
An interface which every listener interface has to extend

Listener Interfaces



- Clicking a button fires an Action Event
- Clicking a mouse fires Mouse Event
- Typing a key fires Key Event.

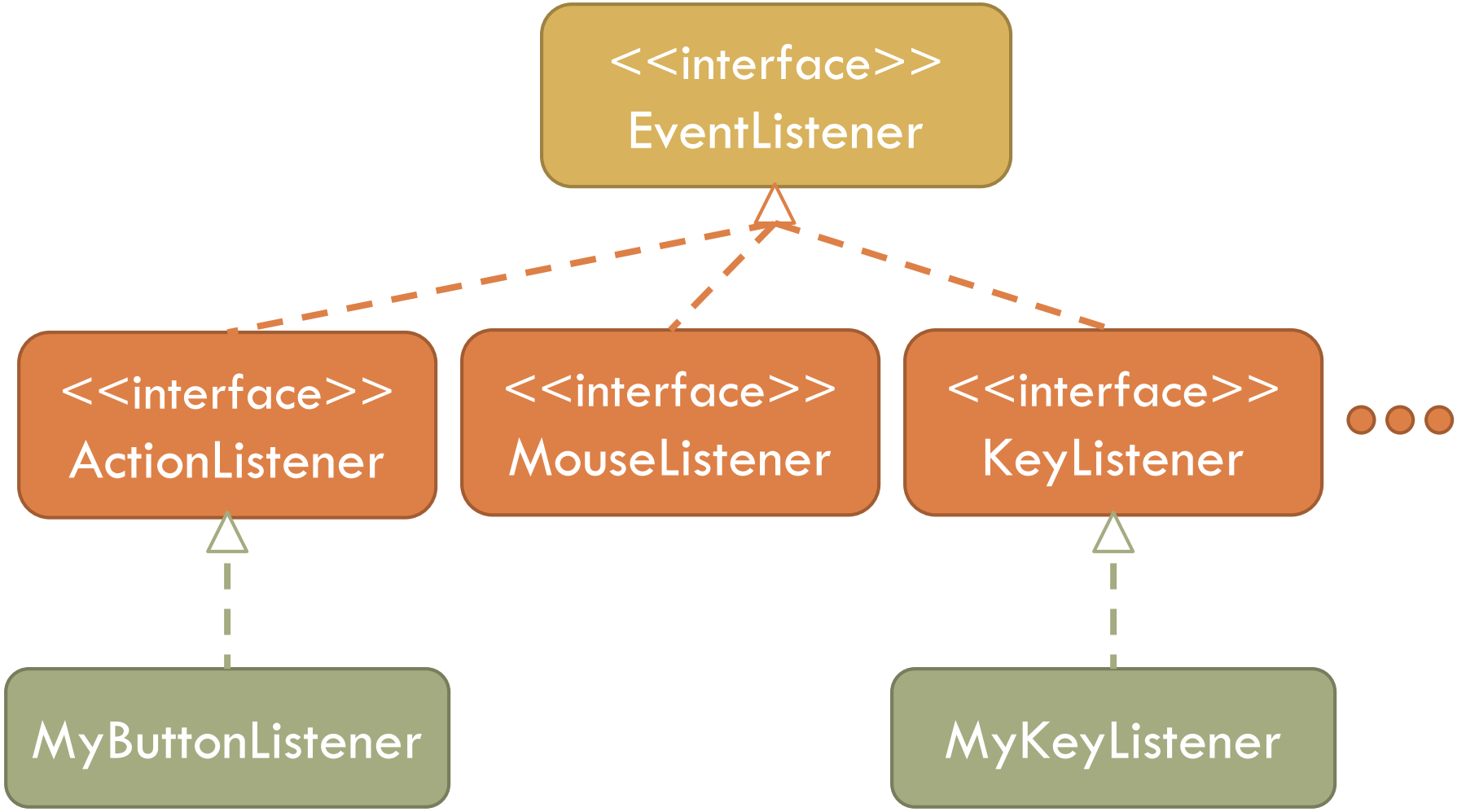
Listener Interfaces



These are the most commonly used ones.

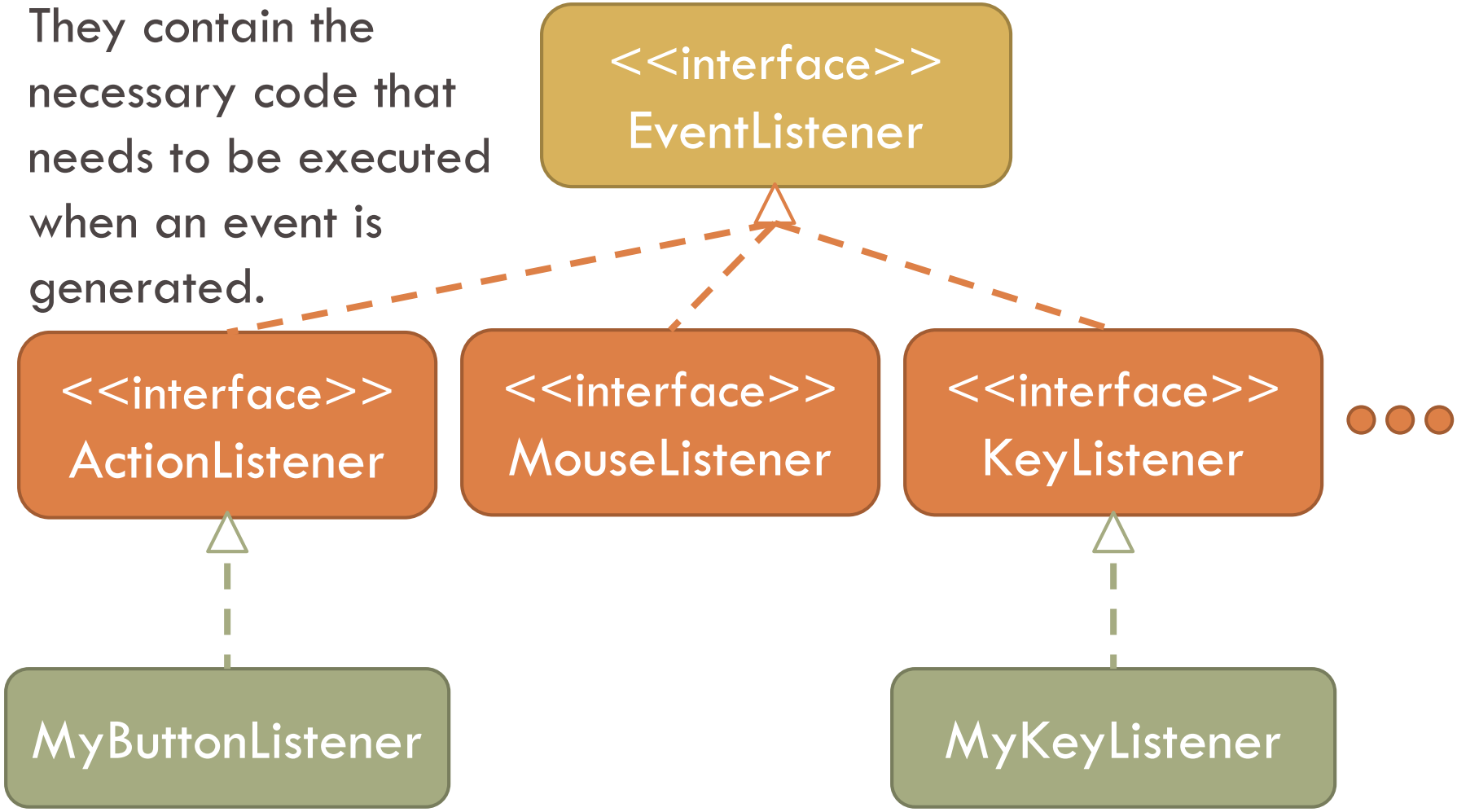
There are also some others.

Listener Interfaces

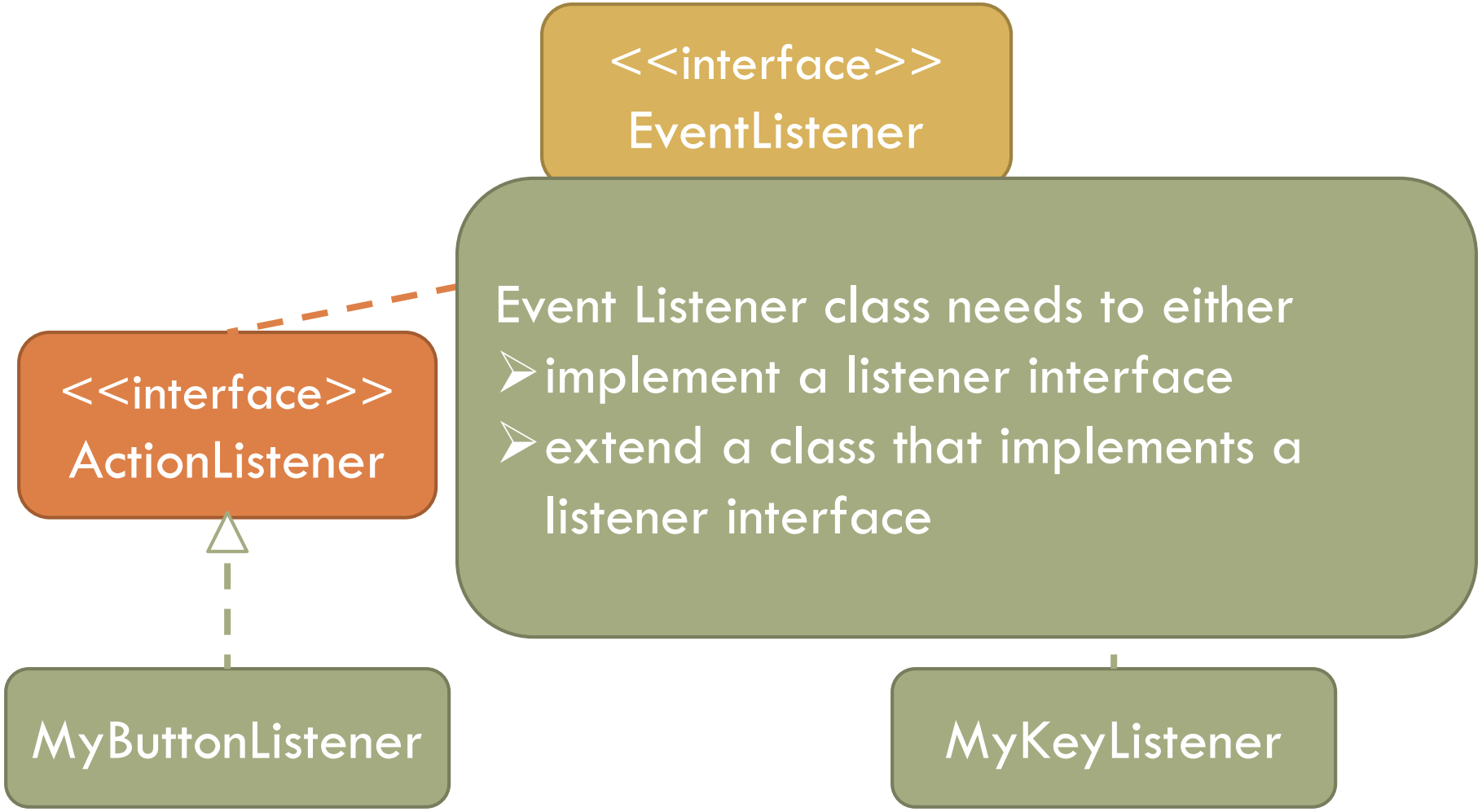


Listener Interfaces

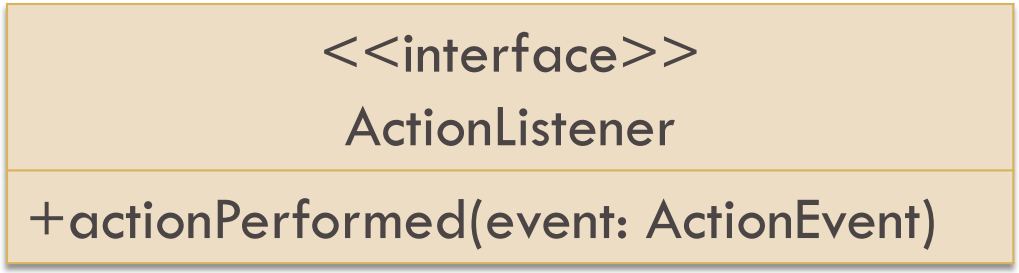
They contain the necessary code that needs to be executed when an event is generated.



Listener Interfaces



ActionListener Interface



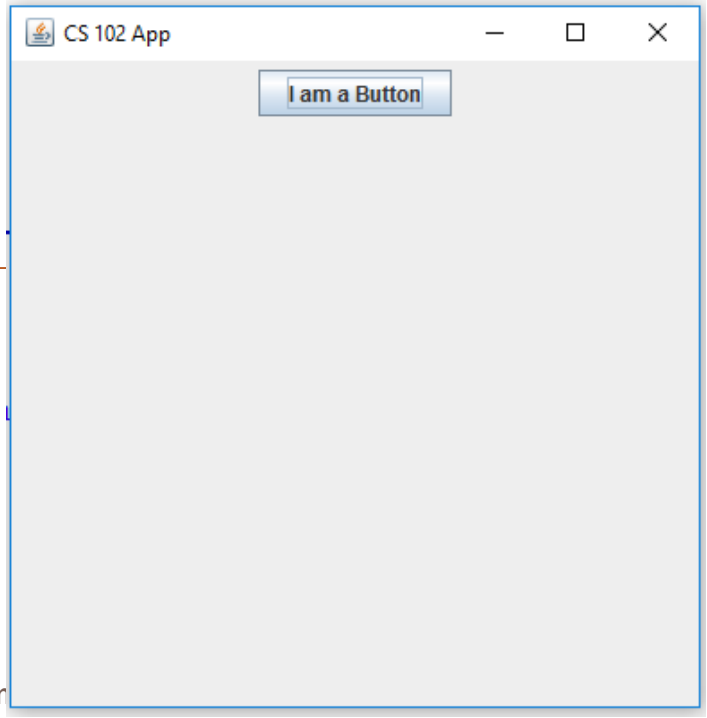
- ActionListener interface contains `actionPerformed` function.
- A listener implementing ActionListener needs to implement this method.
- EventSource calls this method and sends the event as a parameter.

Action Listener Examples

- EventExample1
- EventExample2
- EventExample3
- RadioButtonExample
- CheckBoxExample
- DynamicCheckBoxExample

EventExample1


```
public static void main(String[] args) {
    JFrame frame = new JFrame("CS 102 App");
    frame.setSize(400, 400);
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    JPanel mainPanel = new JPanel();
    frame.add(mainPanel);

    JButton button = new JButton("I am a Button");
    mainPanel.add(button);

    frame.setVisible(true);
}
```



- An example from last lecture.
- Lets extend it so that an event occurs when we click the button

EventExample1


```
class MyListener implements ActionListener {

    public void actionPerformed(ActionEvent e) {
        String msg = "Button is clicked!";
        System.out.println(msg);
    }
};
```

EventExample1


```java
public static void main(String[] args) {
    JFrame frame = new JFrame("CS 102 App");
    frame.setSize(400, 400);
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    JPanel mainPanel = new JPanel();
    frame.add(mainPanel);

    JButton button = new JButton("I am a Button");
    mainPanel.add(button);

    button.addActionListener(new MyListener());

    frame.setVisible(true);
}
```

```java
class MyListener implements ActionListener {

    public void actionPerformed(ActionEvent e) {
        String msg = "Button is clicked!";
        System.out.println(msg);
    }
};
```

Implementing Event Handlers

- There are three steps:

1. An event handler (event listener) class needs to be declared
2. An instance of the event listener needs to be registered to one or more components
3. The event handler class must implement the methods of the interface

EventExample1


```
public static void main(String[] args) {
    JFrame frame = new JFrame("CS 102 App");
    frame.setSize(400, 400);
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    JPanel mainPanel = new JPanel();
    frame.add(mainPanel);

    JButton button = new JButton("I am a Button");
    mainPanel.add(button);

    button.addActionListener(new MyListener());

    frame.setVisible(true);
}
```

```
class MyListener implements ActionListener {

    public void actionPerformed(ActionEvent e) {
        String msg = "Button is clicked!";
        System.out.println(msg);
    }
};
```

EventExample1


```
public static void main(String[] args) {
    JFrame frame = new JFrame("CS 102 App");
    frame.setSize(400, 400);
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    JPanel mainPanel = new JPanel();
    frame.add(mainPanel);

    JButton button = new JButton("I am a Button");
    mainPanel.add(button);

    button.addActionListener(new MyListener());

    frame.setVisible(true);
}
```

```
class MyListener implements ActionListener {

    public void actionPerformed(ActionEvent e) {
        String msg = "Button is clicked!";
        System.out.println(msg);
    }
};
```

EventExample1


```
public static void main(String[] args) {
    JFrame frame = new JFrame("CS 102 App");
    frame.setSize(400, 400);
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    JPanel mainPanel = new JPanel();
    frame.add(mainPanel);

    JButton button = new JButton("I am a Button");
    mainPanel.add(button);

    button.addActionListener(new MyListener());

    frame.setVisible(true);
```

```
stener {

onEvent e) {
ked!";

};
```

`addActionListener(ActionListener e)` functionality of JButton to register for Action Listeners

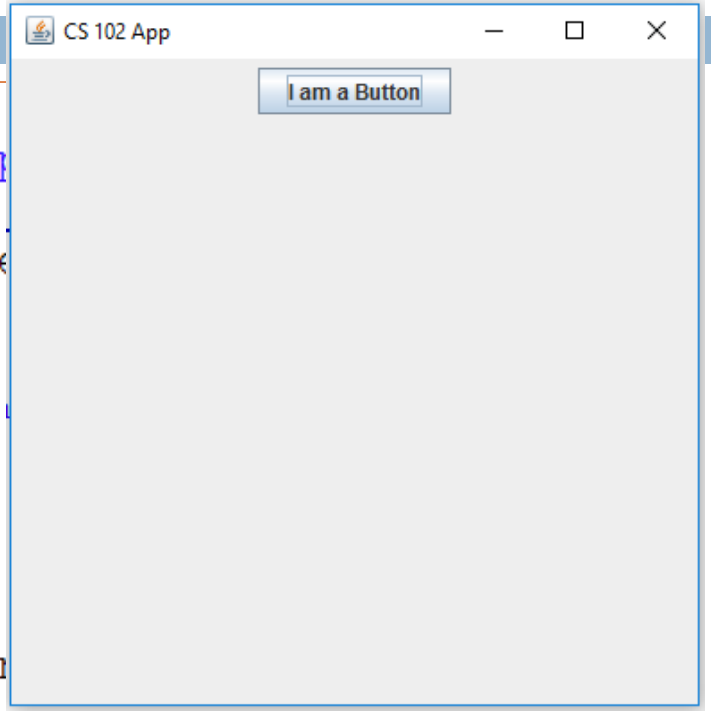
EventExample1


```
public static void main(String[] args) {
    JFrame frame = new JFrame("CS 102 App");
    frame.setSize(400, 400);
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    JPanel mainPanel = new JPanel();
    frame.add(mainPanel);

    JButton button = new JButton("I am a Button");
    mainPanel.add(button);

    button.addActionListener(new MyListener());

    frame.setVisible(true);
}
```

```
class MyListener implements ActionListener {

    public void actionPerformed(ActionEvent e) {
        String msg = "Button is clicked!";
        System.out.println(msg);
    }
};
```

Example (Before button click)


```
public static void main(String[] args) {
    JFrame frame = new JFrame("CS 102 App
    frame.setSize(400, 400);
    frame.setDefaultCloseOperation(JFrame

    JPanel mainPanel = new JPanel();
    frame.add(mainPanel);

    JButton button = new JButton("I am a
    mainPanel.add(button);

    button.addActionListener(new MyLister

    frame.setVisible(true);
}
```




```
class MyListener implements ActionListener {

    public void actionPerformed(ActionEvent e) {
        String msg = "Button is clicked!";
        System.out.println(msg);
    }
};
```

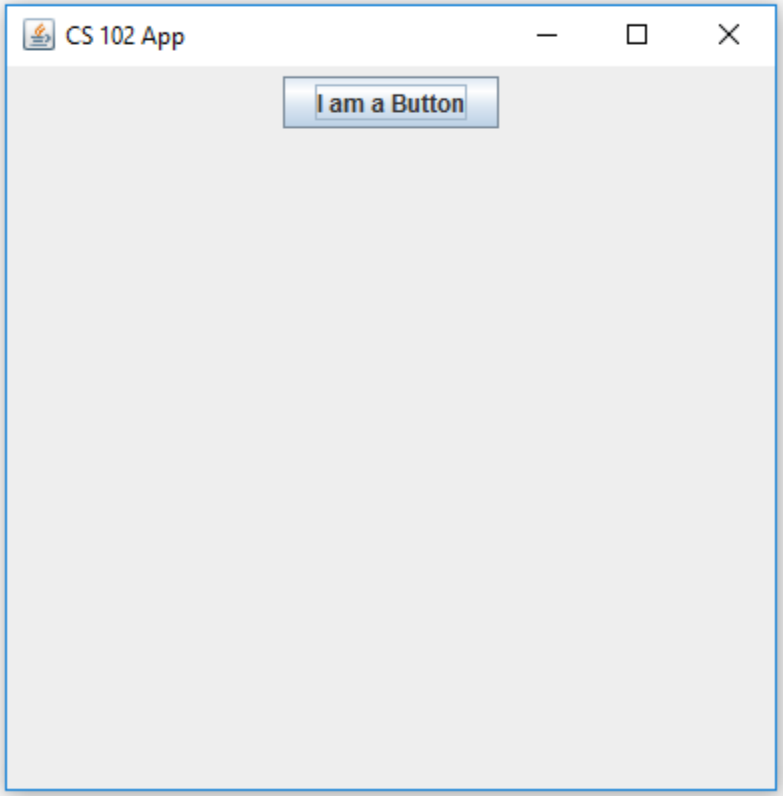
Example (After button click)


```
public static void main(String[]
    JFrame frame = new JFrame("C
    frame.setSize(400, 400);
    frame.setDefaultCloseOperati

    JPanel mainPanel = new JPane
    frame.add(mainPanel);

    JButton button = new JButtor
    mainPanel.add(button);

    button.addActionListener(new

    frame.setVisible(true);
}
```

```
class MyLis

    public
        Str
        System.out.println(msg);
    }
};
```

Javadoc | Declaration | Console

```
eventExample [Java Application] C:\Program Files\Java\jre1.8.0_73\bin\javaw.exe (10 M
Button is clicked!
```



JOptionPane

- A simple dialog box for graphical input or output

```
class MyListener implements ActionListener {

    public void actionPerformed(ActionEvent e) {
        String msg = "Button is clicked!";
        JOptionPane.showMessageDialog(null, msg);
    }
};
```

- showMessageDialog displays a message similar to System.out.println
- `public static void showMessageDialog( Component parent, Object message)`

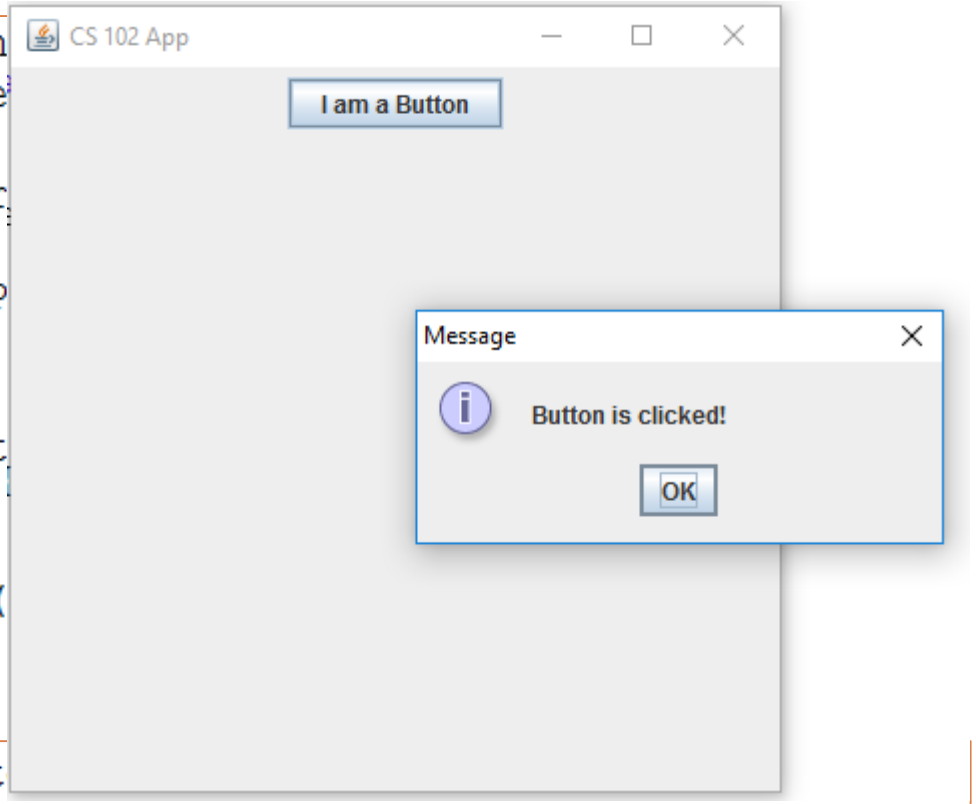
Example with JOptionPane


```
public static void main(String[] args) {
    JFrame frame = new JFrame("CS 102 App");
    frame.setSize(400, 400);
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    JPanel mainPanel = new JPanel();
    frame.add(mainPanel);

    JButton button = new JButton("I am a Button");
    mainPanel.add(button);

    button.addActionListener(new MyListener());

    frame.setVisible(true);
}
```

```
class MyListener implements ActionListener {

    public void actionPerformed(ActionEvent e) {
        String msg = "Button is clicked!";
        JOptionPane.showMessageDialog(null, msg);
    }
};
```

Example (After button click)


```
public static void main(Strin
    JFrame frame = new JFrame
    frame.setSize(400, 400);
    frame.setDefaultCloseOper

    JPanel mainPanel = new JP
    frame.add(mainPanel);

    JButton button = new JBut
    mainPanel.add(button);

    button.addActionListener(

    frame.setVisible(true);
}
```

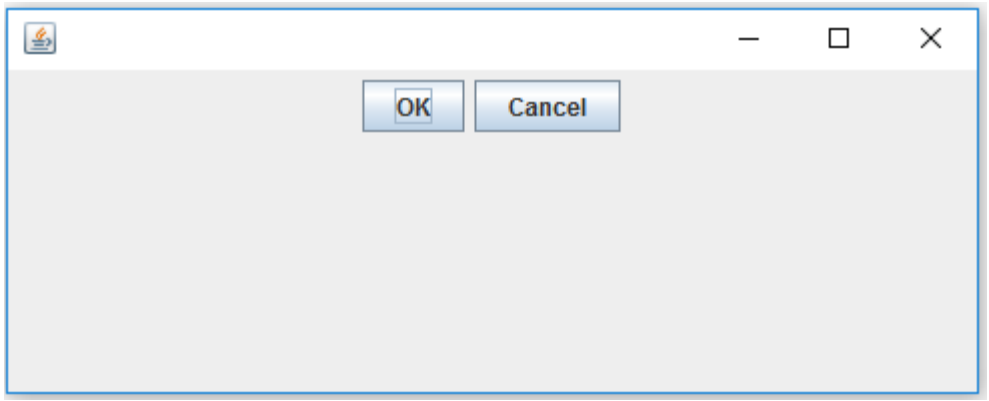



```
class MyList

    public void actionPerformed(ActionEvent e) {
        String msg = "Button is clicked!";
        JOptionPane.showMessageDialog(null, msg);
    }
};
```

Action Listener Examples

- EventExample1
- **EventExample2**
- EventExample3
- RadioButtonExample
- CheckBoxExample
- DynamicCheckBoxExample

EventExample2



- When  is clicked

```
Listener: 1 from OK
Listener: 2 from OK
```

- When  is clicked

```
Listener: 3 from Cancel
Listener: 2 from Cancel
```

Multiple Listeners

- There are 2 buttons and 3 listeners.
  - Button1 is registered to 2 listeners
    - Listener1
    - Listener2
  - Button2 is registered to 2 listeners
    - Listener2
    - Listener3
- Remember an event source can register to multiple event listeners.

Multiple Listener

- We have the same event listener class and three listeners use this one.
- We need to use an id to diffirentiate them.

```
class AnotherActionListener implements ActionListener {
    private int id;

    public AnotherActionListener(int id) {
        this.id = id;
    }

    public void actionPerformed(ActionEvent e) {

    }
}
```

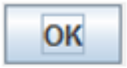
Multiple Listeners


```
public static void main(String[] args) {
    JFrame frame = new JFrame();
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setSize(500, 200);
    frame.setLayout(new FlowLayout());

    JButton button1 = new JButton("OK");
    JButton button2 = new JButton("Cancel");

    AnotherActionListener listener1 = new AnotherActionListener(1);
    AnotherActionListener listener2 = new AnotherActionListener(2);
    AnotherActionListener listener3 = new AnotherActionListener(3);

    button1.addActionListener(listener2);
    button1.addActionListener(listener1);
    button2.addActionListener(listener2);
    button2.addActionListener(listener3);

    frame.add(button1);
    frame.add(button2);
    frame.setVisible(true);
}
```

Multiple Listeners

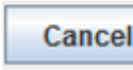
- `actionPerformed` function needs to be implemented

```
class AnotherActionListener implements ActionListener {
    private int id;

    public AnotherActionListener(int id) {
        this.id = id;
    }

    public void actionPerformed(ActionEvent e) {

    }
}
```

When 

is clicked

```
Listener: 1 from OK
Listener: 2 from OK
```

When 

is clicked

```
Listener: 3 from Cancel
Listener: 2 from Cancel
```

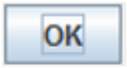
Multiple Listeners

- `actionPerformed` function needs to be implemented

```
class AnotherActionListener implements ActionListener {
    private int id;
```

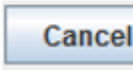
- We need the listener id and the text inside the button.
  - The listener id is a class instance
  - The text of the button can be retrieved from event object

```
}
```

When 

is clicked

```
Listener: 1 from OK
Listener: 2 from OK
```

When 

is clicked

```
Listener: 3 from Cancel
Listener: 2 from Cancel
```

Event Objects

- Every event is a subclass of EventObject
- EventObject is an abstract class
  - getSource() returns the event object
  - It returns the object reference, we need to cast it before using it.

EventObject
#source: Object
+getSource(): Object +toString(): String

- Subclasses can add their own methods

Multiple Listeners

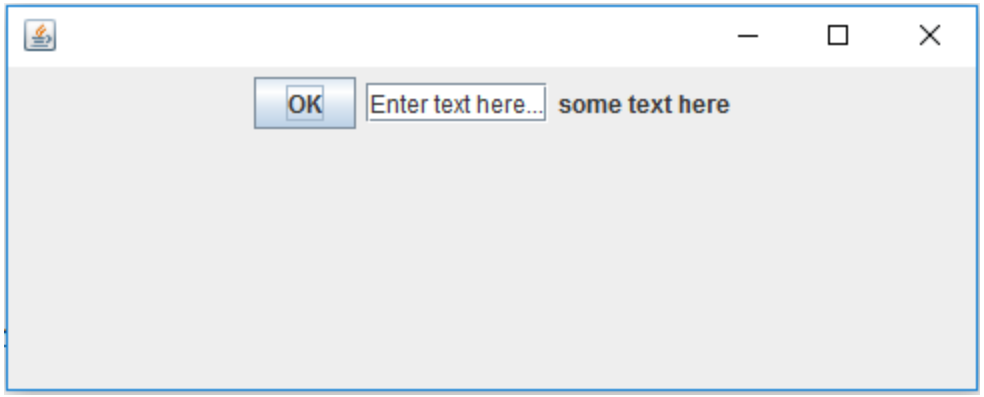

```
class AnotherActionListener implements ActionListener {
    private int id;

    public AnotherActionListener(int id) {
        this.id = id;
    }

    public void actionPerformed(ActionEvent e) {
        if (e.getSource() instanceof JButton) {
            JButton button = (JButton)e.getSource();

            System.out.println("Listener: " + id
                + " from " + button.getText());
        }
    }
}
```

Action Listener Examples

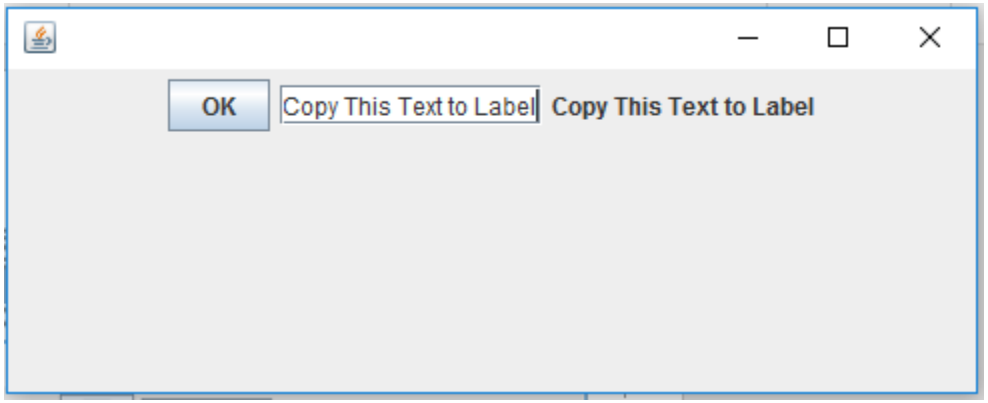
- EventExample1
- EventExample2
- **EventExample3**
- RadioButtonExample
- CheckBoxExample
- DynamicCheckBoxExample

EventExample3

- Initial state



- When clicked to OK button or pressed ENTER in textbox



EventExample3

- The event will copy the text inside textbox to the label.
- Both the textField and label need to be accessed from the event handler class.

EventExample3


```
public static void main(String[] args) {
    JFrame frame = new JFrame();
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setSize(500, 200);
    frame.setLayout(new FlowLayout());

    JButton button = new JButton("OK");
    JTextField textField = new JTextField("Enter text here...");
    JLabel label = new JLabel("some text here");

    MyButtonListener listener = new MyButtonListener(textField, label);
    button.addActionListener(listener);
    textField.addActionListener(listener);

    frame.add(button);
    frame.add(textField);
    frame.add(label);
    frame.setVisible(true);
}
```

EventExample3


```
public static void main(String[] args) {
    JFrame frame = new JFrame();
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setSize(500, 200);
    frame.setLayout(new FlowLayout());

    JButton button = new JButton("OK");
    JTextField textField = new JTextField("Enter text here...");
    JLabel label = new JLabel("some text here");

    MyButtonListener listener = new MyButtonListener(textField, label);
    button.addActionListener(listener);
    textField.addActionListener(listener);

    frame.add(button);
    frame.add(textField);
    frame.add(label);
    frame.setVisible(true);
}
```

- Why?

EventExample3


```
class MyButtonListener implements ActionListener {

    private JTextField textField;
    private JLabel label;

    public MyButtonListener(JTextField textField, JLabel label) {
        this.textField = textField;
        this.label = label;
    }

    public void actionPerformed(ActionEvent e) {
        label.setText(textField.getText());
    }
}
```

Action Listener Examples

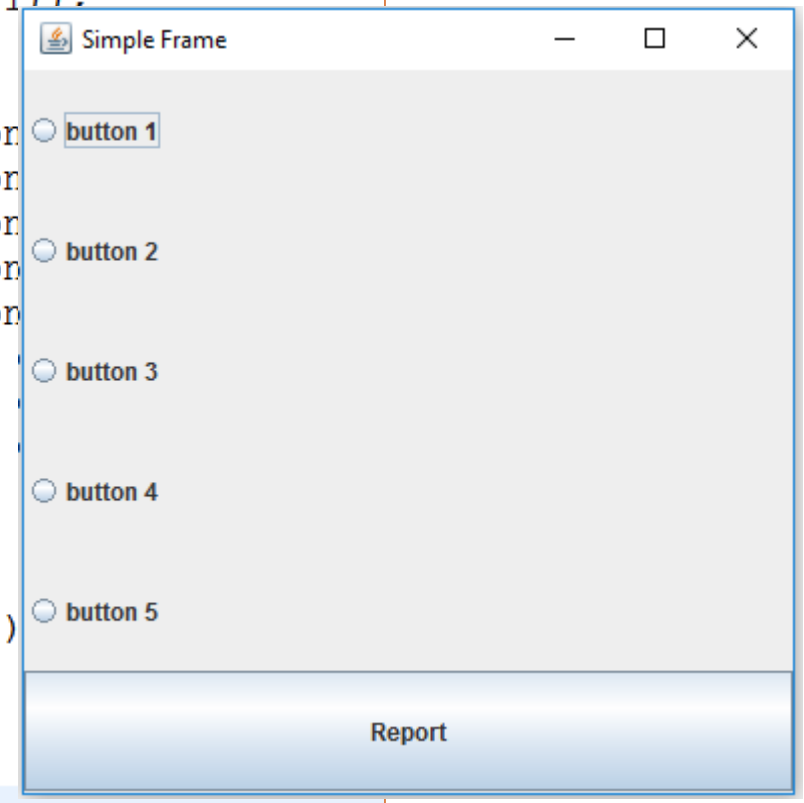
- EventExample1
- EventExample2
- EventExample3
- **RadioButtonExample**
- CheckBoxExample
- DynamicCheckBoxExample

RadioButton Example


```
public static void main(String[] args) {
    JFrame frame = new JFrame("Simple Frame");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setSize(400,400);
    frame.setLayout(new BorderLayout());

    JPanel mainPanel = new JPanel();
    mainPanel.setLayout(new GridLayout(6,1));
    frame.add(mainPanel);

    JRadioButton radio1 = new JRadioButton("button 1");
    JRadioButton radio2 = new JRadioButton("button 2");
    JRadioButton radio3 = new JRadioButton("button 3");
    JRadioButton radio4 = new JRadioButton("button 4");
    JRadioButton radio5 = new JRadioButton("button 5");
    mainPanel.add(radio1);
    mainPanel.add(radio2);
    mainPanel.add(radio3);
    mainPanel.add(radio4);
    mainPanel.add(radio5);

    JButton button = new JButton("Report");
    mainPanel.add(button);

    frame.setVisible(true);
}
```

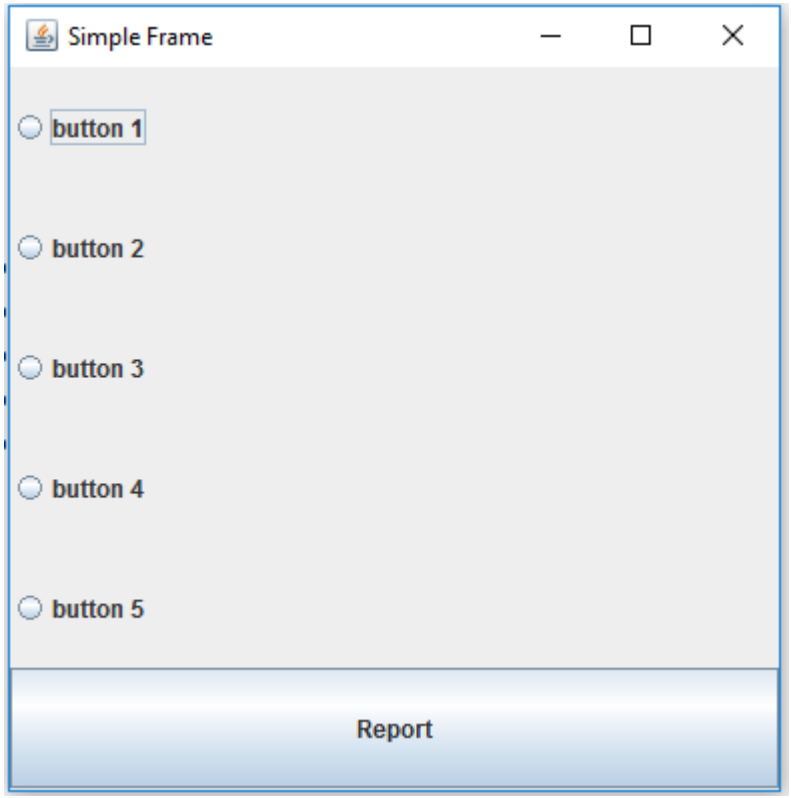
RadioButton Example

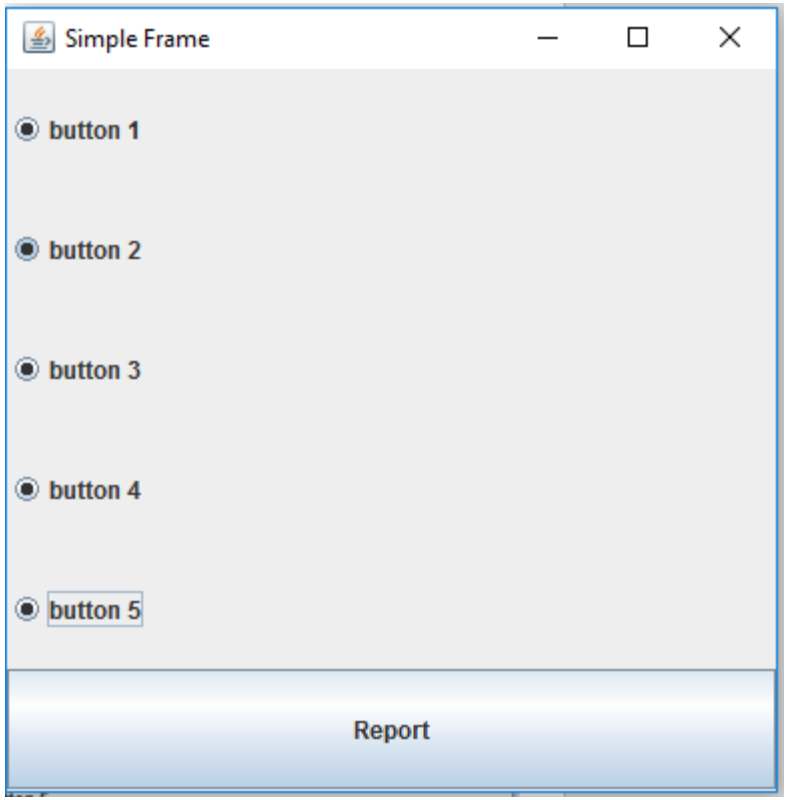

```
public static void main(String[] args) {
    JFrame frame = new JFrame("Simple Frame");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setSize(400,400);
    frame.setLayout(new BorderLayout());

    JPanel mainPanel = new JPanel();
    mainPanel.setLayout(new GridLayout(6,1));
    frame.add(mainPanel);

    JRadioButton radio1 = new JRadioButton
    JRadioButton radio2 = new JRadioButton
    JRadioButton radio3 = new JRadioButton
    JRadioButton radio4 = new JRadioButton
    JRadioButton radio5 = new JRadioButton
    mainPanel.add(radio1);
    mainPanel.add(radio2);
    mainPanel.add(radio3);
    mainPanel.add(radio4);
    mainPanel.add(radio5);

    JButton button = new JButton("Report")
    mainPanel.add(button);

    frame.setVisible(true);
}
```



RadioButtons

- User can select all radiobuttons. How can restrict this?





ButtonGroup

- `ButtonGroup` component makes sure that only button (radio button as well) can be selected at a time.

```
JRadioButton radio1 = new JRadioButton("button 1");
ButtonGroup group1 = new ButtonGroup();
group1.add(radio1);
```

RadioButton Example


```
JRadioButton radio1 = new JRadioButton("button 1");
JRadioButton radio2 = new JRadioButton("button 2");
JRadioButton radio3 = new JRadioButton("button 3");
JRadioButton radio4 = new JRadioButton("button 4");
JRadioButton radio5 = new JRadioButton("button 5");

ButtonGroup group1 = new ButtonGroup();
group1.add(radio1);
group1.add(radio2);
group1.add(radio3);
ButtonGroup group2 = new ButtonGroup();
group2.add(radio4);
group2.add(radio5);

mainPanel.add(radio1);
mainPanel.add(radio2);
mainPanel.add(radio3);
mainPanel.add(radio4);
mainPanel.add(radio5);
```

RadioButton Example

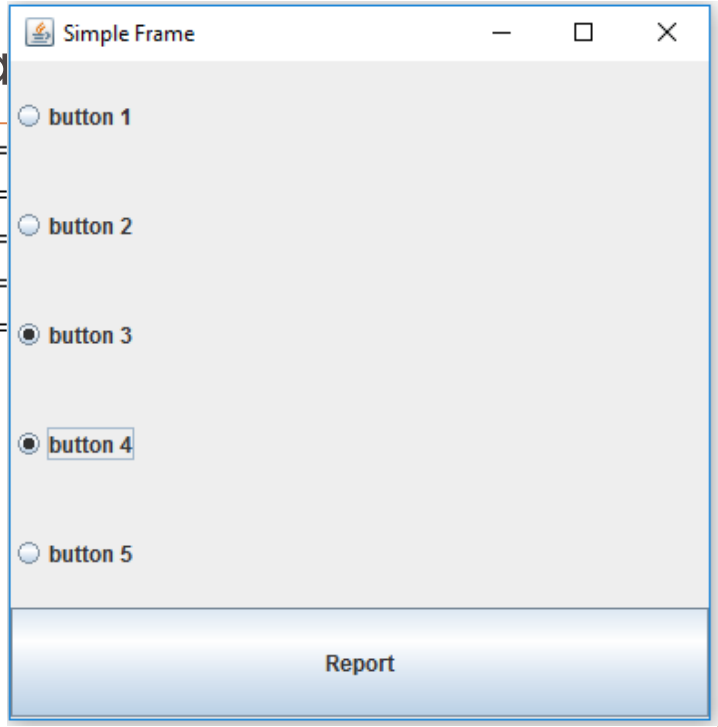
- How many combinations of RadioButtons can a user select?

```
JRadioButton radio1 = new JRadioButton("button 1");
JRadioButton radio2 = new JRadioButton("button 2");
JRadioButton radio3 = new JRadioButton("button 3");
JRadioButton radio4 = new JRadioButton("button 4");
JRadioButton radio5 = new JRadioButton("button 5");

ButtonGroup group1 = new ButtonGroup();
group1.add(radio1);
group1.add(radio2);
group1.add(radio3);
ButtonGroup group2 = new ButtonGroup();
group2.add(radio4);
group2.add(radio5);

mainPanel.add(radio1);
mainPanel.add(radio2);
mainPanel.add(radio3);
mainPanel.add(radio4);
mainPanel.add(radio5);
```

RadioButton Example

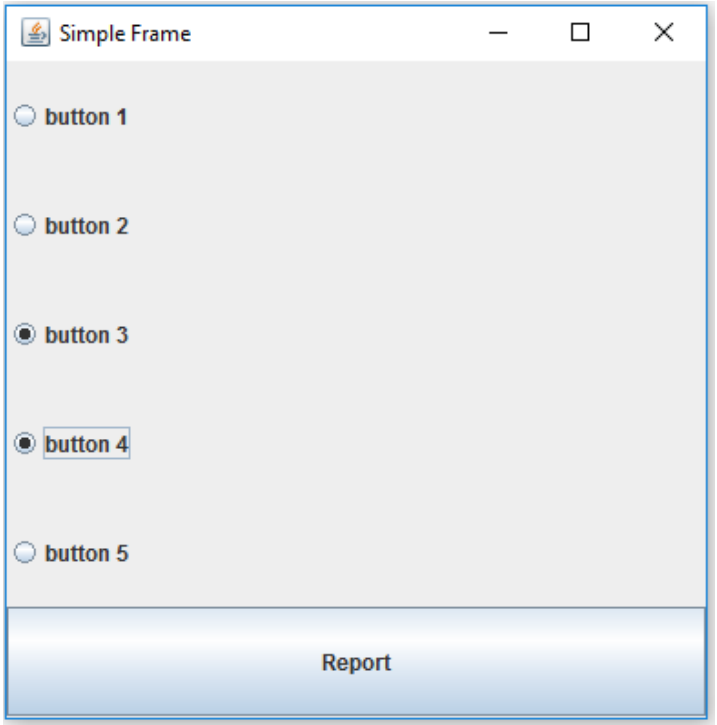
- How many combinations of Ra
  select?

```
JRadioButton radio1 =
JRadioButton radio2 =
JRadioButton radio3 =
JRadioButton radio4 =
JRadioButton radio5 =

ButtonGroup group1 =
group1.add(radio1);
group1.add(radio2);
group1.add(radio3);
ButtonGroup group2 =
group2.add(radio4);
group2.add(radio5);

mainPanel.add(radio1);
mainPanel.add(radio2);
mainPanel.add(radio3);
mainPanel.add(radio4);
mainPanel.add(radio5);
```



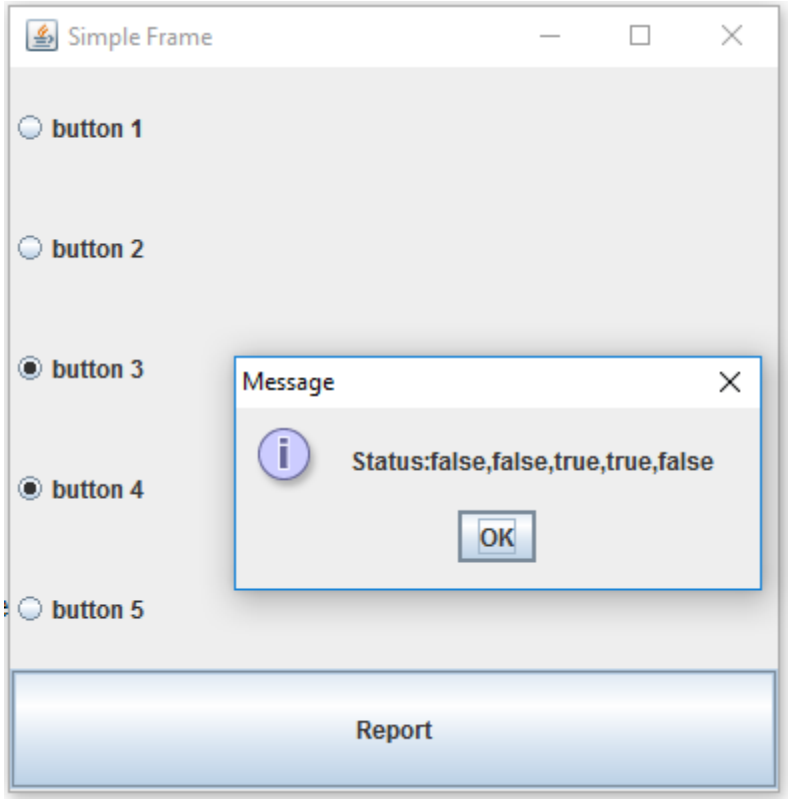
RadioButton Example

- At this point report button does not do anything.
- Lets implement an event handler for the button.



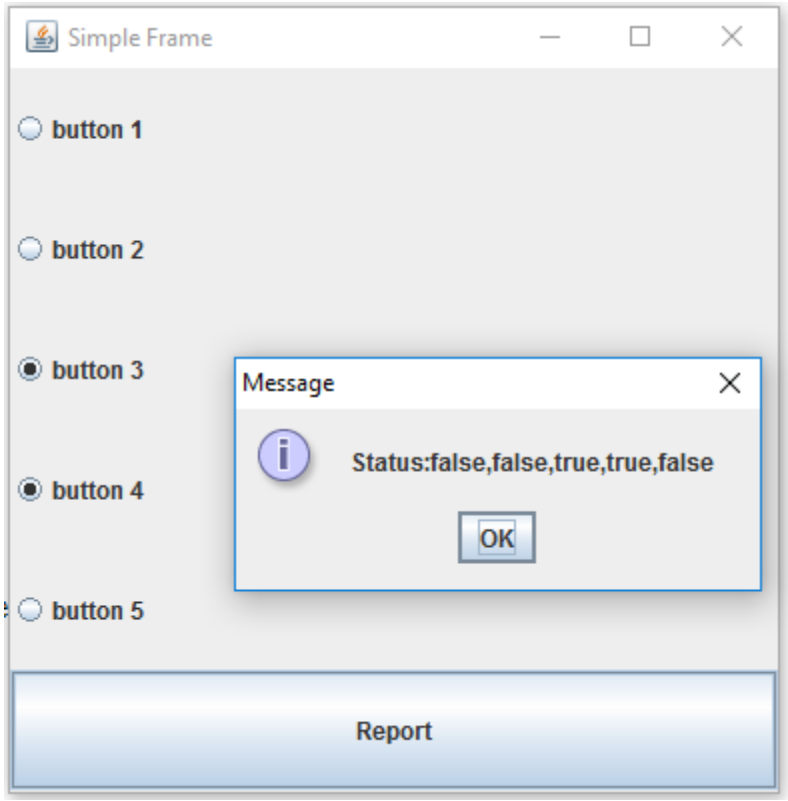
RadioButton Example

- When the report button is clicked display the selected and unselected radiobuttons.



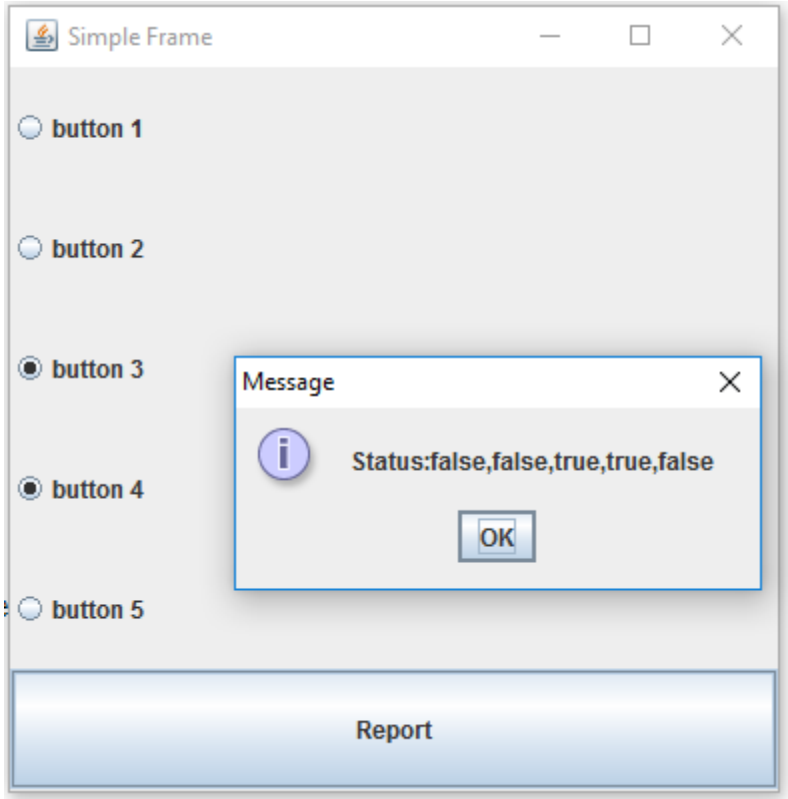
RadioButton Example

- When the report button is clicked display the selected and unselected radiobuttons.
- In our ActionListener class we need to know which radiobuttons are selected.



RadioButton Example

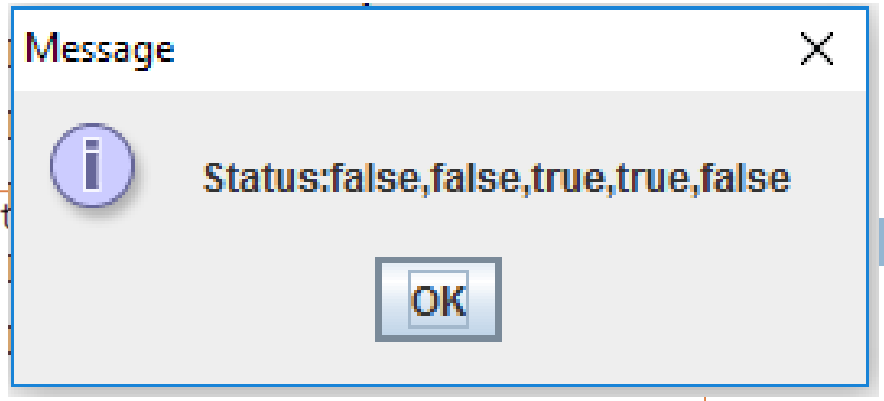
- When the report button is clicked display the selected and unselected radiobuttons.
- In our ActionListener class we need to know which radiobuttons are selected.
- Therefore, we need to send them to listener class.



RadioButton Example


```
class ButtonListener implements ActionListener {
    private JRadioButton radio1;
    private JRadioButton radio2;
    private JRadioButton radio3;
    private JRadioButton radio4;
    private JRadioButton radio5;

    public ButtonListener(JRadioButton r1, JRadioButton r2,
            JRadioButton r3, JRadioButton r4, JRadioButton r5) {
        radio1 = r1;
        radio2 = r2;
        radio3 = r3;
        radio4 = r4;
        radio5 = r5;
    }

    public void actionPerformed(ActionEvent e) {
        String msg = "Status:";
        msg += radio1.isSelected();
        msg += "," + radio2.isSelected();
        msg += "," + radio3.isSelected();
        msg += "," + radio4.isSelected();
        msg += "," + radio5.isSelected();
        JOptionPane.showMessageDialog(null, msg);
    }
};
```

RadioButton Exam




```
class ButtonListener implements Act
    private JRadioButton radio1;
    private JRadioButton radio2;
    private JRadioButton radio3;
    private JRadioButton radio4;
    private JRadioButton radio5;

    public ButtonListener(JRadioButton r1, JRadioButton r2,
            JRadioButton r3, JRadioButton r4, JRadioButton r5) {
        radio1 = r1;
        radio2 = r2;
        radio3 = r3;
        radio4 = r4;
        radio5 = r5;
    }

    public void actionPerformed(ActionEvent e) {
        String msg = "Status:";
        msg += radio1.isSelected();
        msg += "," + radio2.isSelected();
        msg += "," + radio3.isSelected();
        msg += "," + radio4.isSelected();
        msg += "," + radio5.isSelected();
        JOptionPane.showMessageDialog(null, msg);
    }
};
```

RadioButton Example

- Do not forget to register the listener to the report button.

```
JButton button = new JButton("Report");
mainPanel.add(button);

button.addActionListener(new ButtonListener(radio1, radio2, radio3, radio4, radio5));

frame.setVisible(true);
```

Action Listener Examples

- EventExample1
- EventExample2
- EventExample3
- RadioButtonExample
- **CheckBoxExample**
- DynamicCheckBoxExample

CheckBox Example

- We can use CheckBox instead of RadioButton as well.

CheckBox Example


```
public static void main(String[] args) {
    JFrame frame = new JFrame("Simple Frame");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setSize(400,400);

    JPanel mainPanel = new JPanel();
    mainPanel.setLayout(new GridLayout(6,1));
    frame.add(mainPanel);

    JCheckBox check1 = new JCheckBox("check 1");
    JCheckBox check2 = new JCheckBox("check 2");
    JCheckBox check3 = new JCheckBox("check 3");
    JCheckBox check4 = new JCheckBox("check 4");
    JCheckBox check5 = new JCheckBox("check 5");
    ButtonGroup group1 = new ButtonGroup();
    group1.add(check1);
    group1.add(check2);
    group1.add(check3);
    group1.add(check4);
    group1.add(check5);
    mainPanel.add(check1);
    mainPanel.add(check2);
    mainPanel.add(check3);
    mainPanel.add(check4);
    mainPanel.add(check5);
```

CheckBox Example

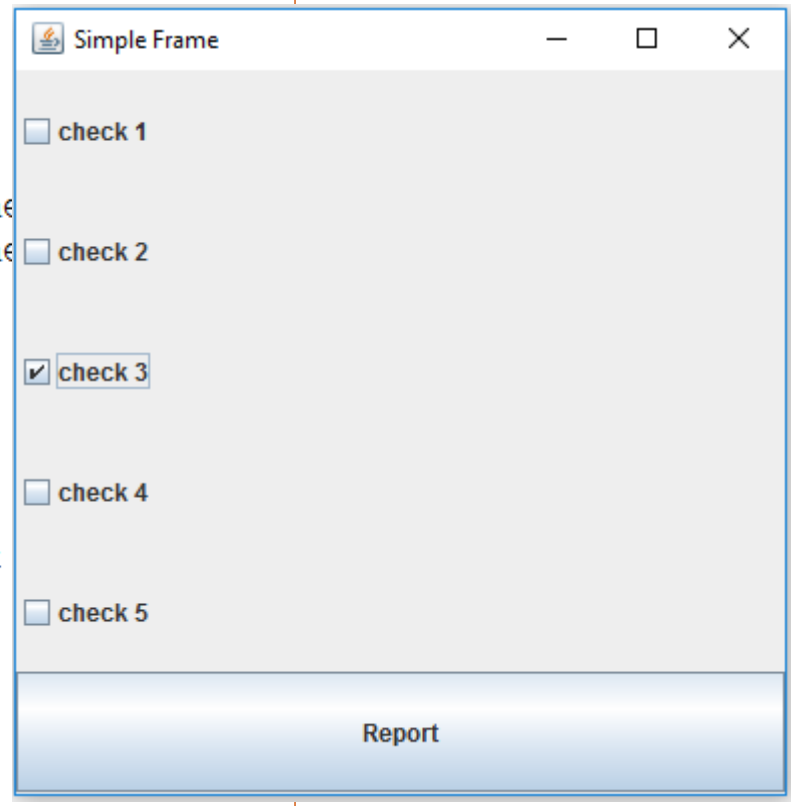

```
    JButton button = new JButton("Report");
    mainPanel.add(button);

    button.addActionListener(new ButtonListener(check1, check2, check3, check4, check5));

    frame.setVisible(true);
}
```

CheckBox Example


```
class ButtonListener implements ActionListener {
    private JCheckBox check1;
    private JCheckBox check2;
    private JCheckBox check3;
    private JCheckBox check4;
    private JCheckBox check5;

    public ButtonListener(JCheckBox r1, JCheckBox r2,
            JCheckBox r3, JCheckBox r4, JCheckBox r5) {
        check1 = r1;
        check2 = r2;
        check3 = r3;
        check4 = r4;
        check5 = r5;
    }
    public void actionPerformed(ActionEvent e) {
        String msg = "Status:";
        msg += check1.isSelected();
        msg += "," + check2.isSelected();
        msg += "," + check3.isSelected();
        msg += "," + check4.isSelected();
        msg += "," + check5.isSelected();

        JOptionPane.showMessageDialog(null, msg);
    }
};
```

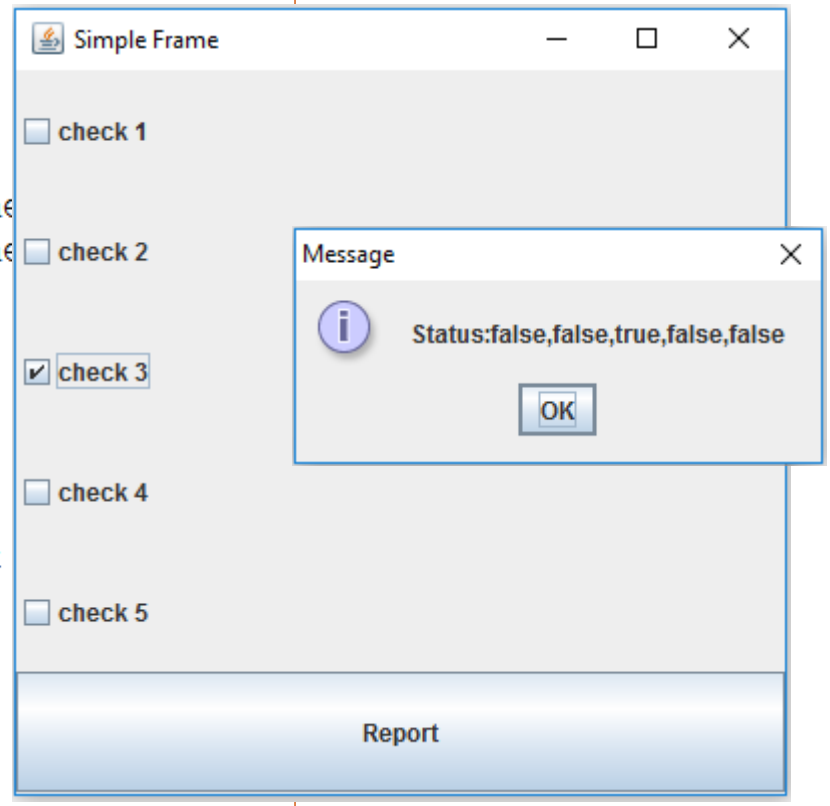
CheckBox Example


```
class ButtonListener implements ActionListener {
    private JCheckBox check1;
    private JCheckBox check2;
    private JCheckBox check3;
    private JCheckBox check4;
    private JCheckBox check5;

    public ButtonListener(JCheckBox r1, JChe
            JCheckBox r3, JCheckBox r4, JChe
        check1 = r1;
        check2 = r2;
        check3 = r3;
        check4 = r4;
        check5 = r5;
    }
    public void actionPerformed(ActionEvent
        String msg = "Status:";
        msg += check1.isSelected();
        msg += "," + check2.isSelected();
        msg += "," + check3.isSelected();
        msg += "," + check4.isSelected();
        msg += "," + check5.isSelected();

        JOptionPane.showMessageDialog(null, msg);
    }
};
```



CheckBox Example


```
class ButtonListener implements ActionListener {
    private JCheckBox check1;
    private JCheckBox check2;
    private JCheckBox check3;
    private JCheckBox check4;
    private JCheckBox check5;

    public ButtonListener(JCheckBox r1, JChe
            JCheckBox r3, JCheckBox r4, JChe
        check1 = r1;
        check2 = r2;
        check3 = r3;
        check4 = r4;
        check5 = r5;
    }
    public void actionPerformed(ActionEvent
        String msg = "Status:";
        msg += check1.isSelected();
        msg += "," + check2.isSelected();
        msg += "," + check3.isSelected();
        msg += "," + check4.isSelected();
        msg += "," + check5.isSelected();

        JOptionPane.showMessageDialog(null, msg);
    }
};
```



CheckBox Example

- If we have not 5 but more CheckBoxes.
- Are we going to have more of the followings?

```
JCheckBox check1 = new JCheckBox("check 1");
JCheckBox check2 = new JCheckBox("check 2");
JCheckBox check3 = new JCheckBox("check 3");
JCheckBox check4 = new JCheckBox("check 4");
JCheckBox check5 = new JCheckBox("check 5");

mainPanel.add(check1);
mainPanel.add(check2);
mainPanel.add(check3);
mainPanel.add(check4);
mainPanel.add(check5);
```

- Lets do the same thing dynamically...

Action Listener Examples

- EventExample1
- EventExample2
- EventExample3
- RadioButtonExample
- CheckBoxExample
- **DynamicCheckBoxExample**

DynamicCheckBox Example

week08/src/gui04/DynamicCheckBoxExample.java

```
public static void main
    JFrame frame = new JFrame("Simple Frame");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setSize(400,400);

    JPanel mainPanel = new JPanel();
    mainPanel.setLayout(new GridLayout(6, 1));
    frame.add(mainPanel);

    ArrayList<JCheckBox> checkboxes = new ArrayList<JCheckBox>();
    ButtonGroup group1 = new ButtonGroup();
    for(int i=0; i<5; i++)
    {
        JCheckBox checkbox = new JCheckBox("check" + (i+1));
        checkboxes.add(checkbox);
        mainPanel.add(checkbox);
        group1.add(checkbox);
    }

    JButton button = new JButton("Report");
    mainPanel.add(button);

    button.addActionListener(new ButtonListener(checkboxes));

    frame.setVisible(true);
}
```

DynamicCheckBox Example

week08/src/gui04/DynamicCheckBoxExample.java

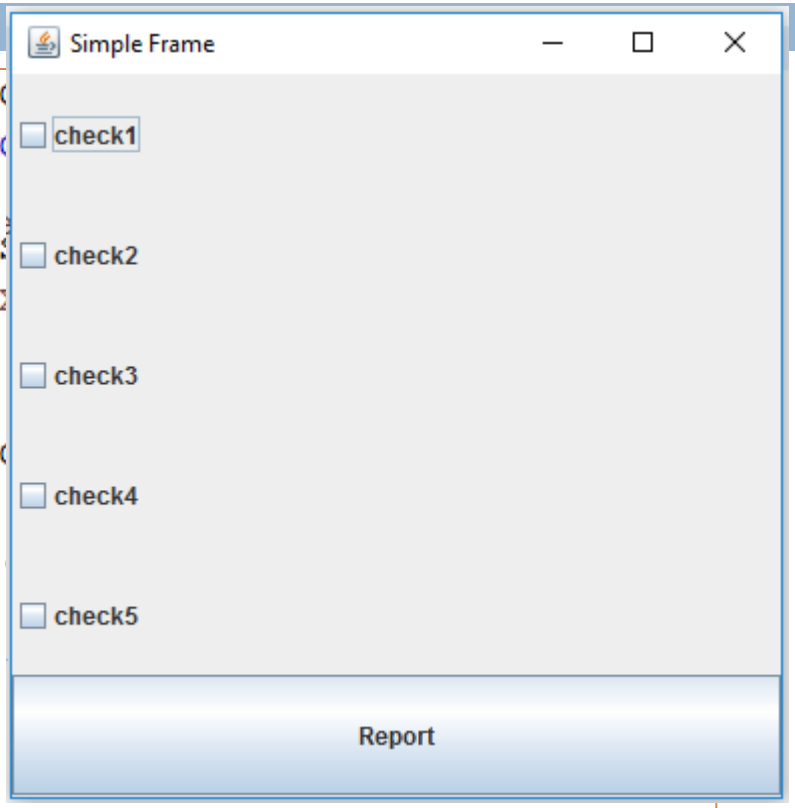
```
public static void main
    JFrame frame = new JFrame("Simple Frame");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setSize(400,400);

    JPanel mainPanel = new JPanel();
    mainPanel.setLayout(new GridLayout(6, 1));
    frame.add(mainPanel);

    ArrayList<JCheckBox> checkboxes = new ArrayList<JCheckBox>();
    ButtonGroup group1 = new ButtonGroup();
    for(int i=0; i<5; i++)
    {
        JCheckBox checkbox = new JCheckBox("check" + (i+1));
        checkboxes.add(checkbox);
        mainPanel.add(checkbox);
        group1.add(checkbox);
    }

    JButton button = new JButton("Report");
    mainPanel.add(button);

    button.addActionListener(new ButtonListener(checkboxes));

    frame.setVisible(true);
}
```

DynamicCheckBox Example


```
class ButtonListener implements ActionListener {
    private ArrayList<JCheckBox> checkboxes;

    public ButtonListener(ArrayList<JCheckBox> checkboxes) {
        this.checkboxes = checkboxes;
    }

    public void actionPerformed(ActionEvent e) {
        String msg = "Status:";
        for(int i=0; i<checkboxes.size(); i++)
        {
            if(i > 0)
                msg += ",";
            msg += checkboxes.get(i).isSelected();
        }

        JOptionPane.showMessageDialog(null, msg);
    }
};
```

DynamicCheckBox Example


```
class ButtonListener implements Ac
    private ArrayList<JCheckBox> c

    public ButtonListener(ArrayLis
        this.checkboxes = checkbox
    }

    public void actionPerformed(Ac
        String msg = "Status:";
        for(int i=0; i<checkboxes
        {
            if(i > 0)
                msg += ",";
            msg += checkboxes.get
        }

        JOptionPane.showMessageDialog(null, msg);
    }
};
```



MouseListener Interface

<<interface>>
MouseListener

+mousePressed(event: MouseEvent)
+mouseReleased(event: MouseEvent)
+mouseClicked(event: MouseEvent)
+mouseEntered(event: MouseEvent)
+mouseExited(event: MouseEvent)

MouseListener

<<interface>>
MouseListener

+mousePressed(event: MouseEvent)
+mouseReleased(event: MouseEvent)
+mouseClicked(event: MouseEvent)
+mouseEntered(event: MouseEvent)
+mouseExited(event: MouseEvent)

mousePressed	Invoked when a mouse button has been pressed on a component.
mouseReleased	Invoked when a mouse button has been released on a component.
mouseClicked	Invoked when the mouse button has been clicked (pressed and released) on a component.
mouseEntered	Invoked when the mouse enters a component.
mouseExited	Invoked when the mouse exits a component.

MouseListener Interface

<<interface>>
MouseListener

+mousePressed(event: MouseEvent)
+mouseReleased(event: MouseEvent)
+mouseClicked(event: MouseEvent)
+mouseEntered(event: MouseEvent)
+mouseExited(event: MouseEvent)

- Do we need to implement them all?

Event Adapters

- Adapter classes provide empty implementations to all the methods of interfaces.
- These abstract classes make interface usage easier
- Most event listener interfaces have their corresponding adapter classes that have all of the interface methods implemented.

Mouse Adapter

- This abstract class implements MouseListener interface with empty bodies.

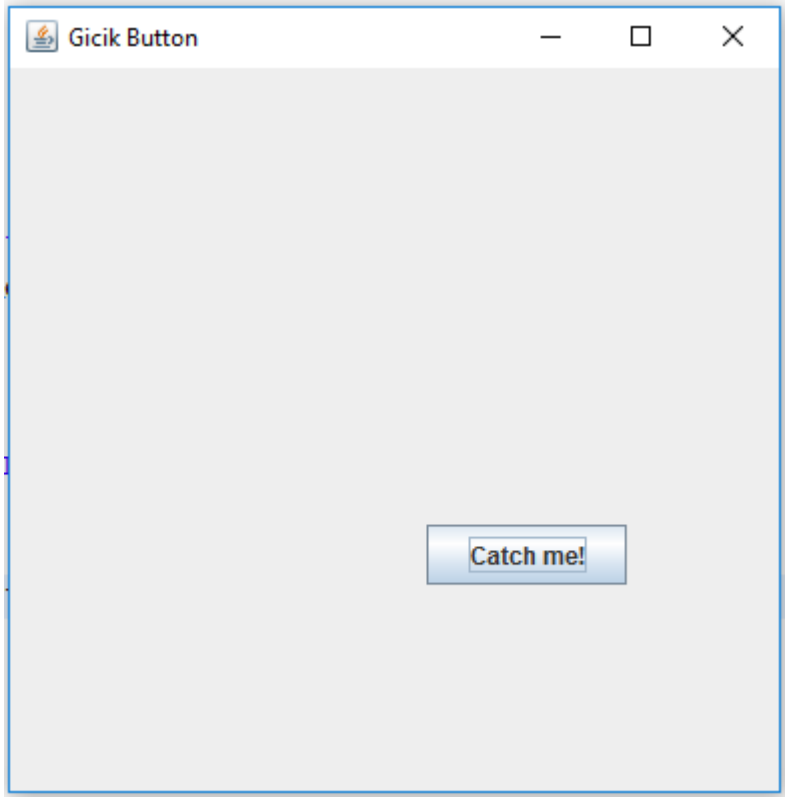
Mouse Listener Examples

GicikButton

MouseClickCoordinates

GicikButton

- Run the code to see how it works.



GicikButton


```
public class GicikButton {
    public static void main(String[] args) {
        JFrame frame = new JFrame("Gicik Button");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(400,400);

        frame.setLayout(null);
        JButton button = new JButton("Catch me!");
        button.setBounds(200, 200, 100, 30);

        button.addMouseListener(new GicikButtonHandler());

        frame.add(button);
        frame.setVisible(true);
    }
}
```

GicikButton


```
class GicikButtonHandler extends MouseAdapter {
    public void mouseEntered(MouseEvent event) {
        if (event.getSource() instanceof JButton) {
            JButton button = (JButton)event.getSource();
            Random r = new Random();
            int x = r.nextInt(350);
            int y = r.nextInt(350);
            button.setLocation(x,y);
        }
    }
}
```

- `setLocation` function invokes `setBounds` function but it uses the component's current width and height.

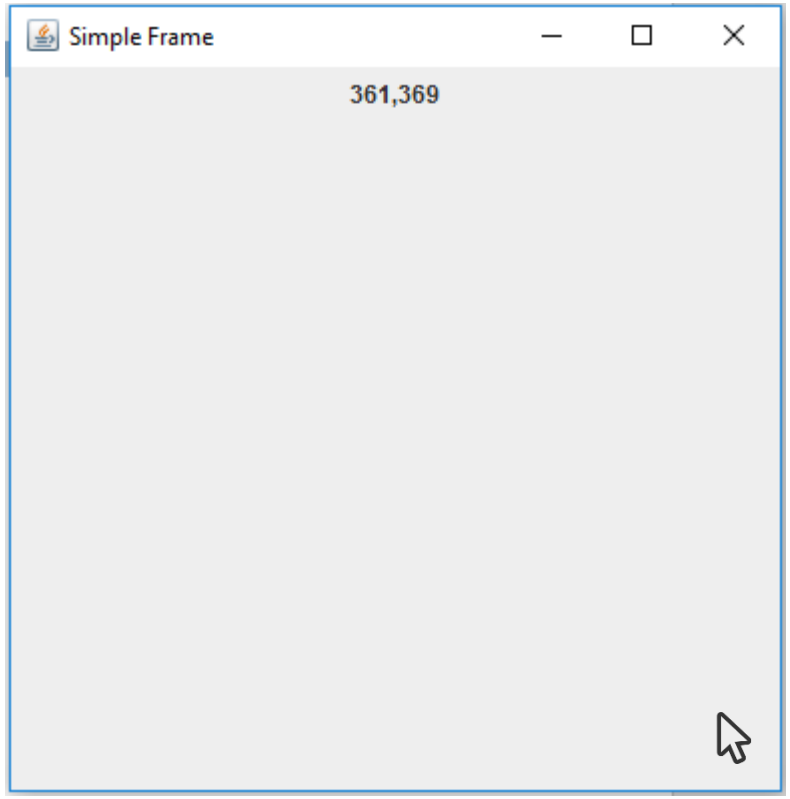
Mouse Listener Examples

GicikButton

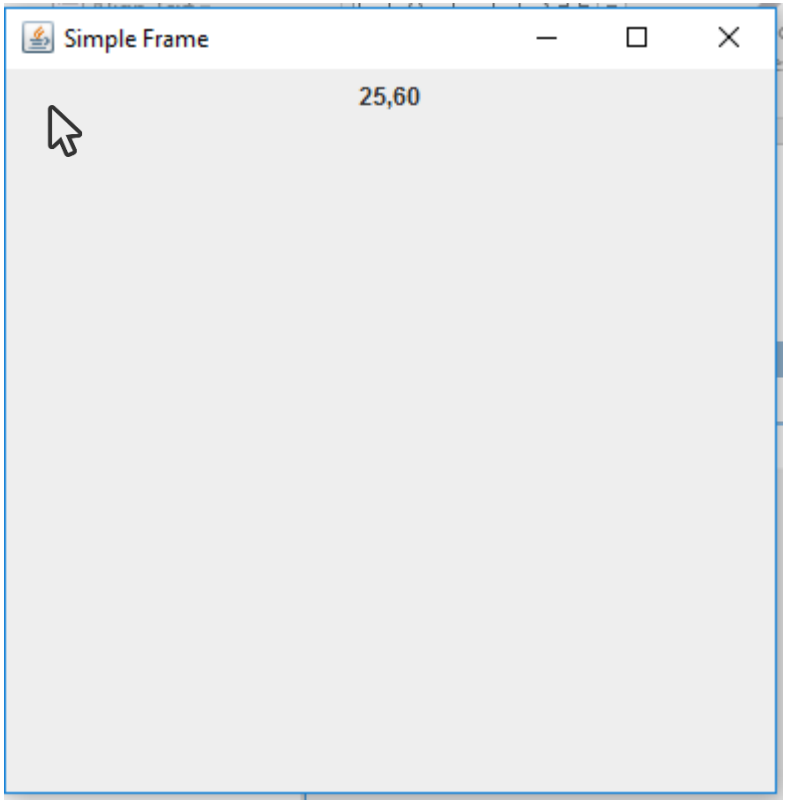
MouseClickCoordinates

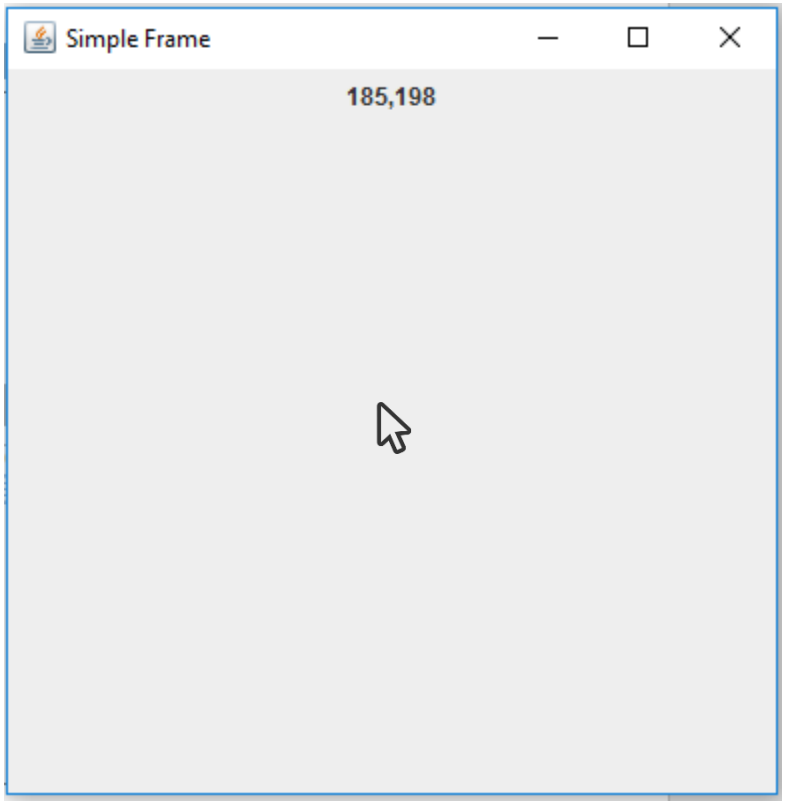
MouseClickCoordinates

- A 400x400 frame
- Whereever a mouse clicks on it, it prints out the x,y coordinates of the clicked location.



MouseClickCoordinates






```
Clicked at 25, 60 on mouse button 1 for 1 time(s).
Clicked at 185, 198 on mouse button 1 for 1 time(s).
Clicked at 361, 369 on mouse button 1 for 1 time(s).
```

MouseClickCoordinates


```
public static void main(String[] args) {
    JFrame frame = new JFrame("Simple Frame");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setSize(400,400);
    frame.setLayout(new FlowLayout());

    JLabel label = new JLabel("-,-");

    MyMouseListener listener = new MyMouseListener(label);
    frame.addMouseListener(listener);

    frame.add(label);
    frame.setVisible(true);
}
```

MouseClickCoordinates


```
class MyMouseListener extends MouseAdapter {
    private JLabel label;

    public MyMouseListener(JLabel label) {
        this.label = label;
    }
    public void mouseClicked(MouseEvent e) {
        System.out.print("Clicked at " + e.getX() + ", " + e.getY());
        System.out.print(" on mouse button " + e.getButton());
        System.out.println(" for " + e.getClickCount() + " time(s).");
    }
    public void mousePressed(MouseEvent e) {
        label.setText(e.getX() + "," + e.getY());
    }
}
```

MouseClickCoordinates


```
class MyMouseListener extends MouseAdapter {
    private JLabel label;

    public MyMouseListener(JLabel label) {
        this.label = label;
    }
    public void mouseClicked(MouseEvent e) {
        System.out.print("Clicked at " + e.getX() + ", " + e.getY());
        System.out.print(" on mouse button " + e.getButton());
        System.out.println(" for " + e.getClickCount() + " time(s).");
    }
    public void mousePressed(MouseEvent e) {
```

int `getX()` function returns the X position of the event

int `getY()` function returns the Y position of the event

int `getButton()` function returns the mouse button which has changed the state

int `getClickCount()` function returns the number of mouse clicks associated with the event within a certain time interval

Any Questions ?