Events

CPSC 1181 – O.O.

Jeremy Hilliker

Spring 2017

Overview

- Events
- Sources
- Listeners

- GUI example

- 2D example

Events, Sources, and Listeners

- An ***event*** is generated when something (of interest) occurs.
- A ***listener*** object ***registers*** itself with a ***source*** to receive events from that source***.***
- Events are ***fired*** by the source to send it to all registered listeners (almost always sequentially).
- Events are received/handled via *IoC* through a *callback*.

```
public interface ActionListener
extends EventListener
```

The listener interface for receiving action events. The class that is interested in processing an action event implements this interface, and the object created with that class is registered with a component, using the component's `addActionListener` method. When the action event occurs, that object's `actionPerformed` method is invoked.

Since:

`1.1`

See Also:

`ActionEvent, How to Write an Action Listener`

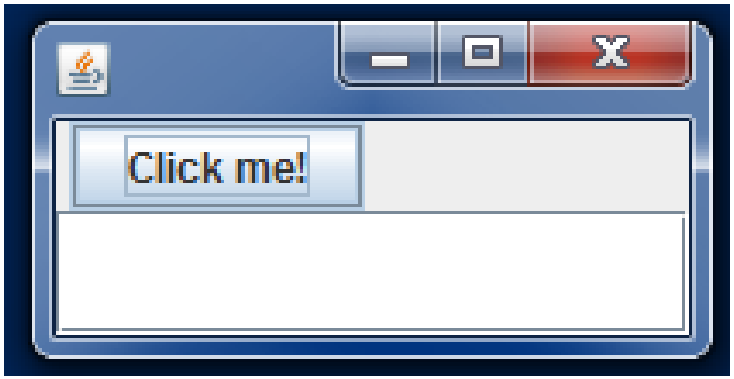
Method Summary

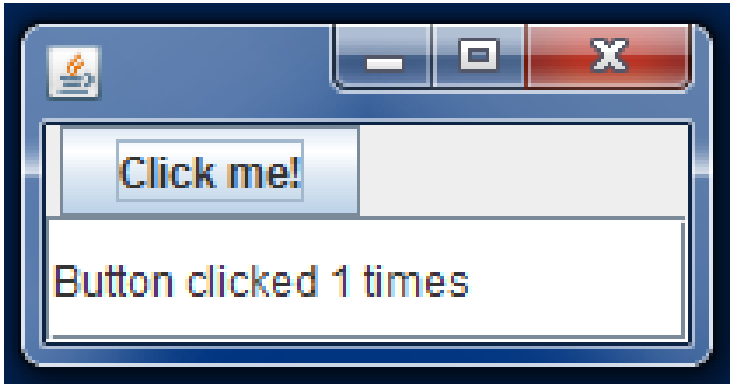
All Methods | **Instance Methods** | **Abstract Methods**

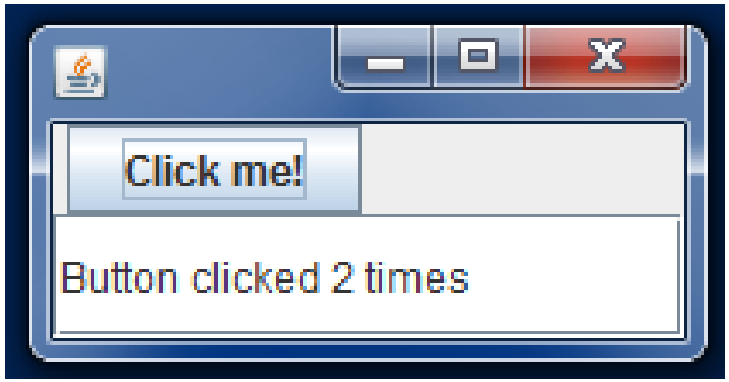
Modifier and Type	Method and Description
`void`	`actionPerformed(ActionEvent e)` Invoked when an action occurs.

Ex: Button Click Count GUI

```
13  public class ButtonListenerDemo implements ActionListener {
14
15      private int numClicks = 0;
16      private final JTextField text;
17
18      private ButtonListenerDemo(JTextField aText) {
19          text = aText;
20      }
21
22      public void actionPerformed(ActionEvent e) {
23          numClicks++;
24          text.setText("Button clicked " + numClicks + " times");
25      }
26
27      public static void main(String[] args) {
28          JFrame frame = new JFrame();
29          frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
30          frame.setLayout(new BoxLayout(
31              frame.getContentPane(), BoxLayout.Y_AXIS));
32
33          // add components
34          JButton button = new JButton("Click me!");
35          frame.add(button);
36          JTextField text = new JTextField();
37          frame.add(text);
38
39          // attach listeners
40          ActionListener bl = new ButtonListenerDemo(text);
41          button.addActionListener(bl);
42
43          // make visible
44          frame.setSize(200, 100);
45          frame.setVisible(true);
46      }
47  }
```


Click me!

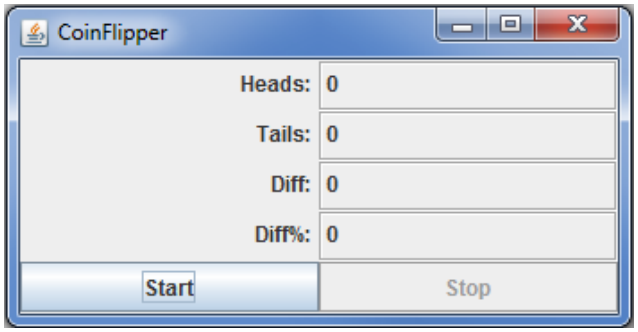

Click me!
Button clicked 1 times


Click me!
Button clicked 2 times

Ex: CoinFlip GUI

- A button will fire an event when clicked
- A listener will receive that event
  - And do “something”

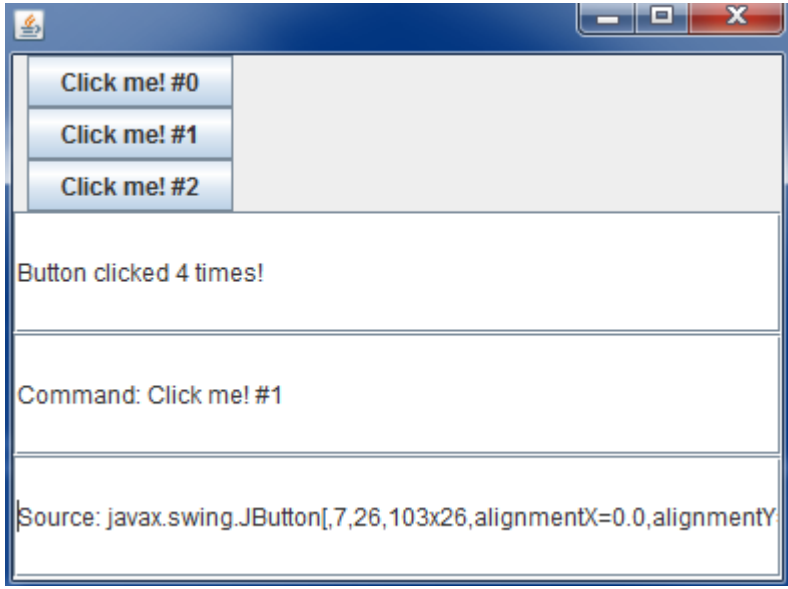
```
 9  public class CoinFlipperView extends JFrame implements ActionListener {
100
101      private JButton makeButton(String s) {
102          JButton b = new JButton(s);
103          b.setActionCommand(s);
104          b.addActionListener(this);
105          return b;
106      }
 48
 49      public void actionPerformed(ActionEvent e) {
 50          boolean start = btnStart == e.getSource();
 51          btnStart.setEnabled(!start);
 52          btnStop.setEnabled(start);
 53          if(start) {
 54              coinFlipTask = new CoinFlipTask();
 55              coinFlipTask.execute();
 56          } else {
 57              coinFlipTask.cancel(true);
 58              coinFlipTask = null;
 59          }
 60      }
```



Ex: GUI x3

- 3 buttons will fire events when clicked
- 1 listener will receive those event
  - And do “something”

```
18      public ThreeButtonListenerDemo(
19          JTextField aMsg, JTextField aCmd, JTextField aSource) {
20
21          super(aMsg);
22          cmd = aCmd;
23          src = aSource;
24      }
25
26      public void actionPerformed(ActionEvent e) {
27          super.actionPerformed(e);
28          cmd.setText("Command: " + e.getActionCommand());
29          src.setText("Source: " + e.getSource());
30      }
31
32      public static void main(String[] args) {
33          JFrame frame = new JFrame();
34          frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
35          frame.setLayout(new BoxLayout(
36              frame.getContentPane(), BoxLayout.Y_AXIS));
37
38          // create components
39          JButton[] buttons = new JButton[3];
40          JTextField[] texts = new JTextField[3];
41          for(int i = 0; i < 3; i++) {
42              buttons[i] = new JButton("Click me! #" + i);
43              texts[i] = new JTextField();
44          }
45
46          // add components, attach listeners
47          ActionListener bl = new ThreeButtonListenerDemo(
48              texts[0], texts[1], texts[2]);
49          for(JButton b : buttons) {
50              frame.add(b);
51              b.addActionListener(bl);
52          }
53          for(JTextField tf: texts) {
54              frame.add(tf);
55          }
56
57          // make visible
58          frame.setSize(400, 300);
59          frame.setVisible(true);
60      }
```



Ex: 2D Graphics

- Recall our animated clock?
  - Used IoC and a callback on a Timer

```
1  import java.awt.*;
2  import javax.swing.JComponent;
3  import javax.swing.Timer;
4  import java.time.LocalTime;
5
6  public class ClockComponent extends JComponent {
7
8      public ClockComponent() {
9          new Timer(1000/60, (a) -> {this.repaint();}).start();
10     }
```

- Next: We'll listen for
  - mouse clicks
  - key presses

java.awt.event MouseListener

Methods

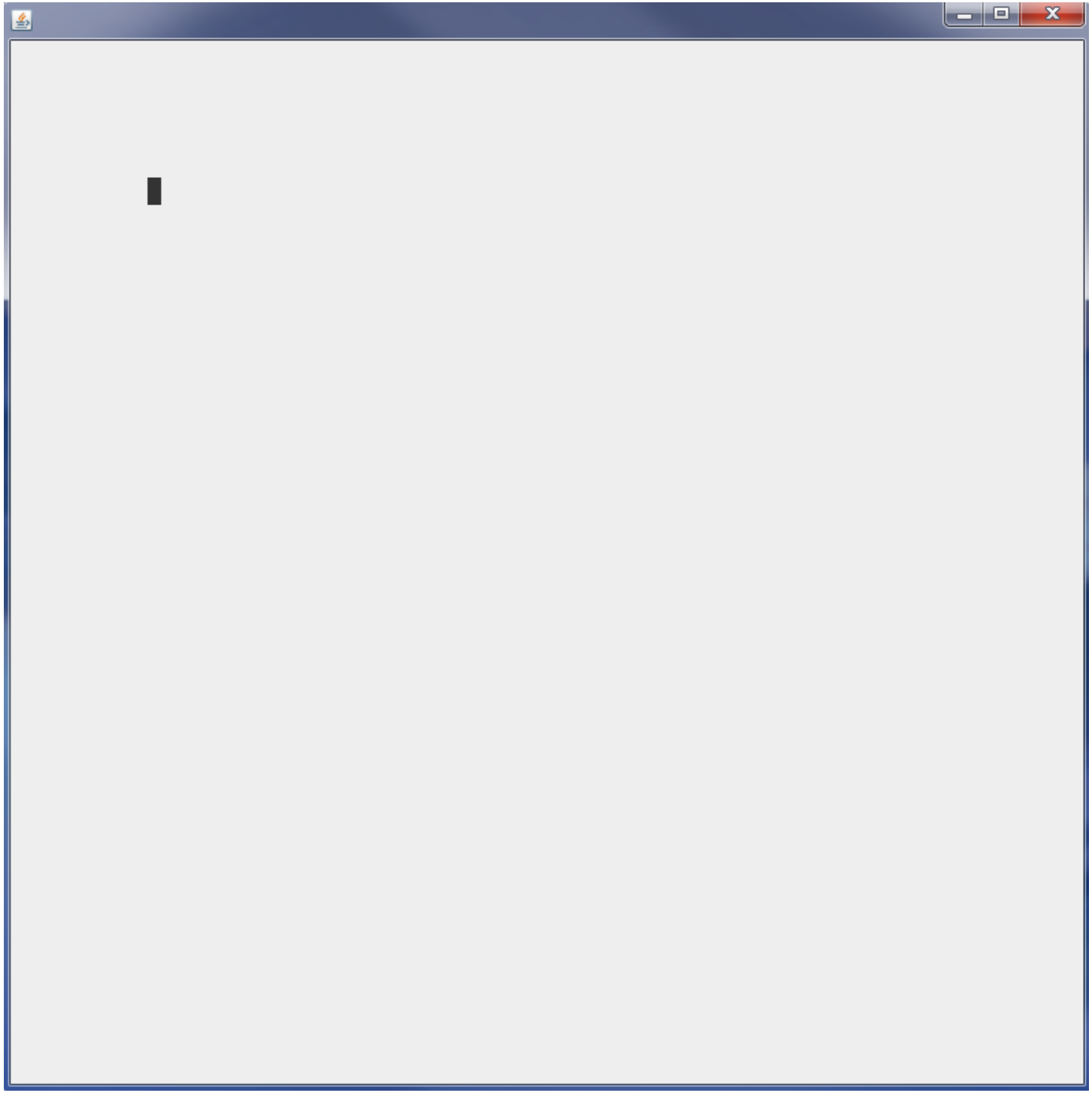
Modifier and Type	Method and Description
void	**mouseClicked**(MouseEvent e) Invoked when the mouse button has been clicked (pressed and released) on a component.
void	**mouseEntered**(MouseEvent e) Invoked when the mouse enters a component.
void	**mouseExited**(MouseEvent e) Invoked when the mouse exits a component.
void	**mousePressed**(MouseEvent e) Invoked when a mouse button has been pressed on a component.
void	**mouseReleased**(MouseEvent e) Invoked when a mouse button has been released on a component.

java.awt.event MouseAdapter

```
public abstract class MouseAdapter
extends Object
implements MouseListener, MouseWheelListener, MouseMotionListener
```

Methods

Modifier and Type	Method and Description
void	**mouseClicked**(MouseEvent e) Invoked when the mouse button has been clicked (pressed and released) on a component.
void	**mouseDragged**(MouseEvent e) Invoked when a mouse button is pressed on a component and then dragged.
void	**mouseEntered**(MouseEvent e) Invoked when the mouse enters a component.
void	**mouseExited**(MouseEvent e) Invoked when the mouse exits a component.
void	**mouseMoved**(MouseEvent e) Invoked when the mouse cursor has been moved onto a component but no buttons have been pushed.
void	**mousePressed**(MouseEvent e) Invoked when a mouse button has been pressed on a component.
void	**mouseReleased**(MouseEvent e) Invoked when a mouse button has been released on a component.
void	**mouseWheelMoved**(MouseWheelEvent e) Invoked when the mouse wheel is rotated.



```
4  public class MovingBoxComponent extends JComponent {
5
6      private final static int BOX_X = 100;
7      private final static int BOX_Y = 100;
8      public final static int BOX_WIDTH = 10;
9      public final static int BOX_HEIGHT = 20;
10
11     private final Rectangle box;
12
13     public MovingBoxComponent() {
14         box = new Rectangle(BOX_X, BOX_Y, BOX_WIDTH, BOX_HEIGHT);
15     }
16
17     public void paintComponent(Graphics g) {
18         super.paintComponent(g);
19         ((Graphics2D) g).fill(box);
20     }
21
22     public void moveBy(int dx, int dy) {
23         box.translate(dx, dy);
24         repaint();
25     }
26
27     public void moveTo(int x, int y) {
28         box.setLocation(x, y);
29         repaint();
30     }
31
32     public boolean boxContains(Point p) {
33         return box.contains(p);
34     }
35 }
```

```
6  public class MovingBoxGame {
7
8      private final static int FRAME_WIDTH = 800;
9      private final static int FRAME_HEIGHT = 800;
10
11     public static void main(String[] args) {
12         JFrame frame = new JFrame();
13         frame.setSize(FRAME_WIDTH, FRAME_HEIGHT);
14         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
15
16         final MovingBoxComponent gameField = new MovingBoxComponent();
17         frame.add(gameField);
18
19         gameField.addMouseListener(new MouseAdapter() {
20             private final Random rand = new Random();
21             public void mousePressed(MouseEvent e) {
22                 if(gameField.boxContains(e.getPoint())) {
23                     gameField.moveTo(
24                         rand.nextInt(FRAME_WIDTH - gameField.BOX_WIDTH),
25                         rand.nextInt(FRAME_HEIGHT - gameField.BOX_HEIGHT));
26                 }
27                 gameField.requestFocusInWindow();
28             }
29         });
```

```
        gameField.addKeyListener(new KeyAdapter() {
            public void keyPressed(KeyEvent e) {
                System.err.println(e);
                int dx = 0;
                int dy = 0;
                switch (e.getKeyCode()) {
                    case KeyEvent.VK_KP_UP:
                    case KeyEvent.VK_UP:
                        dy = -1;
                        break;
                    case KeyEvent.VK_KP_DOWN:
                    case KeyEvent.VK_DOWN:
                        dy = +1;
                        break;
                    case KeyEvent.VK_KP_LEFT:
                    case KeyEvent.VK_LEFT:
                        dx = -1;
                        break;
                    case KeyEvent.VK_KP_RIGHT:
                    case KeyEvent.VK_RIGHT:
                        dx = +1;
                        break;
                    default:
                }
                if(e.isShiftDown()) {
                    dx *= 2;
                    dy *= 2;
                }
                gameField.moveBy(dx, dy);
            }
        });

        frame.setVisible(true);
        gameField.requestFocusInWindow();
    }
}
```

Recap

- Events
  - ActionEvent
- Sources
- Listeners
  - ActionListener

- GUI example
  - JButton
- 2D example
  - MouseListener
  - MouseAdapter
  - KeyAdapter