

Events and Listeners

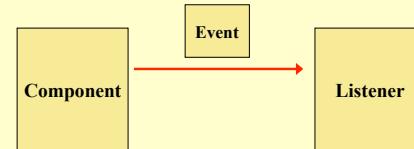
CSC 2014 – Java Bootcamp

Dr. Mary-Angela Papalaskari
Department of Computing Sciences
Villanova University

Course website:

www.csc.villanova.edu/~map/2014/

Events and Listeners



A component object generates an event

A corresponding listener object is designed to respond to the event

When the event occurs, the component calls the appropriate method of the listener, passing an object that describes the event

Buttons

- A *push button* is defined by the `JButton` class
- It generates an *action event*
- The `PushCounter` example displays a push button that increments a counter each time it is pushed
- See `PushCounter.java`
- See `PushCounterPanel.java`

```
*****  
// PushCounter.java    Authors: Lewis/Loftus  
// Demonstrates a graphical user interface and an event listener.  
*****  
  
import javax.swing.JFrame;  
  
public class PushCounter  
{  
    // Creates the main program frame.  
    public static void main(String[] args)  
    {  
        JFrame frame = new JFrame("Push Counter");  
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
  
        frame.getContentPane().add(new PushCounterPanel());  
  
        frame.pack();  
        frame.setVisible(true);  
    }  
}
```

```

//***** PushCounterPanel.java      Authors: Lewis/Loftus *****
// Demonstrates a graphical user interface and an event listener.
//***** Push Counter Example *****

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class PushCounterPanel extends JPanel
{
    private int count;
    private JButton push;
    private JLabel label;

    //-----
    // Constructor: Sets up the GUI.
    //-----
    public PushCounterPanel()
    {
        count = 0;

        push = new JButton("Push Me!");
        push.addActionListener(new ButtonListener());
    }

    continue
}

```

Push Me! Pushes: 7

```

continue

label = new JLabel("Pushes: " + count);

add(push);
add(label);

setPreferredSize(new Dimension(300, 40));
setBackground(Color.cyan);
}

//***** Button Listener *****

// Represents a listener for button push (action) events.
private class ButtonListener implements ActionListener
{
    //-----
    // Updates the counter and label when the button is pushed.
    //-----
    public void actionPerformed(ActionEvent event)
    {
        count++;
        label.setText("Pushes: " + count);
    }
}

```

Push Me! Pushes: 7

Push Counter Example

- The `ButtonListener` class is the listener for the action event generated by the button
- It is implemented as an *inner class*, which means it is defined within the body of another class
- That facilitates the communication between the listener and the GUI components
- Inner classes should only be used in situations where there is an intimate relationship between the two classes and the inner class is not needed in any other context

The Timer Class

- The Timer class of the javax.swing package is a GUI component, but it has no visual representation
- A Timer object generates an action event at specified intervals
- Timers can be used to manage any events that are based on a timed interval, such as an animation
- To create the illusion of movement, we use a timer to change the scene after an appropriate delay

The Timer Class

- The start and stop methods of the Timer class start and stop the timer
- The delay can be set using the Timer constructor or using the setDelay method
- See Rebound.java
- See ReboundPanel.java

```
/*
 * Rebound.java      Author: Lewis/Loftus
 *
 * Demonstrates an animation and the use of the Timer class.
 */
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class Rebound
{
    // Displays the main frame of the program.
    public static void main(String[] args)
    {
        JFrame frame = new JFrame("Rebound");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        frame.getContentPane().add(new ReboundPanel());
        frame.pack();
        frame.setVisible(true);
    }
}
```

```
//*****
// ReboundPanel.java      Author: Lewis/Loftus
//
// Represents the primary panel for the Rebound program.
//***** 

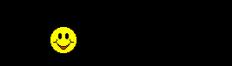
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class ReboundPanel extends JPanel
{
    private final int WIDTH = 300, HEIGHT = 100;
    private final int DELAY = 20, IMAGE_SIZE = 35;

    private ImageIcon image;
    private Timer timer;
    private int x, y, moveX, moveY;

    continue
}

```



```
continue

//-----+
// Sets up the panel, including the timer for the animation.
//-----+
public ReboundPanel()
{
    timer = new Timer(DELAY, new ReboundListener());
    image = new ImageIcon("happyFace.gif");

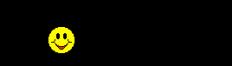
    x = 0;
    y = 40;
    moveX = moveY = 3;

    setPreferredSize(new Dimension(WIDTH, HEIGHT));
    setBackground(Color.black);
    timer.start();
}

//-----+
// Draws the image in the current location.
//-----+
public void paintComponent(Graphics page)
{
    super.paintComponent(page);
    image.paintIcon(this, page, x, y);
}

continue

```



```
continue

//*****
// Represents the action listener for the timer.
//***** 
private class ReboundListener implements ActionListener
{
    //-----
    // Updates the position of the image and possibly the direction
    // of movement whenever the timer fires an action event.
    //-----

    public void actionPerformed(ActionEvent event)
    {
        x += moveX;
        y += moveY;

        if (x <= 0 || x >= WIDTH-IMAGE_SIZE)
            moveX = moveX * -1;

        if (y <= 0 || y >= HEIGHT-IMAGE_SIZE)
            moveY = moveY * -1;

        repaint();
    }
}
```

