

**Lecture 22. Listening to events on a GUI
(and development of a loop)**

Sec. 17.4 contains this material. Corresponding lectures on ProgramLive CD is a better way to learn the material.

Why men think “computer” should be a feminine word

1. No one but their creator understands their internal logic.
2. The native language they use to talk with other computers is incomprehensible to everyone else.
3. Even the smallest mistakes are stored in long term memory for possible later retrieval.
4. As soon as you commit to one, half your paycheck goes for accessories for it.

Why women think “computer” should be a masculine word

1. In order to do anything with them, you have to turn them on.
2. They have a lot of data but still can't think for themselves.
3. They are supposed to help you solve problems, but half the time they ARE the problem.
4. As soon as you commit to one, you realize that if you had waited a little longer, you could have gotten a better model.

**Lecture 22. Listening to events on a GUI
(and development of a loop)**

Sec. 17.4 contains this material. Corresponding lectures on ProgramLive CD is a better way to learn the material.

Why men think “computer” should be a feminine word

1. No one but their creator understands their internal logic.
2. The native language they use to talk with other computers is incomprehensible to everyone else.
3. Even the smallest mistakes are stored in long term memory for possible later retrieval.
4. As soon as you commit to one, half your paycheck goes for accessories for it.

Why women think “computer” should be a masculine word

1. In order to do anything with them, you have to turn them on.
2. They have a lot of data but still can't think for themselves.
3. They are supposed to help you solve problems, but half the time they ARE the problem.
4. As soon as you commit to one, you realize that if you had waited a little longer, you could have gotten a better model.

**Lecture 22. Listening to events on a GUI
(and development of a loop)**

Sec. 17.4 contains this material. Corresponding lectures on ProgramLive CD is a better way to learn the material.

Why men think “computer” should be a feminine word

1. No one but their creator understands their internal logic.
2. The native language they use to talk with other computers is incomprehensible to everyone else.
3. Even the smallest mistakes are stored in long term memory for possible later retrieval.
4. As soon as you commit to one, half your paycheck goes for accessories for it.

Why women think “computer” should be a masculine word

1. In order to do anything with them, you have to turn them on.
2. They have a lot of data but still can't think for themselves.
3. They are supposed to help you solve problems, but half the time they ARE the problem.
4. As soon as you commit to one, you realize that if you had waited a little longer, you could have gotten a better model.

- Lecture 22. Listening to events on a GUI
(and development of a loop)**

Sec. 17.4 contains this material. Corresponding lectures on ProgramLive CD is a better way to learn the material.

Why men think “computer” should be a feminine word

 1. No one but their creator understands their internal logic.
 2. The native language they use to talk with other computers is incomprehensible to everyone else.
 3. Even the smallest mistakes are stored in long term memory for possible later retrieval.
 4. As soon as you commit to one, half your paycheck goes for accessories for it.

Why women think “computer” should be a masculine word

 1. In order to do anything with them, you have to turn them on.
 2. They have a lot of data but still can't think for themselves.
 3. They are supposed to help you solve problems, but half the time they ARE the problem.
 4. As soon as you commit to one, you realize that if you had waited a little longer, you could have gotten a better model.

**Lecture 22. Listening to events on a GUI
(and development of a loop)**

Sec. 17.4 contains this material. Corresponding lectures on ProgramLive CD is a better way to learn the material.

Why men think “computer” should be a feminine word

1. No one but their creator understands their internal logic.
2. The native language they use to talk with other computers is incomprehensible to everyone else.
3. Even the smallest mistakes are stored in long term memory for possible later retrieval.
4. As soon as you commit to one, half your paycheck goes for accessories for it.

Why women think “computer” should be a masculine word

1. In order to do anything with them, you have to turn them on.
2. They have a lot of data but still can't think for themselves.
3. They are supposed to help you solve problems, but half the time they ARE the problem.
4. As soon as you commit to one, you realize that if you had waited a little longer, you could have gotten a better model.

- Lecture 22. Listening to events on a GUI
(and development of a loop)**

Sec. 17.4 contains this material. Corresponding lectures on ProgramLive CD is a better way to learn the material.

Why men think “computer” should be a feminine word

 1. No one but their creator understands their internal logic.
 2. The native language they use to talk with other computers is incomprehensible to everyone else.
 3. Even the smallest mistakes are stored in long term memory for possible later retrieval.
 4. As soon as you commit to one, half your paycheck goes for accessories for it.

Why women think “computer” should be a masculine word

 1. In order to do anything with them, you have to turn them on.
 2. They have a lot of data but still can't think for themselves.
 3. They are supposed to help you solve problems, but half the time they ARE the problem.
 4. As soon as you commit to one, you realize that if you had waited a little longer, you could have gotten a better model.

Developing a loop

```
// Set x to length of largest segment of equal values in b[0..n-1].  
// Precondition: x is sorted (in ascending order)
```

0

1 1 2 3 3 3 5 5 5 7 8 8 8 8 9

n

n

14

x

```
// x = length of largest segment of equal values in b[0..n-1].
```

2

Developing a loop

```
// Set x to length of largest segment of equal values in b[0..n-1].  
// Precondition: x is sorted (in ascending order)
```

0

1 1 2 3 3 3 5 5 5 7 8 8 8 8 9

n

n

14

x

```
// x = length of largest segment of equal values in b[0..n-1].
```

2

Developing a loop

```
// Set x to length of largest segment of equal values in b[0..n-1].  
// Precondition: x is sorted (in ascending order)
```

0

1 1 2 3 3 3 5 5 5 7 8 8 8 8 9

n

n

14

x

```
// x = length of largest segment of equal values in b[0..n-1].
```

2

Developing a loop

```
// Set x to length of largest segment of equal values in b[0..n-1].  
// Precondition: x is sorted (in ascending order)
```

0

1 1 2 3 3 3 5 5 5 7 8 8 8 8 9

n

n

14

x

```
// x = length of largest segment of equal values in b[0..n-1].
```

2

- **Listening to events:** mouseclick, mouse movement into or out of a window, a keystroke, etc.
- An **event** is a mouseclick, a mouse movement into or out of a window, a keystroke, etc.
- To be able to “listen to” a kind of event, you have to
 1. Write a method that will listen to the event.
 2. Let Java know that the method is defined in the class.
 3. Register an instance of the class that contains the method as a *listener* for the event.

We show you how to do this for clicks on buttons, clicks on components, and keystrokes.

- **Listening to events:** mouseclick, mouse movement into or out of a window, a keystroke, etc.
 - An **event** is a mouseclick, a mouse movement into or out of a window, a keystroke, etc.
 - To be able to “listen to” a kind of event, you have to
 1. Write a method that will listen to the event.
 2. Let Java know that the method is defined in the class.
 3. Register an instance of the class that contains the method as a *listener* for the event.
- We show you how to do this for clicks on buttons, clicks on components, and keystrokes.

- **Listening to events:** mouseclick, mouse movement into or out of a window, a keystroke, etc.
- An **event** is a mouseclick, a mouse movement into or out of a window, a keystroke, etc.
- To be able to “listen to” a kind of event, you have to
 1. Write a method that will listen to the event.
 2. Let Java know that the method is defined in the class.
 3. Register an instance of the class that contains the method as a *listener* for the event.

We show you how to do this for clicks on buttons, clicks on components, and keystrokes.

1. Write the procedure to be called when button is clicked:

```
/** Process click of button */  
public void actionPerformed(ActionEvent ae) {  
    ...  
}
```

**Listening to
a Button**
3. Have class implement interface ActionListener:

```
public class C extends JFrame implements  
    ActionListener {  
    ...  
}
```
4. Add instance of this class as an “action listener” for button:

```
button.addActionListener(this);
```

4

1. Write the procedure to be called when button is clicked:

```
/** Process click of button */  
public void actionPerformed(ActionEvent ae) {  
    ...  
}
```

**Listening to
a Button**
3. Have class implement interface ActionListener:

```
public class C extends JFrame implements  
    ActionListener {  
    ...  
}
```
4. Add instance of this class as an “action listener” for button:

```
button.addActionListener(this);
```

4

1. Write the procedure to be called when button is clicked:

```
/** Process click of button */  
public void actionPerformed(ActionEvent ae) {  
    ...  
}
```

**Listening to
a Button**
3. Have class implement interface ActionListener:

```
public class C extends JFrame implements  
    ActionListener {  
    ...  
}
```
4. Add instance of this class as an “action listener” for button:

```
button.addActionListener(this);
```

4

1. Write the procedure to be called when button is clicked:

```
/** Process click of button */  
public void actionPerformed(ActionEvent ae) {  
    ...  
}
```

**Listening to
a Button**
3. Have class implement interface ActionListener:

```
public class C extends JFrame implements  
    ActionListener {  
    ...  
}
```
4. Add instance of this class as an “action listener” for button:

```
button.addActionListener(this);
```

4

1. Write the procedure to be called when button is clicked:

```
/** Process click of button */  
public void actionPerformed(ActionEvent ae) {  
    ...  
}
```

**Listening to
a Button**
3. Have class implement interface ActionListener:

```
public class C extends JFrame implements  
    ActionListener {  
    ...  
}
```
4. Add instance of this class as an “action listener” for button:

```
button.addActionListener(this);
```

4

1. Write the procedure to be called when button is clicked:

```
/** Process click of button */  
public void actionPerformed(ActionEvent ae) {  
    ...  
}
```

**Listening to
a Button**
3. Have class implement interface ActionListener:

```
public class C extends JFrame implements  
    ActionListener {  
    ...  
}
```
4. Add instance of this class as an “action listener” for button:

```
button.addActionListener(this);
```

4

```

/** An instance has two buttons. Exactly one is always enabled. */
public class ButtonDemo1 extends JFrame
    implements ActionListener {

    /** Class invariant: exactly one of eastB and westB is enabled */
    private JButton westB = new JButton("west");
    private JButton eastB = new JButton("east");

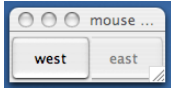
    /** Constructor: frame with title t & two buttons */
    public ButtonDemo1(String t) {
        super(t);
        Container cp = getContentPane();
        cp.add(westB, BorderLayout.WEST);
        cp.add(eastB, BorderLayout.EAST);

        westB.setEnabled(false);
        eastB.setEnabled(true);
        westB.addActionListener(this);
        eastB.addActionListener(this);

        pack();
        setVisible(true);
    }
}

```

Listening to a Button



Process a click of a button */

```

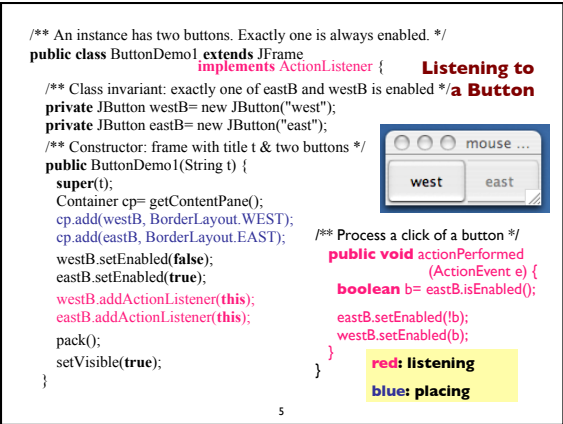
public void actionPerformed
    (ActionEvent e) {
    boolean b = eastB.is_enabled();

    eastB.setEnabled(!b);
    westB.setEnabled(b);
}

```

red: listening
blue: placing

5



```

/** An instance has two buttons. Exactly one is always enabled. */
public class ButtonDemo1 extends JFrame
    implements ActionListener {

    /** Class invariant: exactly one of eastB and westB is enabled */
    private JButton westB = new JButton("west");
    private JButton eastB = new JButton("east");

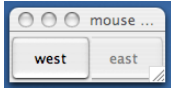
    /** Constructor: frame with title t & two buttons */
    public ButtonDemo1(String t) {
        super(t);
        Container cp = getContentPane();
        cp.add(westB, BorderLayout.WEST);
        cp.add(eastB, BorderLayout.EAST);

        westB.setEnabled(false);
        eastB.setEnabled(true);
        westB.addActionListener(this);
        eastB.addActionListener(this);

        pack();
        setVisible(true);
    }
}

```

Listening to a Button



Process a click of a button */

```

public void actionPerformed
    (ActionEvent e) {
    boolean b = eastB.is_enabled();

    eastB.setEnabled(!b);
    westB.setEnabled(b);
}

```

red: listening
blue: placing

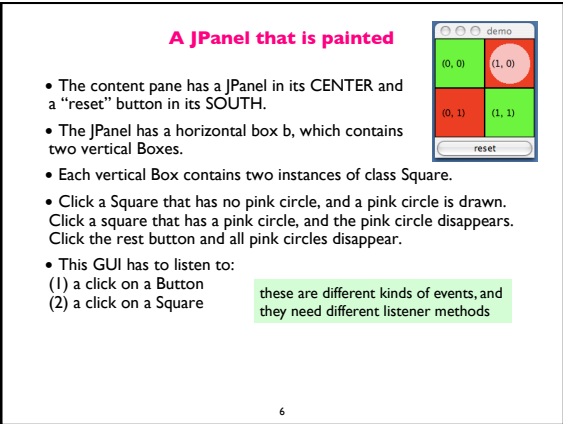
5

A JPanel that is painted

- The content pane has a JPanel in its CENTER and a "reset" button in its SOUTH.
- The JPanel has a horizontal box b, which contains two vertical Boxes.
- Each vertical Box contains two instances of class Square.
- Click a Square that has no pink circle, and a pink circle is drawn. Click a square that has a pink circle, and the pink circle disappears. Click the reset button and all pink circles disappear.
- This GUI has to listen to:
 - (1) a click on a Button
 - (2) a click on a Square

these are different kinds of events, and they need different listener methods

- ## A JPanel that is painted
- The content pane has a JPanel in its CENTER and a "reset" button in its SOUTH.
 - The JPanel has a horizontal box b, which contains two vertical Boxes.
 - Each vertical Box contains two instances of class Square.
 - Click a Square that has no pink circle, and a pink circle is drawn. Click a square that has a pink circle, and the pink circle disappears. Click the reset button and all pink circles disappear.
 - This GUI has to listen to:
 - (1) a click on a Button
 - (2) a click on a Square
- these are different kinds of events, and they need different listener methods



A JPanel that is painted

- The content pane has a JPanel in its CENTER and a "reset" button in its SOUTH.
- The JPanel has a horizontal box b, which contains two vertical Boxes.
- Each vertical Box contains two instances of class Square.
- Click a Square that has no pink circle, and a pink circle is drawn. Click a square that has a pink circle, and the pink circle disappears. Click the reset button and all pink circles disappear.
- This GUI has to listen to:
 - (1) a click on a Button
 - (2) a click on a Square

these are different kinds of events, and they need different listener methods

/** An instance is a JPanel of size (WIDTH,HEIGHT). Green or red depending on whether the sum of constructor parameters is even or odd. ... */

Class Square

```
public class Square extends JPanel {
    public static final int HEIGHT= 70; // height and
    public static final int WIDTH= 70; // width of square

    private int x, y; // Coordinates of square on board
    private boolean hasDisk= false; // = "square has pink disk"

    /** Constructor: a square at (x,y) */
    public Square(int x, int y) {
        this.x= x;    this.y= y;
        setPreferredSize(new Dimension(WIDTH,HEIGHT));
    }

    /** Complement the "has pink disk" property */
    public void complementDisk() {
        hasDisk= ! hasDisk;
        repaint(); // Ask the system to repaint the square
    }
}
```



continued on next page

7

continuation of class Square

Class Square

```
/* paint this square using g. System calls
   paint whenever square has to be redrawn.*/
public void paint(Graphics g) {
    if ((x+y)%2 == 0) g.setColor(Color.green);
    else g.setColor(Color.red);
    g.fillRect(0, 0, WIDTH-1, HEIGHT-1);

    if (hasDisk) {
        g.setColor(Color.pink);
        g.fillOval(7, 7, WIDTH-14, HEIGHT-14);
    }

    g.setColor(Color.black);
    g.drawRect(0, 0, WIDTH-1, HEIGHT-1);
    g.drawString(""+x+", "+y+", ", 10, 5+HEIGHT/2);
}
}
```

```
/** Remove pink disk
    (if present) */
public void clearDisk() {
    hasDisk= false;
    // Ask system to
    // repaint square
    repaint();
}
```



8

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

A class that listens to a mouseclick in a Square

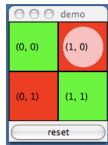
red: listening
blue: placing

```
/** Contains a method that responds to a
    mouse click in a Square */
public class MouseEvents
    extends MouseInputAdapter {
    // Complement "has pink disk" property
    public void mouseClicked(MouseEvent e) {
        Object ob= e.getSource();
        if (ob instanceof Square) {
            ((Square)ob).complementDisk();
        }
    }
}
```

This class has several methods (that do nothing) that process mouse events:

- mouse click
- mouse press
- mouse release
- mouse enters component
- mouse leaves component
- mouse dragged beginning in component

Our class overrides only the method that processes mouse clicks



9

```
public class MouseDemo2 extends JFrame
    implements ActionListener {
    Box b= new Box(BoxLayout.X_AXIS);
    Box leftC= new Box(BoxLayout.Y_AXIS);
    Square b00= new Square(0,0);
    Square b01= new Square(0,1);
    Box riteC= new Box(BoxLayout.Y_AXIS);
    Square b10= new Square(1,0);
    Square b11= new Square(1,1);
    JButton jb= new JButton("reset");
    MouseEvents me= new MouseEvents();

    /** Constructor: ... */
    public MouseDemo2() {
        super();
        leftC.add(b00); leftC.add(b01);
        riteC.add(b10); riteC.add(b11);
        b.add(leftC); b.add(riteC);
        Container cp= getContentPane();
        cp.add(b, BorderLayout.CENTER);
        cp.add(jb, BorderLayout.SOUTH);

        jb.addActionListener(this);
        b00.addMouseListener(me);
        b01.addMouseListener(me);
        b10.addMouseListener(me);
        b11.addMouseListener(me);
        pack(); setVisible(true);
        setResizable(false);

        public void actionPerformed(
            ActionEvent e) {
            b00.clearDisk(); b01.clearDisk();
            b10.clearDisk(); b11.clearDisk();
        }
    }
}
```

red: listening
blue: placing

Class MouseDemo2



10

Listening to the keyboard

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

Class AllCaps

```
public class AllCaps extends KeyAdapter {
    JFrame capsFrame= new JFrame();
    JLabel capsLabel= new JLabel();

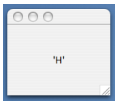
    public AllCaps() {
        capsLabel.setHorizontalAlignment(SwingConstants.CENTER);
        capsLabel.setText("");
        capsFrame.setSize(200,200);
        Container c= capsFrame.getContentPane();
        c.add(capsLabel);
        capsFrame.addKeyListener(this);
        capsFrame.show();
    }

    public void keyPressed(KeyEvent e) {
        char typedChar= e.getKeyChar();
        capsLabel.setText(""+typedChar+"").toUpperCase();
    }
}
```

1. Extend this class.

2. Override this method. It is called when a key stroke is detected.

3. Add this instance as a key listener for the frame



11