## Listening to events on GUIs

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

**Top finalists from a real-life "Dilbert quotes contest"**

As of tomorrow, employees will be able to access the building only using individual security cards. Pictures will be taken next Wednesday and employees will receive their cards in two weeks." (Fred Dales, Microsoft)

I need an exact list of specific unknown problems we might encounter. (Lykes Lines Shipping)

Email is not to be used to pass on information or data. It should be used only for company business. (Accounting manager, Electric Boat Company)

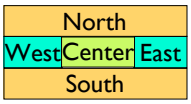
This project is so important, we can't let things that are more important interfere with it. (Advertising/Marketing manager, United Parcel Service)

Doing it right is no excuse for not meeting the schedule. (Plant manager, Delco Corporation)



## Layout manager: An instance controls the placement of components

- JFrame layout manager default: BorderLayout.
- BorderLayout layout manager:
- Can place 5 components:



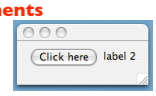

```
Container cp= getContentPane();
JButton jb= new JButton("Click here");
JButton jl= new JLabel( "label 2");

cp.add(jb, BorderLayout.EAST);
cp.add(jl, BorderLayout.WEST);

pack();
setVisible(true);
```

Place this in a constructor of a subclass of JFrame



## Layout manager: An instance controls the placement of components

- JPanel layout manager default: FlowLayout.
- Box layout manager default: BoxLayout.




```
Container cp= getContentPane();
JPanel p= new JPanel();
JButton b= new JButton("Click here");
JLabel jl= new JLabel("label 2");

p.add(b);
p.add(jl);

cp.add(p, BorderLayout.CENTER);
pack();
setVisible(true);
```

Place this in a constructor of a subclass of JFrame

Components are placed in a row in the order in which they were added. With FlowLayout, if the window is too narrow, components flow into the next row(s). With BoxLayout, they don't.



## 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 a button is clicked.:

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

**Listening to a Button**

2. Have the class implement interface ActionListener --write the class heading as

```
public class C extends JFrame  implements ActionListener {
   ...
}
```

We have not discussed interfaces, and we won't. Wait for CS 211!

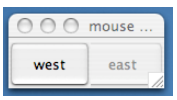
3. Add an instance of this class as an "action listener" for the button:

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



```
import javax.swing.*; import java.awt.*; import java.awt.event.*;
/** 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);
  }
  /** Process a click of a button */
  public void actionPerformed
                (ActionEvent e) {
    boolean b= eastB.isEnabled();
    eastB.setEnabled(!b);
    westB.setEnabled(b);
  }
}
```

**Listening to a Button**




**red: listening**

**blue: placing**

## 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 rest 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



## Class Square

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



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();
}
```




## A class that listens to a mouseclick in a Square

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

/** 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();
    }
  }
}
```

red: listening
blue: placing


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



```
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(t);
    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



## Listening to the keyboard

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

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());
  }
}
```

red: listening
blue: placing

1. Extend this class.

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

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