12. Listening to events. Inner and anonymous classes

Why men think "computer" should be a feminine word	Why women think "computer" should be a masculine word
1. No one but their creator understands their internal logic.	1. In order to do anything with them, you have to turn them on.
2. The native language they use to talk with other computers is incomprehensible to everyone else.	2. They have a lot of data but still can't think for themselves.
3. Even the smallest mistakes are stored in long term memory for possible later retrieval.	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, half your paycheck goes for accessories for it.	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.

1

Prelim: Thursday, 7 March: 7:30–9:00. Place: TBA
Makeup for those with approved conflicts: 5:30-7:00. Place: TBA
Can't make either? Contact Prof. Birman.
Review sessions: Sunday, 3 March: 1-3PM and 3-5PM, Olin 155

Summary of what exam covers:
Basically, everything **before** GUIs (i.e. before this week)
More detail will be provided in a document to be this evening on the course website and Piazza.

Previous exams: On course website (click "Exams")

Note: Today's slides have been updated since posting on course website. Today's version will be put on website this afternoon.

A1. 164 graded. 40 got 100. mean 92.3, median 94, stdev. 7.7

2

Listening to events: mouse click, mouse movement into or out of a window, a keystroke, etc.

• An event is a mouse click, 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. Have some class C implement an interface IN connected with the event.

  2. In class C, override methods required to implement IN; these methods are generally called when the event happens.

  3. Register an object of class C as a *listener* for the event. That object's methods will be called when event happens.

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

3

What is a JButton?
Instance: associated with a "button" on the GUI, which can be clicked to do something

```
jb1= new JButton()        // jb1 has no text on it
jb2= new JButton("first")  // jb2 has label "first" on it

jb2.isEnabled()           // true iff a click on button can be
                          // detected
jb2.setEnabled(b);        // Set enabled property

jb2.addActionListener(object); // object must have a method,
          // which is called when button jb2 clicked (next page)
```

At least 100 more methods; these are most important

JButton is in package javax.swing

4

Listening to a JButton

1. Implement interface ActionListener:
```
public class C extends JFrame  implements
        ...                          ActionListener {
}
```

2. In class C override actionPerformed, which is to be called when button is clicked:
```
/** Process click of button */
public void actionPerformed(ActionEvent e) {
    ...
}
```

3. Add an instance of class C an "action listener" for button:
```
button.addActionListener(this);
```

5

```
/** Object has two buttons. Exactly one is enabled. */
class ButtonDemo1 extends JFrame
                    implements ActionListener {
    /** Class inv: exactly one of eastB, westB is enabled */
    JButton westB= new JButton("west");
    JButton eastB= new JButton("east");

    public ButtonDemo1(String t) {
        super(t);
        Container cp= getContentPane();
        cp.add(westB, BLayout.WEST);
        cp.add(eastB, BLayout, EAST);

        westB.setEnabled(false);
        eastB.setEnabled(true);
        westB.addActionListener(this);
        eastB.addActionListener(this);
        pack(); setVisible(true);
    }

    public void actionPerformed
                (ActionEvent e) {
        boolean b=
            eastB.isEnabled();
        eastB.setEnabled(!b);
        westB.setEnabled(b);
    }
}
```

red: listening
blue: placing

Listening to a Button

6

1

Slide 7
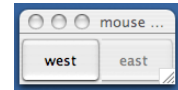
A JPanel that is painted

- The JFRame 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 Button reset
(2) a click on a Square (a Box)

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

7

Slide 8
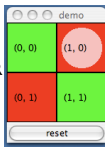
```
/** Instance: JPanel of size (WIDTH, HEIGHT).
          Green or red: */
public class Square extends JPanel {
  public static final int HEIGHT= 70;
  public static final int WIDTH= 70;
  private int x, y; // Panel is at (x, y)
  private boolean hasDisk= false;
  /** Const: square at (x, y). Red/green? Parity of 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
  }
```

Class Square

continued on later

8

Slide 9

Class Graphics

An object of abstract class Graphics has methods to draw on a component (e.g. on a JPanel, or canvas).

Major methods:

drawString("abc", 20, 30);	drawLine(x1, y1, x2, y2);
drawRect(x, y, width, height);	fillRect(x, y, width, height);
drawOval(x, y, width, height);	fillOval(x, y, width, height);
setColor(Color.red);	getColor()
getFont()	setFont(Font f);

More methods

You won't create an object of Graphics; you will be given one to use when you want to paint a component

Graphics is in package java.awt

9

Slide 10

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();
}
```

10

Slide 11

Listen to mouse event
(click, press, release, enter, leave on a component)

```
public interface MouseListener {
    void mouseClicked(MouseEvent e);
    void mouseEntered(MouseEvent e);
    void mouseExited(MouseEvent e);
    void mousePressed(MouseEvent e);
    void mouseReleased(MouseEvent e);
}
```
In package java.awt.event

Having to write all of these in a class that implements MouseListener, even though you don't want to use all of them, can be a pain. So, a class is provided that implements them in a painless.
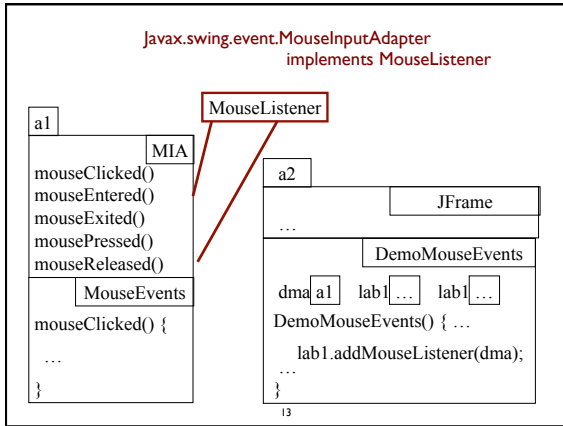
11

Slide 12

Listen to mouse event
(click, press, release, enter, leave on a component)

In package java.swing.event

```
public class MouseInputAdaptor
      implements MouseListener, MouseInputListener {
  public void mouseClicked(MouseEvent e) {}
  public void mouseEntered(MouseEvent e) {}
  public void mouseExited(MouseEvent e) {}
  public void mousePressed(MouseEvent e) {}
  public void mouseReleased(MouseEvent e) {}
  … others …
```

So, just write a subclass of MouseInputAdaptor and override only the methods appropriate for the application
}

12

Slide 13

Javax.swing.event.MouseInputAdapter
implements MouseListener

MouseListener

a1

MIA

mouseClicked()
mouseEntered()
mouseExited()
mousePressed()
mouseReleased()

a2

JFrame

…

DemoMouseEvents

MouseEvents

mouseClicked() {

…

}

dma a1   lab1 …   lab1 …

DemoMouseEvents() { …

   lab1.addMouseListener(dma);

…

}

13

Slide 14

```
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

demo
(0, 0)   (1, 0)
(0, 1)   (1, 1)
reset

/** 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

14

Slide 15

```
public class MD2 extends JFrame
          implements ActionListener {
  Box b= new Box(…X_AXIS);
  Box leftC= new Box(…Y_AXIS);
  Square b00, b01= new squares;
  Box riteC= new Box(..Y_AXIS);
  Square b10, b01= new squares;
  JButton jb= new JButton("reset");

  MouseEvents me=
          new MouseEvents();
/** Constructor: … */
public MouseDemo2() {
  super(t);
  place components on content pane;
  pack, make unresizeable, visible;
```

```
  jb.addActionListener(this);
  b00.addMouseListener(me);
  b01.addMouseListener(me);
  b10.addMouseListener(me);
  b11.addMouseListener(me);
}

public void actionPerformed (
        ActionEvent e) {
   call clearDisk() for
   b00, b01, b10, b11
}
```

red: listening

blue: placing

demo
(0, 0)   (1, 0)
(0, 1)   (1, 1)
reset

Class MouseDemo2

15

Slide 16

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.

'H'

16

Slide 17

```
public class BDemo3 extends JFrame  implements ActionListener {
    private JButton wButt, eButt …;

    public ButtonDemo3() {
        Add buttons to content pane, enable
           ne, disable the other
        wButt.addActionListener(this);
        eButt.addActionListener(new BeListener()); }

    public void actionPerformed(ActionEvent e) {
        boolean b= eButt.isEnabled();
        eButt.setEnabled(!b); wButt.setEnabled(b);  }
}
```

Have a different listener for each button

A listener for eastButt
```
class BeListener implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        boolean b= eButt.isEnabled();
        eButt.setEnabled(!b); wButt.setEnabled(b);
    }
}
```

Doesn't work! Can't reference eButt, wButt

17

Slide 18

Solution to problem: Make BeListener an inner class.

```
public class BDemo3 extends Jframe
          implements ActionListener {
    private JButton wButt, eButt …;

    public ButtonDemo3() { … }

    public void actionPerformed(ActionEvent

    private class BeListener implements ActionListener { … }
```

Just as you can declare variables and methods within a class, you can declare a class within a class

Inside-out rule says that methods in here
Can reference all the fields and methods

We demo this using ButtonDemo3

18

Problem: can't give a function as a parameter:

```
public void m() { …
    eButt.addActionListener(aP);
}

public void aP(ActionEvent e) { body }
```

Why not just give eButt the function to call?
Can't do it in Java!
Can in some other languages

```
public void m() { …
    eButt.addActionListener(new C());
}

public class C implements IN {
    public void aP(ActionEvent e) { body }
}
```

Java says: provide class C that wraps method; give eButt an object of class C

C must implement interface IN that has abstract method aP

19

Have a class for which only one object is created?
Use an **anonymous class**.
Use sparingly, and only when the anonymous class has 1 or 2 methods in it, because the syntax is ugly, complex, hard to understand.

```
public class BDemo3 extends JFrame  implements ActionListener {
    private JButton wButt, eButt …;

    public ButtonDemo3() { …
        eButt.addActionListener(new BeListener());
    }

    public void actionPerformed(ActionEvent e) { … }

    private class BeListener implements ActionListener {
        public void actionPerformed(ActionEvent e) { body }
    }
}
```

1 object of BeListener created. Ripe for making anonymous

20

Making class anonymous will replace **new BeListener()**

Expression that creates object of BeListener

```
eButt.addActionListener(  new BeListener ()   );

    private class BeListener implements ActionListener
        { declarations in class }
}
```

1. Write **new**

2. Write **new ActionListener**

3. Write **new ActionListener ()**

4. Write **new ActionListener ()**
    { declarations in class }

5. Replace **new** BeListener() by new-expression

2. Use name of interface that BeListener implements

3. Put in arguments of constructor call

4. Put in class body

21