Lecture 25 Listening to buttons and mice

Quotes by Tony Hoare

There are two ways of constructing a software design: (1) make it so simple that there are obviously no deficiencies and (2) make it so complicated that there are no obvious deficiencies.

Inside every large program is a small program trying to get to out.

The unavoidable price of reliability is simplicity.

I was eventually persuaded of the need to design programming notations so as to maximize the number of errors which cannot be made, or if made, can be reliably detected at compile time.

You cannot teach beginners top-down programming, because they don't know which end is up.

Lecture 25 Listening to buttons and mice

Reading material is the same as the previous lecture.

The programs we demo today are on the website. Download them and experiment with them!

Look at java.awt and javax.swing as examples of how one designs a large system of classes. This is not the only way to do GUIs. This is how Java does it.

Container: superclass of all Components that can hold other components.

Components are generally added to a Container c:

- c.add(new JButton("yes"));
- c.add(new JButton("no"), "north");

Basic top-level containers:
- JFrame: top-level window with no border
- JDialog: top-level window with border, menubar
- JDialog: top-level window used for a dialog

JPanel, primary use: as a container of other components. Allows one to organize objects into a unit, often to simplify layouts. See this on the next slides.

JPanel, secondary use: paint it with graphics commands --lines, circles, text, etc. (instead of using class Canvas).

Review of lecture 23:

Lecture 24. We:

1. Looked at different components, like JButton, JPanel.
2. Saw three different layout managers:
   - FlowLayout
   - BorderLayout
   - BoxLayout

They are used to lay out components as they are added to a container.

(3) Saw how to add components to a container. End with pack() and show().

Listening to a button

public class ButtonDemo1 extends JFrame implements ActionListener {
    private JButton westButton= new JButton("west");
    private JButton eastButton= new JButton("east");

    /** Constructor: invisible frame; title t, 2 buttons */
    public ButtonDemo1(String t) {
        super(t);
        Container cp= getContentPane();
        cp.add(westButton,BorderLayout.WEST);
        cp.add(eastButton,BorderLayout.EAST);
        westButton.setEnabled(false);
        eastButton.setEnabled(true);
        addActionListener(this);
        pack();
    }

    public void actionPerformed(ActionEvent e) {
        boolean b= eastButton.isEnabled();
        eastButton.setEnabled(!b);
        westButton.setEnabled(!b);
    }

    1. write procedure actionPerformed
    2. implement ActionListener
    3. Register this object with the button
/** Demo use of a mouse listener. */

    Instance is a JFrame that is a 2x2 grid of squares, each of class Square, and a button with title "reset". Clicking an empty square draws a pink disk on it. Clicking again removes it. Clicking button "reset" removes all pink disks.

    This class listens to clicks of the button. Clicks on a square are listened to in class Square.  

/** Module Demo extends JFrame */
    Box bbl=new Box(BoxLayout.Y_AXIS);    Square b00=new Square(0,0);    Square b01=new Square(0,1);    Box b= new Box(BoxLayout.Y_AXIS);    Square b10=new Square(1,0);    Square b11=new Square(1,1);    JButton jb=new JButton("reset");    Box b2=new Box(BoxLayout.X_AXIS);

// continued on next page //

Processing mouse clicks

/** A red or green square, possibly with a pink disk */
    public class Square extends JPanel {    ...       g.setColor(Color.green);    }    // continued on next slide //

/** Class Square (continued) //    /** Complement the "has pink disk" property */    public void complementDisk() {    hasDisk= !hasDisk;    repaint();    }    // also, call inherited method repaint()    /** Remove pink disk (if present) */    public void clearDisk() {    hasDisk= false;    repaint();    }

/** Contains methods that process mouse events */
    public class MouseEvents extends MouseInputAdapter {    /** Complement the "has pink disk" property */    public void mouseClicked(MouseEvent e) {    complementDisk();    }

}    API class MouseInputAdapter has several methods for processing mouse clicks, mouse presses, mouse releases, etc. We override only one of them here.

That's how we get the program to listen to mouse events.

// class MouseDemo continued from last page */

/** Constructor: an invisible JFrame with title t */
    public MouseDemo(String t) {    super();    br.add(b00);    br.add(b01);    br.add(b10);    br.add(b11);    b.add(br);    Container cp=new ContentPanel();    cp.add(b, BorderLayout.CENTER);    cp.add(jb, BorderLayout.SOUTH);    jb.addActionListener(new ButtonListener());    pack();    setResizable(false);    // don’t let user resize!!!    }

// An inner class so that it can refer to fields b00, etc. */
    /** Respond to mouse click on button jb */
    public class ButtonListener implements ActionListener {    public void actionPerformed(ActionEvent e) {    b00.clearDisk();    b01.clearDisk();    }

    }

// Class Square (continued) //