4.4.1

The main... **Event**

Programs that interact frequently with users must generally respond to **events**.

Each of the following produces an event.

- a click on an interface button.
- dragging a slider

Such events must be handled by the executing program by so-called **event**.

In Java events are handled by methods.

4.4.2

**The ThreeButtons Driver**

The ThreeButtons class supports a special kind of Driver class.

This Driver draws a window like the one shown.

This Driver handles three kinds of events, corresponding to the three buttons.
Requirements for a 3-Button Driver

1) The following two words must be inserted after “public class Driver”

2) The Driver class must include the following three parameterless public methods, to handle events from the corresponding buttons.

   leftAction
   midAction
   rightAction

(Note that public methods use the word “public” in place of “private”.)

3) The Driver class should use a ThreeButtonFrame, instead of JFrame.

   new ThreeButtonFrame("example string")

Such ThreeButtonFrame objects have a width of 600 and a height of 500.

The Shell of a 3-Button-Driver

public class Driver extends ThreeButtons {
   private ThreeButtonFrame window;
   // other private declarations belong here.

   public Driver() {
      window = new ThreeButtonFrame("Button Window");
      // other Driver instructions belong here.
   }

   public void leftAction() {
      // The code for handling LEFT button clicks belongs here.
   }

   public void midAction() {
      // The code for handling MID button clicks belongs here.
   }

   public void rightAction() {
      // The code for handling RIGHT button clicks belongs here.
   }
}

© 2006 Pearson Addison-Wesley. All rights reserved
Example

import java.awt.Color;
public class Driver extends ThreeButtons {
    private ThreeButtonFrame window;
    private Oval spot;

    public Driver() {
        window = new ThreeButtonFrame("Button Window");
        spot = new Oval(250, 200, 100, 100);
        spot.setBackground(Color.green);
        window.add(spot, 0);
        window.repaint();
    }

    public void leftAction() {
        spot.setBackground(Color.red);
        spot.repaint();
    }

    public void rightAction() {
        spot.setBackground(Color.blue);
        spot.repaint();
    }

    public void midAction() { /*this does nothing*/ }
}

Variation 1

import java.awt.Color;
public class Driver extends ThreeButtons {
    private ThreeButtonFrame window;

    public Driver() {
        Oval spot;
        window = new ThreeButtonFrame("Button Window");
        spot = new Oval(250, 200, 100, 100);
        spot.setBackground(Color.green);
        window.add(spot, 0);
        window.repaint();
    }

    public void leftAction() {
        spot.setBackground(Color.red);
        spot.repaint();
    }

    public void rightAction() {
        spot.setBackground(Color.blue);
        spot.repaint();
    }

    public void midAction() { /*this does nothing*/ }
}

Why doesn’t this work?
import java.awt.Color;
public class Driver extends ThreeButtons {
    private ThreeButtonFrame window;
    private Oval spot;
    public Driver() {
        window = new ThreeButtonFrame("Button Window");
        spot = new Oval(250, 200, 100, 100);
        spot.setBackground(Color.green);
        window.add(spot, 0);
        window.repaint();
    }
    public void leftAction() {
        Rectangle blueRect;
        blueRect = new Rectangle(0, 0, 20, 10);
        blueRect.setBackground(Color.blue);
        window.add(blueRect, 0);
        blueRect.repaint();
        window.repaint();
    }
    public void midAction() { /*this does nothing*/ }
    public void rightAction() { /*this does nothing*/ }
}

Event-driven Execution

The program begins by executing the “initialization” code. (e.g., Driver)

Program execution suspends, awaiting the occurrence of some event.

[leftAction()](LEFT button clicked) [rightAction()](RIGHT button clicked) [midAction()](MID button clicked)