Concepts of Programming Languages

Lecture 20 - Event-Driven Programming

Patrick Donnelly

Montana State University

Spring 2014

Administrivia

Assignments:

Programming #4 : due 04.28

Reading:

Chapter 14

Of all men's miseries the bitterest is this, to know so much and to have control over nothing.

Herodotus (484-432 BC)

Event-Driven Programming

A conventional model of computation has the program prescribe the exact order of input.

Programs terminate once the input is exhausted.

Event-driven programs do not control the sequence in which input events occur.

Event Handling

Definition

An ***event*** is a notification that something specific has occurred, such as a mouse click on a graphical button.

Definition

The ***event handler*** is a segment of code that is executed in response to an event.

Examples

GUI applications: Model-View-Controller design

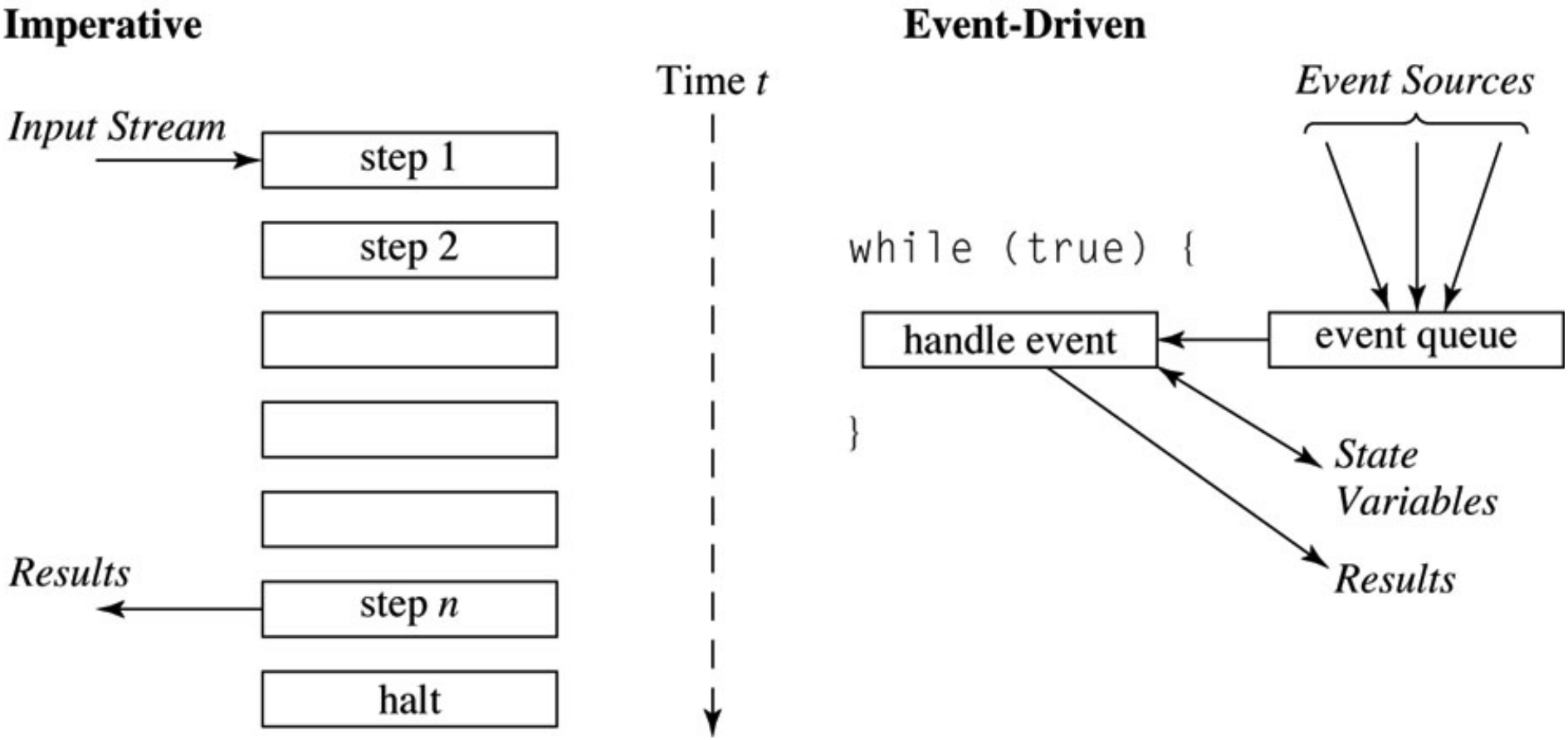
Embedded applications:

- cell phones
- car engines
- airplanes

Computation as interaction [Stein, 1998]:

> *Computation is a community of persistent entities coupled together by their ongoing interactive behavior . . .*
> *Beginning and end, when present, are special cases that can often be ignored.*

Imperative and Event-Driven Paradigms Contrasted



Event Sources

Input to an event-driven program comes from autonomous event sources.

Events occur asynchronously.

Example:

- human,
- robot sensors,
- engine sensors

Event Properties

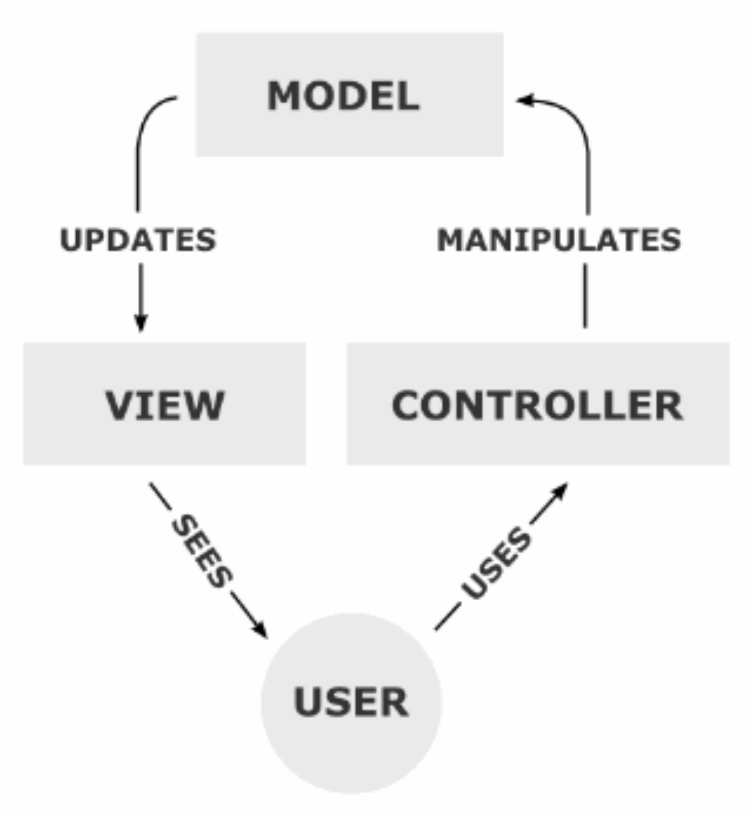
1. An event-driven program has no perceived stopping point.
2. The traditional read-eval-print loop does not explicitly appear.
3. An application processes an input and exits.

Model-View-Controller (MVC)

Model: the object being implemented. Ex: game, calculator.

Controller: input mechanisms. Ex: buttons, menus, combo boxes.

View: output.



Ex: Tic-Tac-Toe Model

Whose turn is it?

State of the board.

Has someone won?

Are there no empty squares?

Java GUI Application

Definition

A ***GUI*** application is a program that runs in its own window and communicates with users using buttons, menus, mouse clicks, etc.

A GUI application often has a paint method, which is invoked whenever the application needs to repaint itself.

Java Swing GUI Components

Text box is an object of class `JTextField`

Radio button is an object of class `JRadioButton`

Applet's display is a frame, a multilayered structure

Content pane is one layer, where applets put output

GUI components can be placed in a frame

Layout manager objects are used to control the placement of components

The Java Event Model

Definition

User interactions with GUI components create events that can be caught by event handlers, called ***event listeners***.

An event generator tells a listener of an event by sending a message

An interface is used to make event-handling methods conform to a standard protocol

A class that implements a listener must implement an interface for the listener

Events in Java

Subclasses of `AWTEvent`

Event sources in Swing are subclasses of `JComponent`

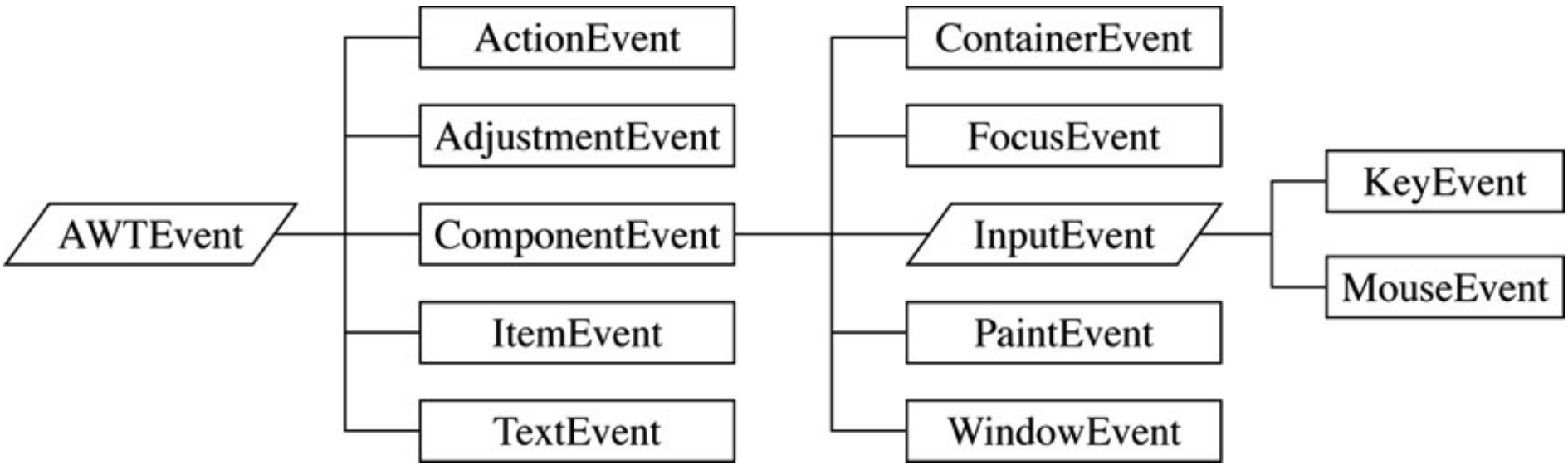
Program must listen for events

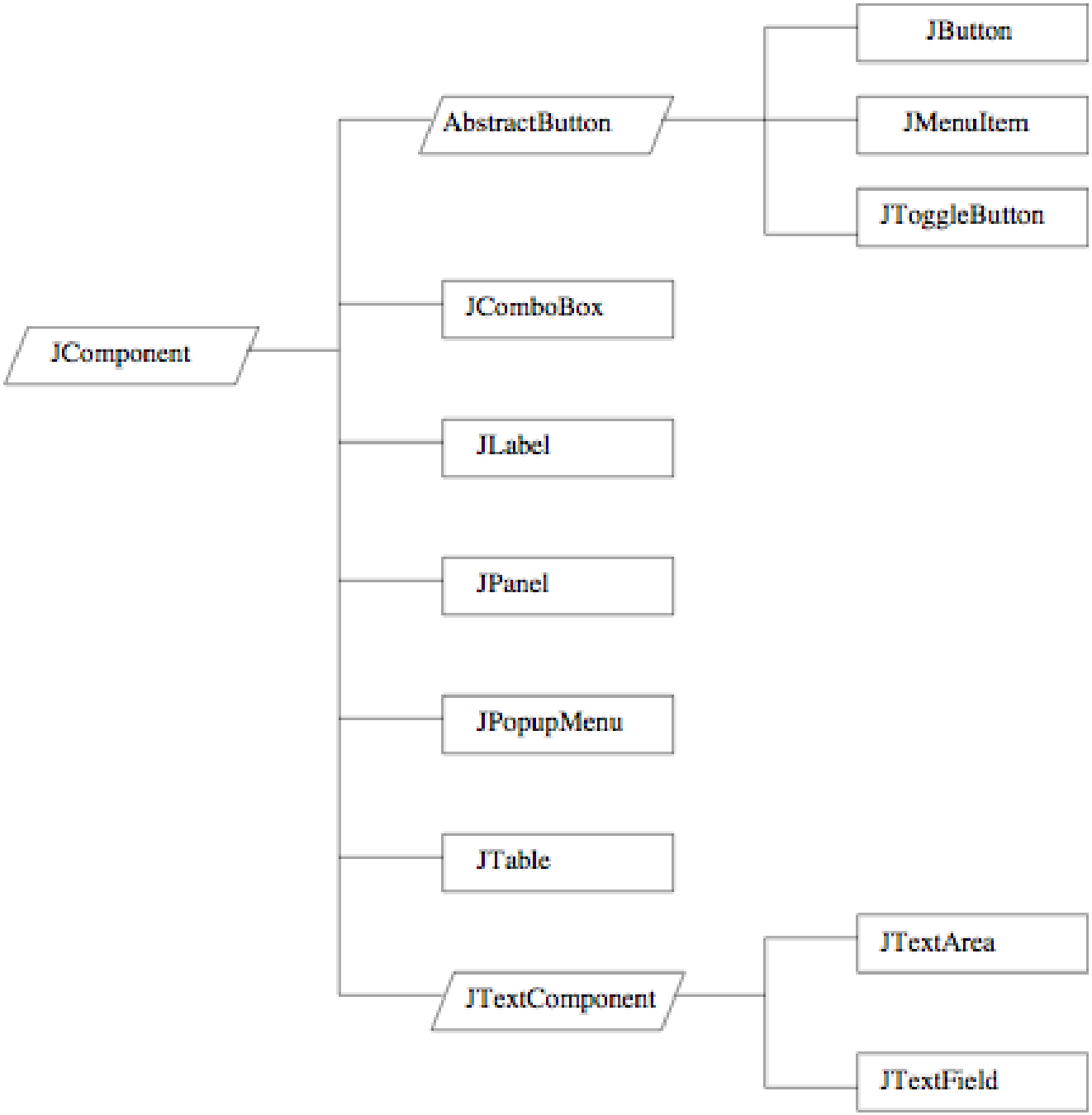
Example

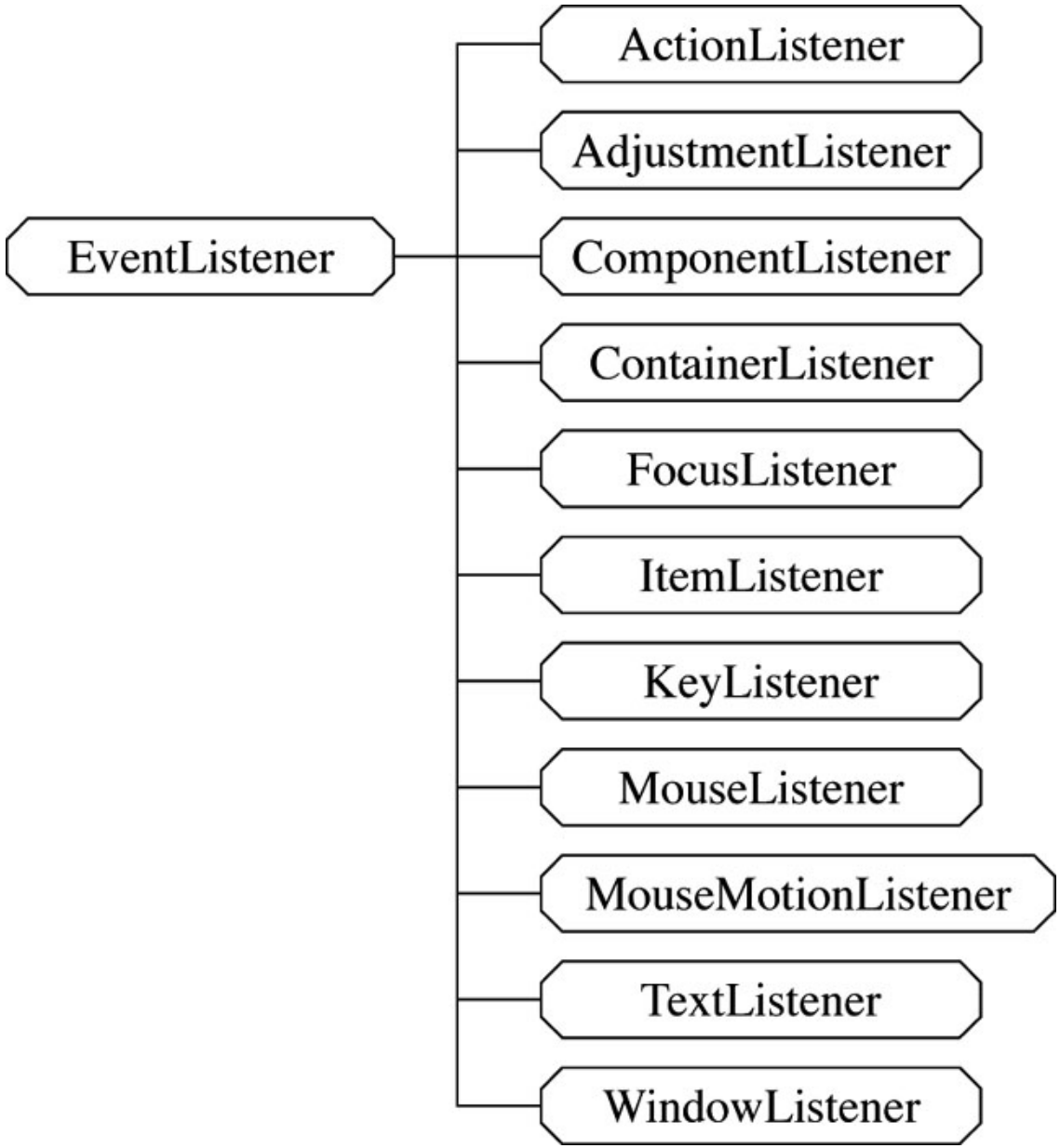
for a `JButton` b:

```
b.addActionListener(listener)
```

Java Class AWTEvent and Its Subclasses




JComponent
AbstractButton
JButton
JMenuItem
JToggleButton
JComboBox
JLabel
JPanel
JPopupMenu
JTable
JTextComponent
JTextArea
JTextField


EventListener
ActionListener
AdjustmentListener
ComponentListener
ContainerListener
FocusListener
ItemListener
KeyListener
MouseListener
MouseMotionListener
TextListener
WindowListener

The Java Event Model

One class of events is `ItemEvent`, which is associated with the event of clicking a checkbox, a radio button, or a list item

The `ItemListener` interface prescribes a method, `itemStateChanged`, which is a handler for `ItemEvent` events

The listener is created with `addItemListener`

Components and their Event Handlers

Widget	Listener	Interface
JButton	ActionListener	actionPerformed(ActionEvent e)
JComboBox	ActionListener	actionPerformed(ActionEvent e)
JLabel	MouseListener	mouseClicked(MouseEvent e) mouseEntered(MouseEvent e) mouseExited(MouseEvent e) mousePressed(MouseEvent e) mouseReleased(MouseEvent e)
	MouseMotionListener	mouseDragged(MouseEvent e) mouseMoved(MouseEvent e)
JTextArea	ActionListener	actionPerformed(ActionEvent e)
JTextField	ActionListener	actionPerformed(ActionEvent e)

GUI Example

```
import javax.swing.JFrame;
public class GUIApp {
    public static void main (String[ ] args) {
        JFrame frame = new JFrame();
        frame.setDefaultCloseOperation(
            JFrame.EXIT_ON_CLOSE);
        MyApp app = new MyApp( ); // JPanel
        frame.getContentPane().add(app);
        frame.show( );
    }
}
```

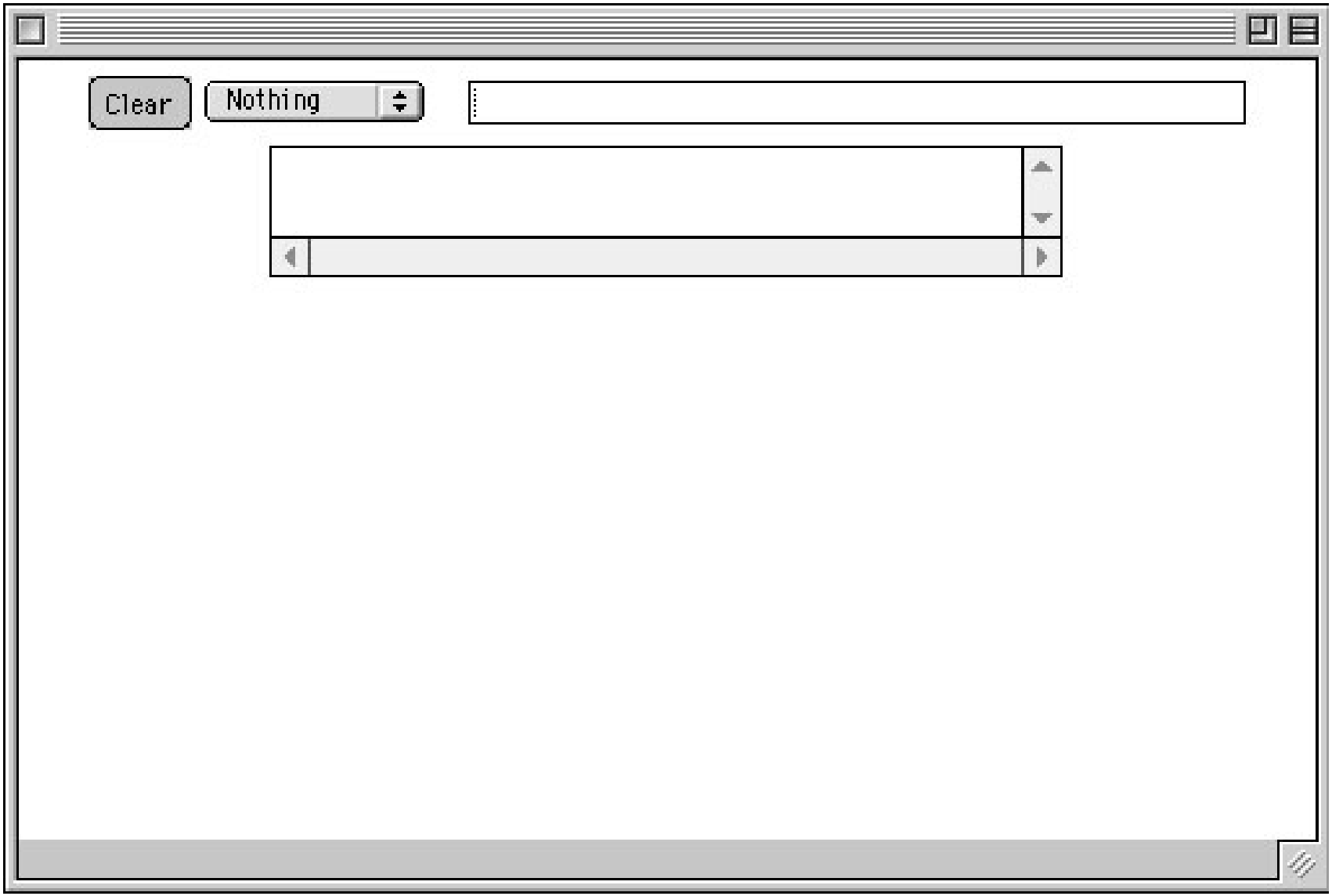
GUI Example using MVC

`combo` : User selects Nothing, Rectangle, Message

`echoArea` : Report events

`typing` : Enter user messages

GUI Design



Instance Variables

```
// first click's x-y coordinates
private int lastX = 0;
private int lastY = 0;
private int clickNumber = 0;
private JComboBox combo;
private String[ ] choices =
      {"Nothing", "Rectangle", "Message"};
private JTextArea echoArea;
private JTextField typing;
```

Initialization Code (1/2)

```
public Skeleton( ) {
    // Set the background color
    // and mouse listener
    setBackground(Color.white);
    addMouseListener(new MouseHandler());

    // Add a button to the Panel.
    JButton clearButton = new JButton("Clear");
    clearButton.setForeground(Color.black);
    clearButton.setBackground(Color.lightGray);
    add(clearButton);
    clearButton.addActionListener(
            new ClearButtonHandler());
```

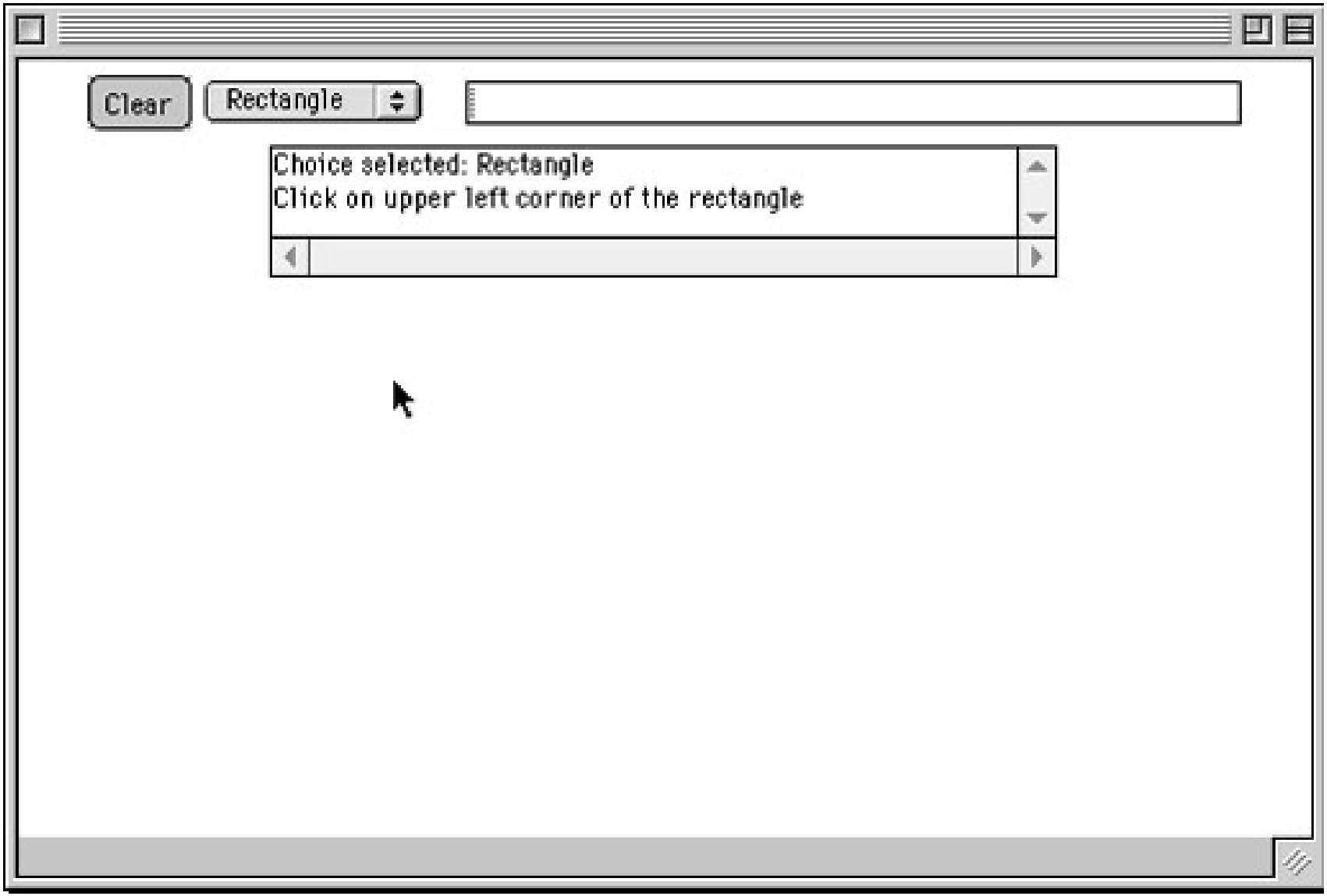
Initialization Code (2/2)

```
        // Create a menu of user combos and add it
        combo = new JComboBox(choices);
        add(combo);
        combo.addActionListener(
            new ComboHandler());
        // Add a TextField and a TextArea
        typing = new JTextField(20);
        add(typing);
        typing.addActionListener(new TextHandler());
        echoArea = new JTextArea(2, 40);
        echoArea.setEditable(false);
        add(echoArea);
    }
```

Action Listener

```
private class ComboHandler
                implements ActionListener {
    public void actionPerformed (ActionEvent e) {
        String c =  (String)
                        (combo.getSelectedItem());
        echoArea.setText("Combo selected: " + c);
        clickNumber = 0;
        if (c.equals("Rectangle"))
            echoArea.append("\nClick to set upper "
                + " left corner of the rectangle");
        else if (c.equals("Message"))
            echoArea.append(
                "\nEnter message in the text area");
    }
}
```

The User Selects Rectangle from the Menu



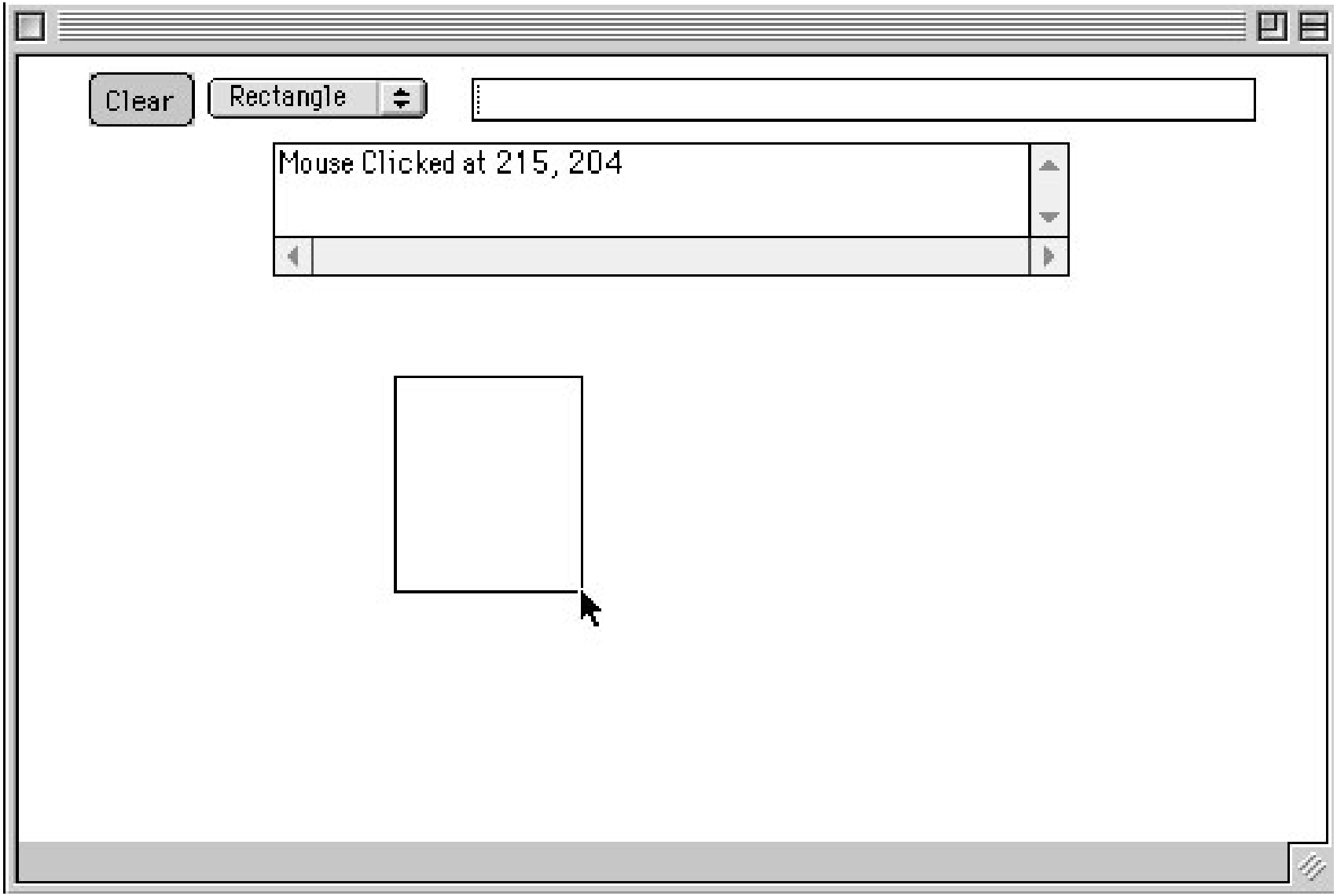
mouseClicked Handler (1/2)

```
private class MouseHandler extends MouseAdapter {
  public void mouseClicked(MouseEvent e) {
     int x = e.getX();
     int y = e.getY();
     echoArea.setText("Mouse Clicked at " +
          e.getX() + ", " + e.getY() + "\n");
     Graphics g = getGraphics();
     if (combo.getSelectedItem().
          equals("Rectangle")) {
        clickNumber = clickNumber + 1;
```

mouseClicked Handler (2/2)

```
    // is it the first click?
    if (clickNumber % 2 == 1) {
        echoArea.append("Click to set lower right"
                + "  corner of the rectangle");
        lastX = x;
        lastY = y;
    }
    // or the second?
    else g.drawRect(lastX, lastY,
            Math.abs(x-lastX), Math.abs(y-lastY));
    }
  else if (combo.getSelectedItem().equals(
                                    "Message"))
        // for a message, display it
        g.drawString(typing.getText(), x, y);
  } // mouseClicked
}
```

Selecting Rectangle Choice and ClickingTwice



`main` method

```
public static void main(String args[])   {
    JFrame frame = new JFrame();
    frame.setDefaultCloseOperation(
            JFrame.EXIT_ON_CLOSE);

    Skeleton panel = new Skeleton( );

    frame.getContentPane().add(panel);
    frame.setSize(500, 500);
    frame.show();
}
```

Event Handling in C#

Event handling in C# (and the other .NET languages) is similar to that in Java

.NET has two approaches, Windows Forms and Windows Presentation Foundation-we cover only the former (which is the original approach)

An application subclasses the `Form` predefined class (defined in `System.Windows.Forms`)

There is no need to create a frame or panel in which to place the GUI components

`Label` objects are used to place text in the window

Radio buttons are objects of the `RadioButton` class

Event Handling in C#

Components are positioned by assigning a new `Point` object to the Location property of the component

```
private RadioButton plain = new RadioButton();
plain.Location = new Point(100, 300);
plain.Text = ``Plain'';
controls.Add(plain);
```

All C# event handlers have the same protocol, the return type is void and the two parameters are of types object and `EventArgs`

Event Handling in C#

An event handler can have any name

A radio button is tested with the Boolean Checked property of the button

```
private void rb_CheckedChanged (object o,
                                EventArgs e) {
    if (plain.Checked)
    ...
}
```

To register an event, a new `EventHandler` object must be created and added to the predefined delegate for the event.

Event Handling in C#

When a radio button changes from unchecked to checked, the `CheckedChanged` event is raised

The associated delegate is referenced by the name of the event

If the handler was named `rb_CheckedChanged`, we could register it on the radio button named plain with:

```
plain.CheckedChanged +=
      new EventHandler (rb_CheckedChanged);
```

Summary

An event is a notification that something has occurred that requires handling by an event handler

Java event handling is defined on the Swing components

C# event handling is the .NET model, which is similar to the Java model