GUI Programming

- Reading: Savitch, Chapter 14

Components, Containers, and Layout Managers in Swing

- Either AWT or Swing can be used in GUI programming.

- The Swing packages are built based on AWT. Consequently, Swing inherits from AWT.

- Generally, if a component or a contains is defined in AWT as ###, then its counterpart in Swing is J###. For example

  JFrame, JButton, JPanel, JTextArea, JTextField, JMenuBar, JMenu and JMenuItem.

  Those classes maintain the major functionality of their counterparts' in AWT, but also provide extra functions.

- Swing also has interfaces and classes which have no counterparts in AWT. For example,

  MenuListener, MouseInputAdapter, JComboBox

JButton, JTextField, JLabel and JFrame

The following example demonstrates how frames, buttons, text fields and labels are built by using Swing.

Example

```
//ButtonApplet.java
import java.applet.Applet;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
/* Allow the user to move a rectangle by specifying
   the x- and y-position. */
public class ButtonApplet extends Applet {
   private Rectangle box;
   private static final int BOX_X = 100;
   private static final int BOX_Y = 100;
   private static final int BOX_WIDTH = 20;
   private static final int BOX_HEIGHT = 30;
```

```
public ButtonApplet() {
    box = new Rectangle(BOX_X, BOX_Y,
      BOX_WIDTH, BOX_HEIGHT);

    // the text fields for entering the x- and y-coordinates
    final JTextField xField = new JTextField(5);
    final JTextField yField = new JTextField(5);

    // the button to move the rectangle
    JButton moveButton = new JButton("Move");
```

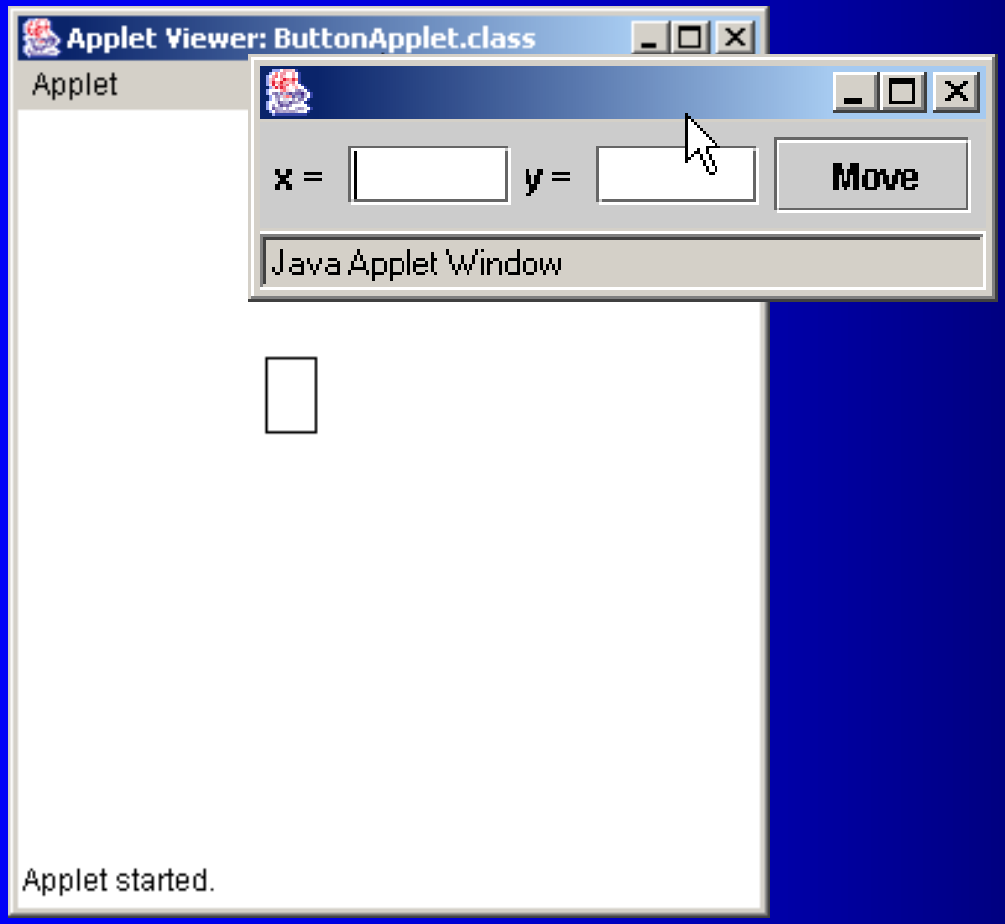
```
class MoveButtonListener implements ActionListener {
      public void actionPerformed(ActionEvent event) {
         int x = Integer.parseInt(xField.getText());
         int y = Integer.parseInt(yField.getText());
         box.setLocation(x, y);
         repaint();
      }
   }

   ActionListener listener = new MoveButtonListener();
   moveButton.addActionListener(listener);
```

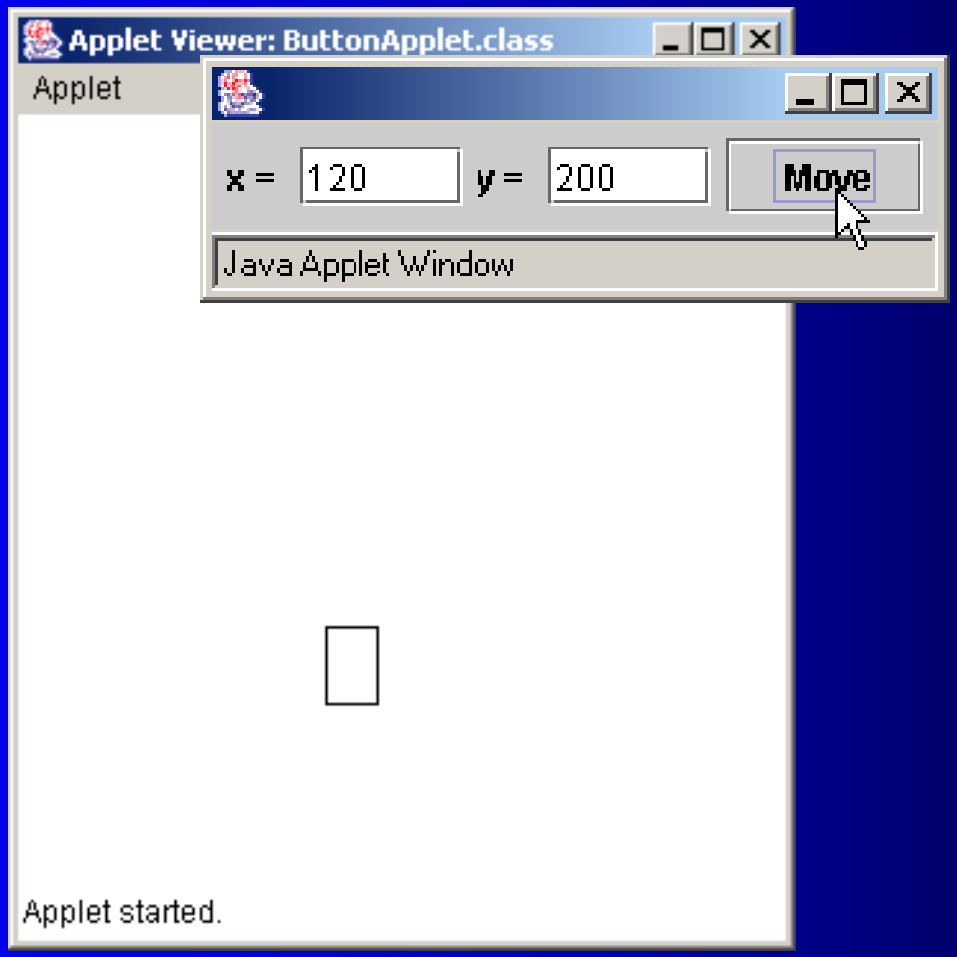
```
// the labels for labelling the text fields
 JLabel xLabel = new JLabel("x = ");
 JLabel yLabel = new JLabel("y = ");

 // the panel for holding the user interface components
 JPanel panel = new JPanel();

 panel.add(xLabel);
 panel.add(xField);
 panel.add(yLabel);
 panel.add(yField);
 panel.add(moveButton);
```

```
        // the frame for holding the component panel
        JFrame frame = new JFrame();
        frame.setContentPane(panel);
        frame.pack();
        frame.show();
    }

    public void paintComponent(Graphics g) {
        super.paintComponent(g);
        Graphics2D g2 = (Graphics2D)g;
        g2.draw(box);
    }
}
```

Program execution

appletviewer ButtonApplet.html




Applet Viewer: ButtonApplet.class
Applet
x = 120
y = 200
Move
Java Applet Window
Applet started.

Some notes

- JFrame is a descendent class of Frame, and therefore inherits methods from Frame.
- Unlike AWT, where we use add() to add components into a frame, in Swing, we use setContentPane().

  For example:

```
JFrame frame = new JFrame();
frame.setContentPane(panel);
```

JMenuItem, JMenu and JMenuBar

Example

```
//MemoGUI.java:  Adapted example from Savitch
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class MemoGUI extends JFrame implements
    ActionListener {
```

```
public static final int WIDTH = 600;
public static final int HEIGHT = 300;
public static final int X = 100;
public static final int Y = 80;


public static final int LINES = 10;
public static final int CHAR_PER_LINE = 40;

private JTextArea theText;
private String memo1 = "No Memo 1.";
private String memo2 = "No Memo 2.";

public MemoGUI( ) {
    setSize(WIDTH, HEIGHT);
    setLocation(X,Y);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setTitle("Memo Saver");
```

```
Container contentPane = getContentPane( );
contentPane.setLayout(new BorderLayout( ));

JMenu memoMenu = new JMenu("Memos");
JMenuItem m;

m = new JMenuItem("Save Memo 1");
m.addActionListener(this);
memoMenu.add(m);

m = new JMenuItem("Save Memo 2");
m.addActionListener(this);
memoMenu.add(m);

m = new JMenuItem("Get Memo 1");
m.addActionListener(this);
memoMenu.add(m);
```

```
m = new JMenuItem("Get Memo 2");
m.addActionListener(this);
memoMenu.add(m);

m = new JMenuItem("Clear");
m.addActionListener(this);
memoMenu.add(m);


m = new JMenuItem("Exit");
m.addActionListener(this);
memoMenu.add(m);
```

```
        JMenuBar mBar = new JMenuBar( );
        mBar.add(memoMenu);
        setJMenuBar(mBar);

        JPanel textPanel  = new JPanel( );
        textPanel.setBackground(Color.BLUE);
        theText = new JTextArea(LINES,  CHAR_PER_LINE);
        theText.setBackground(Color.WHITE);
        textPanel.add(theText);
        contentPane.add(textPanel,  BorderLayout.CENTER);
    }
```

```
public void actionPerformed(ActionEvent e) {
        String actionCommand = e.getActionCommand( );
        if (actionCommand.equals("Save Memo 1"))
            memo1 = theText.getText( );
        else if (actionCommand.equals("Save Memo 2"))
            memo2 = theText.getText( );
        else if (actionCommand.equals("Clear"))
            theText.setText("");
        else if (actionCommand.equals("Get Memo 1"))
            theText.setText(memo1);
        else if (actionCommand.equals("Get Memo 2"))
            theText.setText(memo2);
        else if (actionCommand.equals("Exit"))
            System.exit(0);
        else theText.setText("Error in memo interface");
    }
```

```
    public static void main(String [ ] args) {
        MemoGUI gui = new MemoGUI( );
        gui.setVisible(true);
    }
}
```

The way we use JMenuItem, JMenu and JMenuBar is the same as we use MenuItem, Menu and MenuBar.

A JFrame knows how to respond when a user signals that they would like to close a window (such as when they press the **X** button).

For this reason when creating a Swing application we can replace the:

```
addWindowListener(new WindowDestroyer())
```

command (and the corresponding WindowDestroyer class definition) with the single line of code:

```
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE)
```