AWT CLASSES

Prepared by

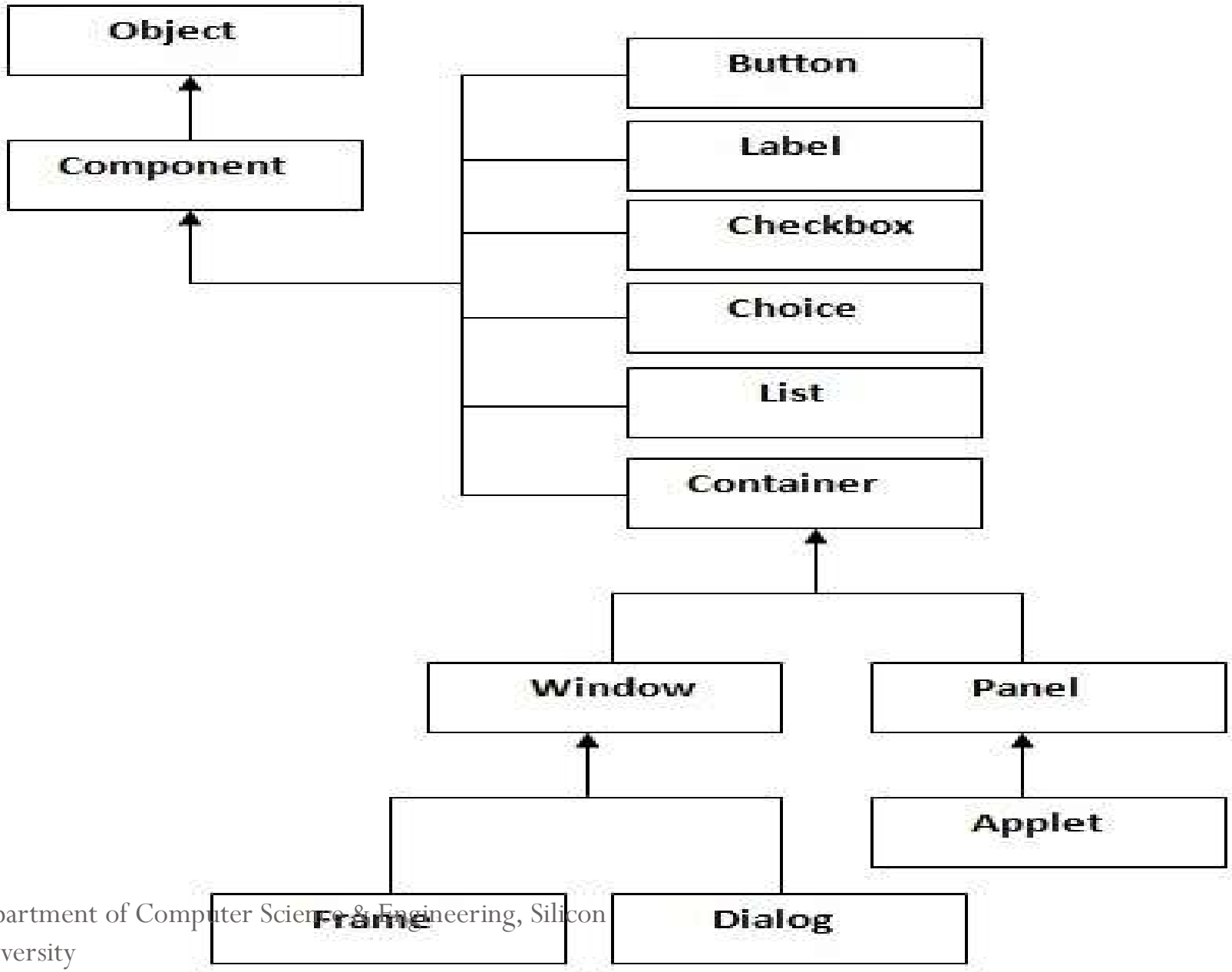
Dr. Rajesh Kumar Ojha
Asst. Prof., CSE, Silicon University

Difference between AWT and Swing

There are many differences between java awt and swing that are given below.

No.	Java AWT	Java Swing
1)	AWT components are **platform-dependent**.	Java swing components are **platform-independent**.
2)	AWT components are **heavyweight**.	Swing components are **lightweight**.
3)	AWT **doesn't support pluggable look and feel**.	Swing **supports pluggable look and feel**.
4)	AWT provides **less components** than Swing.	Swing provides **more powerful components** such as tables, lists, scrollpanes, colorchooser, tabbedpane etc.
5)	AWT **doesn't follows MVC**(Model View Controller) where model represents data, view represents presentation and controller acts as an interface between model and view.	Swing **follows MVC**.

AWT HIERARCHY



Component

- Component is an abstract class that encapsulates all of the attributes of a visual component.

- It defines over a hundred public methods that are responsible for managing events, such as mouse and keyboard input, positioning and sizing the window, and repainting.

Container

- A container is responsible for laying out (that is, positioning) any components that it contains. It does this through the use of various layout managers
- The Container is a component in AWT that can contain another components like buttons, textfields, labels etc. The classes that extends Container class are known as container such as Frame, Dialog and Panel.

Panel

- Panel class is a concrete subclass of Container.
- Panel is the superclass for Applet.
- Panel is a window that does not contain a title bar, menu bar, or border. This is why you don't see these items when an applet is run inside a browser.

Window

- Window class creates a top-level window.
- It sits directly on the desktop.
- The window is the container that have no borders and menu bars. You must use frame, dialog or another window for creating a window.
- We won't create Window objects directly. Instead, you will use a subclass of Window called Frame.

Frame

- It is a subclass of Window and has a title bar, menu bar, borders, and resizing corners.

- When a Frame window is created by a program rather than an applet, a normal window is created.

Canvas

- Canvas encapsulates a blank window upon which you can draw.

Useful Methods of Component Class

Method	Description
public void add(Component c)	inserts a component on this component.
public void setSize(int width,int height)	sets the size (width and height) of the component.
public void setLayout(LayoutManager m)	defines the layout manager for the component.
public void setVisible(boolean status)	changes the visibility of the component, by default false.

- To create simple awt example, you need a frame. There are two ways to create a frame in AWT.
  - By extending Frame class (inheritance)
  - By creating the object of Frame class (association)

Frame Windows

- Frame's constructors:

```
Frame( )
Frame(String title)
```

- setSize( ) method is used to set the dimensions of the window.

```
void setSize(int newWidth, int newHeight)
void setSize(Dimension newSize)
```

- getSize( ) method is used to obtain the current size of a window.

```
Dimension getSize( )
```

- After a frame window has been created, it will not be visible until you call setVisible( ).

  void setVisible(boolean *visibleFlag*)

- You can change the title in a frame window using setTitle( )

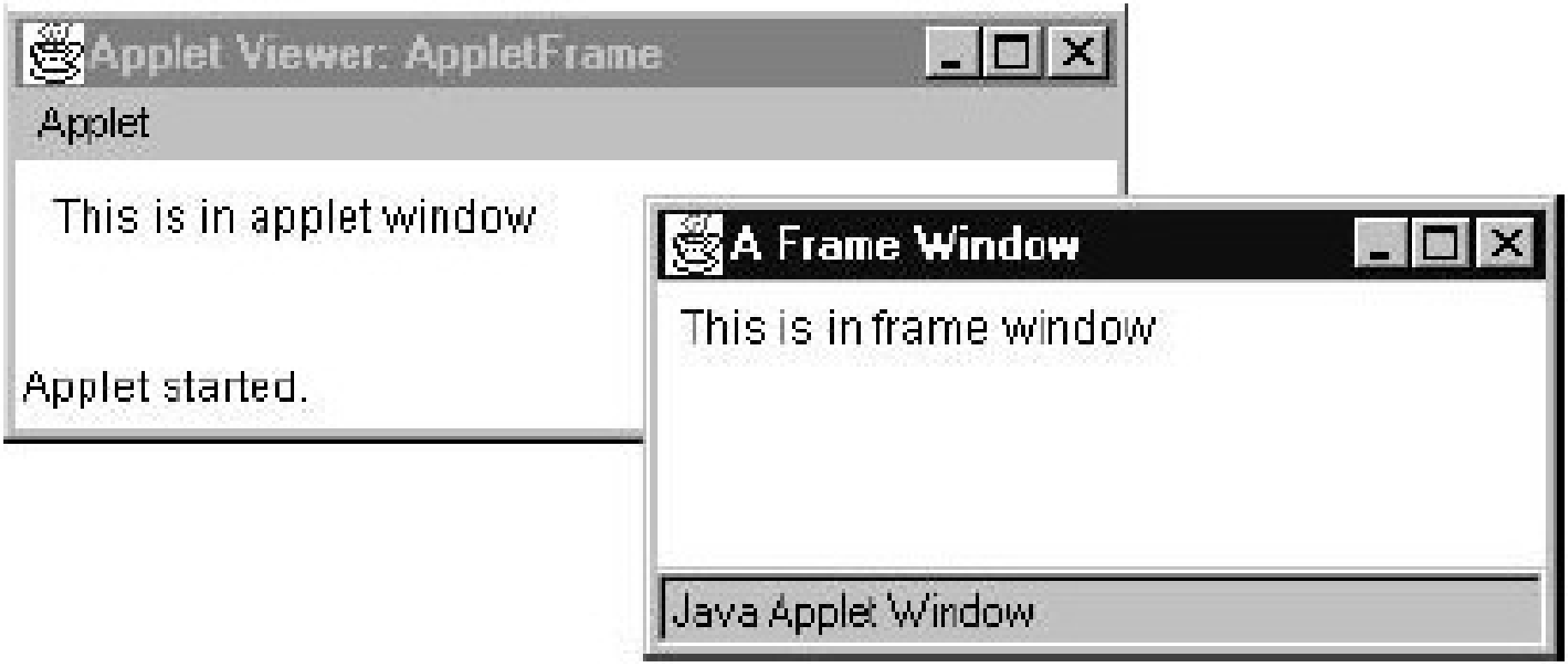
  void setTitle(String *newTitle*)

Creating a Frame Window in an Applet

```
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
/*
  <applet code="AppletFrame" width=300 height=50>
  </applet>
*/

// Create a subclass of Frame.
class SampleFrame extends Frame {
  SampleFrame(String title) {
    super(title);
    // create an object to handle window events
    MyWindowAdapter adapter = new MyWindowAdapter(this);
    // register it to receive those events
```

```
        addWindowListener(adapter);
    }
    public void paint(Graphics g) {
        g.drawString("This is in frame window", 10, 40);
    }
}

class MyWindowAdapter extends WindowAdapter {
    SampleFrame sampleFrame;
    public MyWindowAdapter(SampleFrame sampleFrame) {
        this.sampleFrame = sampleFrame;
    }
    public void windowClosing(WindowEvent we) {
        sampleFrame.setVisible(false);
    }
}
```

```
// Create frame window.
public class AppletFrame extends Applet {
  Frame f;
  public void init() {
    f = new SampleFrame("A Frame Window");

    f.setSize(250, 250);
    f.setVisible(true);
  }
  public void start() {
    f.setVisible(true);
  }
  public void stop() {
    f.setVisible(false);
  }
  public void paint(Graphics g) {
    g.drawString("This is in applet window", 10, 20);
  }
}
```


Applet Viewer: AppletFrame
Applet
This is in applet window
Applet started.
A Frame Window
This is in frame window
Java Applet Window

Adapter Classes

- Adapter class provides an empty implementation of all methods in an event listener interface.

- Adapter classes are useful when you want to receive and process only some of the events that are handled by a particular event listener interface.

Adapter Class	Listener Interface
ComponentAdapter	ComponentListener
ContainerAdapter	ContainerListener
FocusAdapter	FocusListener
KeyAdapter	KeyListener
MouseAdapter	MouseListener
MouseMotionAdapter	MouseMotionListener
WindowAdapter	WindowListener

```
// Handle mouse events in both child and applet windows.
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
/*
  <applet code="WindowEvents" width=300 height=50>
  </applet>
*/

// Create a subclass of Frame.
class SampleFrame extends Frame
  implements MouseListener, MouseMotionListener {

  String msg = "";
  int mouseX=10, mouseY=40;
  int movX=0, movY=0;
```

```
SampleFrame(String title) {
  super(title);
  // register this object to receive its own mouse events
  addMouseListener(this);
  addMouseMotionListener(this);
  // create an object to handle window events
  MyWindowAdapter adapter = new MyWindowAdapter(this);
  // register it to receive those events
  addWindowListener(adapter);
}

// Handle mouse clicked.
public void mouseClicked(MouseEvent me) {
}

// Handle mouse entered.
public void mouseEntered(MouseEvent evtObj) {
  // save coordinates
  mouseX = 10;
  mouseY = 54;
  msg = "Mouse just entered child.";
  repaint();
}
```

```
// Handle mouse exited.
public void mouseExited(MouseEvent evtObj) {
  // save coordinates
  mouseX = 10;
  mouseY = 54;
  msg = "Mouse just left child window.";
  repaint();
}

// Handle mouse pressed.
public void mousePressed(MouseEvent me) {
  // save coordinates
  mouseX = me.getX();
  mouseY = me.getY();
  msg = "Down";
  repaint();
}
```

```
public void mouseReleased(MouseEvent me) {
  // save coordinates
  mouseX = me.getX();
  mouseY = me.getY();
  msg = "Up";
  repaint();
}

// Handle mouse dragged.
public void mouseDragged(MouseEvent me) {
  // save coordinates
  mouseX = me.getX();
  mouseY = me.getY();
  movX = me.getX();
  movY = me.getY();
  msg = "*";
  repaint();
}

// Handle mouse moved.
public void mouseMoved(MouseEvent me) {
  // save coordinates
  movX = me.getX();
  movY = me.getY();
  repaint(0, 0, 100, 60);
}
```

```
  public void paint(Graphics g) {
    g.drawString(msg, mouseX, mouseY);
    g.drawString("Mouse at " + movX + ", " + movY, 10, 40);
  }
}

class MyWindowAdapter extends WindowAdapter {
  SampleFrame sampleFrame;
  public MyWindowAdapter(SampleFrame sampleFrame) {
    this.sampleFrame = sampleFrame;
  }
  public void windowClosing(WindowEvent we) {
    sampleFrame.setVisible(false);
  }
}
```

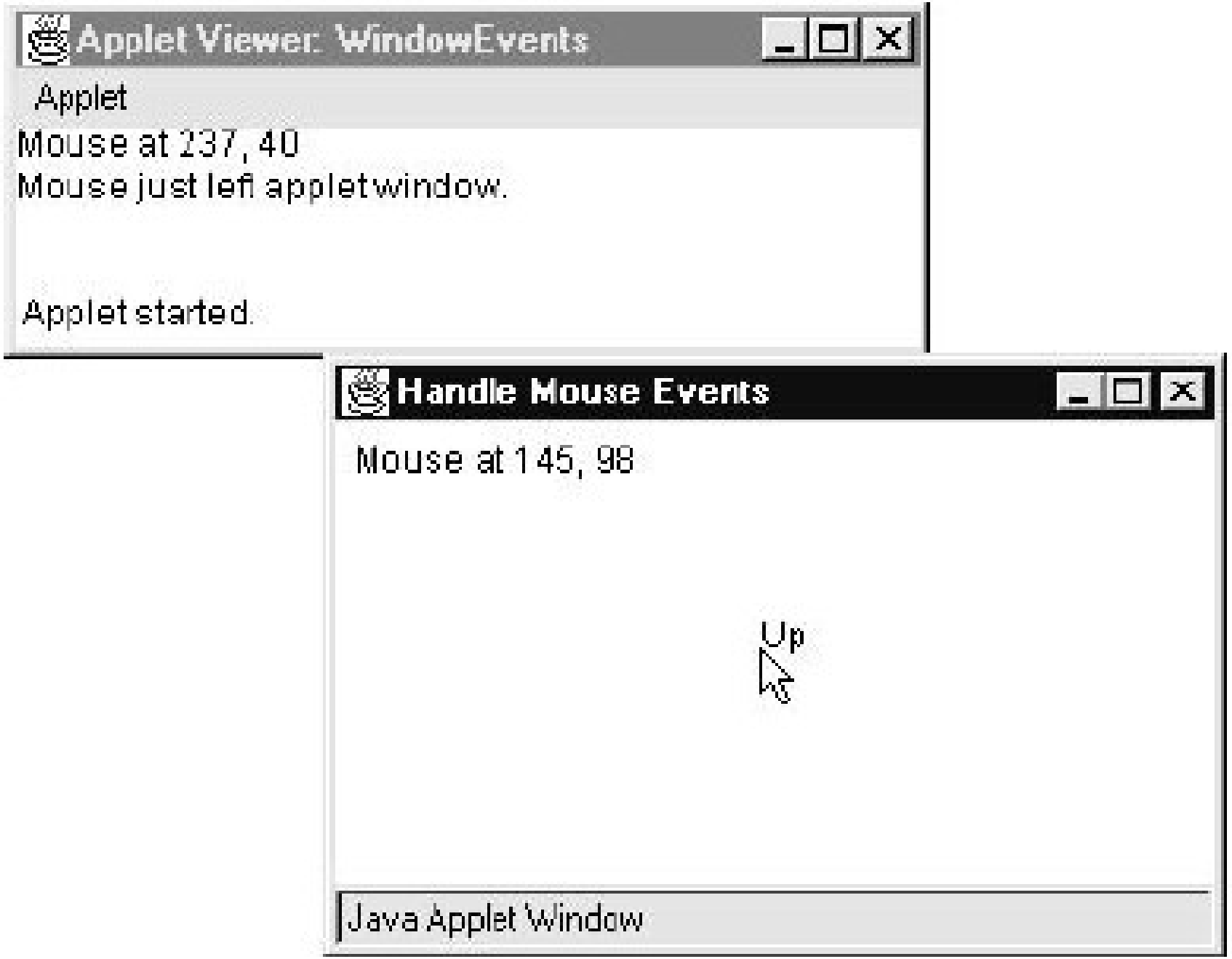
```
// Applet window.
public class WindowEvents extends Applet
  implements MouseListener, MouseMotionListener {

  SampleFrame f;
  String msg = "";
  int mouseX=0, mouseY=10;
  int movX=0, movY=0;

  // Create a frame window.
  public void init() {
    f = new SampleFrame("Handle Mouse Events");
    f.setSize(300, 200);
    f.setVisible(true);

    // register this object to receive its own mouse events
    addMouseListener(this);
    addMouseMotionListener(this);
  }

  // Remove frame window when stopping applet.
  public void stop() {
    f.setVisible(false);
  }
```

```
    // Show frame window when starting applet.
    public void start() {
      f.setVisible(true);
    }

    // Handle mouse clicked.
    public void mouseClicked(MouseEvent me) {
    }

    // Handle mouse entered.
    public void mouseEntered(MouseEvent me) {
      // save coordinates
      mouseX = 0;
      mouseY = 24;
      msg = "Mouse just entered applet window.";
      repaint();
    }
}
```

```
// Handle mouse exited.
public void mouseExited(MouseEvent me) {
  // save coordinates
  mouseX = 0;
  mouseY = 24;
  msg = "Mouse just left applet window.";
  repaint();
}

// Handle button pressed.
public void mousePressed(MouseEvent me) {
  // save coordinates
  mouseX = me.getX();
  mouseY = me.getY();
  msg = "Down";
  repaint();
}
```

```
// Handle button released.
public void mouseReleased(MouseEvent me) {
  // save coordinates
  mouseX = me.getX();
  mouseY = me.getY();
  msg = "Up";
  repaint();
}

// Handle mouse dragged.
public void mouseDragged(MouseEvent me) {
  // save coordinates
  mouseX = me.getX();
  mouseY = me.getY();
  movX = me.getX();
  movY = me.getY();
  msg = "*";
  repaint();
}

// Handle mouse moved.
public void mouseMoved(MouseEvent me) {
```

```
    // save coordinates
    movX = me.getX();
    movY = me.getY();
    repaint(0, 0, 100, 20);
  }

  // Display msg in applet window.
  public void paint(Graphics g) {
    g.drawString(msg, mouseX, mouseY);
    g.drawString("Mouse at " + movX + ", " + movY, 0, 10);
  }
}
```


Applet Viewer: WindowEvents
Applet
Mouse at 237, 40
Mouse just left applet window.
Applet started.
Handle Mouse Events
Mouse at 145, 98
Up
Java Applet Window

Drawing Lines

- Lines are drawn by means of the drawLine( ) method.

void drawLine(int *startX*, int *startY*, int *endX*, int *endY*)

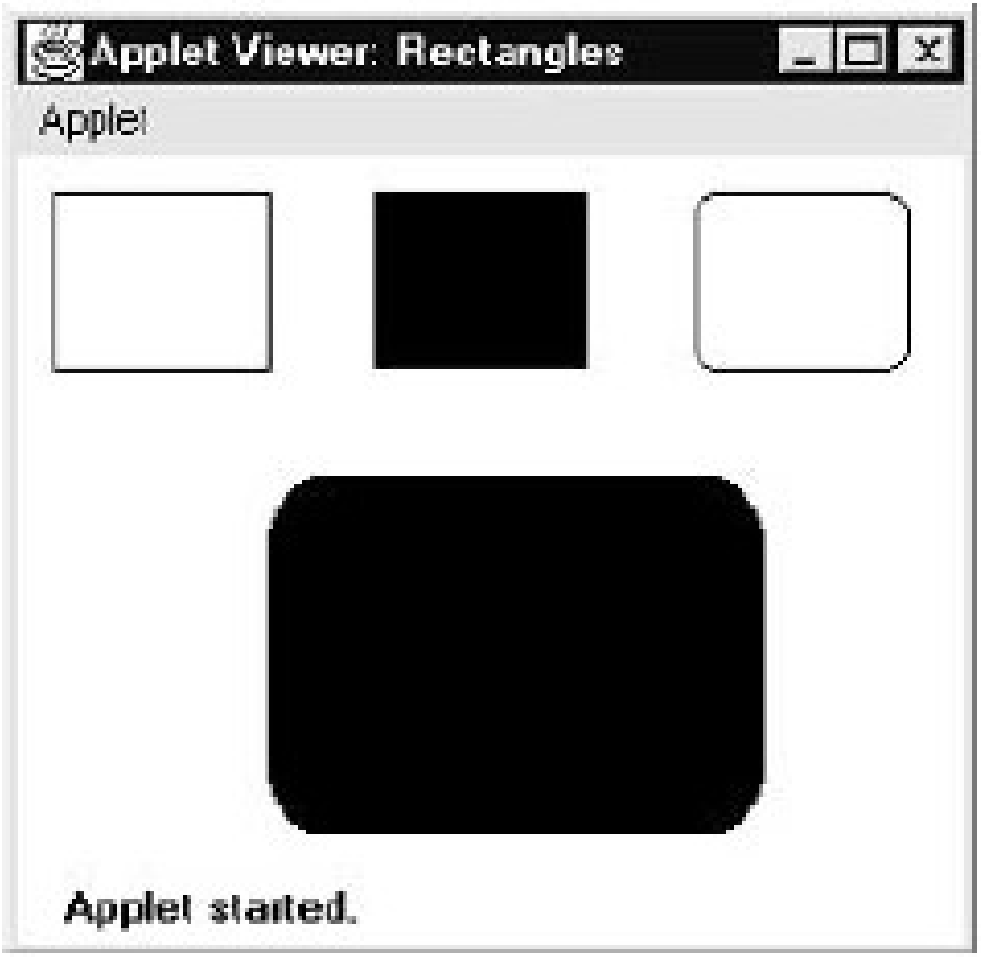
```
import java.awt.*;
import java.applet.*;
/*
<applet code="Lines" width=300 height=200>
</applet>
*/
public class Lines extends Applet {
  public void paint(Graphics g) {
    g.drawLine(0, 0, 100, 100);
    g.drawLine(0, 100, 100, 0);
    g.drawLine(40, 25, 250, 180);
  }
}
```

Drawing Rectangles

- **drawRect( )** and **fillRect( )** methods display an outlined and filled rectangle, respectively.

```
void drawRect(int top, int left, int width, int height)
void fillRect(int top, int left, int width, int height)

void drawRoundRect(int top, int left, int width, int height,
                   int xDiam, int yDiam)

void fillRoundRect(int top, int left, int width, int height,
                   int xDiam, int yDiam)
```

```
import java.awt.*;
import java.applet.*;
/*
<applet code="Rectangles" width=300 height=200>
</applet>
*/

public class Rectangles extends Applet {
  public void paint(Graphics g) {
    g.drawRect(10, 10, 60, 50);
    g.fillRect(100, 10, 60, 50);
    g.drawRoundRect(190, 10, 60, 50, 15, 15);
    g.fillRoundRect(70, 90, 140, 100, 30, 40);
  }
}
```


Applet Viewer: Rectangles
Applet
Applet started.

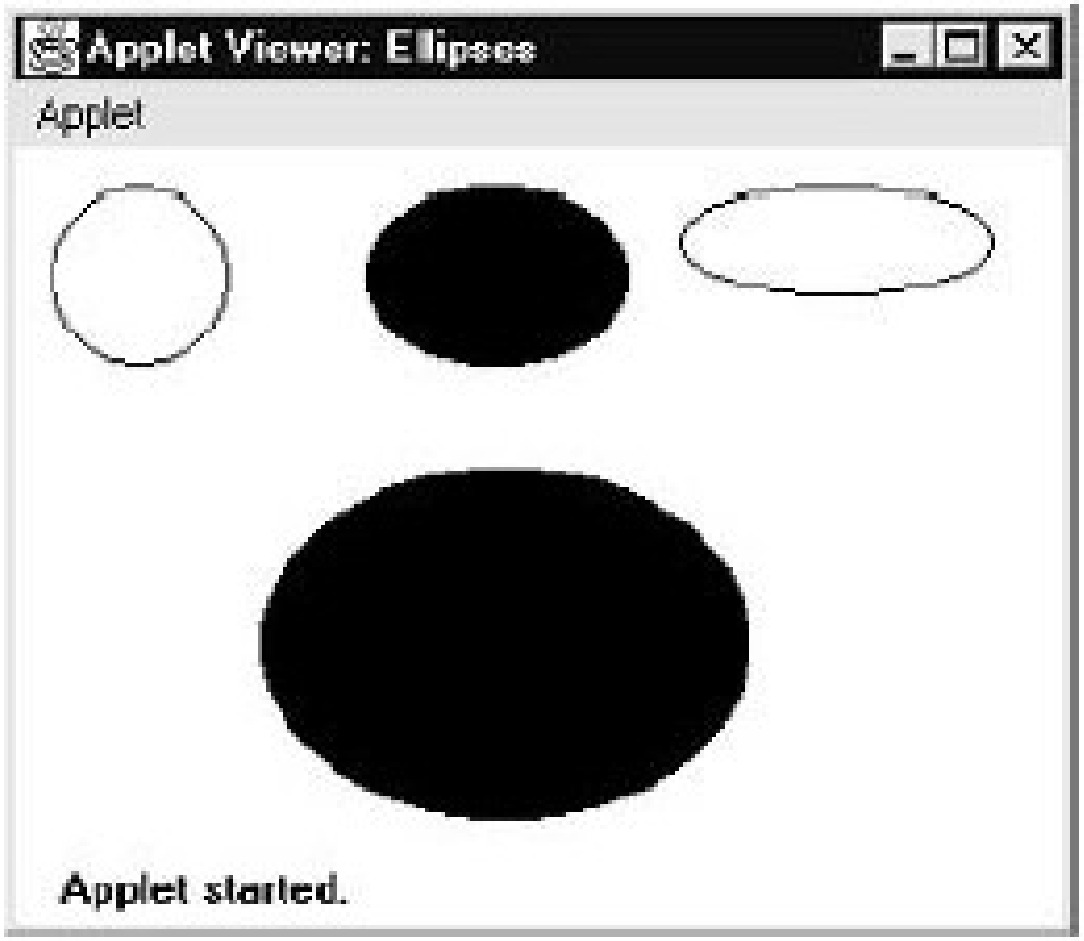
Drawing Ellipses and Circles

- To draw an ellipse, use **drawOval( )**.

- To fill an ellipse, use **fillOval( ).**

void drawOval(int *top*, int *left*, int *width*, int *height*)
void fillOval(int *top*, int *left*, int *width*, int *height*)

```
import java.awt.*;
import java.applet.*;
/*
<applet code="Ellipses" width=300 height=200>
</applet>
*/

public class Ellipses extends Applet {
  public void paint(Graphics g) {
    g.drawOval(10, 10, 50, 50);
    g.fillOval(100, 10, 75, 50);
    g.drawOval(190, 10, 90, 30);
    g.fillOval(70, 90, 140, 100);
  }
}
```

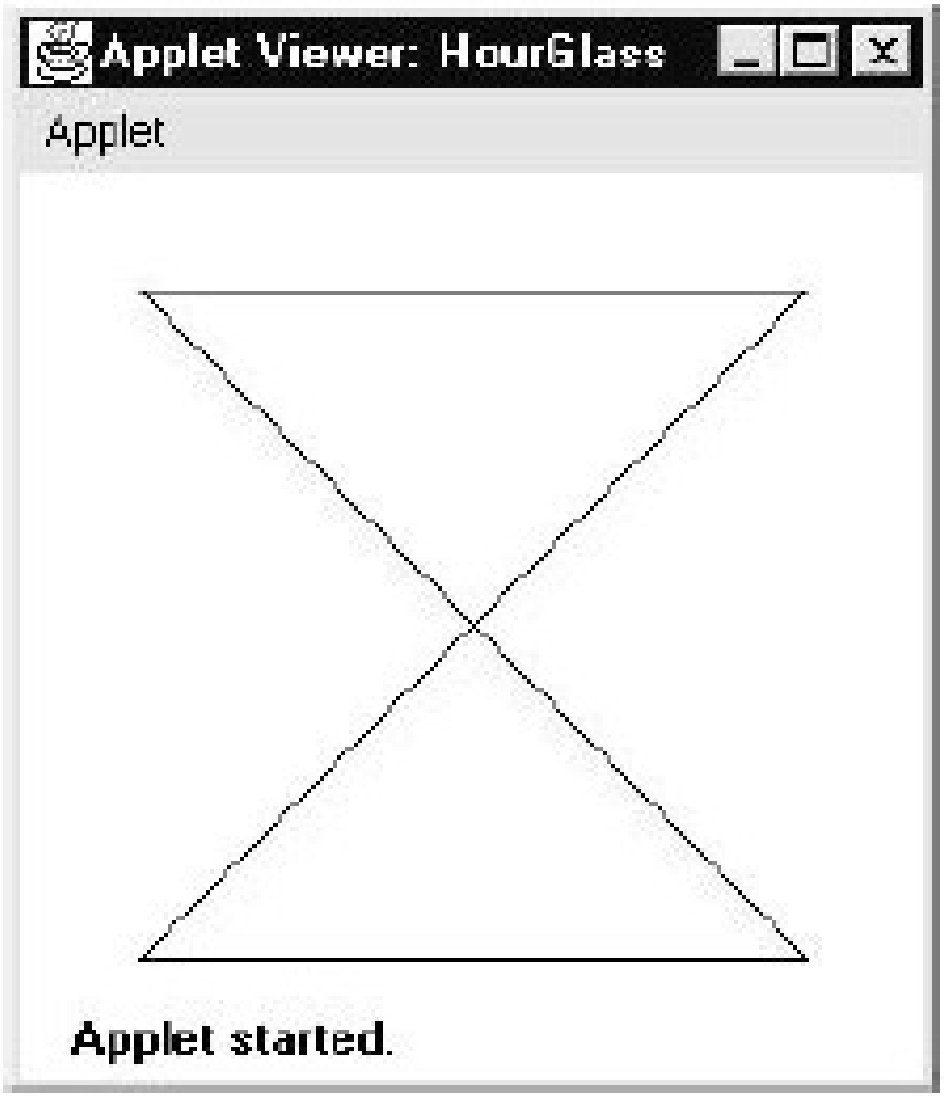

Applet Viewer: Ellipses
Applet
Applet started.

Drawing Polygons

```
void drawPolygon(int x[ ], int y[ ], int numPoints)
void fillPolygon(int x[ ], int y[ ], int numPoints)
```

- Polygon's endpoints are specified by the coordinate pairs contained within the x and y arrays. The number of points defined by x and y is specified by numPoints.

```
import java.awt.*;
import java.applet.*;
/*
<applet code="HourGlass" width=230 height=210>
</applet>
*/

public class HourGlass extends Applet {
  public void paint(Graphics g) {
    int xpoints[] = {30, 200, 30, 200, 30};
    int ypoints[] = {30, 30, 200, 200, 30};
    int num = 5;

    g.drawPolygon(xpoints, ypoints, num);

    }
  }
```


Applet Viewer: HourGlass
Applet
Applet started.

Control Fundamentals

- AWT supports the following types of controls:
  - Labels
  - Push buttons
  - Check boxes
  - Choice lists
  - Lists
  - Scroll bars
  - Text editing

Adding and Removing Controls

- You must first create an instance of the desired control and then add it to a window by calling add( ), which is defined by **Container.**

```
Component add(Component compObj)
```

- To remove a control from a window when the control is no longer needed. To do this, call **remove( ).**

```
void remove(Component obj)
```

Labels

- Labels are passive controls that do not support any interaction with the user.

- A label is an object of type **Label**, and it contains a string, which it displays.

- Constructors:

```
Label( )
Label(String str)
Label(String str, int how)
```

- The v these three constants: **Label.LEFT, Label.RIGHT, or Label.CENTER.**

- To set or change the text in a label by using the **setText( )** method. To obtain the current label use **getText( ).**

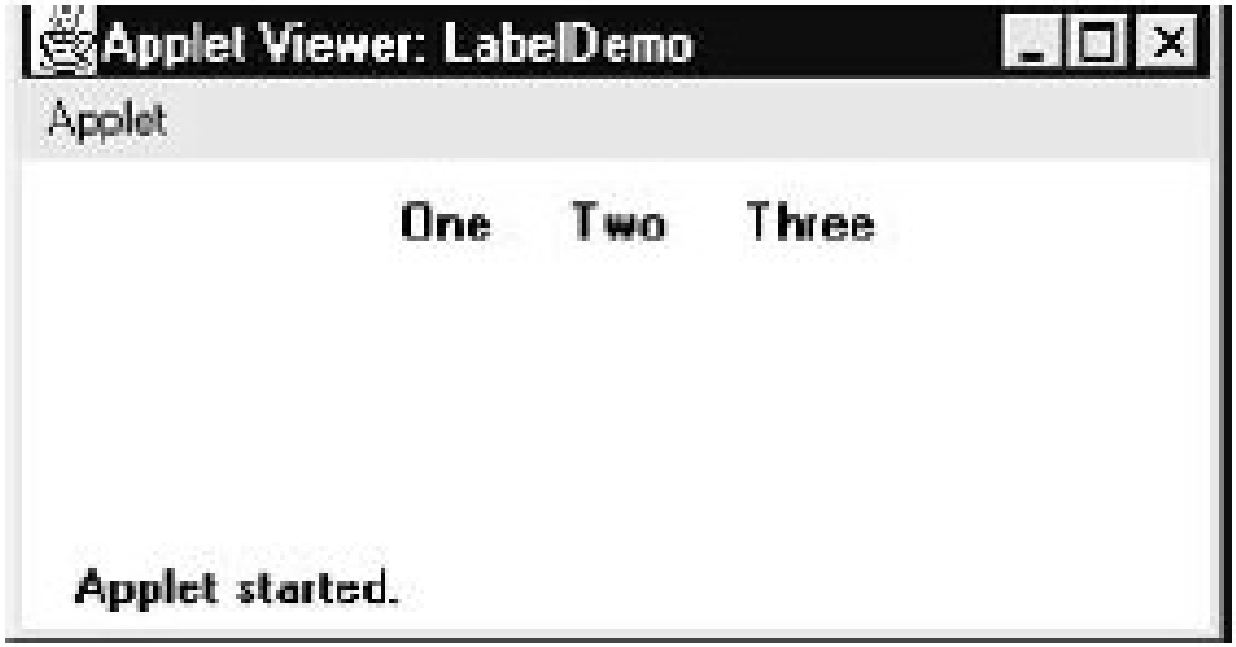
```
void setText(String str)
String getText( )
```

- To set the alignment of the string within the label use **setAlignment( ).** To obtain the current alignment, call **getAlignment( ).**

```
void setAlignment(int how)
int getAlignment( )
```

```
import java.awt.*;
import java.applet.*;
/*
<applet code="LabelDemo" width=300 height=200>
</applet>
*/

public class LabelDemo extends Applet {
  public void init() {
    Label one = new Label("One");
    Label two = new Label("Two");
    Label three = new Label("Three");

    // add labels to applet window
    add(one);
    add(two);
    add(three);
  }
}
```

```
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
/*
  <applet code="ButtonDemo" width=250 height=150>
  </applet>
*/

public class ButtonDemo extends Applet implements ActionListener {
  String msg = "";
  Button yes, no, maybe;

  public void init() {
    yes = new Button("Yes");
    no = new Button("No");
    maybe = new Button("Undecided");
```


Applet Viewer: LabelDemo
Applet
One Two Three
Applet started.

Using Buttons

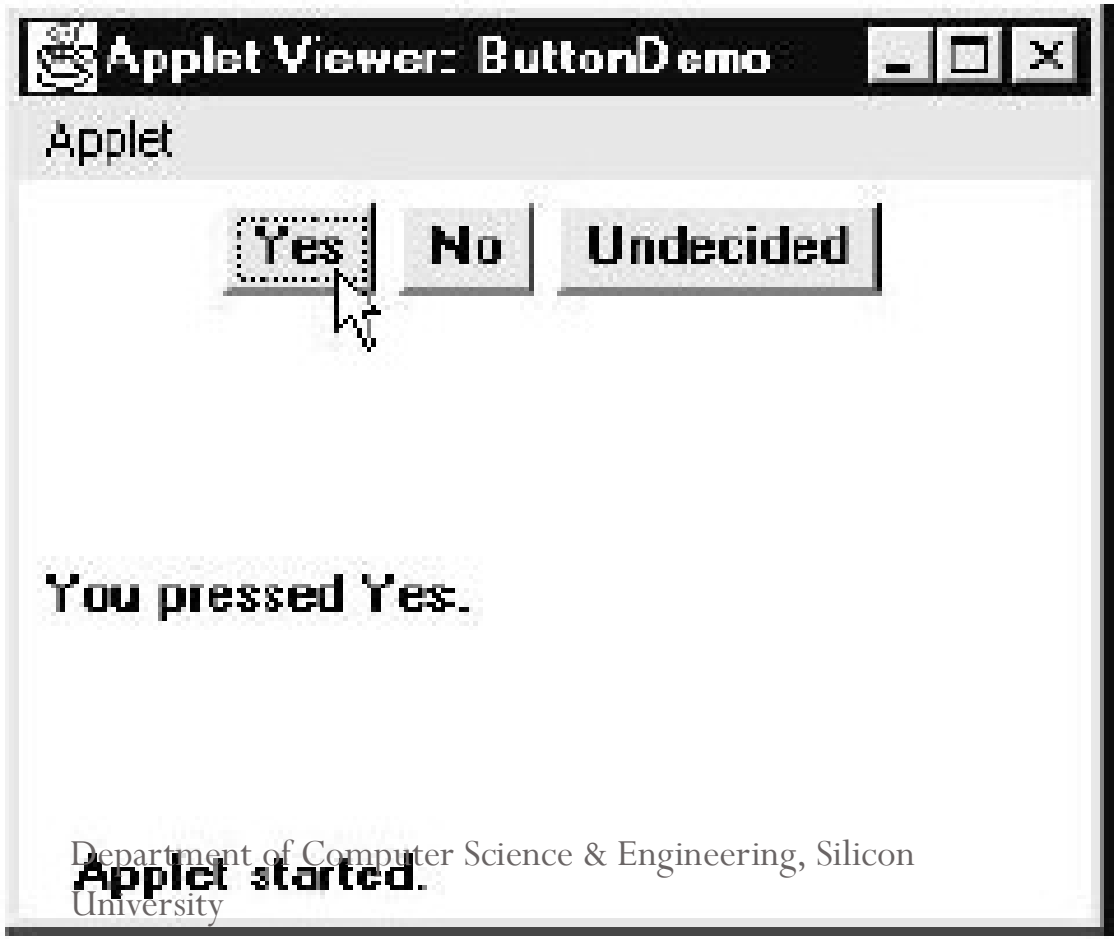
- Push button is a component that contains a label and that generates an event when it is pressed.

- Button defines these two constructors:

```
Button( )
Button(String str)
```

```
    add(yes);
    add(no);
    add(maybe);

    yes.addActionListener(this);
    no.addActionListener(this);
    maybe.addActionListener(this);
  }

  public void actionPerformed(ActionEvent ae) {
    String str = ae.getActionCommand();
    if(str.equals("Yes")) {
      msg = "You pressed Yes.";
    }
    else if(str.equals("No")) {
      msg = "You pressed No.";
    }
    else {
      msg = "You pressed Undecided.";
    }
```

```
    repaint();
  }

  public void paint(Graphics g) {
    g.drawString(msg, 6, 100);
  }
}
```



Check Boxes

- Check box is a control that is used to turn an option on or off.
- It consists of a small box that can either contain a check mark or not.
- Checkbox supports these constructors:

Checkbox( )
Checkbox(String *str*)
Checkbox(String *str*, boolean *on*)
Checkbox(String *str*, boolean *on*, CheckboxGroup *cbGroup*)
Checkbox(String *str*, CheckboxGroup *cbGroup*, boolean *on*)

- To retrieve the current state of a check box, call **getState( ).**
- To set its state, call **setState( ).**
- To obtain the current label associated with a check box call **getLabel( ).**
- To set the label, call **setLabel( ).**

```
boolean getState( )
void setState(boolean on)
String getLabel( )
void setLabel(String str)
```

```
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
/*
  <applet code="CheckboxDemo" width=250 height=200>
  </applet>
*/

public class CheckboxDemo extends Applet implements ItemListener {
  String msg = "";
  Checkbox Win98, winNT, solaris, mac;

  public void init() {
    Win98 = new Checkbox("Windows 98/XP", null, true);
    winNT = new Checkbox("Windows NT/2000");
    solaris = new Checkbox("Solaris");
    mac = new Checkbox("MacOS");
```

```
    add(Win98);
    add(winNT);
    add(solaris);
    add(mac);

    Win98.addItemListener(this);
    winNT.addItemListener(this);
    solaris.addItemListener(this);
    mac.addItemListener(this);
  }

  public void itemStateChanged(ItemEvent ie) {
    repaint();
  }

  // Display current state of the check boxes.
  public void paint(Graphics g) {
```

```
        msg = "Current state: ";
        g.drawString(msg, 6, 80);
        msg = "  Windows 98/XP: " + Win98.getState();
        g.drawString(msg, 6, 100);
        msg = "  Windows NT/2000: " + winNT.getState();
        g.drawString(msg, 6, 120);
        msg = "  Solaris: " + solaris.getState();
        g.drawString(msg, 6, 140);
        msg = "  MacOS: " + mac.getState();
        g.drawString(msg, 6, 160);
    }
}
```


Applet Viewer: CheckboxDemo
Applet
Windows 98/XP
Windows NT/2000
Solaris
MacOS
Current state:
Windows 98/XP: true
Windows NT/2000: false
Solaris: false
MacOS: false
Applet started.

CheckboxGroup

Checkbox getSelectedCheckbox( )
void setSelectedCheckbox(Checkbox *which*)

```
// Demonstrate check box group.
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
/*
  <applet code="CBGroup" width=250 height=200>
  </applet>
*/

public class CBGroup extends Applet implements ItemListener {
  String msg = "";
  Checkbox Win98, winNT, solaris, mac;
  CheckboxGroup cbg;
```

```
public void init() {
   cbg = new CheckboxGroup();
   Win98 = new Checkbox("Windows 98/XP", cbg, true);
   winNT = new Checkbox("Windows NT/2000", cbg, false);
   solaris = new Checkbox("Solaris", cbg, false);
   mac = new Checkbox("MacOS", cbg, false);

   add(Win98);
   add(winNT);
   add(solaris);
   add(mac);

   Win98.addItemListener(this);
```

```
    winNT.addItemListener(this);
    solaris.addItemListener(this);
    mac.addItemListener(this);
  }

  public void itemStateChanged(ItemEvent ie) {
    repaint();
  }

  // Display current state of the check boxes.
  public void paint(Graphics g) {
    msg = "Current selection: ";
    msg += cbg.getSelectedCheckbox().getLabel();
    g.drawString(msg, 6, 100);
  }
}
```


Applet Viewer: CBGroup
Applet
Windows 98/XP
Windows NT/2000
Solaris
MacOS
Current selection: Windows 98/XP
Applet started

Choice Controls

- **Choice** class is used to create a pop-up list of items from which the user may choose.
- Choice control is a form of menu.
- Each item in the list is a string that appears as a left-justified label in the order it is added to the **Choice object.**
- Choice only defines the default constructor.
- To add a selection to the list, call **add( ).**

void add(String *name*)

- To determine which item is currently selected, you may call either **getSelectedItem( )** or **getSelectedIndex( )**.

```
String getSelectedItem( )
int getSelectedIndex( )
```

- **getSelectedIndex( )** returns the index of the item. The first item is at index 0. By default, the first item added to the list is selected.

- To obtain the number of items in the list, call **getItemCount( ).**

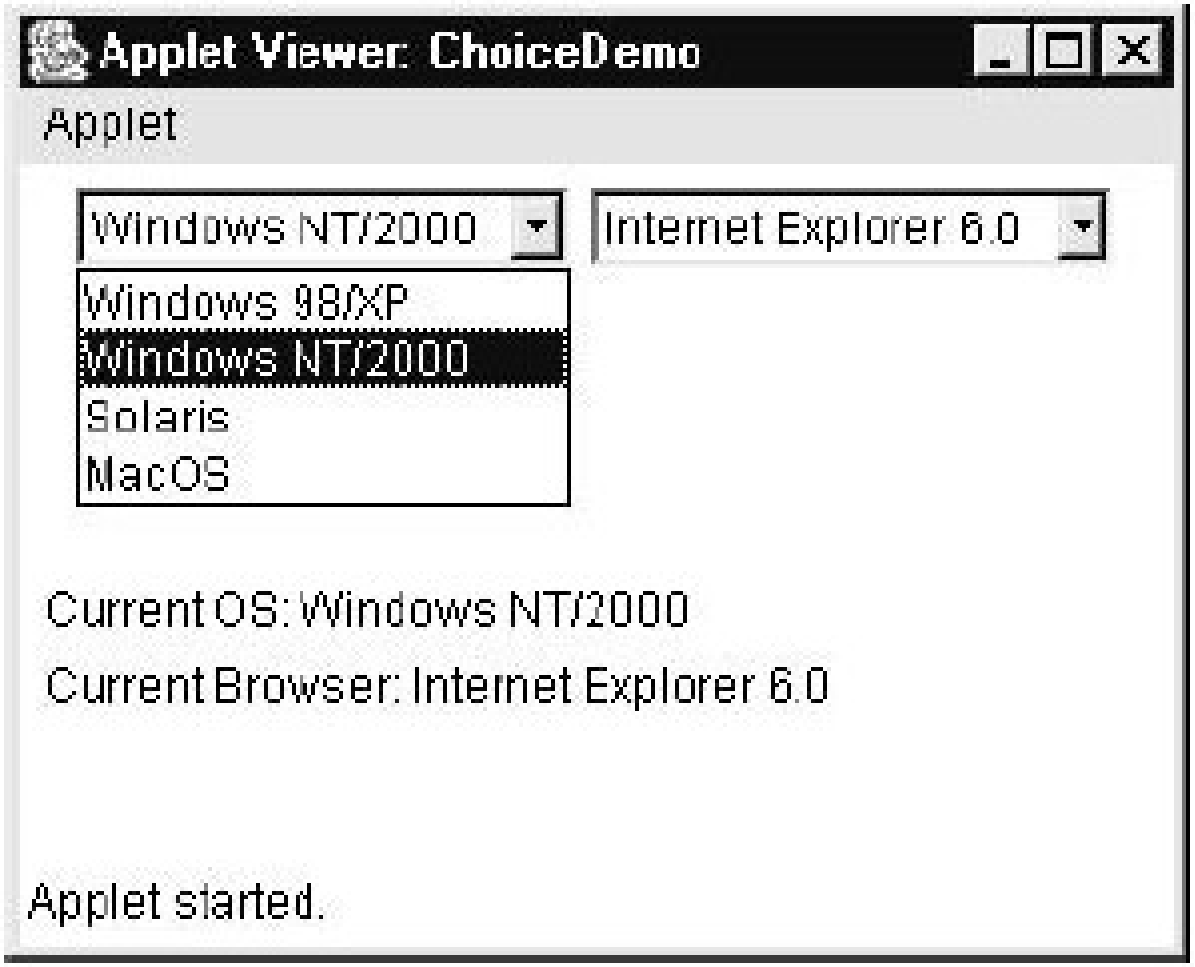
- Set the currently selected item using the **select( )** method with either a zero-based integer index or a string that will match a name in the list.

```
int getItemCount( )
void select(int index)
void select(String name)
```

```
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
/*
  <applet code="ChoiceDemo" width=300 height=180>
  </applet>
*/

public class ChoiceDemo extends Applet implements ItemListener {
  Choice os, browser;
  String msg = "";

  public void init() {
    os = new Choice();
    browser = new Choice();
```

```
// add items to os list
os.add("Windows 98/XP");
os.add("Windows NT/2000");
os.add("Solaris");
os.add("MacOS");

// add items to browser list
browser.add("Netscape 3.x");
browser.add("Netscape 4.x");
browser.add("Netscape 5.x");
browser.add("Netscape 6.x");

browser.add("Internet Explorer 4.0");
browser.add("Internet Explorer 5.0");
browser.add("Internet Explorer 6.0");

browser.add("Lynx 2.4");

browser.select("Netscape 4.x");
```

```
    // add choice lists to window
    add(os);
    add(browser);

    // register to receive item events
    os.addItemListener(this);
    browser.addItemListener(this);
  }

  public void itemStateChanged(ItemEvent ie) {
    repaint();
  }

  // Display current selections.
  public void paint(Graphics g) {
    msg = "Current OS: ";
    msg += os.getSelectedItem();
    g.drawString(msg, 6, 120);
    msg = "Current Browser: ";
    msg += browser.getSelectedItem();
    g.drawString(msg, 6, 140);
  }
}
```


Applet Viewer: ChoiceDemo
Applet
Windows NT/2000
Internet Explorer 6.0
Windows 98/XP
Windows NT/2000
Solaris
MacOS
Current OS: Windows NT/2000
Current Browser: Internet Explorer 6.0
Applet started.

Lists

- List class provides a compact, multiple-choice, scrolling selection list.

- List provides these constructors

```
List( )
List(int numRows)
List(int numRows, boolean multipleSelect)
```

- In the second form, the value of *numRows* specifies the number of entries in the list that will always be visible.

- If multipleSelect is true, then the user may select two or more items at a time.

- To add a selection to the list, call add( ).

```
void add(String name)
void add(String name, int index)
```

- We can determine which item is currently selected by calling either **getSelectedItem( ) or getSelectedIndex( ).**

```
String getSelectedItem( )
int getSelectedIndex( )
String[ ] getSelectedItems( )
int[ ] getSelectedIndexes( )
```

- To obtain the number of items in the list, call **getItemCount( ).**

- We can obtain the name associated with the item at that index by calling **getItem( )**

```
String getItem(int index)
```

```
// Demonstrate Lists.
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
/*
  <applet code="ListDemo" width=300 height=180>
  </applet>
*/

public class ListDemo extends Applet implements ActionListener {
  List os, browser;
  String msg = "";

  public void init() {
    os = new List(4, true);
    browser = new List(4, false);
```

```
// add items to os list
os.add("Windows 98/XP");
os.add("Windows NT/2000");
os.add("Solaris");
os.add("MacOS");

// add items to browser list
browser.add("Netscape 3.x");
browser.add("Netscape 4.x");
browser.add("Netscape 5.x");
browser.add("Netscape 6.x");

browser.add("Internet Explorer 4.0");
browser.add("Internet Explorer 5.0");
browser.add("Internet Explorer 6.0");

browser.add("Lynx 2.4");

browser.select(1);

// add lists to window
add(os);
add(browser);
```

```
    // register to receive action events
    os.addActionListener(this);
    browser.addActionListener(this);
}

public void actionPerformed(ActionEvent ae) {
    repaint();
}

// Display current selections.
public void paint(Graphics g) {
    int idx[];

    msg = "Current OS: ";
    idx = os.getSelectedIndexes();
    for(int i=0; i<idx.length; i++)
      msg += os.getItem(idx[i]) + "   ";
```

```
        g.drawString(msg, 6, 120);
        msg = "Current Browser: ";
        msg += browser.getSelectedItem();
        g.drawString(msg, 6, 140);
    }
}
```

Using a TextField

- TextField class implements a single-line text-entry area.
- TextField is a subclass of TextComponent.
- TextField defines the following constructors:

```
TextField( )
TextField(int numChars)
TextField(String str)
TextField(String str, int numChars)
```

- To obtain the string currently contained in the text field, call getText( ). To set the text, call setText( ).

- Program can obtain the currently selected text by calling getSelectedText( ).

- We can disable the echoing of the characters as they are typed by calling setEchoChar( ).

- We can check a text field to see if it is in this mode with the **echoCharIsSet( )** method.

- We can retrieve the echo character by calling the **getEchoChar( ) method.**

```
void setEchoChar(char ch)
boolean echoCharIsSet( )
char getEchoChar( )
```

```
// Demonstrate text field.
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
/*
<applet code="TextFieldDemo" width=380 height=150>
</applet>
*/

public class TextFieldDemo extends Applet
  implements ActionListener {

  TextField name, pass;

  public void init() {
    Label namep = new Label("Name: ", Label.RIGHT);
    Label passp = new Label("Password: ", Label.RIGHT);
    name = new TextField(12);
    pass = new TextField(8);
    pass.setEchoChar('?');
```

```
    add(namep);
    add(name);
    add(passp);
    add(pass);

    // register to receive action events
    name.addActionListener(this);
    pass.addActionListener(this);
}

// User pressed Enter.
public void actionPerformed(ActionEvent ae) {
    repaint();
}

public void paint(Graphics g) {
    g.drawString("Name: " + name.getText(), 6, 60);
    g.drawString("Selected text in name: "
                 + name.getSelectedText(), 6, 80);
    g.drawString("Password: " + pass.getText(), 6, 100);
}
```


Applet Viewer: TextFieldDemo
Applet
Name:
Herb Schildt
Password:
??????
Name: Herb Schildt
Selected text in name: Schildt
Password: 123test
Applet started.

Using a TextArea

- AWT includes a simple multiline editor called TextArea.

- Constructors for **TextArea:**

```
TextArea( )
TextArea(int numLines, int numChars)
TextArea(String str)
TextArea(String str, int numLines, int numChars)
TextArea(String str, int numLines, int numChars, int sBars)
```

- *numLines* specifies the height, in lines, of the text area, and *numChars* specifies its width, in characters.

- Initial text can be specified by *str*.

- sBars must be one of these values:

SCROLLBARS_BOTH	SCROLLBARS_NONE
SCROLLBARS_HORIZONTAL_ONLY	SCROLLBARS_VERTICAL_ONLY

- TextArea adds the following methods:

```
void append(String str)
void insert(String str, int index)
void replaceRange(String str, int startIndex, int endIndex)
```

- append( ) method appends the string specified by str to the end of the current text.

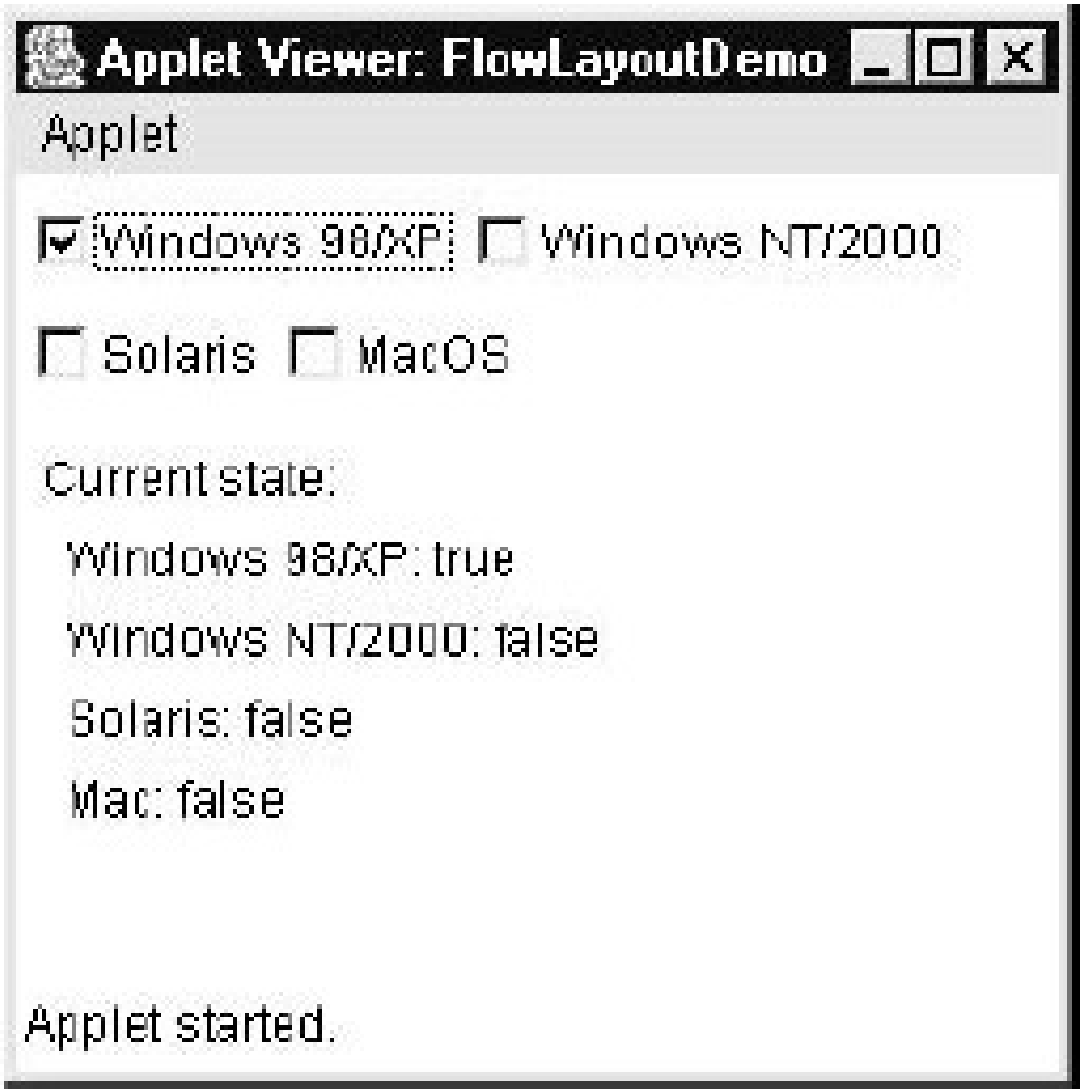
- insert( ) inserts the string passed in str at the specified index. To replace text, call replaceRange( ). It replaces the characters from startIndex to endIndex–1, with the replacement text passed in str.

Layout Managers

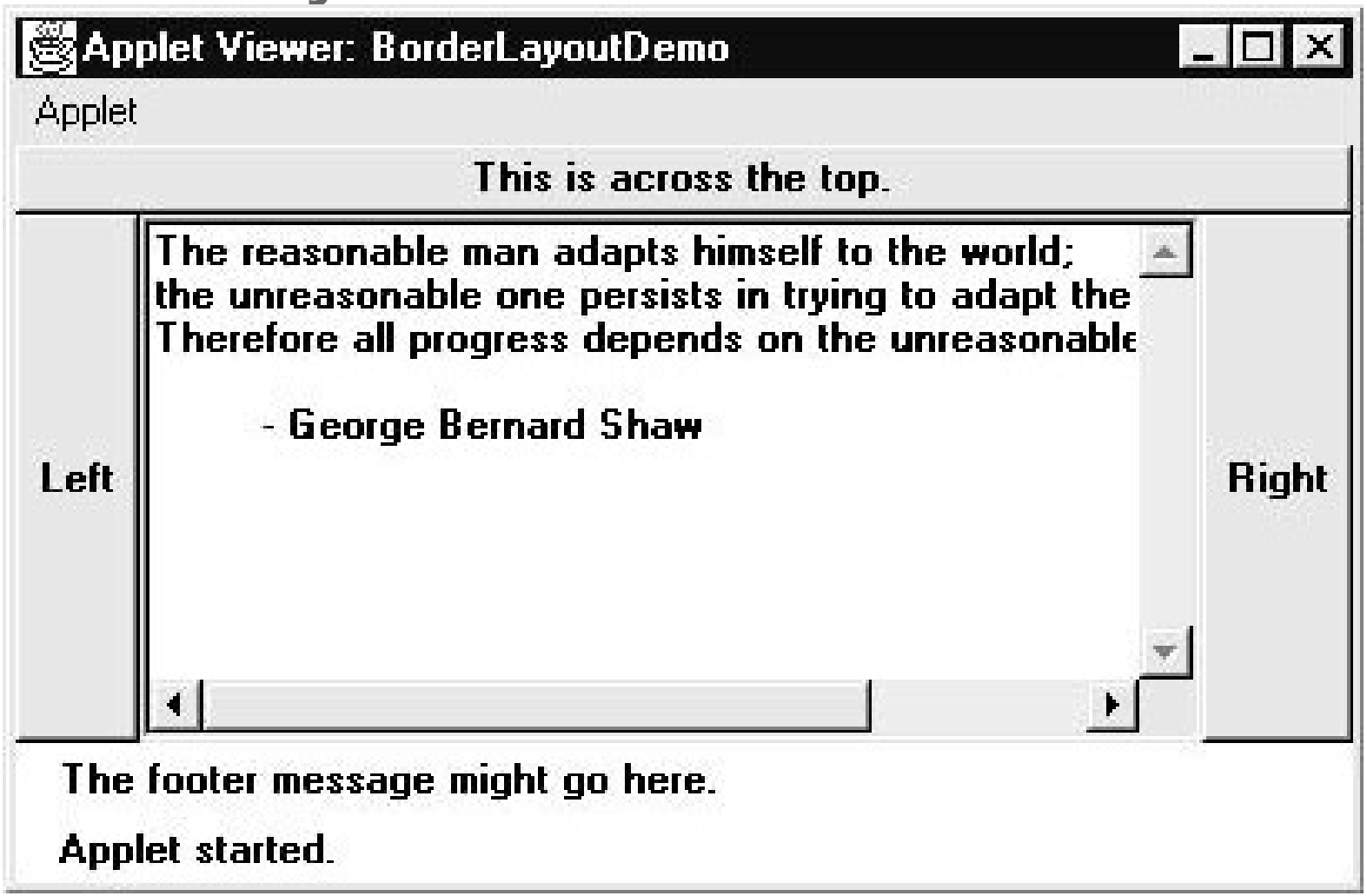
- Each Container object has a layout manager associated with it.
- A layout manager is an instance of any class that implements the LayoutManager interface.
- Layout manager is set by the setLayout( ) method.
- If no call to setLayout( ) is made, then the default layout manager is used.

void setLayout(LayoutManager *layoutObj*)

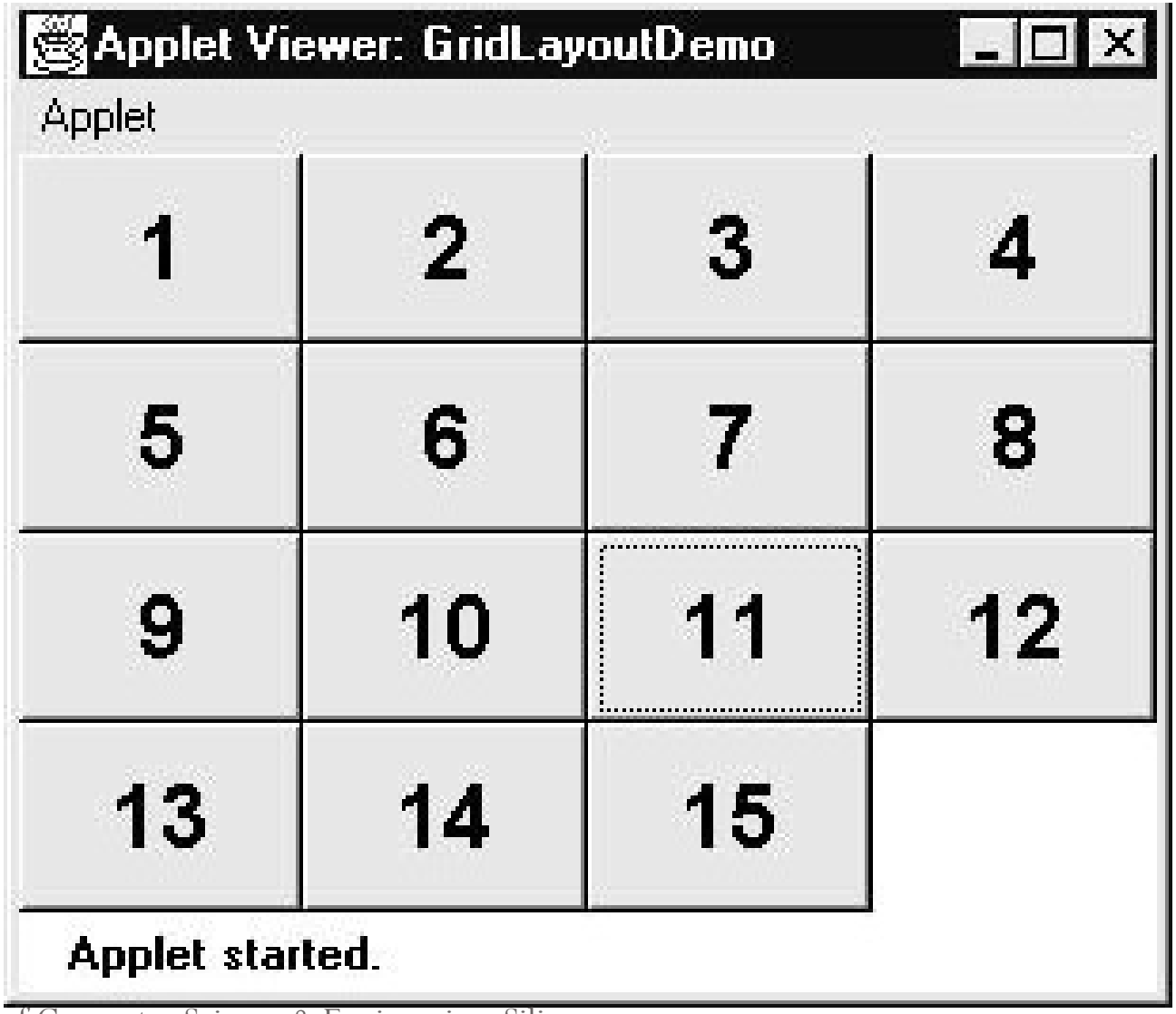
FlowLayout


Applet Viewer: FlowLayoutDemo
Applet
Windows 98/XP
Windows NT/2000
Solaris
MacOS
Current state:
Windows 98/XP: true
Windows NT/2000: false
Solaris: false
Mac: false
Applet started.

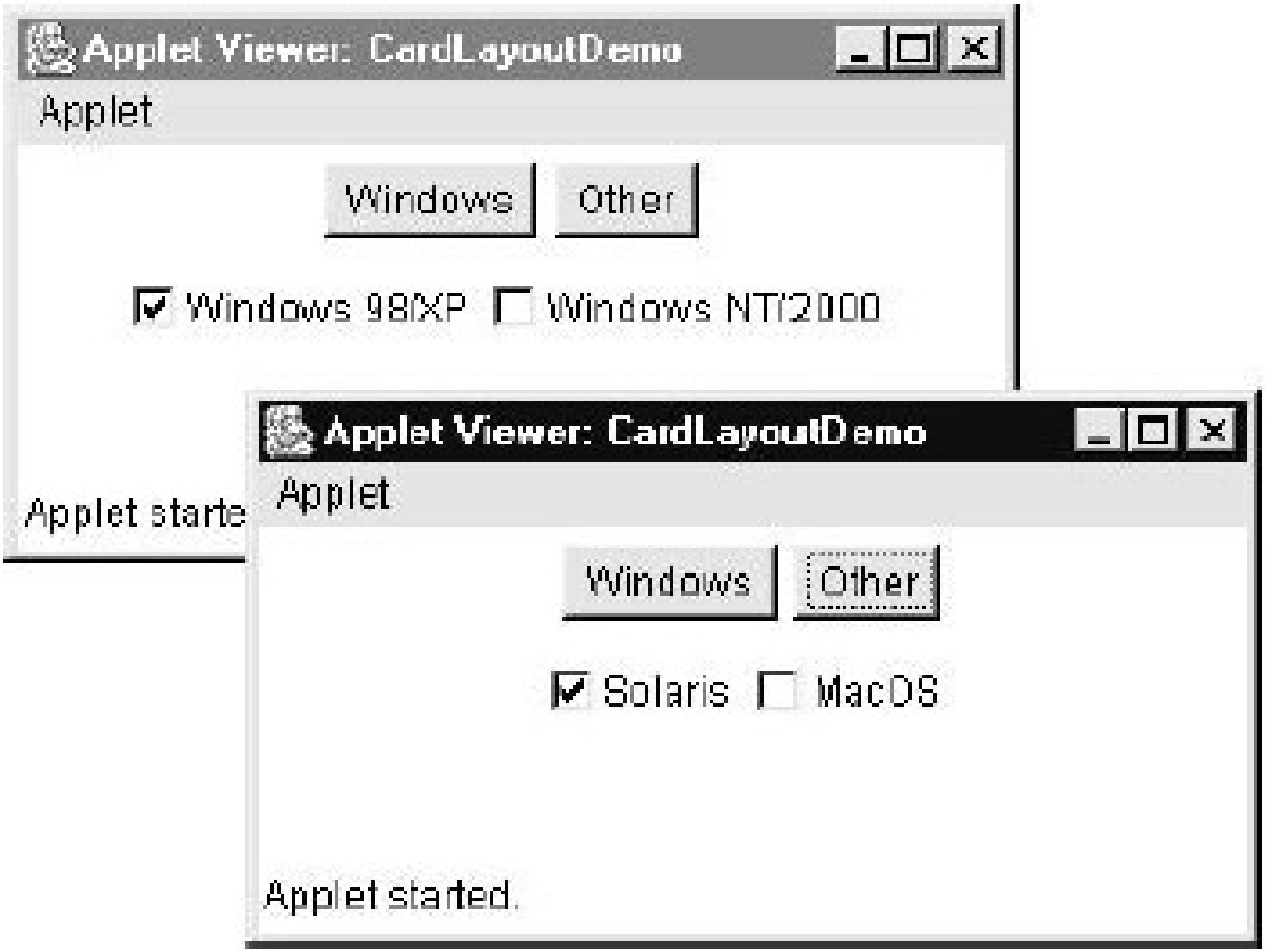
BorderLayout



GridLayout


Applet Viewer: GridLayoutDemo
Applet
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
Applet started.

CardLayout


Applet Viewer: CardLayoutDemo
Applet
Windows
Other
Windows 98/XP
Windows NT/2000
Applet starte
Applet Viewer: CardLayoutDemo
Applet
Windows
Other
Solaris
MacOS
Applet started.

FlowLayout

- FlowLayout is the default layout manager.

- Components are laid out from the upper-left corner, left to right and top to bottom. When no more components fit on a line, the next one appears on the next line.

- Constructors for FlowLayout

```
FlowLayout( )
FlowLayout(int how)
FlowLayout(int how, int horz, int vert)
```

- Valid values for how are as follows:

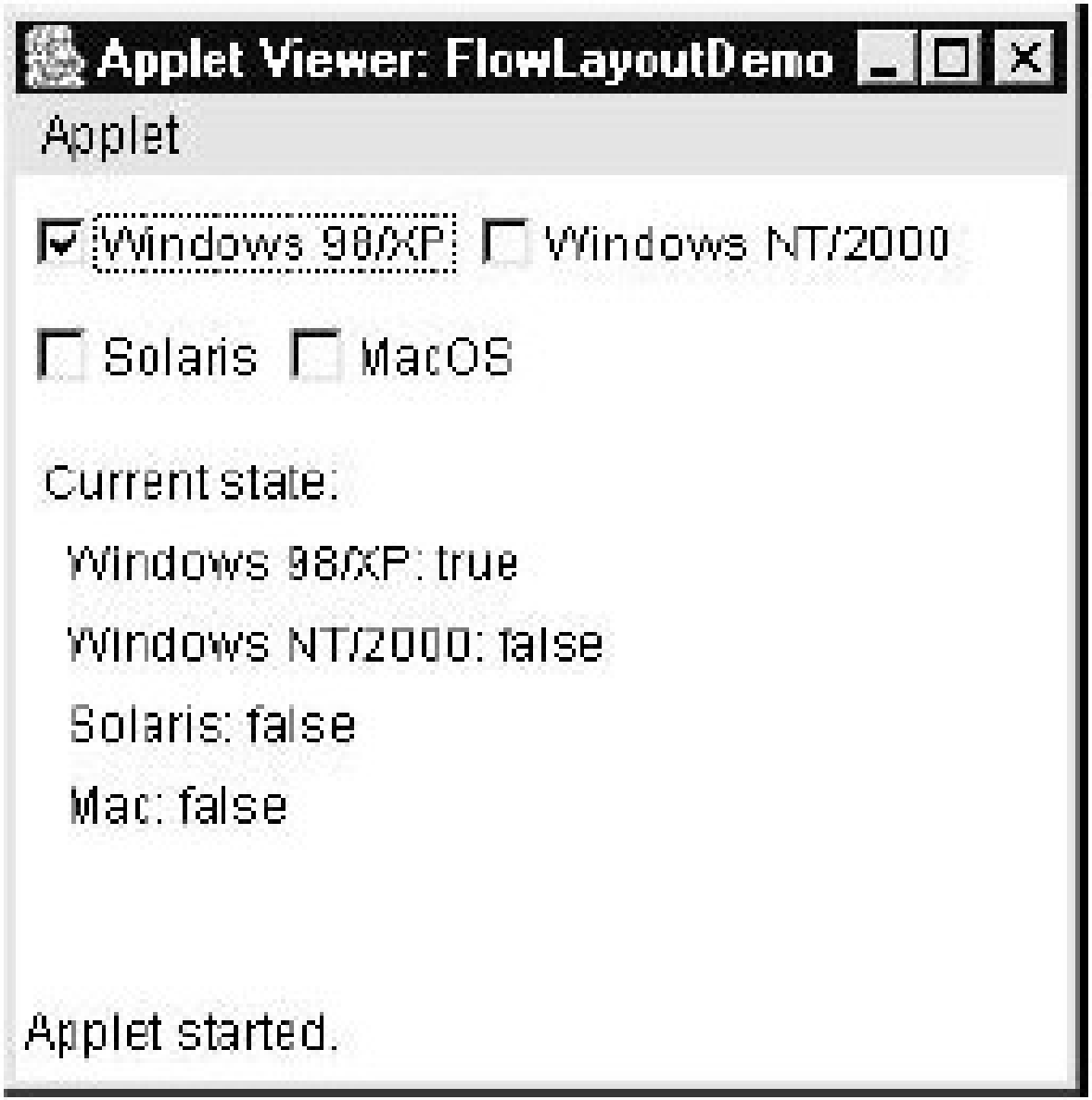
```
FlowLayout.LEFT
FlowLayout.CENTER
FlowLayout.RIGHT
```

- The third form allows you to specify the horizontal and vertical space left between components in *horz and vert*

```
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
/*
  <applet code="FlowLayoutDemo" width=250 height=200>
  </applet>
*/

public class FlowLayoutDemo extends Applet
  implements ItemListener {

  String msg = "";
  Checkbox Win98, winNT, solaris, mac;

  public void init() {
    // set left-aligned flow layout
    setLayout(new FlowLayout(FlowLayout.LEFT));

    Win98 = new Checkbox("Windows 98/XP", null, true);
    winNT = new Checkbox("Windows NT/2000");
    solaris = new Checkbox("Solaris");
    mac = new Checkbox("MacOS");
```

```
    add(Win98);
    add(winNT);
    add(solaris);
    add(mac);

    // register to receive item events
    Win98.addItemListener(this);
    winNT.addItemListener(this);
    solaris.addItemListener(this);
    mac.addItemListener(this);
  }

  // Repaint when status of a check box changes.
  public void itemStateChanged(ItemEvent ie) {
    repaint();
  }

  // Display current state of the check boxes.
  public void paint(Graphics g) {
```

```
        msg = "Current state: ";
        g.drawString(msg, 6, 80);
        msg = "  Windows 98/XP: " + Win98.getState();
        g.drawString(msg, 6, 100);
        msg = "  Windows NT/2000: " + winNT.getState();
        g.drawString(msg, 6, 120);
        msg = "  Solaris: " + solaris.getState();
        g.drawString(msg, 6, 140);
        msg = "  Mac: " + mac.getState();
        g.drawString(msg, 6, 160);
    }
}
```


Applet Viewer: FlowLayoutDemo
Applet
Windows 98/XP
Windows NT/2000
Solaris
MacOS
Current state:
Windows 98/XP: true
Windows NT/2000: false
Solaris: false
Mac: false
Applet started.

BorderLayout

- It has four narrow, fixed-width components at the edges and one large area in the center.

- The four sides are referred to as north, south, east, and west. The middle area is called the center.

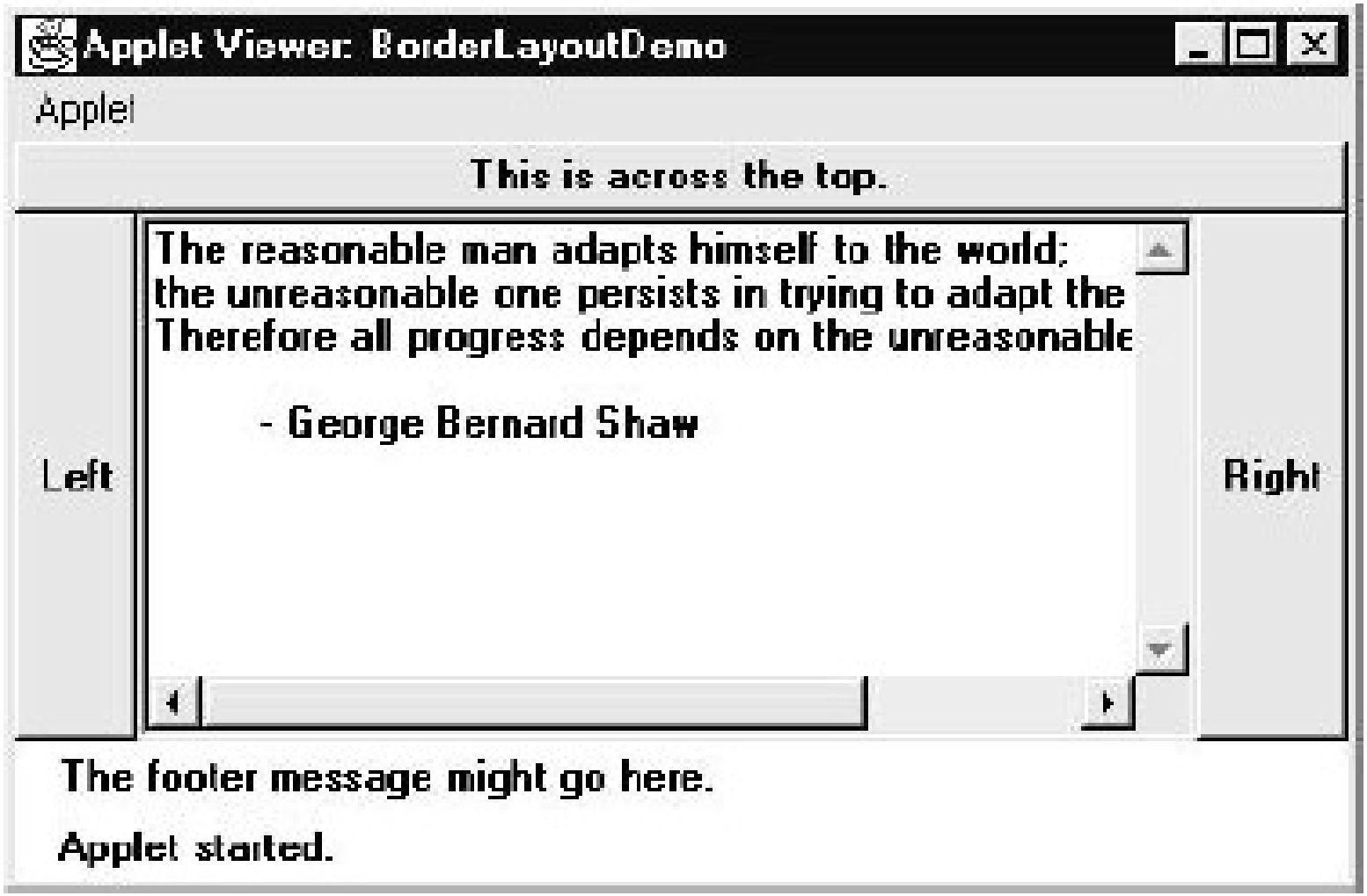
BorderLayout( )
BorderLayout(int *horz*, int *vert*)

- BorderLayout defines the following constants that specify the regions:

BorderLayout.CENTER
BorderLayout.SOUTH
BorderLayout.EAST
BorderLayout.WEST
BorderLayout.NORTH

```
import java.awt.*;
import java.applet.*;
import java.util.*;
/*
<applet code="BorderLayoutDemo" width=400 height=200>
</applet>
*/

public class BorderLayoutDemo extends Applet {
  public void init() {
    setLayout(new BorderLayout());

    add(new Button("This is across the top."),
        BorderLayout.NORTH);
    add(new Label("The footer message might go here."),
        BorderLayout.SOUTH);
    add(new Button("Right"), BorderLayout.EAST);
    add(new Button("Left"), BorderLayout.WEST);
```

```
    String msg = "The reasonable man adapts " +
      "himself to the world;\n" +
      "the unreasonable one persists in " +
      "trying to adapt the world to himself.\n" +

        "Therefore all progress depends " +
        "on the unreasonable man.\n\n" +
        "          - George Bernard Shaw\n\n";

      add(new TextArea(msg), BorderLayout.CENTER);
    }
}
```


Applet Viewer: BorderLayoutDemo
Applet
This is across the top.
The reasonable man adapts himself to the world;
the unreasonable one persists in trying to adapt the
Therefore all progress depends on the unreasonable
- George Bernard Shaw
Left
Right
The footer message might go here.
Applet started.

GridLayout

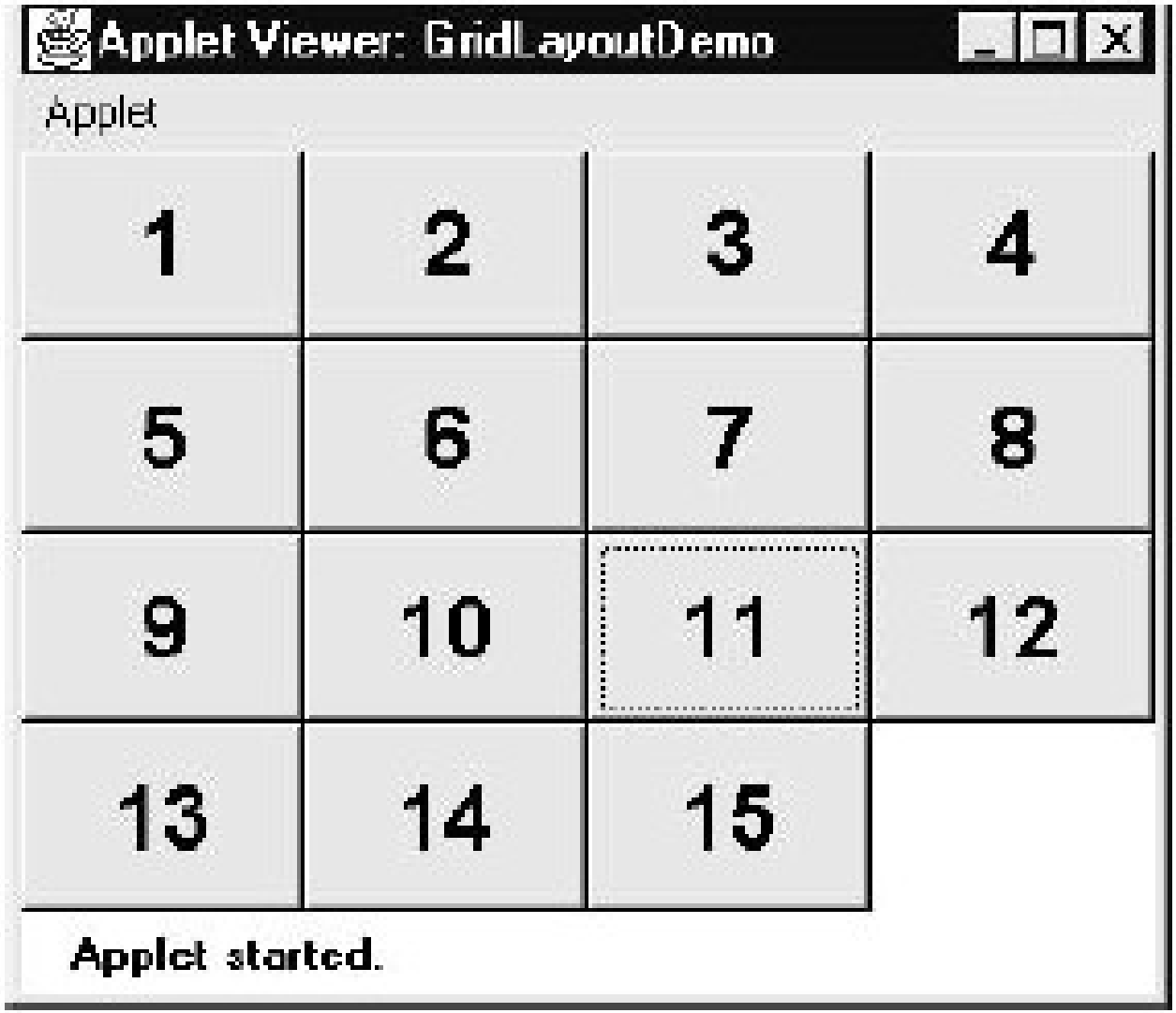
- GridLayout lays out components in a two-dimensional grid.

- Constructors supported by **GridLayout:**

GridLayout( )
GridLayout(int *numRows*, int *numColumns*)
GridLayout(int *numRows*, int *numColumns*, int *horz*, int *vert*)

- The third form allows you to specify the horizontal and vertical space left between components in *horz and vert*, respectively.

```
import java.awt.*;
import java.applet.*;
/*
<applet code="GridLayoutDemo" width=300 height=200>
</applet>
*/

public class GridLayoutDemo extends Applet {
  static final int n = 4;
  public void init() {
    setLayout(new GridLayout(n, n));

    setFont(new Font("SansSerif", Font.BOLD, 24));

    for(int i = 0; i < n; i++) {
      for(int j = 0; j < n; j++) {
        int k = i * n + j;
        if(k > 0)
          add(new Button("" + k));
      }
    }
  }
}
```


Applet Viewer: GridLayoutDemo
Applet
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
Applet started.

CardLayout

- CardLayout provides these two constructors:

```
CardLayout( )
CardLayout(int horz, int vert)
```

- Cards are typically held in an object of type Panel. This panel must have CardLayout selected as its layout manager.

- After you have created a deck, your program activates a card by calling one of the following methods defined by CardLayout:

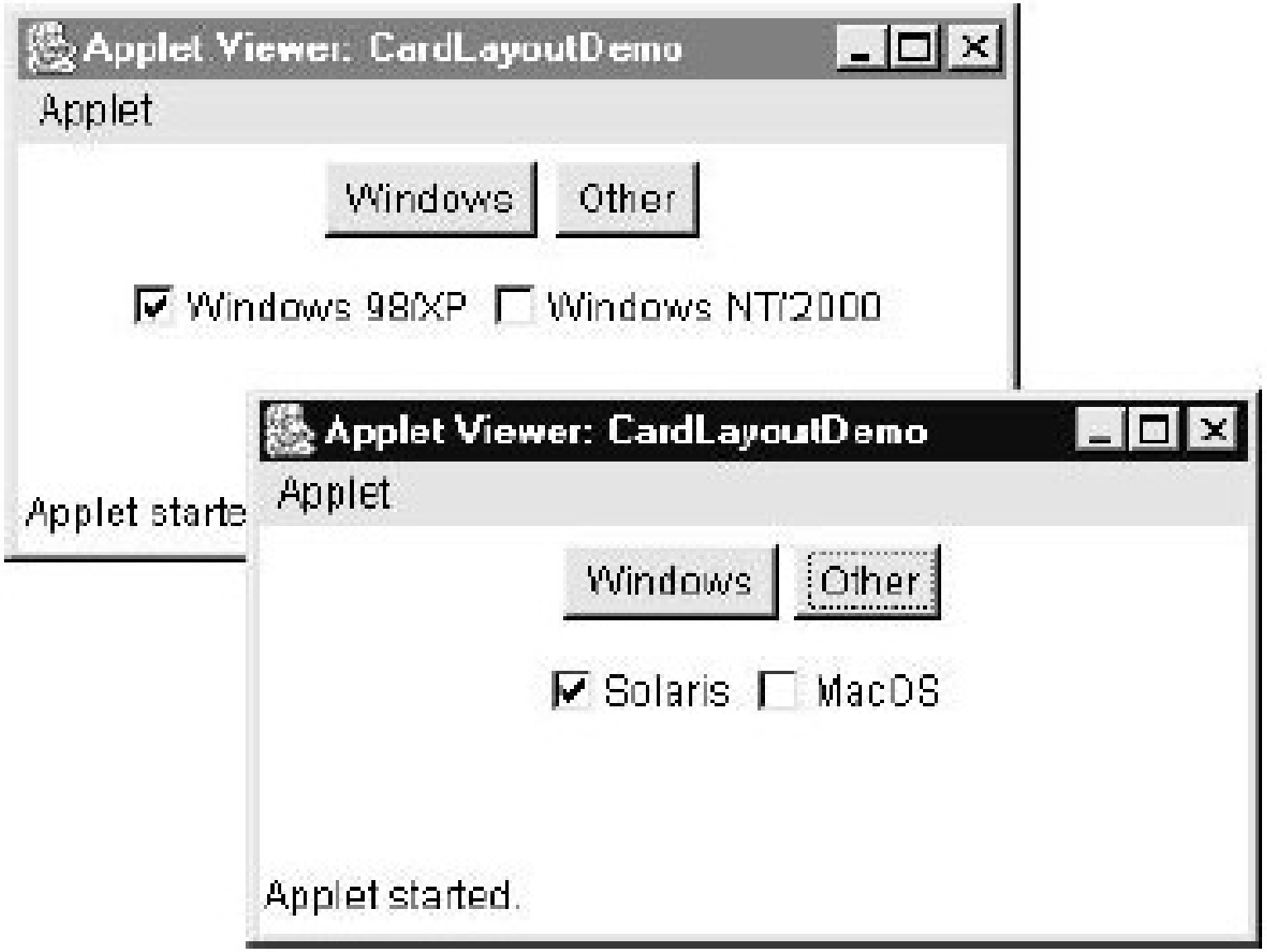
```
void first(Container deck)
void last(Container deck)
void next(Container deck)
void previous(Container deck)
void show(Container deck, String cardName)
```

```
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
/*
  <applet code="CardLayoutDemo" width=300 height=100>
  </applet>
*/

public class CardLayoutDemo extends Applet
  implements ActionListener, MouseListener {

  Checkbox Win98, winNT, solaris, mac;
  Panel osCards;
  CardLayout cardLO;
  Button Win, Other;

  public void init() {
    Win = new Button("Windows");
    Other = new Button("Other");
    add(Win);
    add(Other);
```

```
cardLO = new CardLayout();
osCards = new Panel();
osCards.setLayout(cardLO); // set panel layout to card layout

Win98 = new Checkbox("Windows 98/XP", null, true);
winNT = new Checkbox("Windows NT/2000");
solaris = new Checkbox("Solaris");
mac = new Checkbox("MacOS");

// add Windows check boxes to a panel
Panel winPan = new Panel();
winPan.add(Win98);
winPan.add(winNT);

// Add other OS check boxes to a panel
Panel otherPan = new Panel();
otherPan.add(solaris);
otherPan.add(mac);
```

```
    // add panels to card deck panel
    osCards.add(winPan, "Windows");
    osCards.add(otherPan, "Other");

    // add cards to main applet panel
    add(osCards);

    // register to receive action events
    Win.addActionListener(this);
    Other.addActionListener(this);

    // register mouse events
    addMouseListener(this);
  }

  // Cycle through panels.
  public void mousePressed(MouseEvent me) {
    cardLO.next(osCards);
  }

  // Provide empty implementations for the other MouseListener methods.
  public void mouseClicked(MouseEvent me) {
  }
```

```
  public void mouseEntered(MouseEvent me) {
  }
  public void mouseExited(MouseEvent me) {
  }
  public void mouseReleased(MouseEvent me) {
  }

  public void actionPerformed(ActionEvent ae) {
    if(ae.getSource() == Win) {
      cardLO.show(osCards, "Windows");
    }
    else {
      cardLO.show(osCards, "Other");
    }
  }
}
```


Applet Viewer: CardLayoutDemo
Applet
Windows
Other
Windows 98/XP
Windows NT/2000
Applet starte
Applet Viewer: CardLayoutDemo
Applet
Windows
Other
Solaris
MacOS
Applet started.

Thank you