An Introduction To Graphical User Interfaces

The event-driven model

Building simple graphical user interfaces (GUI's) in Java

Note: GUI Code Cannot Be Run Through A Text-Only Connection: SSH

```
[csb exampleTwo 45 ]> ls
Driver.class*      Driver.java       MyListener.class*  MyListener.java

[csb exampleTwo 46 ]> java Driver
Exception in thread "main" java.lang.InternalError: Can't connect to X11 window server
   using ':0.0' as the value of the DISPLAY variable.
      at sun.awt.X11GraphicsEnvironment.initDisplay(Native Method)
      at sun.awt.X11GraphicsEnvironment.<clinit>(X11GraphicsEnvironment.java:125)
      at java.lang.Class.forName0(Native Method)
      at java.lang.Class.forName(Class.java:140)
      at
   java.awt.GraphicsEnvironment.getLocalGraphicsEnvironment(GraphicsEnvironment.jav
   a:62)
      at java.awt.Window.init(Window.java:223)
      at java.awt.Window.<init>(Window.java:267)
      at java.awt.Frame.<init>(Frame.java:398)
      at java.awt.Frame.<init>(Frame.java:363)
      at Driver.main(Driver.java:7)
```

Components

- They are all types of graphical controls and displays available:
  - Button, Canvas, CheckBox, Dialog, File Dialog, Frame, Label, List, Menu, Panel, PopupMenu, Scrollbar, ScrollPane, TextArea, Window
- Also known as "widgets"
- For Sun's online documentation refer to the url:
  - *http://java.sun.com/j2se/1.4.2/docs/guide/awt/index.html*

Containers

- A special type of component that is used to hold/contain the components (subclass of component)
- Can be used to group components on the screen
- You must have at least one container object for your GUI:
  - Panel, Window, Dialogs, Frame

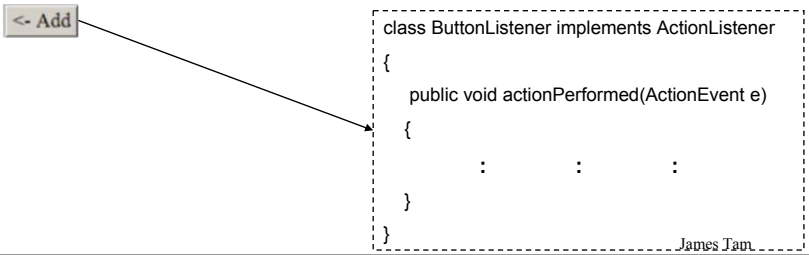
Containers

- A special type of component that is used to hold/contain the components (subclass of component)
- Can be used to group components on the screen
- You must have at least one container object for your GUI:
  - Panel, Window, Dialogs, **Frame**

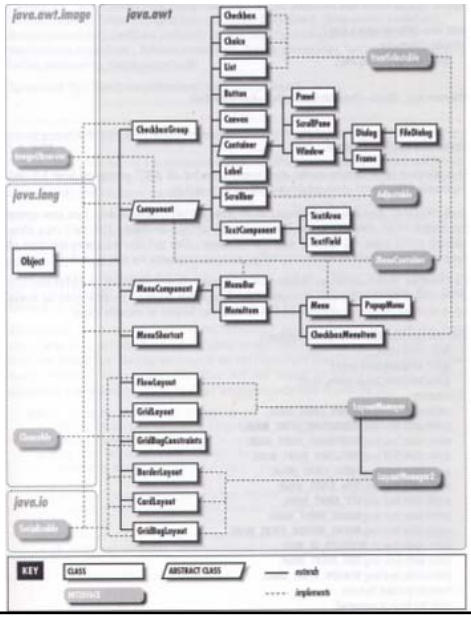
Some Relevant Java GUI packages

1. Java classes for the components and containers
   - e.g., Button class
   - java.awt (import java.awt.*)

2. Java classes with the code to react to user-initiated events
   - e.g., code to react when a button is pressed
   - java.awt.event (import java.awt.event.*)



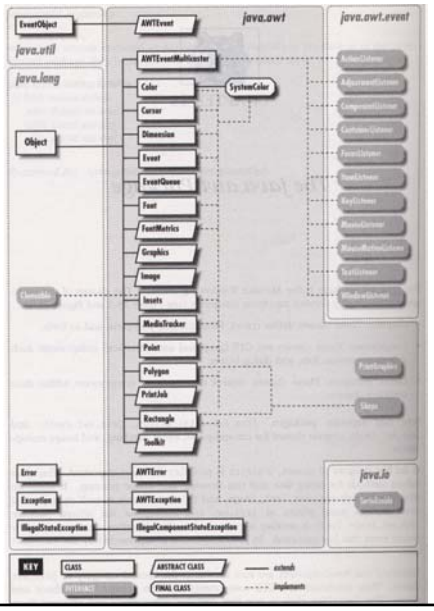
The AWT Package



From "Java in a Nutshell" 2nd Edition p. 281

James Tam

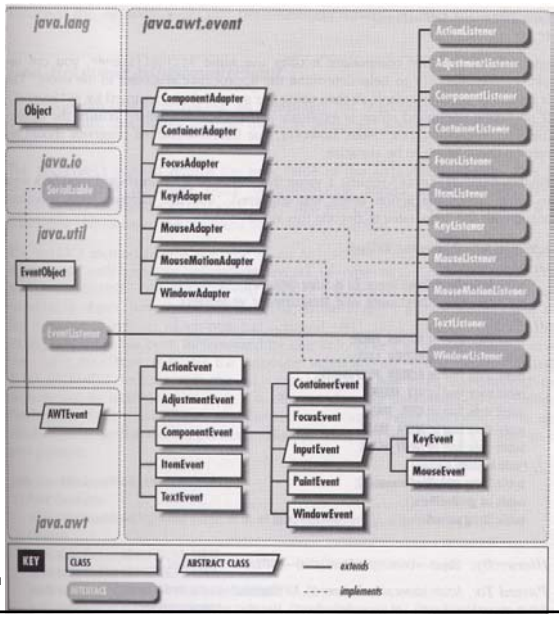
The AWT Package (2)



From "Java in a Nutshell" 2nd Edition p. 280

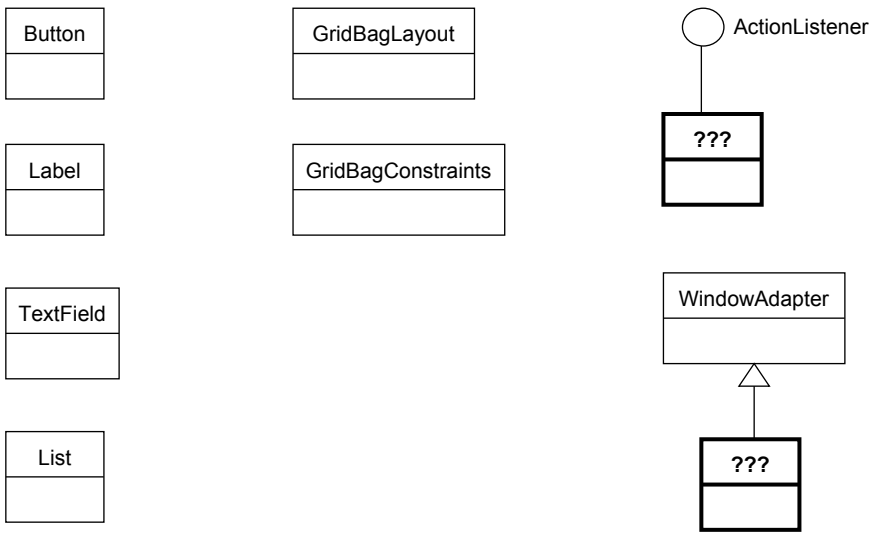
James Tam

The Java AWT Event Package



From "Java in a Nutshell" 2nd Edition p. 341

Some Relevant Java GUI Classes For This Class



Traditional Software

Program control is largely determined by the program through a series of sequential statements.

Example

```
:
if (num >= 0)
{
    // Statements for the body of the if
}
else
{
    // Statements for the body of the else
}
```

When num is non-negative

Num is negative

Traditional Software

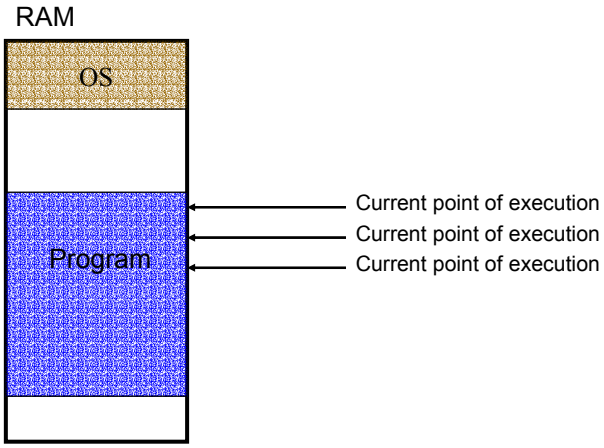
The user can only provide input at places that are specified by the program (when an input statement is encountered).

Example

```
System.out.print("Enter student ID number: ");
studentID = Console.in.readInt();
```

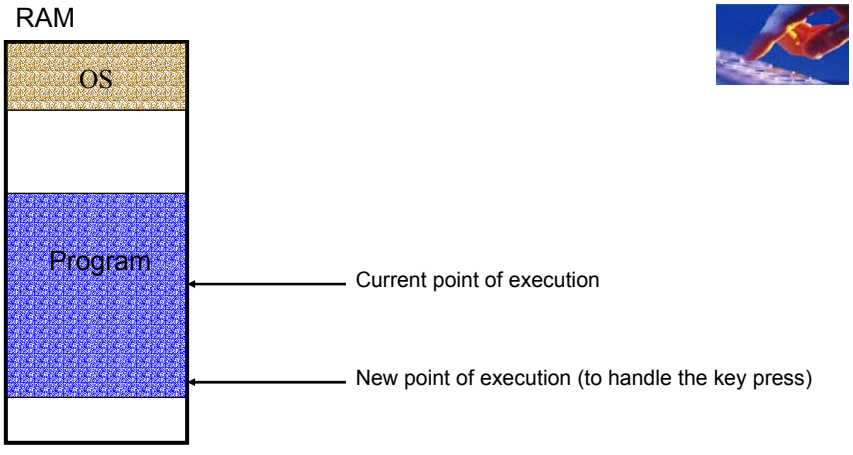
Event-Driven Software

Program control can also be determined be sequential



Event-Driven Software

Program control *can also* be determined by events



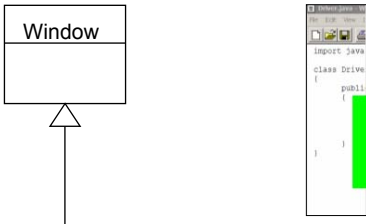
Characteristics Of Event Driven Software

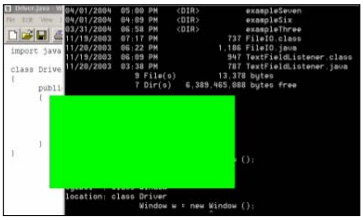
- Program control can be determined by events as well standard program control statements
- A typical source of these events is the user
- These events can occur at any time

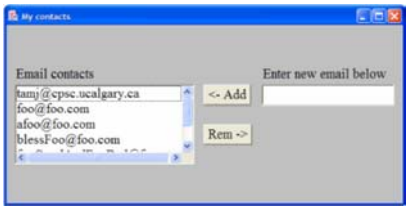
Most Components Can Trigger Events

- Graphical objects can be manipulated by the user to trigger events.
- Each graphical object can have 0, 1 or many events that can be triggered.

Window Classes



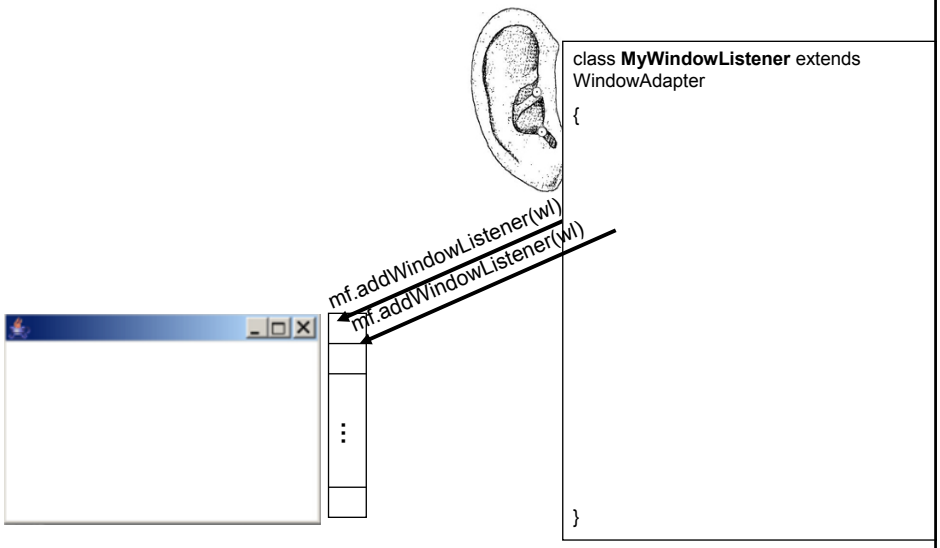




Steps In The Event Model For Handling A Frame Event: Window Closing

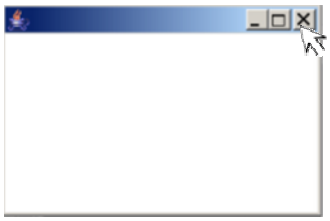
1) The frame must register all interested event listeners.
2) The user triggers the event by closing the window
3) The window sends a message to all listeners of that event.
4) The window event listener runs the code to handle the event.

1. The Frame Must Register All Interested Event Listeners.



James Tam

2. The User Triggers The Event By Closing The Window

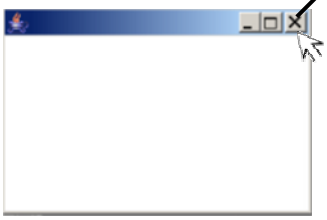


James Tam

3. The Window Sends A Message To All Listeners Of That Event.

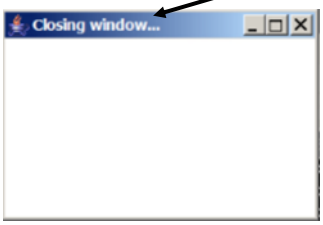
```
class MyWindowListener extends WindowAdapter
{
      public void windowClosing  (WindowEvent e)
      {




      }
}
```


4. The Event Listener Runs The Code To Handle The Event.



```
class MyWindowListener extends WindowAdapter
{
      public void windowClosing  (WindowEvent e)
      {
        Frame f = (Frame) e.getWindow();
        f.setTitle("Closing window...");
        for (int i = 0; i < 500000000; i++);
        f.setVisible(false);
        f.dispose();
        System.exit(0);
      }
}
```

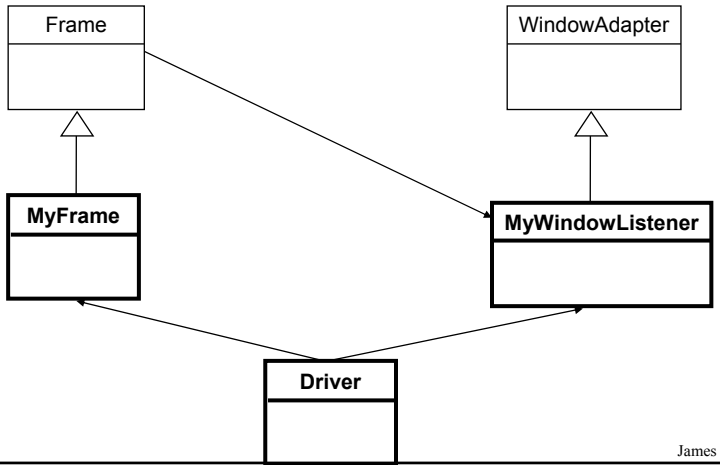
4. The Event Listener Runs The Code To Handle The Event.




```
class MyWindowListener extends WindowAdapter
{
      public void windowClosing  (WindowEvent e)
      {
        Frame f = (Frame) e.getWindow();
        f.setTitle("Closing window...");
        for (int i = 0; i < 500000000; i++);
        f.setVisible(false);
        f.dispose();
        System.exit(0);
      }
}
```


An Example Of Handling A Frame Event

The complete code for this example can be found in Unix under the path:
/home/233/examples/gui/exampleOne



An Example Of Handling A Frame Event: The Driver Class

```
import java.awt.*;

class Driver
{
    public static void main (String [] args)
    {
        MyFrame mf = new MyFrame ();
        MyWindowListener mwl = new MyWindowListener() ;
        mf.addWindowListener(mwl);
        mf.setSize (300,200);
        mf.setVisible(true);
    }
}
```


An Example Of Handling A Frame Event: Class MyFrame

```
import java.awt.*;
class MyFrame extends Frame
{
  // More code will be added in later examples.
}
```

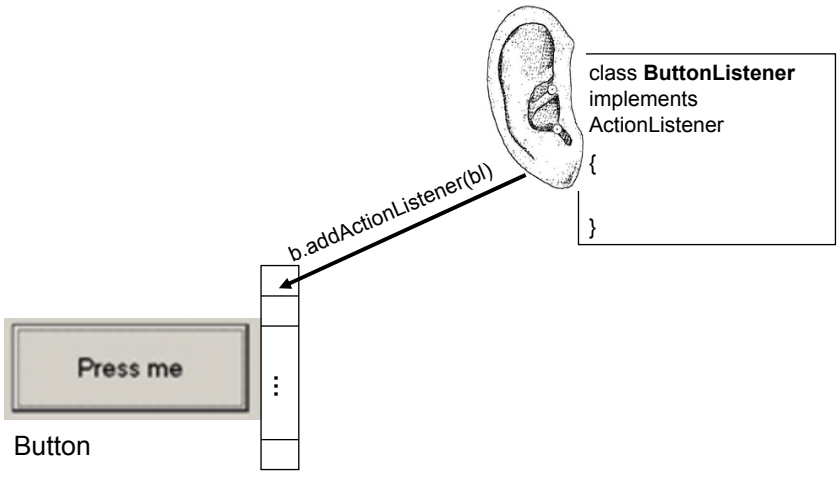
An Example Of Handling A Frame Event: Class MyWindowListener

```
import java.awt.event.*;
import java.awt.*;

class MyWindowListener extends WindowAdapter
{
      public void windowClosing (WindowEvent e)
      {
        Frame f = (Frame) e.getWindow();
        f.setTitle("Closing window...");
        for (int i = 0; i < 500000000; i++);
        f.setVisible(false);
        f.dispose();
        System.exit(0);
      }
}
```


Steps In The Event Model For Handling A Button Event

1) The button must register all interested event listeners.
2) The user triggers an event by pressing a button.
3) The button sends a message to all listeners of the button press event.
4) The button listener runs the code to handle the button press event.

1. The Graphical Component Must Register All Interested Event Listeners.



2. The User Triggers An Event By Pressing The Button

3. The Button Sends A Message To All Registered Listeners For That Event

```
class ButtonListener implements ActionListener
{
    public void actionPerformed (ActionEvent e)
    {

    }
}
```

3. The Component Sends A Message To All Registered Listeners For That Event

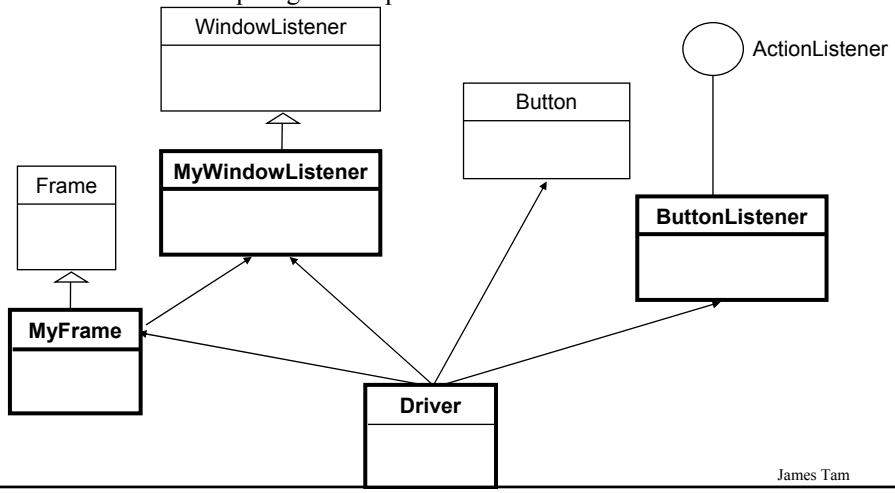
```
class ButtonListener implements ActionListener
{
    public void actionPerformed (ActionEvent e)
    {
        Button b = (Button) e.getSource();
        b.setLabel("Stop pressing me!");
    }
}
```

4. The Event Listener Runs The Code To Handle The Event

```
class ButtonListener implements ActionListener
{
    public void actionPerformed (ActionEvent e)
    {
        Button b = (Button) e.getSource();
        b.setLabel("Stop pressing me!");
    }
}
```

An Example Of Handling A Button Event

The complete code for this example can be found in Unix under the path:
/home/233/examples/gui/exampleTwo



An Example Of Handling A Button Event: The Driver Class

```
import java.awt.*;
class Driver
{
    public static void main (String [] args)
    {
        MyFrame mf = new MyFrame ();
        MyWindowListener mwl = new MyWindowListener();
        mf.addWindowListener(mwl);
        mf.setSize (300,200);
```


An Example Of Handling A Button Event: The Driver Class (2)

```
        Button b = new Button("Press me.");
        ButtonListener bl = new ButtonListener();
        b.addActionListener(bl);
        mf.add(b);
        mf.setVisible(true);
    }
}
```

An Example Of Handling A Button Event: The ButtonListener Class

```
import java.awt.*;
import java.awt.event.*;

class ButtonListener implements ActionListener
{
   public void actionPerformed (ActionEvent e)
   {
        Button b = (Button) e.getSource();
        b.setLabel("Stop pressing me!");
   }
}
```

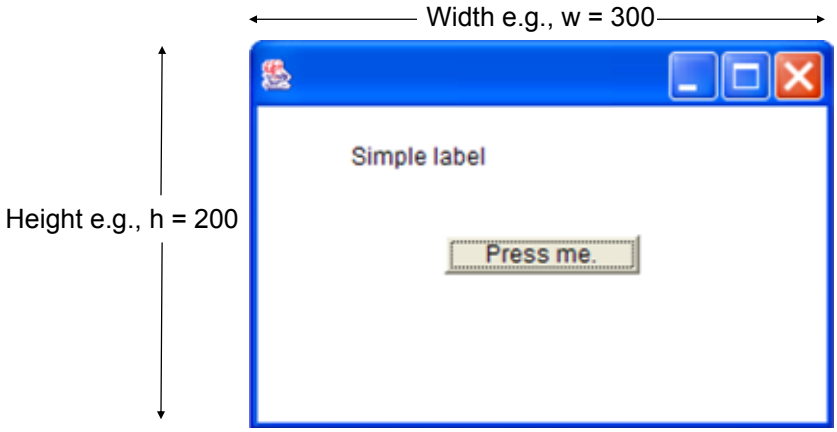

How To Handle The Layout Of Components

1. Manually set the coordinates yourself
2. Use one of Java's built-in layout manager classes

Layout Is Based On Spatial Coordinates

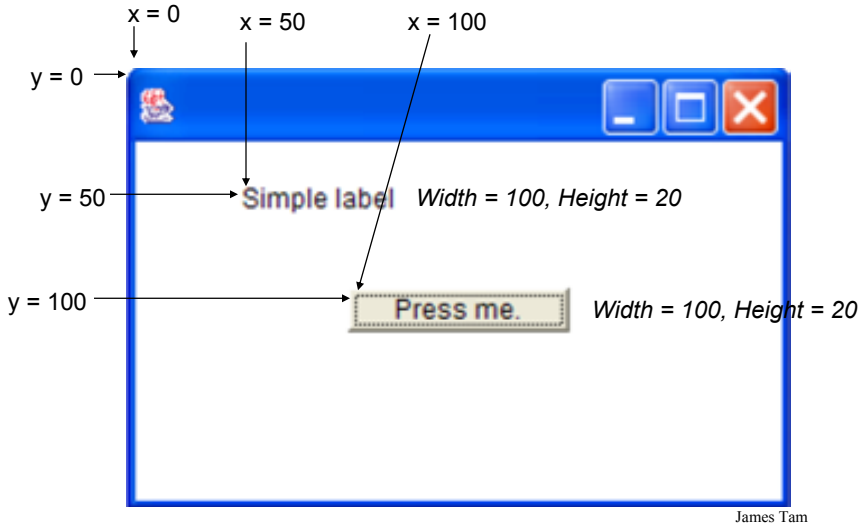
e.g. MyFrame my =new MyFrame ();

my.setSize(300,200);



Layout Is Based On Spatial Coordinates

Coordinates Of Components: Relative To The Container



A Example Showing Manual Layout

The complete code for this example can be found in Unix under the path:
/home/233/examples/gui/exampleThree

An Example Showing Manual Layout: The Driver Class

```
import java.awt.*;

class Driver
{
    public static void main (String [] args)
    {
        MyFrame mf = new MyFrame ();
        MyWindowListener mwl = new MyWindowListener();
        mf.addWindowListener(mwl);
        mf.setLayout(null);
        mf.setSize (300,200);
        Button b1 = new Button("Press me.");
        b1.setBounds(100,100,100,20);
```


An Example Showing Manual Layout: The Driver Class (2)

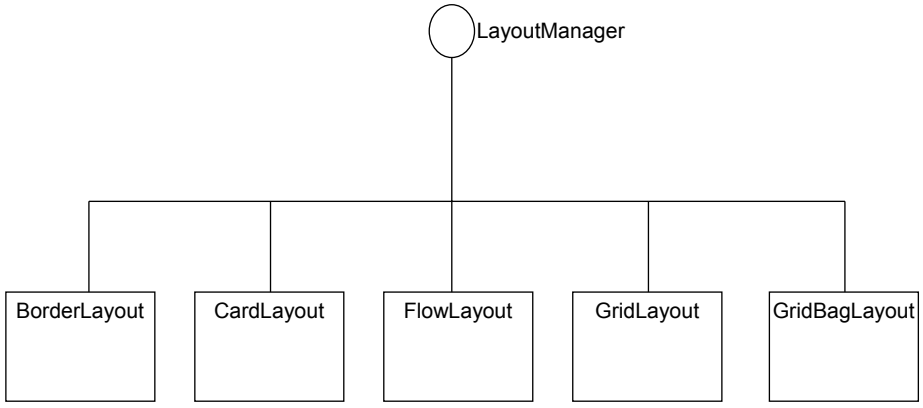
```
        ButtonListener bl = new ButtonListener();
        b1.addActionListener(bl);

        Label l1 = new Label ("Simple label");
        l1.setBounds(50,50,100,20);

        mf.add(b1);
        mf.add(l1);
        mf.setVisible(true);
    }
}
```

How To Handle The Layout Of Components

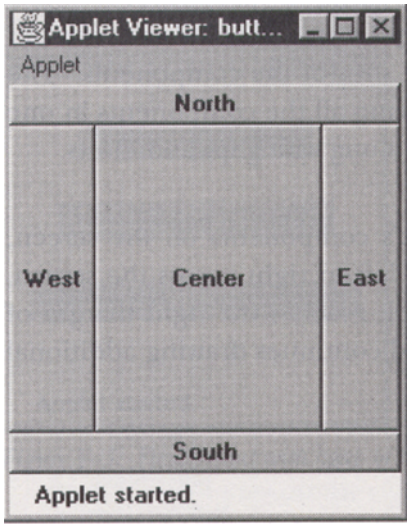
1. Manually set the coordinates yourself
2. Use one of Java's built-in layout manager classes

Java Layout Classes

There are many implementations (this diagram only includes the original classes that were implemented by Sun).

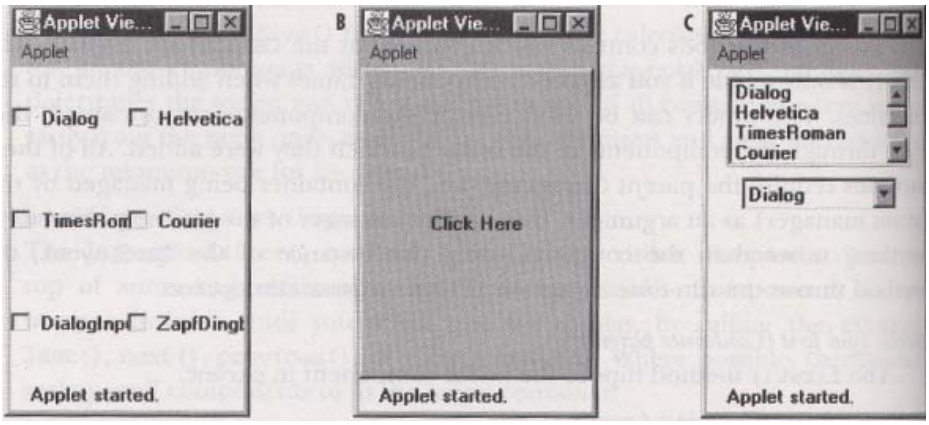


BorderLayout



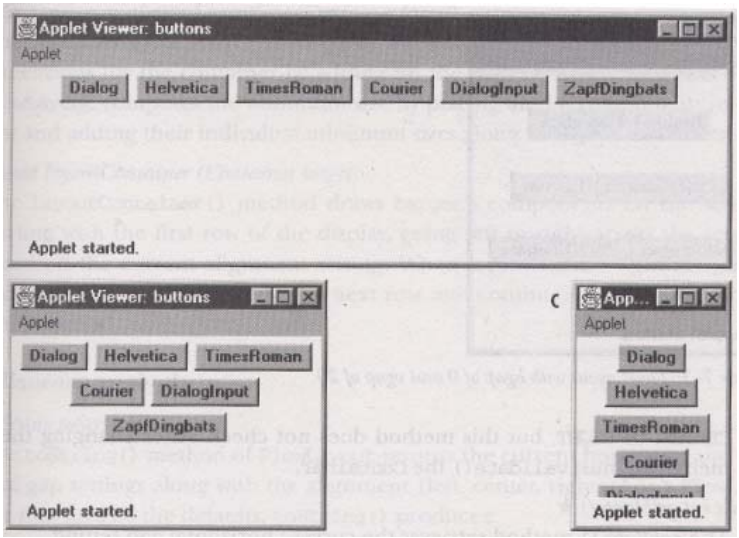
From Java: AWT Reference p. 256

CardLayout



From Java: AWT Reference p. 264

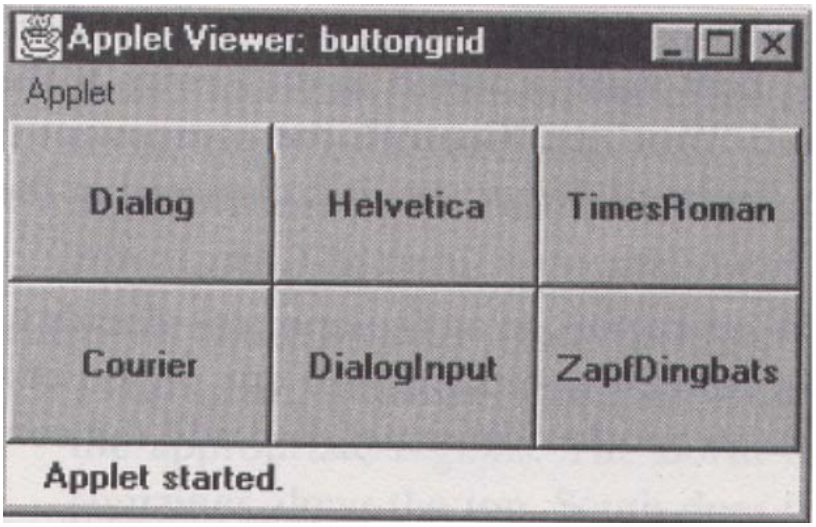
FlowLayout



From Java: AWT Reference p. 253

James Tam

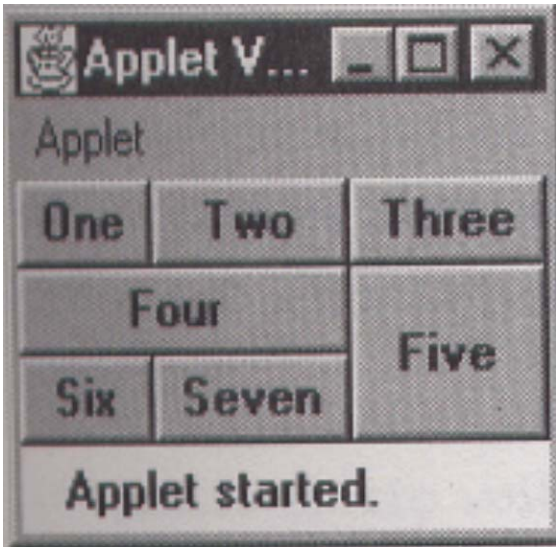
GridLayout



From Java: AWT Reference p. 260

James Tam

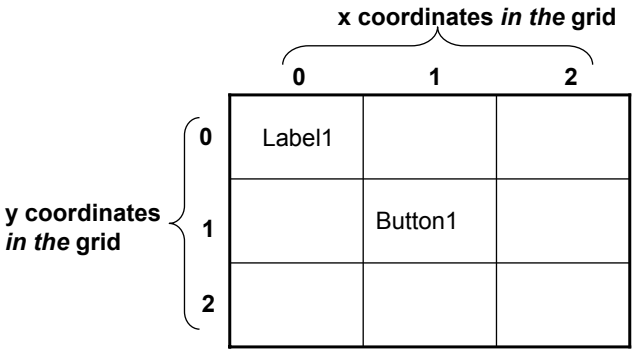
GridBagLayout



From Java: AWT Reference p. 269

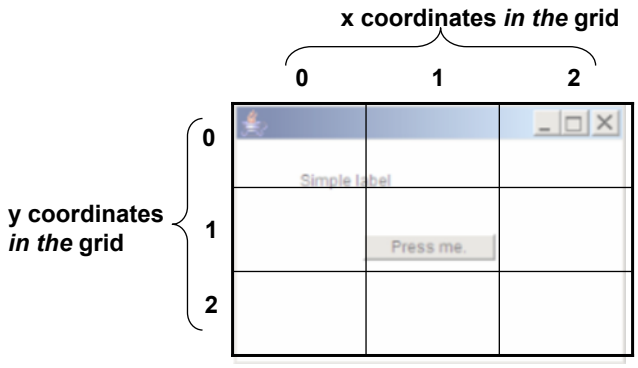
Designing A GUI When Using The GridBagLayout

Use graph paper or draw out a table.



Designing A GUI When Using The GridBagLayout

Use graph paper or draw out a table.



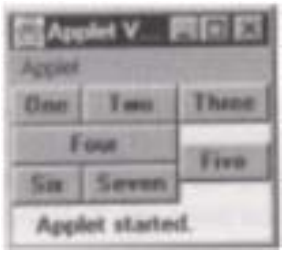
GridBagConstraints

- Goes with the GridBagLayout class
- Because the GridBagLayout doesn't know how to display components you also need GridBagConstraints to constrain things (determine the layout).
- GridBagConstraints indicates how components should be displayed within the GridBag
- For more complete information see:
  - *http://java.sun.com/j2se/1.4.2/docs/api/java/awt/GridBagConstraints.html*

Some Important Parts Of The GridBagConstraints Class

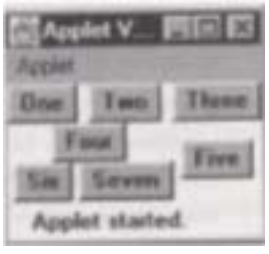
```
class GridBagConstraints
{
  // Used in conjunction with the constants below to determine the resize policy of
  the component
  public int fill;

  // Apply only if there is available space.
  // Determine in which direction (if any) that the component expands to fill the
  // space.
  public final static int NONE;
  public final static int BOTH;
  public final static int HORIZONTAL;
  public final static int VERTICAL;
```


GridBagContraints: Fill Values



Horizontal

Vertical



None

Some Important Parts Of The GridBagConstraints Class (2)

```
// Position within the grid
 public int gridx;
 public int gridy;

 // Number of grid squares occupied by a component
 public int gridwidth;
 public int gridheight;
```


Some Important Parts Of The GridBagConstraints Class (3)

```
// Used in conjunction with the constants below to determine that the component
// drift if the space available is larger than the component.
public int anchor;

// Apply only if the component is smaller than the available space.
// Determine in which direction that the component will be anchored there
public final static int CENTER;
public final static int EAST;
public final static int NORTH;
public final static int NORTHEAST;
public final static int NORTHWEST;
public final static int SOUTH;
public final static int SOUTHEAST;
public final static int SOUTHWEST;
public final static int WEST;
```

An Example Using The GridBagLayout

The complete code for this example can be found in Unix under the path:
/home/233/examples/gui/exampleFour

An Example Using The GridBagLayout: The Driver Class

```
import java.awt.*;

class Driver
{
    public static void main (String [] args)
    {
        MyFrame mf = new MyFrame ();
    }
}
```

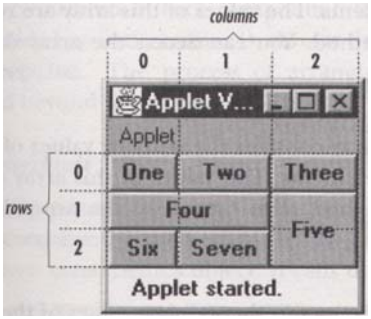
An Example Using The GridBagLayout: Class MyFrame

```
import java.awt.*;
class MyFrame extends Frame
{
   private MyWindowListener mwl;
   private Button button1;
   private ButtonListener bl;
   private Label label1;
   private GridBagLayout gbl;
   public MyFrame ()
   {
      mwl = new MyWindowListener ();
      button1 = new Button("Press me");
      bl = new ButtonListener();
      label1 = new Label("Simple label");
      gbl = new GridBagLayout();
      setLayout(gbl);     // Calling method of super class.
      addWidget(label1, 0, 0, 1, 1);
      addWidget(button1, 2, 2, 1, 1);
   }
```


An Example Using The GridBagLayout: Class MyFrame (2)

```
   public void addWidget (Component widget, int x, int y, int w, int h)
   {
      GridBagConstraints gbc = new GridBagConstraints();
      gbc.gridx = x;
      gbc.gridy = y;
      gbc.gridwidth = w;
      gbc.gridheight = h;
      gbl.setConstraints (widget, gbc);
      add(widget);        // Calling method of super class.
   }
}
```

Advanced Uses Of GridBagLayout



Button	gridx (col)	gridy (row)	grid-width	grid-height
One	0	0	1	1
Two	1	0	1	1
Three	2	0	1	1
Four	0	1	2	1
Five	2	1	1	2
Six	0	2	1	1
Seven	1	2	1	*1*

From Java: AWT Reference p. 269

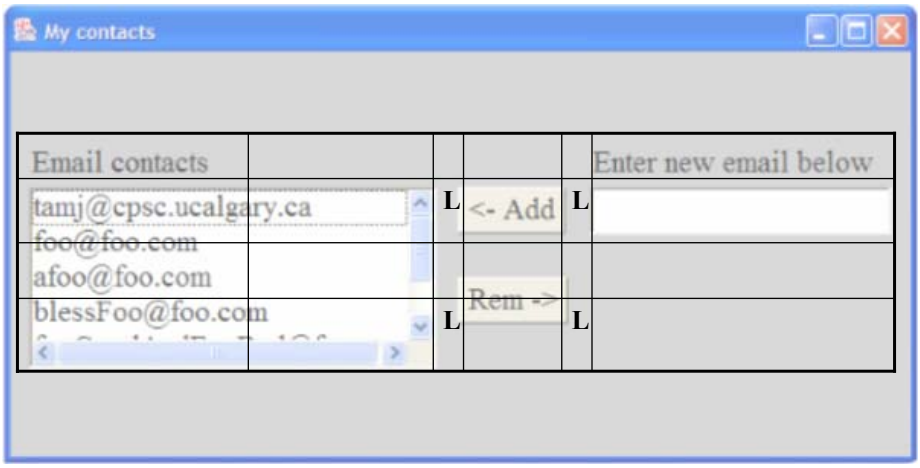
James Tam

The Grid For Your Assignment

	0	1	2	3	4	5
0	Label (“Email contacts”)					Label (“Enter new email below”)
1	List (Personal contacts)	List (Personal contacts)	*Label (empty)*	Button (“Add”)	*Label (empty)*	TextField (For new contact)
2	List (Personal contacts)	List (Personal contacts)				
3	List (Personal contacts)	List (Personal contacts)	*Label (empty)*	Button (“Remove”)	*Label (empty)*	
4						

James Tam

The Grid For Your Assignment (2)



Components Effecting The State Of Other Components

The complete code for this example can be found in Unix under the path:
/home/233/examples/gui/exampleFive

Components Effecting The State Of Other Components: The Driver Class

```
import java.awt.*;

class Driver
{
    public static void main (String [] args)
    {
        MyFrame mf = new MyFrame ();
    }
}
```


Components Effecting The State Of Other Components: Class MyFrame

```
import java.awt.*;

class MyFrame extends Frame
{
   private MyWindowListener mwl;
   private Button himButton;
   private Button herButton;
   private ButtonListener bl;
   private Label label1;
   private GridBagLayout gbl;
```

Components Effecting The State Of Other Components: Class MyFrame (2)

```
public MyFrame ()
{
   mwl = new MyWindowListener ();
   addWindowListener(mwl);

   himButton = new Button("Press her not me.");
   himButton.setActionCommand("him");
   himButton.setBackground(Color.lightGray);

   herButton = new Button("Press him not me");
   herButton.setActionCommand("her");
   herButton.setBackground(Color.lightGray);

   bl = new ButtonListener();
   himButton.addActionListener(bl);
   herButton.addActionListener(bl);
```


Components Effecting The State Of Other Components: Class MyFrame (3)

```
   Label1 = new Label("Simple label");
   gbl = new GridBagLayout();
   setLayout(gbl);     // Calling method of super class.

   addWidget(label1, 0, 0, 1, 1);
   addWidget(himButton, 0, 1, 1, 1);
   addWidget(herButton, 0, 2, 1, 1);
   setSize(300,200);
   setVisible(true);
}
```

Components Effecting The State Of Other Components: Class MyFrame (4)

```
    public void addWidget (Component widget, int x, int y, int w, int h)
    {
       GridBagConstraints gbc = new GridBagConstraints();
       gbc.gridx = x;
       gbc.gridy = y;
       gbc.gridwidth = w;
       gbc.gridheight = h;
       gbl.setConstraints(widget, gbc);
       add(widget);        // Calling method of super class.
    }

    public Button getHerButton () { return herButton; }
    public Button getHimButton () { return himButton; }
}
```


Components Effecting The State Of Other Components: Class ButtonListener

```
import java.awt.*;
import java.awt.event.*;

class ButtonListener implements ActionListener
{
    public void actionPerformed (ActionEvent e)
    {
       Button b = (Button) e.getSource();
       String s = e.getActionCommand();
       MyFrame mf = (MyFrame) b.getParent();
```

Components Effecting The State Of Other Components: Class ButtonListener (2)

```
if (s.equals("her"))
{
        Button himButton = mf.getHimButton();
        himButton.setLabel("Ha! Ha! Ha!");

        himButton.setBackground(Color.green);
        for (int i = 0; i < 500000000; i++);
        himButton.setLabel("Press him not me.");
        himButton.setBackground(Color.lightGray);
}
```


Components Effecting The State Of Other Components: Class ButtonListener (3)

```
 else if (s.equals("him"))
{
        Button herButton = mf.getHerButton();
        herButton.setLabel("Ha! Ha! Ha!");

        herButton.setBackground(Color.green);
        for (int i = 0; i < 500000000; i++);
        herButton.setLabel("Press him not me.");
        herButton.setBackground(Color.lightGray);
}
 else
 {
        :
 }
}
}
```

The List Class

- Used to provide a graphical and interactive control for a list of Strings.
- Scrollbars are automatically included
- For the complete class refer to the url:
  - *http://java.sun.com/j2se/1.4.2/docs/api/java/awt/List.html*

Some Important Parts Of The List Class

```
class List
{
  // The data for the list is stored internally as an array of references to Strings.

  // Creates a scrollable list
  public List ()

  // Creates a list with the specified number of visible rows
  public List (int rows)

  // Adds a new element to the end of the list
  public void add (String item)

  // Adds a listener for list events
  addActionListener (ActionListener l)
```

Some Important Parts Of The List Class (2)

```
   // Returns a count of the number of elements in the list.
   public int getItemCount ()

   // Removes the item at the specified position in the list
   public void remove (int index)
      :              :                    :
      :              :                    :
}
```


An Example Employing A List

The complete code for this example can be found in Unix under the path:
/home/233/examples/gui/exampleSix

An Example Employing A List: The Driver Class

```
import java.awt.*;

class Driver
{
    public static void main (String [] args)
    {
        MyFrame mf = new MyFrame ();
    }
}
```


An Example Employing A List: Class MyFrame

```
import java.awt.*;
class MyFrame extends Frame
{
   private MyWindowListener mwl;
   private Label listLabel;
   private Label textLabel;
   private List list;
   private TextField text;
   private GridBagLayout gbl;
   private ListListener listListener;
```

An Example Employing A List: Class MyFrame (2)

```
public MyFrame ()
{
    mwl = new MyWindowListener ();
    addWindowListener(mwl);

    list = new List();
    initializeList();
    listListener = new ListListener();
    list.addActionListener(listListener);

    text = new TextField();
    text.setText(list.getSelectedItem());

    listLabel = new Label(Integer.toString(list.getItemCount()));
    textLabel = new Label("Currently selected item");
    gbl = new GridBagLayout();
    setLayout(gbl);     // Calling method of super class.
```


An Example Employing A List: Class MyFrame (3)

```
    addWidget(listLabel, 0, 0, 1, 1, GridBagConstraints.NONE);
    addWidget(textLabel, 2, 0, 1, 1, GridBagConstraints.NONE);
    addWidget(list, 0, 1, 1, 3, GridBagConstraints.HORIZONTAL);
    addWidget(text, 2, 1, 1, 1, GridBagConstraints.HORIZONTAL);

    setSize(300,200);
    setVisible(true);
}

public void initializeList ()
{
   int i;
   for (i = 1; i <= 10; i++)
   {
        list.add(new String(Integer.toString(i * 10)));
   }
}
```

An Example Employing A List: Class MyFrame (4)

```
    public void addWidget (Component widget, int x, int y, int w, int h, int fill)
    {
        GridBagConstraints gbc = new GridBagConstraints();
        gbc.gridx = x;
        gbc.gridy = y;
        gbc.gridwidth = w;
        gbc.gridheight = h;
        gbc.fill = fill;
        gbl.setConstraints(widget, gbc);
        add(widget);        // Calling method of super class.
    }

    public TextField getTextField () { return text; }
    public List getList () { return list; }
}
```


An Example Employing A List: Class ListListener

```
import java.awt.*;
import java.awt.event.*;

class ListListener implements ActionListener
{
   public void actionPerformed (ActionEvent e)
   {
       List list = (List) e.getSource ();
       MyFrame mf = (MyFrame) list.getParent();
       TextField text = mf.getTextField();
       text.setText(list.getSelectedItem());
   }
}
```

An Example Employing A List: Class WindowListener

```
import java.awt.event.*;
import java.awt.*;
class MyWindowListener extends WindowAdapter
{
  public void windowClosing (WindowEvent e)
  {
    MyFrame mf = (MyFrame) e.getWindow();
    List list = mf.getList();

    mf.setTitle("Closing window...");
    list.removeAll();
    for (int i = 0; i < 500000000; i++);
    mf.setVisible(false);
    mf.dispose();
    System.exit(0);
  }
}
```


Capturing TextField Events

The complete code for this example can be found in Unix under the path:
/home/233/examples/gui/exampleSeven

Capturing TextFieldEvents: Class Driver

```
import java.awt.*;

class Driver
{
    public static void main (String [] args)
    {
     MyFrame mf = new MyFrame ();
    }
}
```


Capturing TextFieldEvents: Class MyFrame

```
import java.awt.*;

class MyFrame extends Frame
{
   private Label instructions;
   private TextField input;
   private MyWindowListener mwl = new MyWindowListener();
```

Capturing TextFieldEvents: Class MyFrame (2)

```
public MyFrame ()
{
    MyTextFieldListener tfl = new MyTextFieldListener();
    mwl = new MyWindowListener ();
    addWindowListener(mwl);
    setLayout(null);
    setSize(300,200);
    instructions = new Label("Enter some text below and hit return");
    instructions.setBounds(20,100,200,20);
    input = new TextField();
    input.setBounds(20,150,200,20);
    input.addActionListener(tfl);
    add(instructions);
    add(input);
    setVisible(true);
  }
}  // End of class MyFrame
```


Capturing TextFieldEvents: Class MyTextFieldListener

```
import java.awt.event.*;
import java.awt.*;

class MyTextFieldListener implements ActionListener
{
  public void actionPerformed (ActionEvent e)
  {
    TextField tf = (TextField) e.getSource ();
    MyFrame mf = (MyFrame) tf.getParent ();
    mf.setTitle(tf.getText());
    System.out.println(tf.getText());
  }
}
```

Capturing TextFieldEvents: Class WindowListener

```
import java.awt.event.*;
import java.awt.*;

class MyWindowListener extends WindowAdapter
{
   public void windowClosing (WindowEvent e)
   {
      MyFrame mf = (MyFrame) e.getWindow();
      mf.setTitle("Closing window...");
      for (int i = 0; i < 500000000; i++);
      mf.setVisible(false);
      mf.dispose();
      System.exit(0);
   }
}
```


You Now Know

- The difference between traditional and event driven software
- How event-driven software works
- How some basic graphical controls work