

An Introduction To Graphical User Interfaces

The event-driven model

Building simple graphical interfaces in Java

James Tam

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>

James Tam

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

James Tam

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**



James Tam

Java GUI packages

1. Java classes for the components
 - 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.*)

James Tam

Traditional Software

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

James Tam

Traditional Software

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

Example

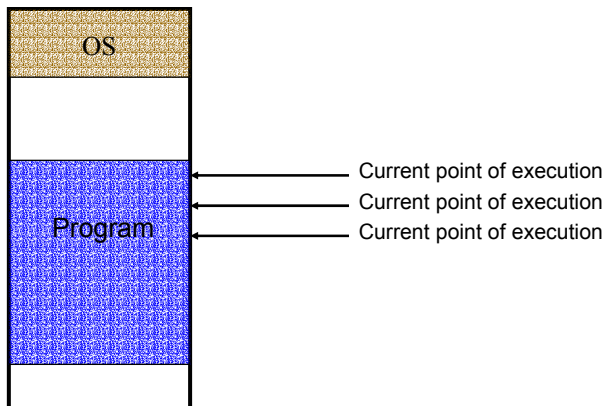
```
System.out.print("Enter student ID number: ");  
studentID = Console.in.readInt();
```

James Tam

Event-Driven Software

Program control *can also* be determined by events

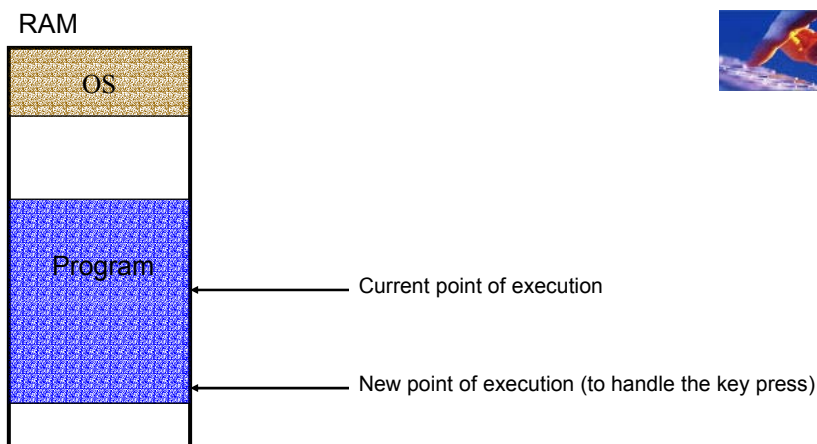
RAM



James Tam

Event-Driven Software

Program control *can also* be determined by events



James Tam

Characteristics Of Event Driven Software

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

James Tam

Most Components Can Trigger Events

Graphical objects can be manipulated by the user to trigger events.

Each object can have 0, 1 or many events that can be triggered.



James Tam

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 provides the code to handle the event.

James Tam

1. The Frame Must Register All Interested Event Listeners.

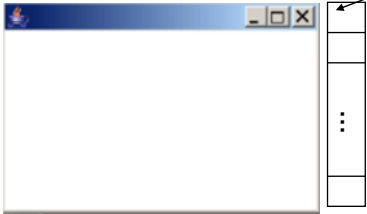


```
class MyWindowListener extends WindowAdapter
```

```
{
```

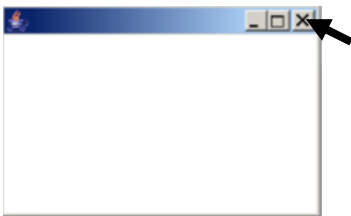
```
}
```

mf.addWindowListener(wl)



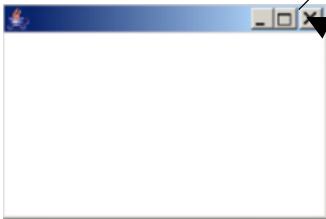
James Tam

2. The User Triggers The Event By Closing The Window



James Tam

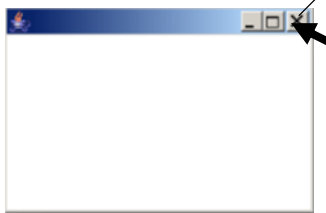
3. The Window Sends A Message To All Listeners Of That Event.



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

James Tam

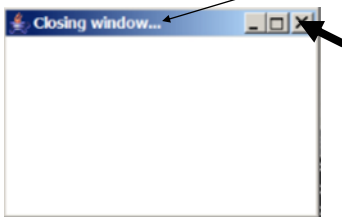
4. The Event Listener Provides The Code To Handle The Event.



```
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);
    }
}
```

James Tam

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

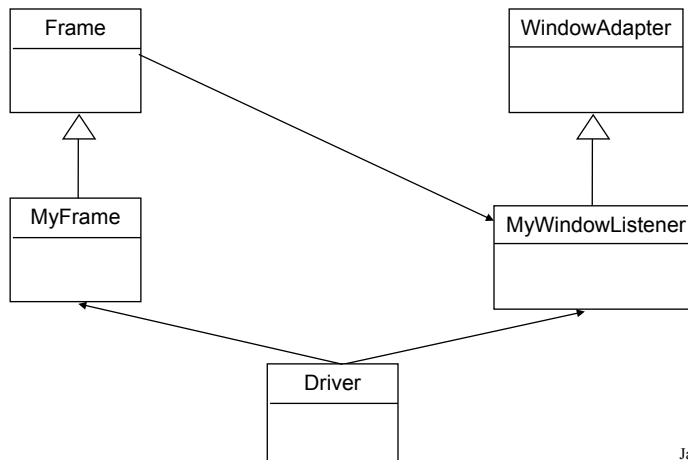


```
class MyWindowListener extends WindowAdapter
{
    public void windowClosing (WindowEvent e)
    {
        Frame f = (Frame) e.getWindow();
        f.setTitle("Closing window...");
        for (int i = 0; i < 500000000; i++);
        f.setVisible(false);
        f.dispose();
        System.exit(0);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

James Tam

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



James Tam

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);
    }
}
```

James Tam

An Example Of Handling A Frame Event: Class MyFrame

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

James Tam

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);
    }
}
```

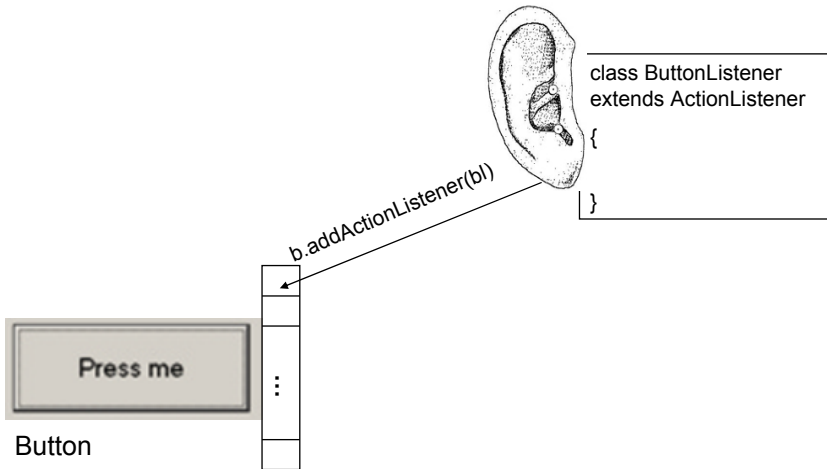
James Tam

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 provides code to handle the button press event.

James Tam

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



James Tam

2. The User Triggers An Event By Pressing The Button



James Tam

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

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



James Tam

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

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



James Tam

4. The Event Listener Provides Code To Handle The Event

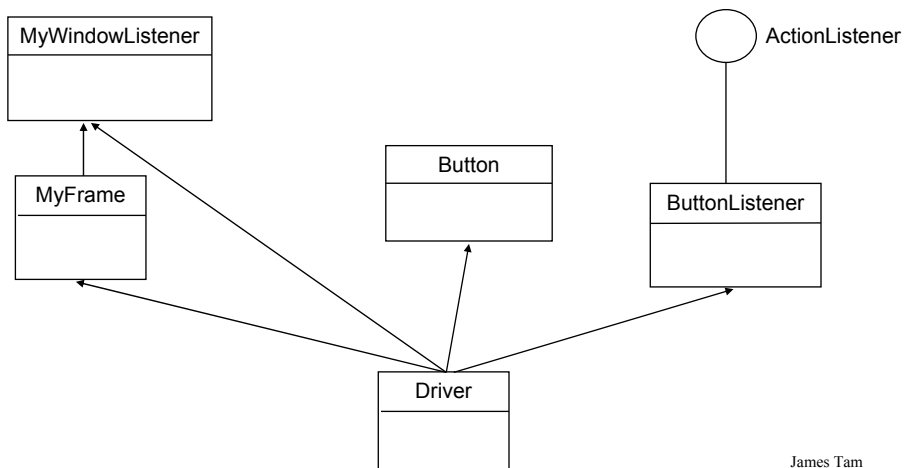
```
class MyListener extends ActionListener
{
    public void actionPerformed (ActionEvent e)
    {
        Button b = (Button) e.getSource();
        b.setLabel("Stop pressing me!");
    }
}
```



James Tam

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



James Tam

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);
    }
}
```

James Tam

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);
    }
}
```

James Tam

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!");
    }
}
```

James Tam

How To Handle The Layout Of Components

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

James Tam

How To Handle The Layout Of Components

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

James Tam

Layout Is Based On Spatial Coordinates

```
e.g. MyFrame my =new MyFrame ();  
my.setSize(300,200);
```

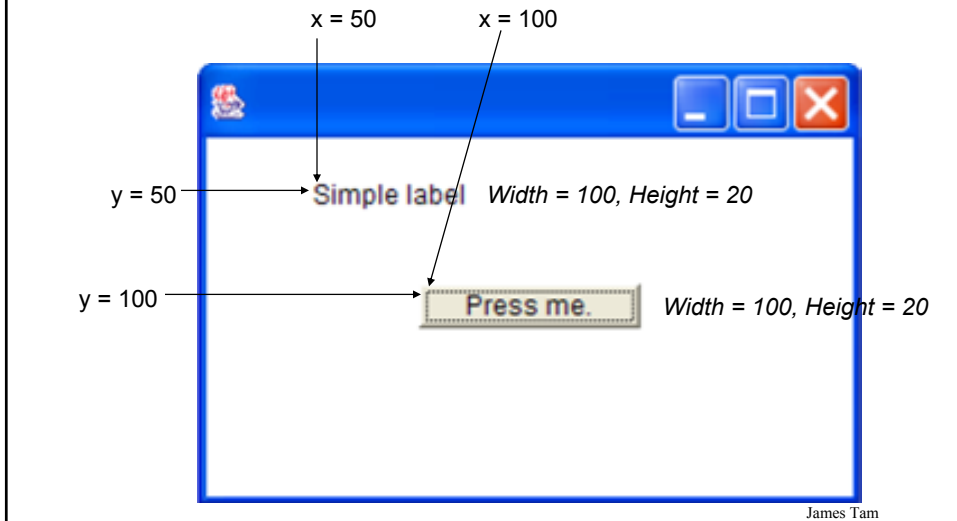


James Tam

Layout Is Based On Spatial Coordinates

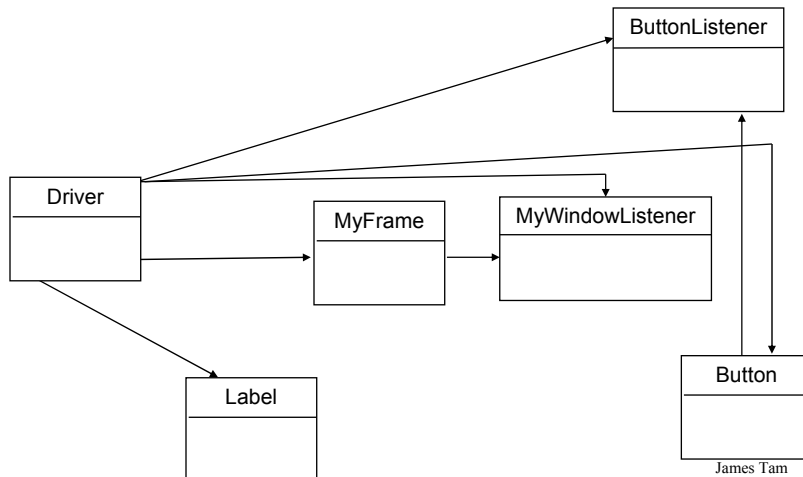


Coordinates Of Components: Relative To The Container



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.*;
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 b = new Button("Press me.");
        b.setBounds(100,100,100,20);
    }
}
```

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

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

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

mf.add(b);
mf.add(l);
mf.setVisible(true);
}
}
```

James Tam

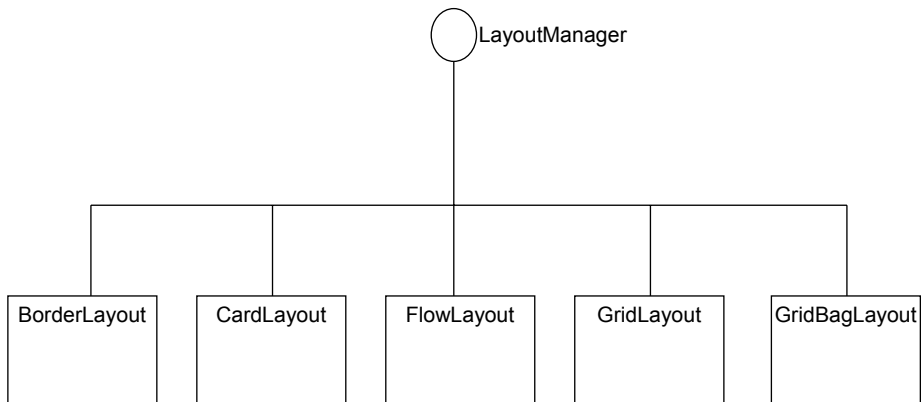
How To Handle The Layout Of Components

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

James Tam

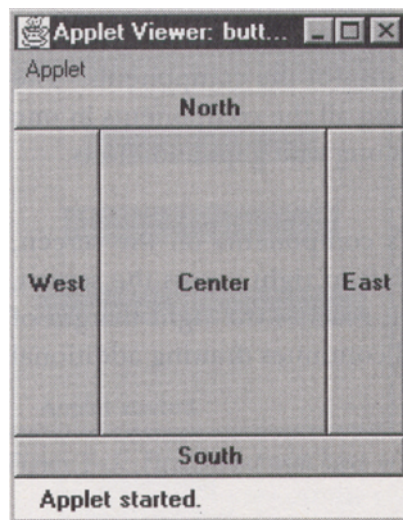
Java Layout Classes

Many implementations (these are only the original classes implemented by Sun).



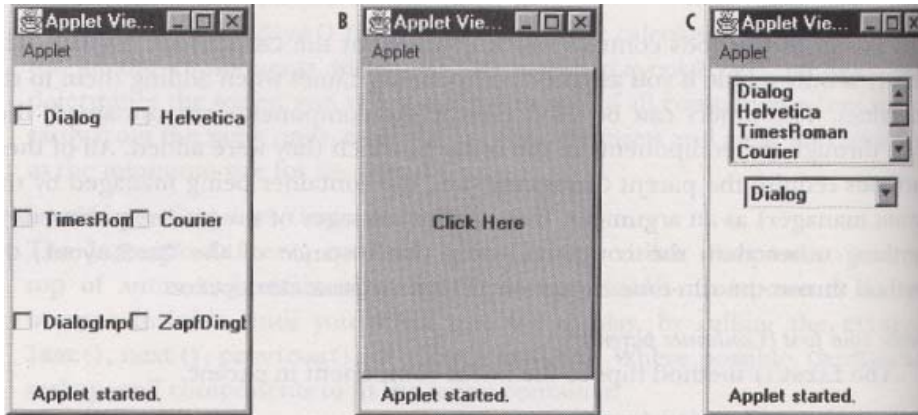
James Tam

BorderLayout



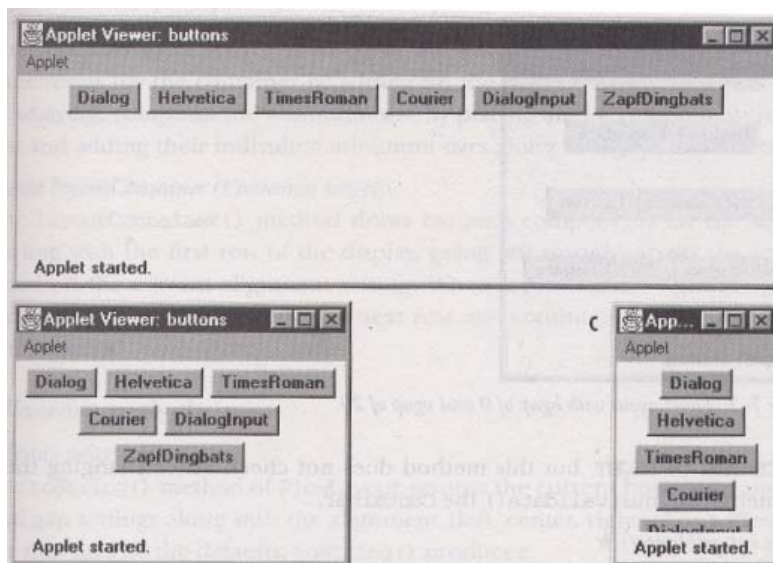
James Tam

CardLayout



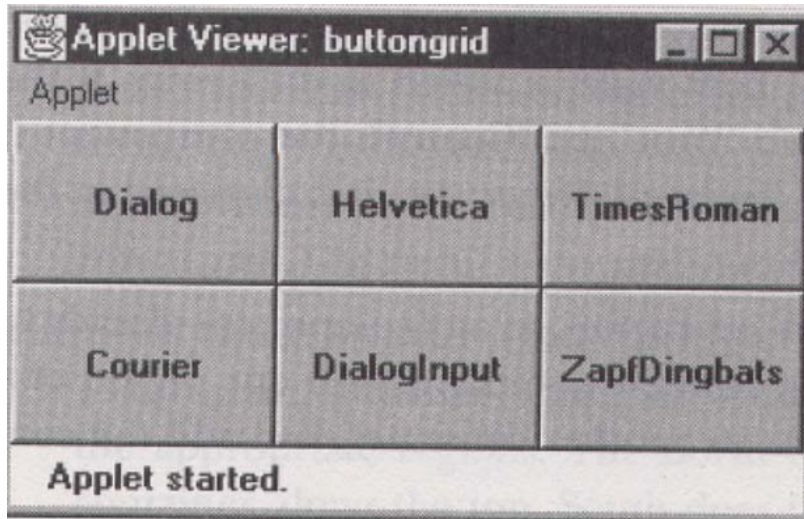
James Tam

FlowLayout



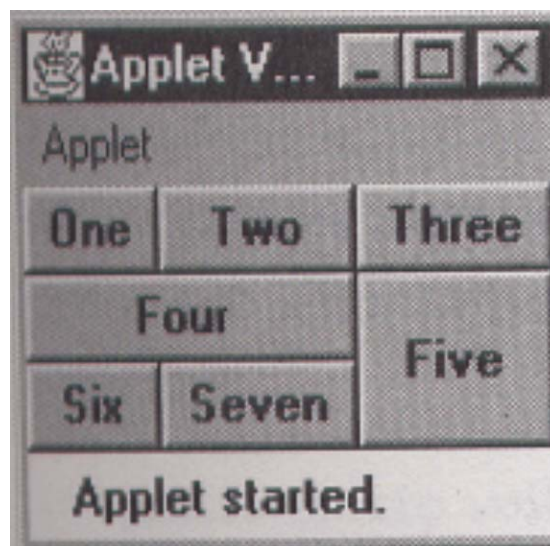
James Tam

GridLayout



James Tam

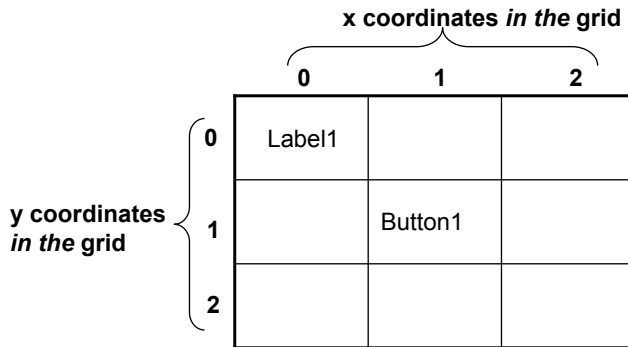
GridBagLayout



James Tam

Designing A GUI When Using The GridBagLayout

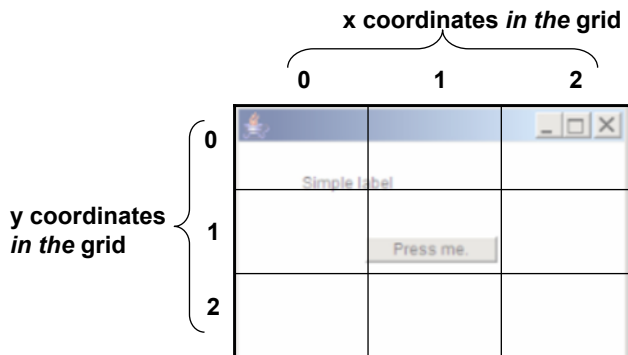
Use graph paper or draw out a table.



James Tam

Designing The GUI When Using The GridBagLayout

Use graph paper or draw out a table.



James Tam

GridBagConstraints

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

James Tam

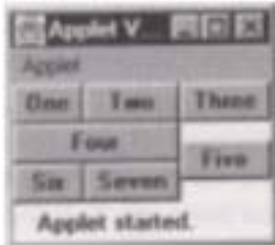
Some Important Parts Of The GridBagConstraints Class

```
class GridBagConstraints
{
    // Used in conjunction with the constants 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;
```

James Tam

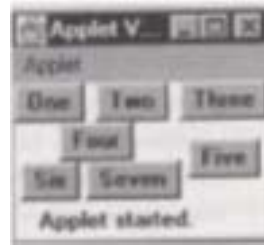
GridBagConstraints: Fill Values



Horizontal



Vertical



None

James Tam

More Important Parts Of The GridBagConstraints Class

// Position within the grid

```
public int gridx;
```

```
public int gridy;
```

// Number of grid squares occupied by a component

```
public int gridwidth;
```

```
public int gridheight;
```

James Tam

Some Important Parts Of The GridBagConstraints Class (3)

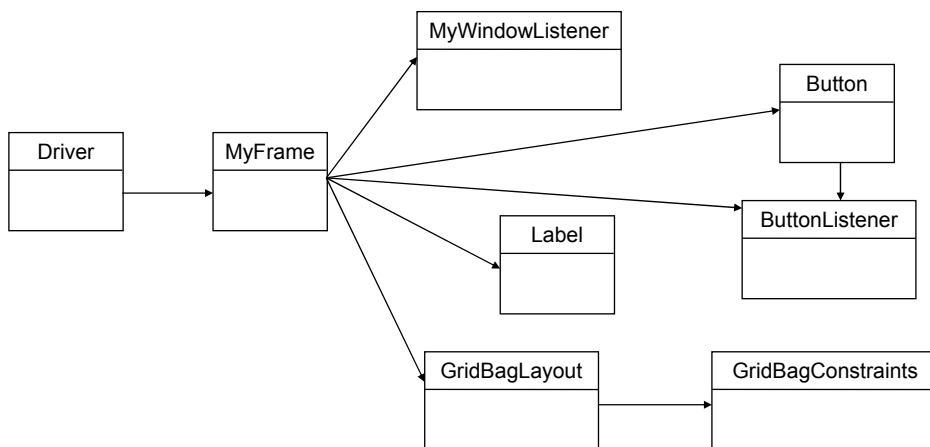
```
// Used in conjunction with the constants below to determine that the component
// drift if the space available is larger than the component.
public int anchor;

// Apply only if the component is smaller than the available space.
// Determine in which direction that the component will be anchored within the
// space.
public final static int CENTER;
public final static int EAST;
public final static int NORTH;
public final static int NORTHEAST;
public final static int ;
public final static int BOTH;
public final static int HORIZONTAL;
public final static int VERTICAL;
```

James Tam

An Example Using The GridBagLayout

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



James Tam

An Example Using The GridBagLayout: The Driver Class

```
import java.awt.*;

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

James Tam

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);
    }
}
```

James Tam

An Example Using The GridBagLayout: Class MyFrame (2)

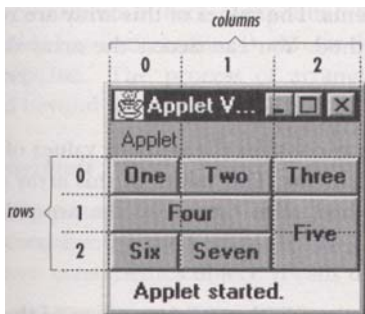
```

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.
}
}

```

James Tam

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	3

James Tam

Grid For Your Assignment

	0	1	2	3
0	Label ("Email contacts")			Label ("Enter new email below")
1	List (Personal contacts)	List (Personal contacts)	Button ("Add")	TextField (For new contact)
2	List (Personal contacts)	List (Personal contacts)		
3	List (Personal contacts)	List (Personal contacts)	Button ("Remove")	

James Tam

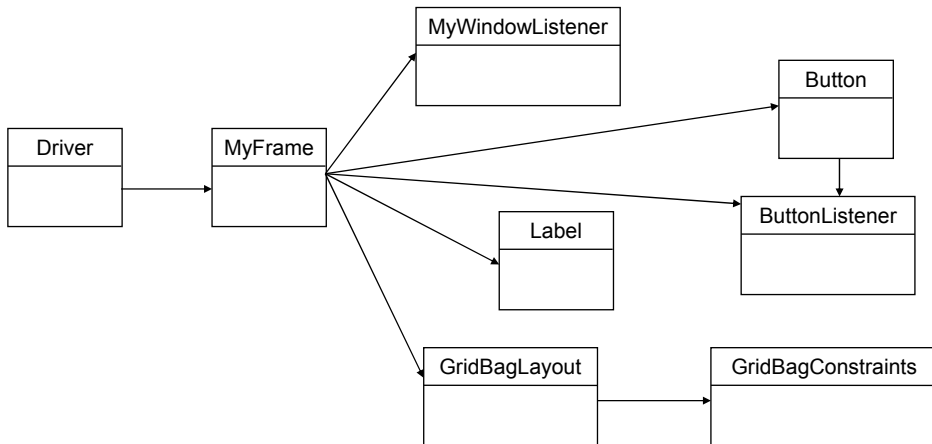
The Grid For Your Assignment (2)



James Tam

Having Controls Effect The State Of Other Components

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



James Tam

Having Controls Effect The State Of Other Components: The Driver Class

```
import java.awt.*;

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

James Tam

Having Controls Effect 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;
```

James Tam

Having Controls Effect The State Of Other Components: Class MyFrame (2)

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

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

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

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

James Tam

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

James Tam

Having Controls Effect 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; }
}
```

James Tam

Having Controls Effect The State Of Other Components: Class ButtonListener

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

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

James Tam

Having Controls Effect 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 her not me.");
    himButton.setBackground(Color.lightGray);
}
```

James Tam

Having Controls Effect 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 her not me.");
    herButton.setBackground(Color.lightGray);
}
}
}
```

James Tam

The List Class

- Used to provide a graphical and interactive 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>

James Tam

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 scrolling list
    public List ()

    // Creates a list with the specified no. 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)
```

James Tam

Some Important Parts Of The List Class (2)

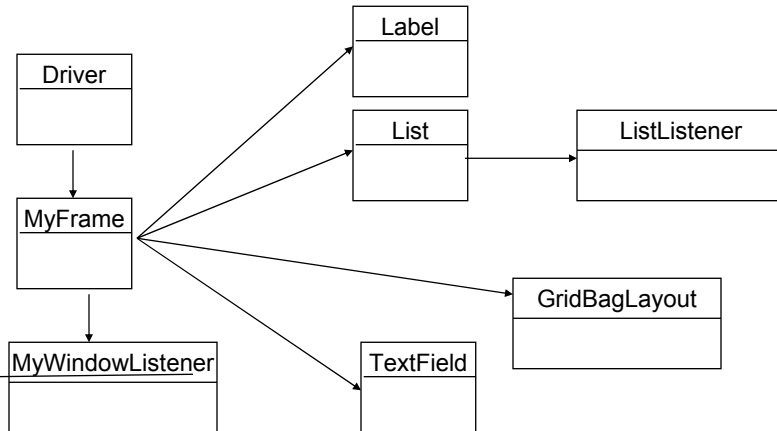
```
// Returns a count of the number of elements in the list.
public int getItemCount ()

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

James Tam

An Example Employing A List

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



James Tam

An Example Employing A List: The Driver Class

```
import java.awt.*;

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

James Tam

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;
```

James Tam

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.getSelectedItemAt());

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

James Tam

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)));
    }
}
```

James Tam

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; }
}
```

James Tam

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());
    }
}
```

James Tam

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);
    }
}
```

James Tam

You Now Know

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