Essentials of Information Technology

PC-CS-305

GUI using Button Panel Layout Manager

Objectives

Layout Managers

- Layout Manager Overview
- BorderLayout Manager
- FlowLayout Manager
- GridLayout Manager
- Other Layout Managers

gauravgambhir.cse@piet.co.in

Create and Add a Button

- Creating a Swing button is easy
  - `JButton b1 = new JButton("Submit");`
  - The parameter is displayed on the button face
- Adding a button to the frame is easy
  - `getContentPane().add(b1);`
  - This makes the button appear on the frame

JFrame Uses a Layout Manager

- The JFrame uses a layout manager to take care of positioning its graphical components
  - The JFrame's default manager is BorderLayout (more details later)
- The way we added the canvas and button did not take account of how the border layout manager works
  - This is why we have the problem

Layout Manager Overview

- A layout manager can be applied to any subclass of the `java.awt.Container` class
- To define the layout manager for the container, use:
  - `setLayout(new <layout class name>())`
- For example:
  - `getContentPane().setLayout(new BorderLayout())`

Layout Managers

What Are Layout Managers?

- Layout Managers are tools that helps ***place controls on a Container*** object.
- There are 5 standard layout managers
  - Flow Layout
  - Border Layout
  - Grid Layout
  - Card Layout
  - GridBag Layout
- Each of these layout managers is represented by a class of the same name.

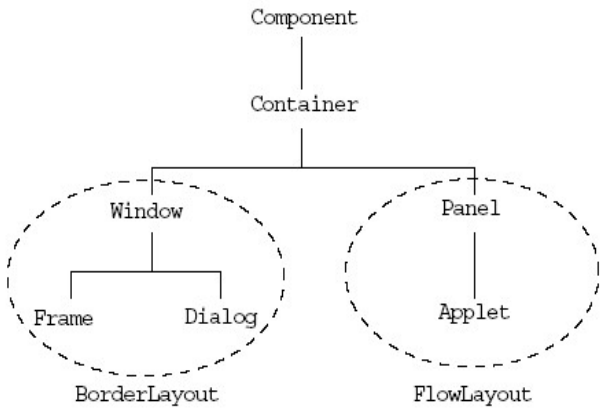
Layout Managers

How To Use Layout Managers?

- Each of all the layout managers is represented by a class of the same name.
- To create a layout manager for your applet :
  1. ***Create an instance*** of the appropriate layout class
  2. ***Call the setLayout() method*** to tell Java which layout object you want to use.

gauravgambhir.cse@piet.co.in

Default Layout Managers



gauravgambhir.cse@piet.co.in

Layout Managers - FlowLayout

- The ***FlowLayout*** manager places controls, in the ***order in which they're added***, one after the other in horizontal rows.
- In its default state, the ***FlowLayout*** manager ***centers*** controls on each row.
- However, you can set the alignment when you create the layout manager for your applet, like this:

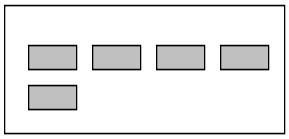
```
FlowLayout layout = new FlowLayout(align, hor, ver);
setLayout(layout);
```

- The ***FlowLayout*** constructor takes three arguments :
  - the alignment (FlowLayout.LEFT, FlowLayout.CENTER, or FlowLayout.RIGHT)
  - the horizontal spacing between components
  - the vertical spacing between components

gauravgambhir.cse@piet.co.in

The FlowLayout Manager

- Adds components from left to right, top to bottom
  - As the size of the container changes, components are redistributed
- The default layout manager for panels
- To add a component, c, to container, ct
  - `ct.add(c);`



gauravgambhir.cse@piet.co.in

Layout Managers - FlowLayout

Using A FlowLayout

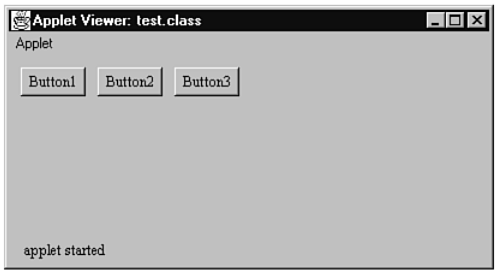
```
FlowLayout layout = new FlowLayout(FlowLayout.LEFT, 10, 10);
setLayout(layout);

button1 = new Button("Button1");
button2 = new Button("Button2");
button3 = new Button("Button3");

add(button1);
add(button2);
add(button3);
```

Example **FlowExample.java**

gauravgambhir.cse@piet.co.in

Layout Managers - FlowLayout

Output will look like this :



gauravgambhir.cse@piet.co.in

Flow Layout Manager

- Default layout for the Panel class
- Components added from left to right
- Default alignment is centered
- Uses components' preferred sizes
- Uses the constructor to tune behavior

gauravgambhir.cse@piet.co.in

Example

```
import java.awt.*;
public class FlowExample {
private Frame f;
private Button button1;
private Button button2;
private Button button3;
public FlowExample(){
        f = new Frame("Flow Layout");
       button1 = new Button("Ok");
       button2 = new Button("Open");
       button3 = new Button("Close");
}
 public void launchFrame(){
        f.setLayout(new FlowLayout());
       f.add(button1);
       f.add(button2);
```

gauravgambhir.cse@piet.co.in

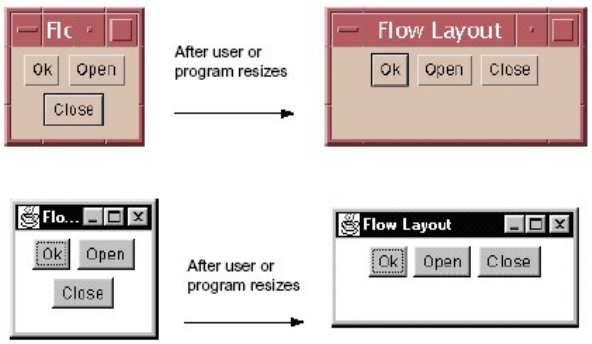
Example

```
    f.add(button3);
    f.setSize(100,100);
    f.setVisible(true);}

    public static void main(String args[]){
        FlowExample guiWindow = new FlowExample();
        guiWindow.launchFrame();
    }
}
```

See FlowExample.java

gauravgambhir.cse@piet.co.in

Output



gauravgambhir.cse@piet.co.in

Layout Managers - GridLayout

- The ***GridLayout*** manager organizes your applet's display into ***a rectangular grid***, similar to the grid used in a spreadsheet.
- Java then places the components you create for the applet into each cell of the grid, working from left to right and top to bottom
- You create a ***GridLayout*** manager like this:

```
GridLayout layout = new GridLayout(rows, cols, hor, ver);
setLayout(layout);
```

- The ***GridLayout*** constructor takes three arguments :
  - the number of rows
  - the number of columns
  - the horizontal space between the grid cells
  - the vertical space between the grid cells

gauravgambhir.cse@piet.co.in

Layout Managers - GridLayout

Using A GridLayout

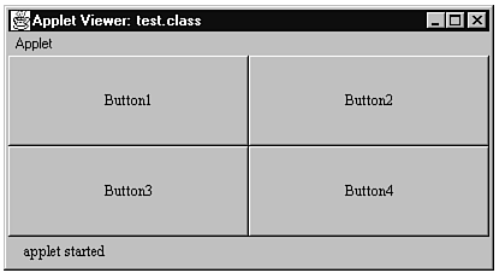
```
GridLayout layout = new GridLayout(2, 2, 0, 0);
setLayout(layout); button1 = new Button("Button1");

button2 = new Button("Button2");
button3 = new Button("Button3");
button4 = new Button("Button4");

add(button1);
add(button2);
add(button3);
add(button4);
```

Example GridExample.java

gauravgambhir.cse@piet.co.in

Layout Managers - GridLayout

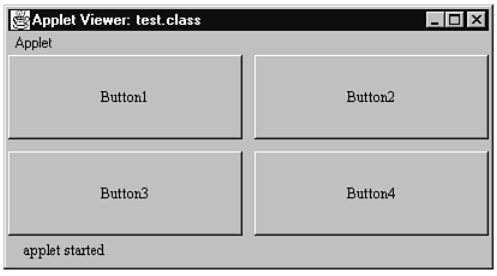
Output will look like this :



gauravgambhir.cse@piet.co.in

Layout Managers - GridLayout

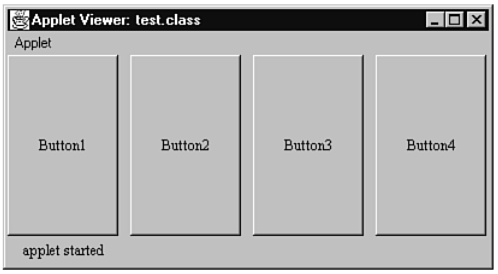
With horizontal and vertical spacing :



gauravgambhir.cse@piet.co.in

Layout Managers - GridLayout

With 1 row and 4 columns :



gauravgambhir.cse@piet.co.in

Layout Managers - BorderLayout

- The ***BorderLayout*** manager enables you to position components using the ***directions north, south, east, west, and center.***
- You create a ***BorderLayout*** manager like this:

```
BorderLayout layout = new BorderLayout(hor, ver);
setLayout(layout);
```

- The ***GridLayout*** constructor takes three arguments :
  - the horizontal spacing between the cells
  - the vertical spacing between the cells

Example BorderExample.java

gauravgambhir.cse@piet.co.in

Layout Managers - BorderLayout

- After you create the ***BorderLayout*** object, you must add the components using a ***different version of the add() method*** :

  add(position, object);

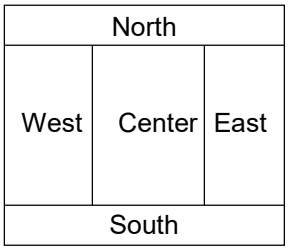
- ***Position*** is where to place the component and must be the string
  - ***North***
  - ***South***
  - ***East***
  - ***West***
  - ***Center***

- ***Object*** is the component you want to add to the applet

gauravgambhir.cse@piet.co.in

The BorderLayout Manager

- Divides the area it controls into five regions
  - North, South, East, West, Center (note the spelling!)
  - The regions vary in size according to what (if anything) they contain
  - Only one component per region



- We must specify the region when adding a component
  - `getContentPane().add("South", changeColour);`
  - Run ButtonDemo2 to see both canvas and button

gauravgambhir.cse@piet.co.in

Layout Managers - BorderLayout

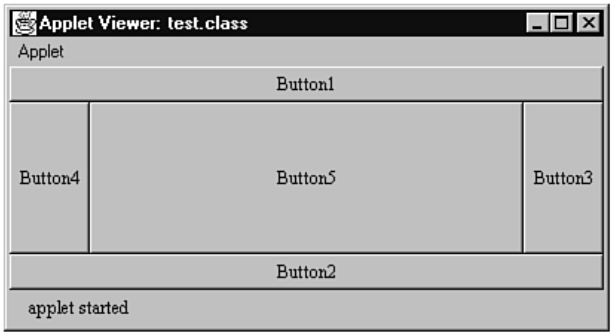
Using A BorderLayout

```
BorderLayout layout = new BorderLayout(0, 0);
setLayout(layout); button1 = new Button("Button1");

button2 = new Button("Button2");
button3 = new Button("Button3");
button4 = new Button("Button4");
button5 = new Button("Button5");

add("North", button1);
add("South", button2);
add("East", button3);
add("West", button4);
add("Center", button5);
```

gauravgambhir.cse@piet.co.in

Layout Managers - BorderLayout

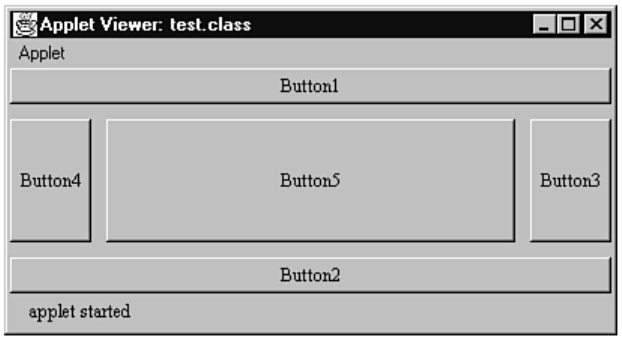
Output will look like this :



gauravgambhir.cse@piet.co.in

Layout Managers - BorderLayout

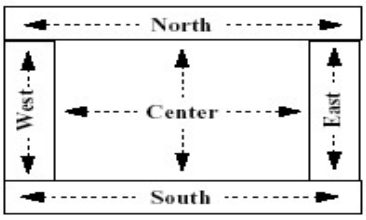
With horizontal and vertical spacing :



gauravgambhir.cse@piet.co.in

Border Layout Manager

- Default layout for the Frame class
- Components added to specific regions
- The resizing behavior:
  - North, South, and Center regions adjust horizontally
  - East, West, and Center regions adjust vertically



gambhir.cse@piet.co.in

Example

```
import java.awt.*;
public class BorderExample {
private Frame f;
private Button bn, bs, bw, be, bc;
public BorderExample(){
f = new Frame("Border Layout");
bn = new Button("B1");
bs = new Button("B2");
bw = new Button("B3");
be = new Button("B4");
bc = new Button("B5");
}
public void launchFrame(){
        f.add(bn, BorderLayout.NORTH);
        f.add(bs, BorderLayout.SOUTH);
        f.add(bw, BorderLayout.WEST);
        f.add(be, BorderLayout.EAST);
        f.add(bc, BorderLayout.CENTER);
```

gauravgambhir.cse@piet.co.in

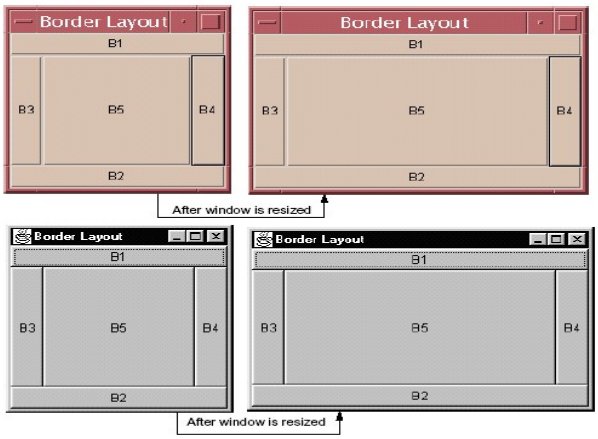
Example

```
f.setSize(200,200);f.setVisible(true);
  public static void main(String args[]){
  BorderExample guiWindow2 = new BorderExample();
guiWindow2.launchFrame();
 }
}
```

See BorderExample.java

gauravgambhir.cse@piet.co.in

Example



gauravgambhir.cse@piet.co.in

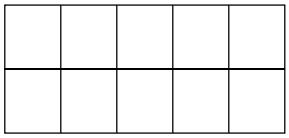
Remember

- Default Layout for Panel is FlowLayout
- Default Layout for Frame is BorderLayout

gauravgambhir.cse@piet.co.in

The GridLayout Manager

- Divides the container area into a specified number of rows and columns of equal size
  - `setLayout(new GridLayout(2, 5));`
- Adds components from left to right, top to bottom
  - Components completely fill each grid element
  - As the size of the container changes, grid elements are resized
- To add a component, c, to container, ct
  - `ct.add(c);`



gauravgambhir.cse@piet.co.in

Grid Layout Manager

- Components are added left to right, top to bottom
- All regions are equally sized
- The constructor specifies the rows and columns

gauravgambhir.cse@piet.co.in

Example

```
import java.awt.*;
public class GridExample {
private Frame f;
private Button b1, b2, b3, b4, b5, b6;
public GridExample() {

f = new Frame("Grid Example");
b1 = new Button("1");
b2 = new Button("2");
b3 = new Button("3");
b4 = new Button("4");
b5 = new Button("5");
b6 = new Button("6");
}
```

See GridEx.java

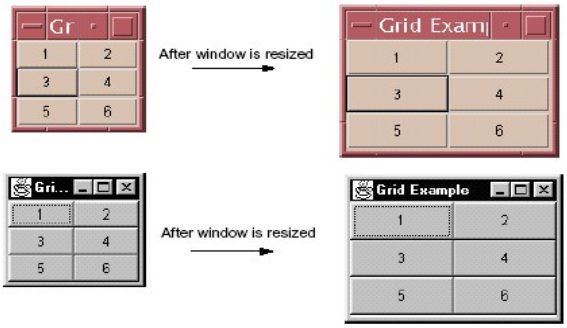
gauravgambhir.cse@piet.co.in

Example (Continued)

```
public void launchFrame(){
  f.setLayout (new GridLayout(3,2));
  f.add(b1);
  f.add(b2);
  f.add(b3);
  f.add(b4);
  f.add(b5);
  f.add(b6);
  f.pack();
  f.setVisible(true);}

  public static void main(String args[]){
     GridExample grid = new GridExample();
     grid.launchFrame();
   }
}
```

gauravgambhir.cse@piet.co.in

Output



gauravgambhir.cse@piet.co.in

Card Layout

The **CardLayout** class is unique among the other layout managers in that it stores several different layouts.

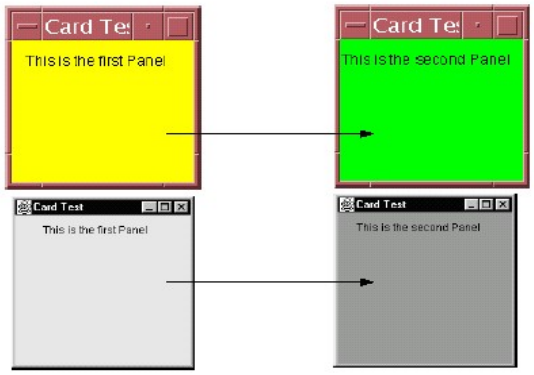
This can be useful for user interfaces with optional components that can be dynamically enabled and disabled upon user input.

The cards are typically held in an object of type **Panel**.

This panel must have **CardLayout** selected as its layout manager.

gauravgambhir.cse@piet.co.in

Output



gauravgambhir.cse@piet.co.in

Example

```
import java.awt.*;
import java.awt.event.*;
public class CardExample implements MouseListener {
private Panel p1, p2, p3, p4, p5;
private Label lb1, lb2, lb3, lb4, lb5;
// Declare a CardLayout object to call its methods.  private CardLayout
myCard;
private Frame f;
public CardExample(){
     f = new Frame ("Card Test");
myCard = new CardLayout();
p1 = new Panel();
p2 = new Panel();
p3 = new Panel();
p4 = new Panel();
p5 = new Panel();
```

gauravgambhir.cse@piet.co.in

Example (Continued)

```
lb1 = new Label("This is the first Panel");
lb2 = new Label("This is the second Panel");
lb3 = new Label("This is the third Panel");
lb4 = new Label("This is the fourth Panel");
lb5 = new Label("This is the fifth Panel");
}
public void launchFrame() {
 f.setLayout(myCard);
 // change the color of each panel, so they are
 // easily distinguishable
 p1.setBackground(Color.yellow);
 p1.add(lb1);
 p2.setBackground(Color.green);
 p2.add(lb2);
 p3.setBackground(Color.magenta);
 p3.add(lb3);
 p4.setBackground(Color.white);
 p4.add(lb4);
 p5.setBackground(Color.cyan);
 p5.add(lb5);
```

gauravgambhir.cse@piet.co.in

Example (Continued)

```
 // Set up the event handling here.
 p1.addMouseListener(this);
 p2.addMouseListener(this);
 p3.addMouseListener(this);
 p4.addMouseListener(this);
 p5.addMouseListener(this);

 // Add each panel to my CardLayout
 f.add(p1, "First");
 f.add(p2, "Second");
 f.add(p3, "Third");
 f.add(p4, "Fourth");
 f.add(p5, "Fifth");
 // Display the first panel.
 myCard.show(f, "First");
 f.setSize(200,200);
 f.setVisible(true)
}
```

gauravgambhir.cse@piet.co.in

Example (Continued)

```
        public void mousePressed(MouseEvent e) {
         myCard.next(f);
         }

         public void mouseReleased(MouseEvent e) { }
         public void mouseClicked(MouseEvent e) { }
         public void mouseEntered(MouseEvent e) { }
         public void mouseExited(MouseEvent e) { }

         public static void main (String args[]) {
         CardExample ct = new CardExample();
         ct.launchFrame();
         }
}
```

gauravgambhir.cse@piet.co.in

Other Layout Managers

- There are three other layout managers
  - java.awt.CardLayout
  - java.awt.GridBagLayout
  - javax.swing.BoxLayout

gauravgambhir.cse@piet.co.in

Other Layout Examples

- CardExample.java
- CardLayoutDemo.java

gauravgambhir.cse@piet.co.in

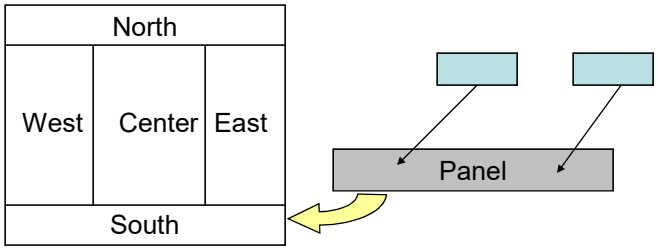
Panels

- We can only add one component to each region of a border layout
- To add more than one component in a region we need to use panels
- We have the choice of
  - `java.awt.Panel`
  - `javax.swing.JPanel`
- General rule: do not mix AWT and Swing components unless you have to

gauravgambhir.cse@piet.co.in

Using a JPanel

- Create the components and a JPanel object
- Add the components to the JPanel object
- Add the JPanel object to the container



gauravgambhir.cse@piet.co.in

A JPanel in the JFrame

- In the constructor function

```
changeColour = new JButton("Change colour");
JPanel p = new JPanel();
p.add(changeColour);
getContentPane().add("South", p);
```

gauravgambhir.cse@piet.co.in

Look and Feel

- In Java Swing, set the “look and feel” of the application before adding components

```
String lookAndFeel =
"com.sun.java.swing.plaf.windows.WindowsLookAndF
eel";

try
{
    UIManager.setLookAndFeel(lookAndFeel);
}
catch (Exception e)
{
    e.printStackTrace();
}
```

gauravgambhir.cse@piet.co.in

Summary of Main Teaching Points

- Described the AWT package and its components.
- Defined the terms containers, components, and layout managers, and how they work together to build a graphical user interface (GUI).
- Used layout managers
- Used the FlowLayout, BorderLayout, GridLayout, and CardLayout managers to achieve a desired dynamic layout.
- Added components to a container

gauravgambhir.cse@piet.co.in

Summary of Main Teaching Points

- Used the Frame and Panel containers appropriately
- Described how complex layouts with nested containers work.
- In a Java program, identified the following:
  - Containers
  - The associated layout managers
  - The layout hierarchy of all components

gauravgambhir.cse@piet.co.in

Question and Answer Session

Q & A

gauravgambhir.cse@piet.co.in