Layout Managers

So far we have let the applet determine the placement of components based on the size of the HTML host file. As you have discovered it has not been easy making your applet look the way you want it to.

To help with placement of components Java provides a Layout Manager that is provided in five different classes.

Layout Managers		
Layout Manager	**Component Handling**	**Features**
FlowLayout	Places components in rows from left to right	Three alignments; Left, Right Center
BorderLayout	Places components at compass Points	Placement of five locations; North, South, East, West and Center
GridLayout	Places components left to Right in a grid	You can specify the number of row and columns
CardLayout	Places components in a stack	Support for methods first() Last(), previous() and next()
GridBagLayout	Places components in a grids Components can very in size	You can specify grid and location of components

FlowLayout

The flowlayout is the default setting for panels and Applets if there hasn't been a layout manager specified. The FlowLayout places components from left to right and the height of the row is determined by the first component in the row.

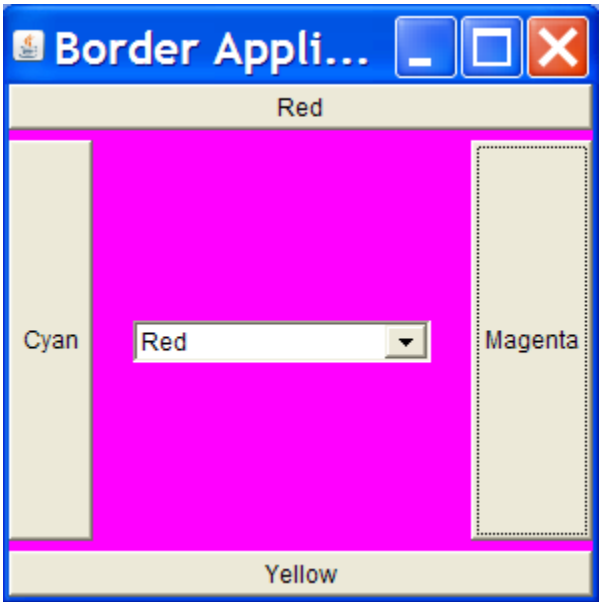
This example uses radio buttons for the components.

On line 13 the ColorButtons class extends Frame. This class is used to make the application GUI based like an applet and is often called a windowed application.

```
 9  import javax.swing.JOptionPane;
10  import java.awt.*;
11  import java.awt.event.*;
12
13  public class ColorButtons extends Frame implements ItemListener
14  {
15      static ColorButtons f = new ColorButtons();
16
17      CheckboxGroup options = new CheckboxGroup();
18          Checkbox blue = new Checkbox("Blue",false,options);
19          Checkbox red = new Checkbox("Red",false,options);
20          Checkbox yellow = new Checkbox("Yellow",false,options);
21          Checkbox pink = new Checkbox("Pink",false,options);
22          Checkbox gray = new Checkbox("Gray",true,options);
23
24
25      public ColorButtons()
26      {
27          this.setLayout(new FlowLayout());
28          add(blue);
29          add(red);
30          add(yellow);
31          add(pink);
32          add(gray);
33          blue.addItemListener(this);
34          red.addItemListener(this);
35          yellow.addItemListener(this);
36          pink.addItemListener(this);
37          gray.addItemListener(this);
38
39          //overriding windowClosing() allows user to click Close button
40          addWindowListener(
41              new WindowAdapter()
42              {
43                 public void windowClosing(WindowEvent e)
44                 {
45                      System.exit(0);
46                 }
47              }
48          );
49
50       } //end of constructor method
51
52       public static void main(String[] args)
53       {
54           //ColorButtons f = new ColorButtons();
55           f.setBounds(200,200,500,100);
56           f.setTitle("What's My Color?");
57           f.setVisible(true);
58       } //end of main
59
60       public void itemStateChanged(ItemEvent choice)
61       {
62           if (blue.getState()) f.setBackground(Color.blue);
63           else if (red.getState()) f.setBackground(Color.red);
64           else if (yellow.getState()) f.setBackground(Color.yellow);
65           else if (pink.getState()) f.setBackground(Color.pink);
66           else if (gray.getState()) f.setBackground(Color.gray);
67
68           repaint();
69
70       }//end of actionPerformed method
71
72  } //end of class
```

BorderLayout

The layout manager BorderLayout places the components in five regions; North, South, East, West, and Center. Components can be added in any order. On line 19 the BorderLayout sets the number of pixels between components, twenty pixels horizontally and five pixels vertically.



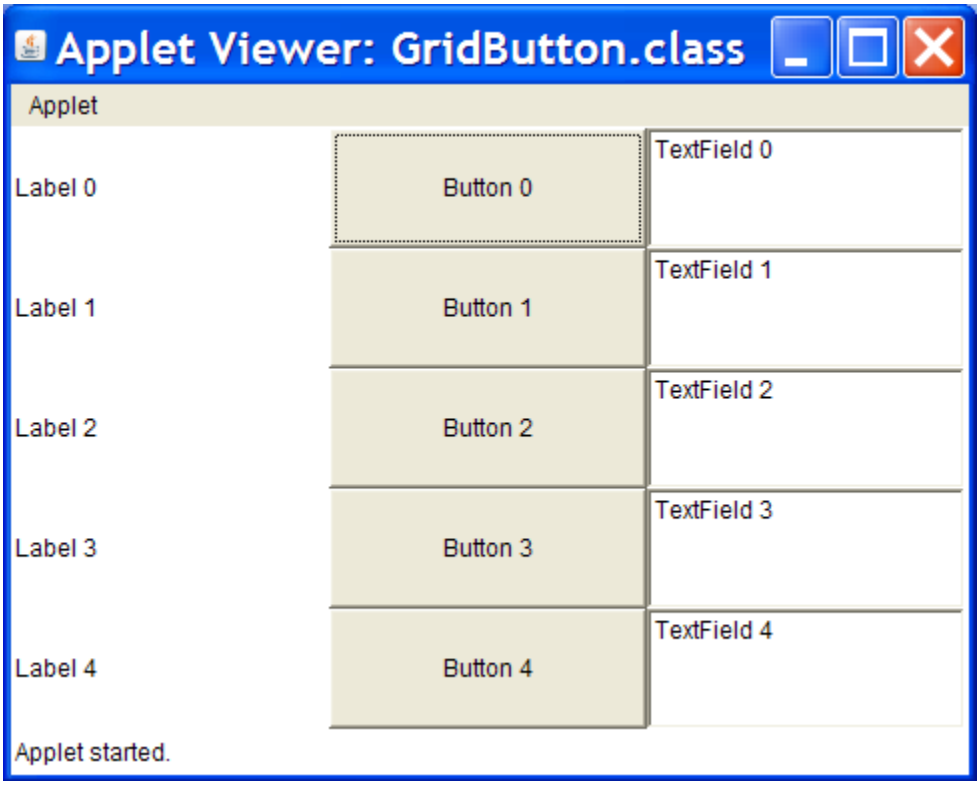

```
 9  import java.awt.*;
10  import java.awt.event.*;
11
12  public class Buttons extends Frame implements ActionListener, ItemListener
13  {
14      Choice colors = new Choice();
15
16      public Buttons()
17      {
18          //set the layout
19          setLayout(new BorderLayout(20,5));
20
21          colors.add("Red");
22          colors.add("Yellow");
23          colors.add("Magenta");
24          colors.add("Cyan");
25          colors.add("White");
26          colors.addItemListener(this);
27
28          //Add buttons
29          Button Red = new Button("Red");
30          Button Yellow = new Button("Yellow");
31          Button Magenta = new Button("Magenta");
32          Button Cyan = new Button("Cyan");
33          //Button White = new Button("White");
34
35          add(Red, BorderLayout.NORTH);
36          add(Yellow, BorderLayout.SOUTH);
37          add(Magenta, BorderLayout.EAST);
38          add(Cyan, BorderLayout.WEST);
39          //add(White, BorderLayout.CENTER);
40          add(colors, BorderLayout.CENTER);
41
42          Red.addActionListener(this);
43          Yellow.addActionListener(this);
44          Magenta.addActionListener(this);
45          Cyan.addActionListener(this);
46          //White.addActionListener(this);
47
```

```
48          //override the windowClosing event
49          addWindowListener(
50              new WindowAdapter()
51                  {
52                  public void windowClosing(WindowEvent e)
53                      {
54                          System.exit(0);
55                      }
56                  }
57          );
58      }
59
60      public static void main(String[] args)
61      {
62          // set frame properties
63          Buttons f = new Buttons();
64          f.setTitle("Border Application");
65          f.setBounds(200,200,300,300);
66          f.setVisible(true);
67          f.setBackground(Color.red);
68      }
69
70        public void actionPerformed(ActionEvent e)
71        {
72          //test for menu item clicks
73          String arg = e.getActionCommand();
74
75          if (arg == "Red")
76              setBackground(Color.red);
77
78          if (arg == "Yellow")
79              setBackground(Color.yellow);
80
81          if (arg == "Magenta")
82              setBackground(Color.magenta);
83
84          if (arg == "Cyan")
85              setBackground(Color.cyan);
86
87
88      }
89
90      public void itemStateChanged(ItemEvent ie)
91      {
92           String arg = colors.getSelectedItem();
93
94           if (arg == "Red")
95               setBackground(Color.red);
96
97           if (arg == "Yellow")
98               setBackground(Color.yellow);
99
100          if (arg == "Magenta")
101              setBackground(Color.magenta);
102
103          if (arg == "Cyan")
104              setBackground(Color.cyan);
105
106          if (arg == "White")
107              setBackground(Color.white);
108     }
109 }
```

GridLayout

The gridlayout manager divides the container into a grid so components can be placed in rows and columns. The grid runs from left to right and top to bottom within the grid.

In this example setLayout(new GridLayout(5, 3). Five rows and three columns. In the code below note that this one is an applet and not an application. This example also uses a loop to define the number of columns.



Pro

```
1  import java.applet.Applet;
2  import java.awt.*;
3  public class GridButton extends Applet
4  {
5    public void init()
6    {
7      // we must explicitly set GridLayout as the manager
8
9      setLayout (new GridLayout(5, 3));     // 5 rows, 3 columns, no gaps
10     for (int row = 0; row < 5; row ++)
11     {
12       add (new Label("Label " + row));
13       add (new Button("Button " + row));
14       add (new TextField("TextField " + row));
15     }
16   }
17 }
18
```

Programming Assignment

1. Recreate the FlowLayout application.
2. Recreate the BorderLayout application
3. Recreate the GridLayout applet
4. Experiment with any of the layouts by adding your own features. One example would be to add an action listener to the buttons in the GridLayout and have them add color or text to the textboxes.