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.

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

**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.
public class ColorButtons extends Frame implements ItemListener
{
    static ColorButtons f = new ColorButtons();
    CheckboxGroup options = new CheckboxGroup();
    Checkbox blue = new Checkbox("Blue",false, options);
    Checkbox red = new Checkbox("Red",false, options);
    Checkbox yellow = new Checkbox("Yellow",false, options);
    Checkbox pink = new Checkbox("Pink",false, options);
    Checkbox gray = new Checkbox("Gray",true, options);

    public ColorButtons()
    {
        this.setLayout(new FlowLayout());
        add(blue);
        add(red);
        add(yellow);
        add(pink);
        add(gray);
        blue.addItemListener(this);
        red.addItemListener(this);
        yellow.addItemListener(this);
        pink.addItemListener(this);
        gray.addItemListener(this);

        //overriding windowClosing() allows user to click Close button
        addWindowListener(
                new WindowAdapter()
                {
                    public void windowClosing(WindowEvent e)
                    {
                        System.exit(0);
                    }
                });
    }

    public static void main(String[] args)
    {
        //ColorButtons f = new ColorButtons();
        f.setBounds(200,200,500,100);
        f.setTitle("What's My Color?");
        f.setVisible(true);
    }

    public void itemStateChanged(ItemEvent choice)
    {
        if (blue.getState()) f.setBackground(Color.blue);
        else if (red.getState()) f.setBackground(Color.red);
        else if (yellow.getState()) f.setBackground(Color.yellow);
        else if (pink.getState()) f.setBackground(Color.pink);
        else if (gray.getState()) f.setBackground(Color.gray);
        repaint();
    }
}

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.

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

public class Buttons extends Frame implements ActionListener, ItemListener
{
    Choice colors = new Choice();

    public Buttons()
    {
        //set the layout
        setLayout(new BorderLayout(20, 5));

        colors.add("Red");
        colors.add("Yellow");
        colors.add("Magenta");
        colors.add("Cyan");
        colors.add("White");
        colors.addItemListener(this);

        //Add buttons
        Button Red = new Button("Red");
        Button Yellow = new Button("Yellow");
        Button Magenta = new Button("Magenta");
        Button Cyan = new Button("Cyan");
        //Button White = new Button("White");

        add(Red, BorderLayout.NORTH);
        add(Yellow, BorderLayout.SOUTH);
        add(Magenta, BorderLayout.EAST);
        add(Cyan, BorderLayout.WEST);
        //add(White, BorderLayout.CENTER);
        add(colors, BorderLayout.CENTER);

        Red.addActionListener(this);
        Yellow.addActionListener(this);
        Magenta.addActionListener(this);
        Cyan.addActionListener(this);
        //White.addActionListener(this);
    }
}
```
//override the windowClosing event
addWindowListener(
    new WindowAdapter()
    {
        public void windowClosing(WindowEvent e)
        {
            System.exit(0);
        }
    }
);

public static void main(String[] args)
{
    // set frame properties
    Buttons f = new Buttons();
    f.setTitle("Border Application");
    f.setBounds(200,200,300,300);
    f.setVisible(true);
    f.setBackground(Color.red);

}

public void actionPerformed(ActionEvent e)
{
    //test for menu item clicks
    String arg = e.getActionCommand();
    if (arg == "Red")
        setBackground(Color.red);
    if (arg == "Yellow")
        setBackground(Color.yellow);
    if (arg == "Magenta")
        setBackground(Color.magenta);
    if (arg == "Cyan")
        setBackground(Color.cyan);

}

public void itemStateChanged(ItemEvent e)
{
    String arg = colors.getSelectedItem();
    if (arg == "Red")
        setBackground(Color.red);
    if (arg == "Yellow")
        setBackground(Color.yellow);
    if (arg == "Magenta")
        setBackground(Color.magenta);
    if (arg == "Cyan")
        setBackground(Color.cyan);
    if (arg == "White")
        setBackground(Color.white);
}
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.

```java
import java.applet.Applet;
import java.awt.*;
public class GridButton extends Applet
{
    public void init()
    {
        // we must explicitly set GridLayout as the manager
        setLayout(new GridLayout(5, 3));  // 5 rows, 3 columns, no gaps
        for (int row = 0; row < 5; row ++)
        {
            add (new Label("Label "+ row));
            add (new Button("Button "+ row));
            add (new TextField("TextField "+ row));
        }
        Applet started.
    }
}
```
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.