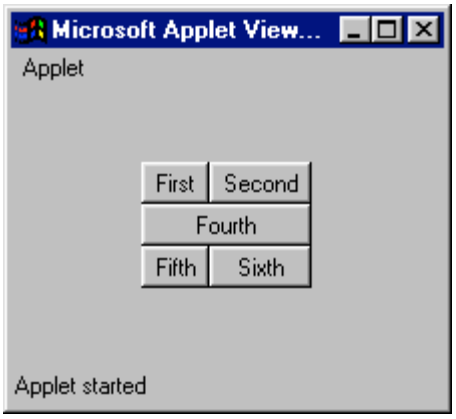
# Java GridBagLayout and Dialogs

Gridlayout allows positioning of components within a grid of columns and rows. GridBagLayout together with its pal GridBagConstraints allows a component to take up more than 1 cell in a grid, it is also possible to set up some empty cells allowing spaces within the grid.

The following code shows variation in a grid using GridBagLayout. The same Button object is placed more than once to illustrate the layout. The comments in **bold** will be used to demonstrate some layout variations

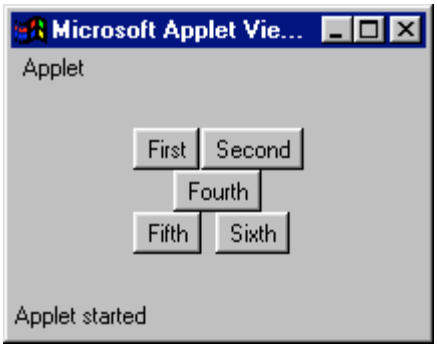
```
import java.awt.*;
import java.applet.*;

public class GridBag extends Applet
{
        public void init()
        {
                GridBagLayout gb = new GridBagLayout();
                GridBagConstraints gbc = new GridBagConstraints();
                //GridBagConstraints governs the x and y position
                //as well as the width within the grid
                Button b;

                setLayout(gb); //the default for an Applet is FlowLayout

                gbc.gridwidth=1;
                gbc.gridheight=1;
                gbc.gridx=0; //start at top corner
                gbc.gridy=0;
                gbc.fill= GridBagConstraints.HORIZONTAL;  1

                b=new Button("First");
                add(b,gbc);

                b= new Button("Second");
                gbc.gridx=1; //move along 1 cell
                gbc.gridwidth=2;  //this component has a double width
                add(b,gbc);

                b= new Button("Third");        2
                gbc.gridx=2;
                gbc.gridwidth=GridBagConstraints.REMAINDER; //fill to the edge
                add(b,gbc);

                b= new Button("Fourth");       3
                gbc.gridy=2; //down 1 row      4
                gbc.gridx=0; //back to x edge
                add(b,gbc);
```

```
            b= new Button("Fifth");
            gbc.gridy=3;  //down another row
            gbc.gridwidth=1; //set gridwidth back to 1
            add(b,gbc);

            b= new Button("Sixth");
            gbc.gridx=1;
            gbc.gridwidth=GridBagConstraints.REMAINDER;
            add(b,gbc);
        }
}
```




This is the output. The above code will be modified to show alternative layouts of the same components.

1. Change the constraints constant to VERTICAL and then NONE.




VERTICAL will not work without an anchor so the layout reverts to the default NONE. The components are centred within their allocated grid space. Fourth is now not in line with any other component.

Add the following line after 1.
gbc.anchor=GridBagConstraints.NORTHWEST;

Also try SOUTHEAST and WEST. Try GridBagConstraints.VERTICAL followed by the anchor for WEST.

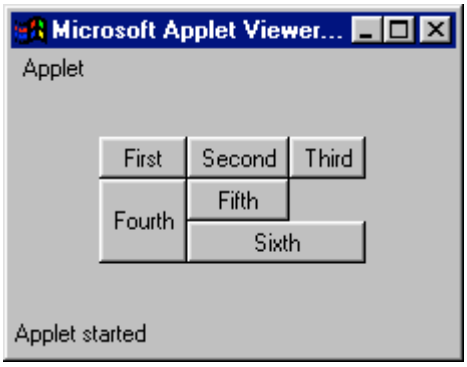
2. The third button is not shown. Modify the layout to show it at the end of the 1st line, three times as wide as first.

3. Make the fourth Button wider by using the padding methods;

```
gbc.ipadx=20;
```

gbc.ipady=20;

have the fifth and sixth Buttons set back to the default (0) padding.




4. Create the following layout

GridBagConstraints.REMAINDER

Will fill the Button fourth to the bottom of the grid, gridx and giddy will also need modifying.

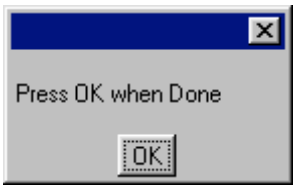
**Dialogs**

These objects require a Frame to run. The general Dialog class is not a lot of help but it can be extended to produce Dialog classes that will capture data or give feedback to the user. Once a suitable Dialog class has been created it, suitable objects can be created as required. GridBagLayout will be used to create a suitable layout for the Dialogs.

**OK Dialog**

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

public class MsgChoice extends Dialog implements ActionListener
{
      protected Button ok;
      protected Label msg;
     public MsgChoice(Frame parent, String message)
     {//constructor - true for modal Dialog
      super(parent, true); //call super class constructor
      GridBagLayout gb= new GridBagLayout();
      GridBagConstraints gbc= new GridBagConstraints();
      ok = new Button("OK");
      msg = new Label(message); //put message in Label

       setLayout(gb);  //GridBagLayout
      //set suitable layout with gb, add controls and ActionListener
     //pack() to have window take up the minimum size to display components
     pack();
    }
    public void actionPerformed(ActionEvent e)
    {
            if (e.getSource()==ok)
                  hide(); //hide dialog
   }
}
```

The new Dialog will have to be called from an Applet or Application using




```
Frame msg;
MsgChoice testD;
testD = new MsgChoice(msg,"Press OK when Done");
```

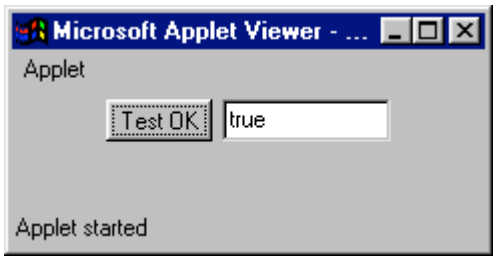
The constructor has used super to call the constructor for Dialog. This is because java expects a default constructor of the form MyClass(). This is usually inherited from the super class and then overridden. Unfortunately there is no appropriate constructor within Dialog and it would be bad form to open it up and write one. Any constructor of the form MsgChoice() will throw an error as there is no suitable method within Dialog to override. The use of super here restricts the ability to further subclass the Message Dialog.

Another problem results from having a Frame within an Applet. Frames are not as secure as Applets so java will put a note on the Frame to that effect.

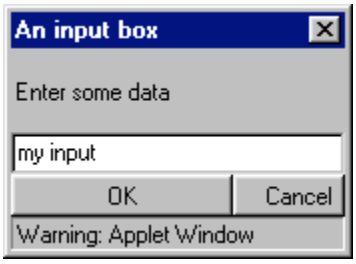
**Modifications**

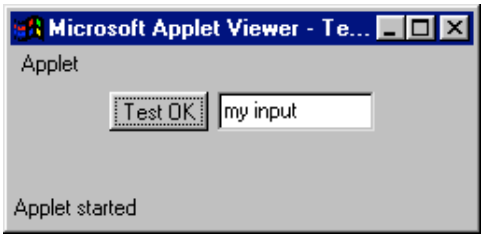
1. The Dialog has no title. The Dialog is displayed within a Frame and Frames have a setTitle(String) method. Allow a 2nd String to be passed to the Dialog, this will set a title on the Dialog.


2. Modify the Dialog to have 2 buttons. The result of pressing each will be passed back to the calling code as a Boolean value. In this example a TextField is used to test the code.


4. Create an input box dialog that will accept a String from the user and pass the data back with a get() method. The Dialog will have to allow for the user pressing cancel. The title and message are passed from the event that created the Dialog.

**Swing**

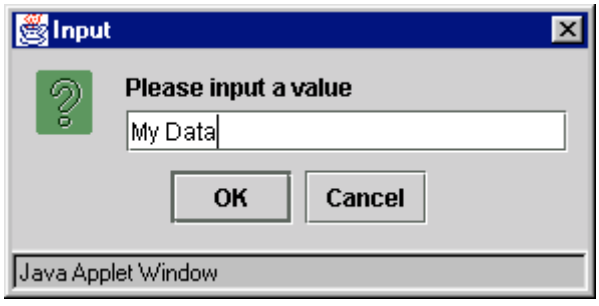
There is an easier way to do this but it requires the use of Swing. As this was introduced in JDK 1.1.6 it will not run in J++. The following code will need to be compiled in Forte or run with a Java 2 JRE. If run in a browser that too must support Java 2, any recent web browser will do so.

Here is the outline code for an applet:

```
import javax.swing.JOptionPane;

public class Dialog2 extends java.applet.Applet
{

   JOptionPane pane;
   public void init ()
   {
      pane = new JOptionPane();
pane.showMessageDialog(null, "alert", "alert", JOptionPane.ERROR_MESSAGE);
   }
}
```


Use the following code to show an input box type window and place the String entered in a TextField:

```
String inputValue = JOptionPane.showInputDialog("Please input a value");
```




The Java help files contain some more ready made examples. If Java help is set up in Forte Ctrl_F1 will offer help on the highlighted text.