GridBagLayout

With the GridBagLayout manager, a Container is divided into a grid of rectangular cells, in which a Component occupies all or a part of a contiguous group of cells.

The layout manager deduces the number of rows and columns of the grid from the layout information provided for the various components.

Position, size, and properties of components are determined by setting the GridBagConstraints of the GridBagLayout to particular values before the component is added to the container.

GridBagConstraints specify:

1. Location and size of the component on the grid.
2. Position and size of the component within the rectangular region specified in 1.
3. Behavior of the component and its region when the container is resized.

Steps for Using a GridBag

- Create a GridBagLayout object, say *gbl*, and set it to be the layout manager of the Container.

```
GridBagLayout gbl = new GridBagLayout();
setLayout(gbl);
```

- Create a GridBagConstraints object, say *gbc*, and provide values for its public instance variables for each Component, in turn, to be placed on the Container.

```
GridBagConstraints gbc = new GridBagConstraints();
```


- Place component using

```
gbl.setConstraints(component, gbc);
add(component);
```

or

```
add(component, gbc);
```

Note: GridBagConstaints object may be used for more than one component.

Instance Variables: java.awt.GridBagConstraints

int gridx (≥ 0)

Starting column of component

int gridy (≥ 0)

Starting row of component

int gridwidth (≥ 1)

Number of columns component occupies

int gridheight (≥ 1)

Number of rows component occupies

int fill

Indicates how the component is to be enlarged to fill its allocated region, one of NONE, BOTH, HORIZONTAL, or VERTICAL.

Default: NONE

If a component dimension is not covered by its fill value, then it takes its preferred size.

int anchor

Tells where the component is positioned in its allocated region, one of CENTER, NORTH, EAST, SOUTH, WEST, NORTHEAST, SOUTHEAST, SOUTHWEST, or NORTHWEST.

Default: CENTER

double weightx (≥ 0.0)

Indicates how to distribute extra horizontal space.

The weight of a column is the maximum *weightx* of its components.

If the layout is smaller horizontally than the allocated width of the region, the extra space is distributed to each column in proportion to its weight.

A column of zero weight receives no extra space.

If all weights are zero, all extra space appears between the grids of the cell and the left and right edges.

Default = 0.0.

double weighty (≥ 0.0)

See *weightx* and substitute vertical, row height, top and bottom.

Insets insets

Specifies the external padding for the component that it applies to, giving the minimum amount of space between it and the boundary of the adjacent components or the edges of the display area.

Use the constructor:

new Insets(**int** top, **int** left, **int** bottm, **int** right)

int ipadx

Specifies the horizontal internal padding of the component by increasing its preferred size.

The width of the component is at least its minimum width plus *2*ipadx* pixels (values may be negative).

int ipady

Specifies the vertical internal padding of the component by increasing its preferred size.

The height of the component is at least its minimum width plus *2*ipady* pixels.

The values *gridx*, *gridy*, *gridwidth*, *gridheigh*t, *anchor*, and *fill* determine the initial positions and sizes of the components.

When the container is enlarged, the extra pixels are allocated to the various components according to their values of *weightx* and *weighty*, which represent the proportion of the extra space each component receives.

That proportion is the *weightx* (or *weighty*) of the component divided by the sum of all of the components in its row (or column).

Regions that should not grow are given the value 0.0, which is the default, for *weightx* and *weighty*.

Example: Testing GridBag Constraints

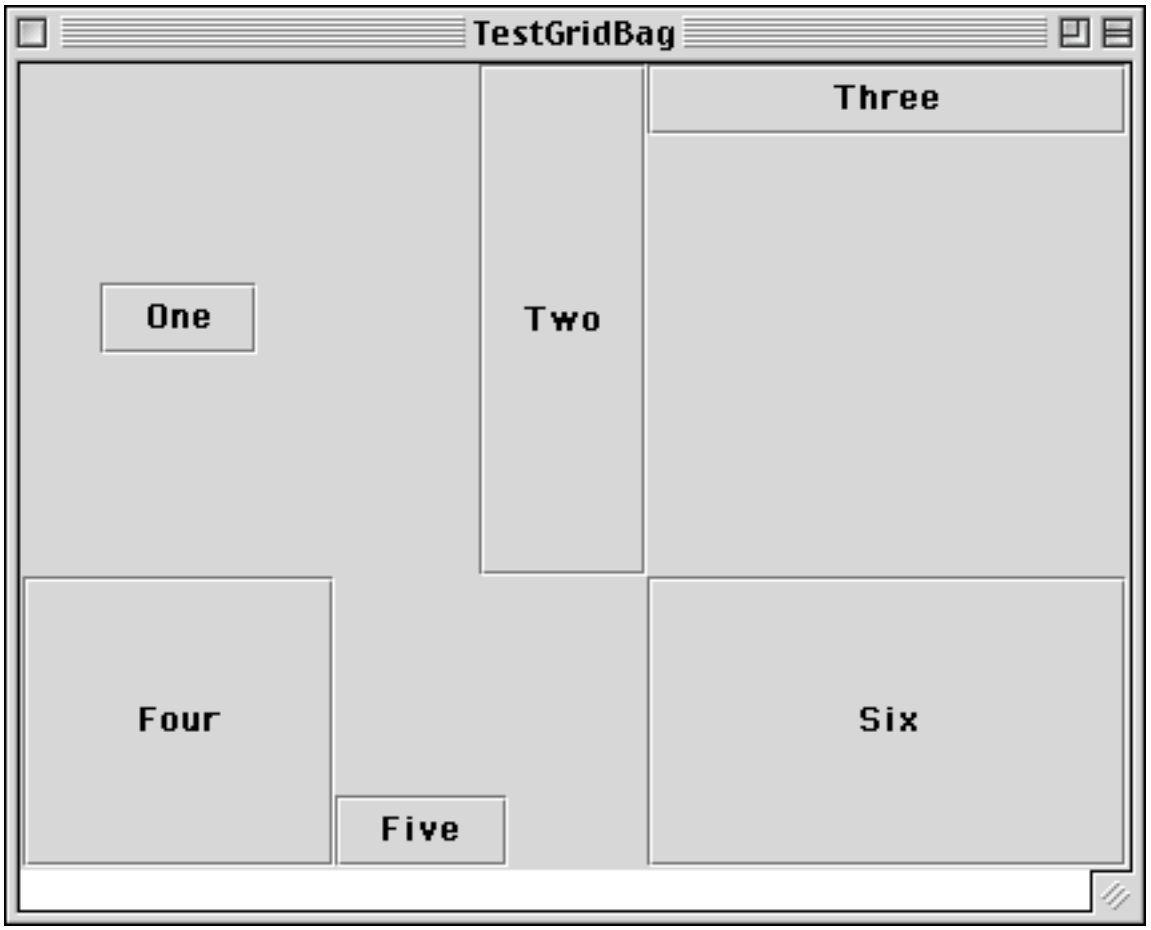
```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
```



```
public class TestGridBag extends JFrame
{
   TestGridBag()
   {
      setTitle("TestGridBag");
      setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      GridBagLayout layout = new GridBagLayout();
      getContentPane().setLayout(layout);
      GridBagConstraints cons = new GridBagConstraints();
      cons.gridx = 0;                  cons.gridy = 0;
      cons.gridwidth = 1;              cons.gridheight = 2;
      cons.fill = GridBagConstraints.NONE;
      cons.anchor =  GridBagConstraints.CENTER;
      cons.weightx = 1;                cons.weighty = 2;
      addButton("One", cons, layout);

      cons.gridx = 1;                  cons.gridy = 0;
      cons.gridwidth = 1;              cons.gridheight = 2;
      cons.fill = GridBagConstraints.VERTICAL;
      cons.anchor = GridBagConstraints.EAST;
      cons.weightx = 1;                cons.weighty = 2;
      addButton("Two", cons, layout);

      cons.gridx = 2;                  cons.gridy = 0;
      cons.gridwidth = 2;              cons.gridheight = 2;
      cons.fill = GridBagConstraints.HORIZONTAL;
      cons.anchor = GridBagConstraints.NORTH;
      cons.weightx = 2;                cons.weighty = 2;
      addButton("Three", cons, layout);

      cons.gridx = 0;                  cons.gridy = 2;
      cons.gridwidth = 1;              cons.gridheight = 1;
      cons.fill = GridBagConstraints.BOTH;
      cons.anchor =  GridBagConstraints.CENTER;
      cons.weightx = cons.weighty = 1;
```



```
        addButton("Four", cons, layout);

        cons.gridx = 1;                 cons.gridy = 2;
        cons.gridwidth = 1;             cons.gridheight = 1;
        cons.fill = GridBagConstraints.NONE;
        cons.anchor = GridBagConstraints.SOUTHWEST;
        cons.weightx = cons.weighty = 1;
        addButton("Five", cons, layout);

        cons.gridx = 2;                 cons.gridy = 2;
        cons.gridwidth = 1;             cons.gridheight = 1;
        cons.fill = GridBagConstraints.BOTH;
        cons.anchor =  GridBagConstraints.CENTER;
        cons.weightx = 2;               cons.weighty = 1;
        addButton("Six", cons, layout);
    }

    public static void main(String [] args)
    {
        JFrame f = new TestGridBag();
        Toolkit theKit = f.getToolkit();
        Dimension wndSize = theKit.getScreenSize();
        f.setBounds(wndSize.width/4, wndSize.height/4,
                    wndSize.width/2, wndSize.height/2);
        f.setVisible(true);
    }

    void addButton(String label,
                           GridBagConstraints constraints,
                                GridBagLayout layout)
    {
        JButton button = new JButton(label);
        layout.setConstraints(button, constraints);
        getContentPane().add(button);
    }
}
```



Example: Planning Retirement

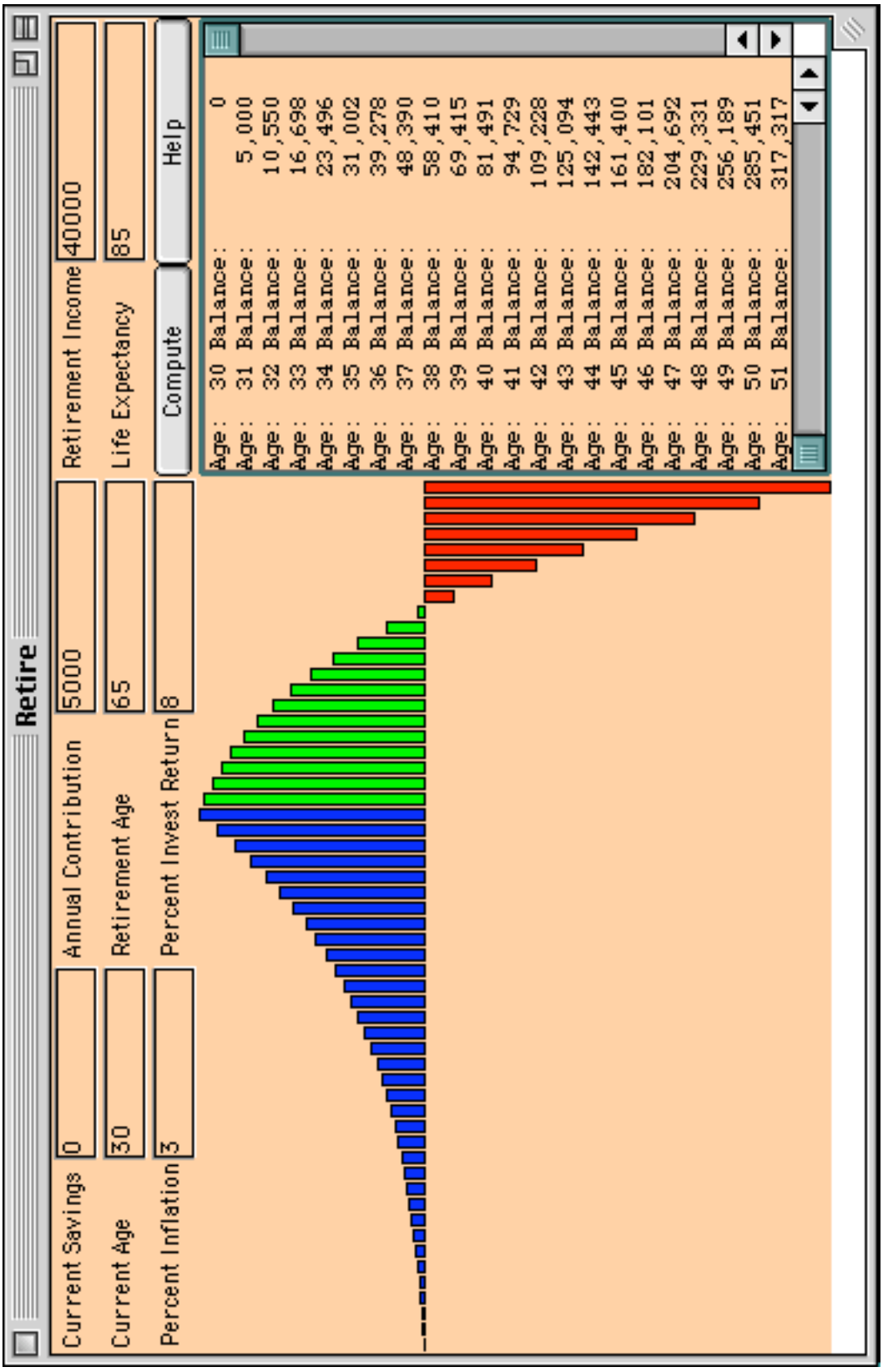
This application is a variation of an applet in the Horstmann and Cornell book (Java 1.1 version).

Changes include

- Writing code as an application instead of an applet.
- Removing the dependence on the package supplied in the text.

  In particular, the Java class DecimalFormat is used to format the numbers in the TextArea and we have our own IntTextField class.
- Computations are carried out so as to provide a more realistic accounting for inflation and investment return.

- Various parts of code are simpler than in the textbook.



The Code

```
import java.awt.*;
import java.awt.event.*;
import java.text.*;
import javax.swing.*;

public class MyRetire extends JFrame
                              implements ActionListener
{
   MyRetire()
   {
      JButton help, compute;

      setTitle("Retire");

      GridBagLayout gbl = new GridBagLayout();
      getContentPane().setLayout(gbl);

      GridBagConstraints gbc = new GridBagConstraints();

      gbc.weightx = 0;              gbc.weighty = 0;
      gbc.fill = GridBagConstraints.HORIZONTAL;

      add(new JLabel(" Current Savings"),gbl,gbc,0,0,1,1);
      add(savingsField, gbl, gbc, 1, 0, 1, 1);
      add(new JLabel(" Annual Contribution"), gbl, gbc, 2,0, 1,1);
      add(contribField, gbl, gbc, 3, 0, 1, 1);
      add(new JLabel(" Retirement Income"), gbl, gbc, 4, 0, 1, 1);
      add(incomeField, gbl, gbc, 5, 0, 1, 1);
//-----------------------------------------------------------------
      add(new JLabel(" Current Age"), gbl, gbc, 0, 1, 1, 1);
      add(currentAgeField, gbl, gbc, 1, 1, 1, 1);
      add(new JLabel(" Retirement Age"), gbl, gbc, 2,1, 1,1);
      add(retireAgeField, gbl, gbc, 3, 1, 1, 1);
      add(new JLabel(" Life Expectancy"), gbl, gbc, 4,1, 1,1);
      add(deathAgeField, gbl, gbc, 5, 1, 1, 1);
//-----------------------------------------------------------------
      add(new JLabel(" Percent Inflation"), gbl, gbc, 0,2, 1,1);
```



```
        add(inflationPercentField, gbl, gbc, 1, 2, 1, 1);
        add(new JLabel(" Prct Invest Return"), gbl, gbc, 2,2, 1,1);
        add(investPercentField, gbl, gbc, 3, 2, 1, 1);
        add(compute = new JButton("Compute"), gbl, gbc, 4,2, 1,1);
        compute.addActionListener(this);
        add(help = new JButton("Help"), gbl, gbc, 5, 2, 1, 1);
        help.addActionListener(this);
//-----------------------------------------------------------------
        gbc.fill = GridBagConstraints.BOTH;
        gbc.weightx = 100;  gbc.weighty = 100;
        add(retireCanvas, gbl, gbc, 0, 3, 4, 8);
        add(retirePane, gbl, gbc, 4, 3, 2, 8);
        text.setEditable(false);
        text.setFont(
                    new Font("Monospaced",Font.PLAIN,10));
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }

    private void add(Component c, GridBagLayout gbl,
            GridBagConstraints gbc, int x, int y, int w, int h)
    {
        gbc.gridx = x;                  gbc.gridy = y;
        gbc.gridwidth = w;              gbc.gridheight = h;
        gbl.setConstraints(c, gbc);
        getContentPane().add(c);
    }

    public void actionPerformed(ActionEvent evt)
    {
        String arg = evt.getActionCommand();
        if (arg.equals("Compute"))
        {
            RetireInfo info = new RetireInfo();
            info.savings = savingsField.getValue();
            info.contrib = contribField.getValue();
            info.income = incomeField.getValue();
            info.currentAge = currentAgeField.getValue();
```



```
      info.retireAge = retireAgeField.getValue();
      info.deathAge = deathAgeField.getValue();
      info.inflationPercent = inflationPercentField.getValue();
      info.investPercent = investPercentField.getValue();
      text.setText("");
      for (int a = info.currentAge; a <= info.deathAge; a++)
      {
         text.append(formatAge(a) +
               formatBalance(info.getBalance(a)) + "\n");
      }
      retireCanvas.redraw(info);
   }
   else if (arg.equals("Help"))
   {  text.setText("This is a modified version\n"
      + "of code by Cornell-Horstmann\n(Core Java).\n\n"
      + "Click Compute button \nafter entering valid data \n"
      + "in textfields.\n\nRetirement income and \n"
      + "annual contribution are\nadjusted to reflect \n “
      + “yearly inflation.\n\nBalances shown in table \n"
      + "and graph are given in \nactual dollars. \n" );

   }
}

String formatAge(int a)
{  String val = "" + a;
   return "Age: " + "  ".substring(0,3-val.length()) + val;
}

String formatBalance(int b)
{  NumberFormat nf =
         new DecimalFormat("##,###,##0;-##,###,##0");
   String val = nf.format(b);
   return " Balance:"
                  + "         ".substring(0,11-val.length()) + val;
}
public static void main(String [] a)
{  JFrame f = new MyRetire();
```



```
      f.setSize(600,350);              f.setVisible(true);
   }
   private IntTextField savingsField = new IntTextField(0, 10);
   private IntTextField contribField =
                                    new IntTextField(5000, 10);
   private IntTextField incomeField =
                                    new IntTextField(40000, 10);
   private IntTextField currentAgeField =
                                        new IntTextField(30, 4);
   private IntTextField retireAgeField =
                                        new IntTextField(65, 4);
   private IntTextField deathAgeField =
                                        new IntTextField(85, 4);
   private IntTextField inflationPercentField =
                                        new IntTextField(3, 4);
   private IntTextField investPercentField =
                                        new IntTextField(8, 4);
   private RetireCanvas retireCanvas = new RetireCanvas();
   private JTextArea text = new JTextArea(10, 25);
   private JScrollPane retirePane = new JScrollPane(text);
}
/*****************************************************************/
class IntTextField extends JTextField
{
   IntTextField(int defval, int size)
   {
      super("" + defval, size);
      lastValue = "" + defval;
   }
   private void checkValue()
   {
      try
      {  Integer.parseInt(getText().trim() + "0");
         lastValue = getText();
      }
```

```
      catch(NumberFormatException e)
      {  setText(lastValue);  }
   }

   int getValue()
   {
      checkValue();
      try
      {  return Integer.parseInt(getText().trim());
      }
      catch (NumberFormatException e)
      {  return 0;  }
   }

   private String lastValue;
}

/*****************************************************************/

class RetireInfo
{                                        // Works correctly only
   int getBalance(int year)              // if called for an increasing
   {                                     // series of consecutive
      if (year == currentAge)            // years starting
         balance = savings;              // with currentAge
      else if (year <= retireAge)
      {
         balance = (int)(balance * (1 + investPercent/100.0));
         balance = balance +
            (int)(contrib*Math.pow((1+inflationPercent/100.0),
                                          year-currentAge-1));
      }
      else
      {
         balance = balance -
         (int)(income * Math.pow((1+inflationPercent/100.0),
                                          year-currentAge-1));
         balance = (int)(balance*(1 + investPercent/100.0));
      }
```



```
      return balance;
   }

   int savings, contrib, income, currentAge;
   int retireAge, deathAge, inflationPercent;
   int investPercent, balance;
}
/*************************************************************/

class RetireCanvas extends JPanel
{
   void redraw(RetireInfo newInfo)
   {
      info = newInfo;        repaint();
   }

   public void paintComponent(Graphics g)
   {
      super.paintComponent(g);
      if (info == null) return;
      int minValue = 0, maxValue = 0;
      for (int yr=info.currentAge; yr<=info.deathAge; yr++)
      {
         int v = info.getBalance(yr);
         if (minValue > v) minValue = v;
         if (maxValue < v) maxValue = v;
      }
      if (maxValue == minValue) return;          // avoid div by 0
      Dimension d = getSize();
      int barWidth =
            d.width/(info.deathAge-info.currentAge+1);
      double scale =
               (double)d.height/(maxValue-minValue);
```



```
      for (int yr=info.currentAge; yr<=info.deathAge; yr++)
      {
         int x1 = (yr - info.currentAge)*barWidth + 1;
         int y1, height;
         int v = info.getBalance(yr);
         int yOrigin = (int)(maxValue*scale);
         if (v >= 0)
         {
            y1 = (int)((maxValue - v) * scale);
            height = yOrigin - y1;
         }
         else
         {  y1 = yOrigin;
            height = (int)(-v * scale);
         }
         if (yr <= info.retireAge) g.setColor(Color.blue);
         else if (v >= 0) g.setColor(Color.green);
         else  g.setColor(Color.red);
         g.fillRect(x1, y1, barWidth - 2, height);
         g.setColor(Color.black);
         g.drawRect(x1, y1, barWidth - 2, height);
      }  // end for yr
   }
   private RetireInfo info = null;
}
```


A GridBag Laboratory

This example provides a mechanism for testing variations in a GridBagLayout.

The initial JFrame allows the user to enter *gridx*, *gridy*, *gridwidth*, *gridheight*, *weightx*, *weighty*, *fill*, and *anchor* values for up to six Button objects.

When the SHOW button is pressed, the layout manager places the buttons onto another JFrame according to the specifications provided.

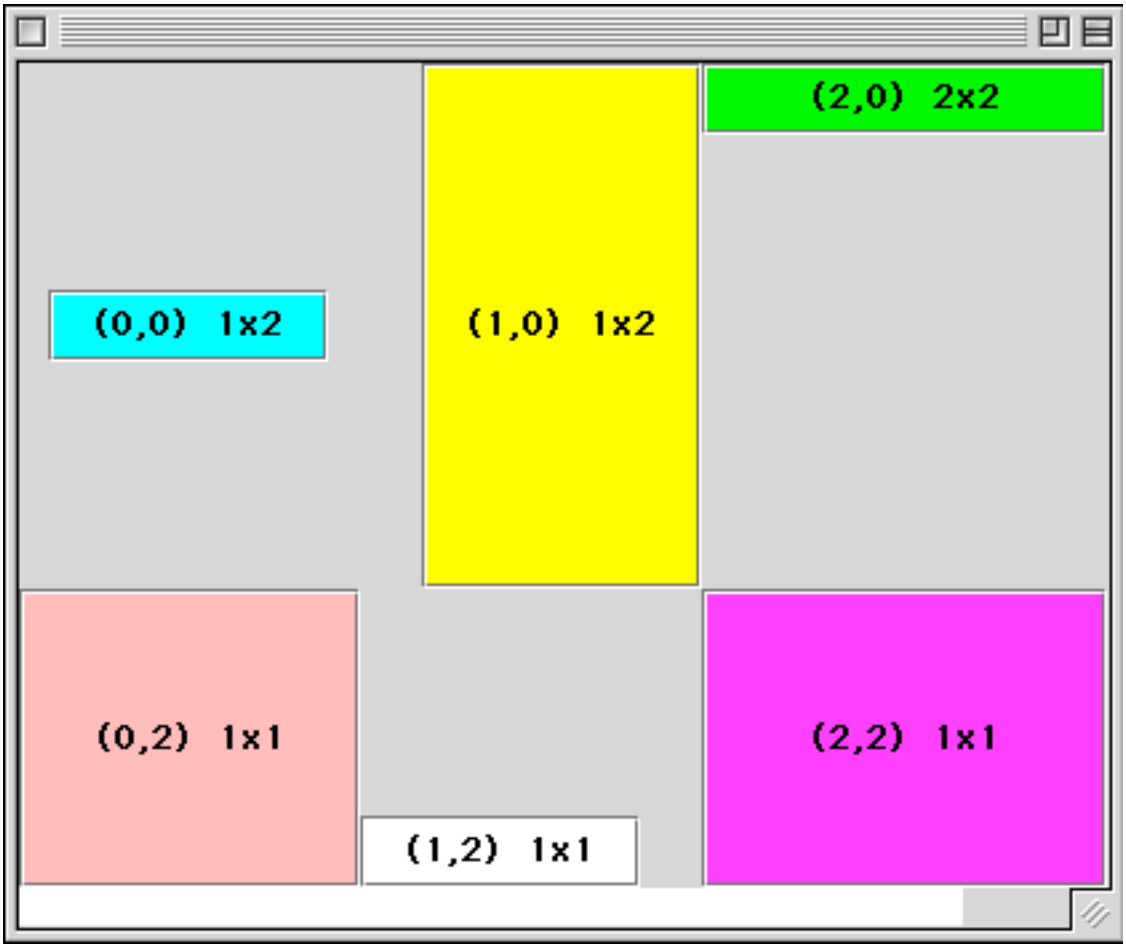
The next page shows the initial frame with values set for all six buttons.

The following page shows the resulting frame.

Each button is labeled by its xy coordinates and its size in grid cells.

The Code

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

public class GridBagLab extends JFrame
                        implements ActionListener
{
   final static Color [] colors =
            {  Color.cyan, Color.yellow, Color.green,
               Color.pink, Color.white, Color.magenta };

   int maxButtons = colors.length;
   JButton showButton;
   GBPanel [] gbpanels;
```

The center of the initial JFrame is a JPanel with six sub-panels arranged in a six by one grid and stored in an array.

The Code Continued

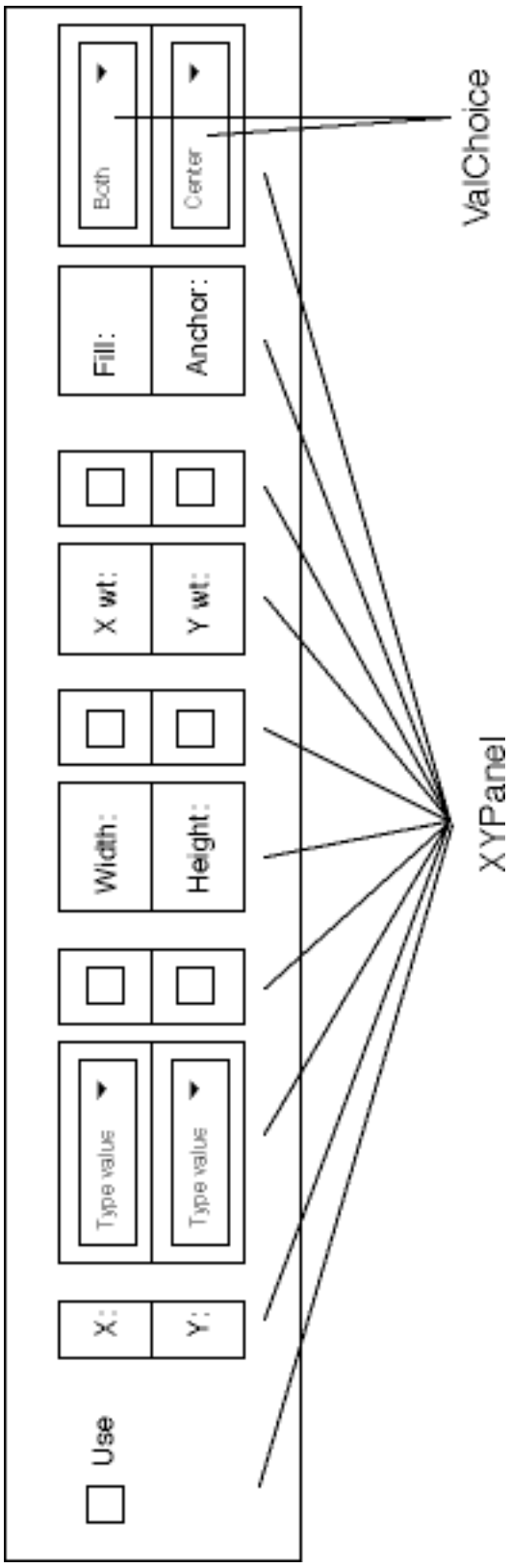
```
GridBagLab()
{
   JPanel p = new JPanel();
   JLabel label = new JLabel(
                     "Configure parameters; click SHOW.");
   label.setFont(new Font("SansSerif", Font.BOLD, 14));
   p.add(label);
   getContentPane().add(p, "North");
```

```
      JPanel gridPanel = new JPanel();
      gridPanel.setLayout(new GridLayout(maxButtons, 1));
      gbpanels = new GBPanel[maxButtons];

      for (int k=0; k<maxButtons; k++)
      {  gbpanels[k] = new GBPanel(colors[k]);
         gridPanel.add(gbpanels[k]);
      }
      getContentPane().add(gridPanel, "Center");

      JPanel controlPanel = new JPanel();
      showButton = new JButton("SHOW");
      showButton.setFont(
                  new Font("SansSerif", Font.PLAIN, 18));
      showButton.addActionListener(this);
      controlPanel.add(showButton);
      getContentPane().add(controlPanel, "South");

      setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
   }

   public void actionPerformed(ActionEvent ev)
   {
      if (ev.getSource() == showButton)
      {
         JFrame frame = new DisposableFrame();
         frame.setBounds(200, 100, 300, 300);
         GridBagLayout gbl = new GridBagLayout();
         JPanel pan = new JPanel();
         pan.setLayout(gbl);
         for (int k=0; k<maxButtons; k++)
         {  GridBagConstraints gbc =
                              gbpanels[k].getConstraints();
            if (gbc == null) continue;

            JButton b = new JButton("(" + gbc.gridx + ","
                  + gbc.gridy + ")" + "  "
                  + gbc.gridwidth + "x" + gbc.gridheight);
            b.setBackground(colors[k]);
            gbl.setConstraints(b, gbc);
```



```
            pan.add(b);
        }
        frame. getContentPane().add(pan, "Center");
        frame.show();
    }
  }

  public static void main(String [] args)
  {
    JFrame f = new GridBagLab();
    f.setSize(600,440);
    f.setVisible(true);
  }
}

class ValChoice extends JComboBox
{
  int [] values;

  ValChoice(String [] labels, int [] values)
  {
    this.values = values;
    setLightWeightPopupEnabled(false);
    for (int k=0; k<labels.length; k++)
          addItem(labels[k]);
  }

  public int getSelectedValue()
  { return values[getSelectedIndex()]; }
}

class XYPanel extends JPanel
{
  XYPanel(Color bgc)
  {
    setBackground(bgc);
    setLayout(new GridLayout(2, 1));
  }
}
```

```
class GBPanel extends JPanel
{
   JCheckBox useCB;
   JComboBox xChoice, yChoice;
   ValChoice fillChoice, anchorChoice;
   JTextField xtf, ytf, wtf, htf, xwttf, ywttf;
   Color bgcolor;
   static String [] fillStrings =
               { "Both","Horizontal","Vertical","None" };
   static int [] fillValues =
               {  GridBagConstraints.BOTH,
                  GridBagConstraints.HORIZONTAL,
                  GridBagConstraints.VERTICAL,
                  GridBagConstraints.NONE };
   static String [] anchorStrings =
               { "Center", "North", "NorthEast",
                  "East", "SouthEast", "South",
                  "SouthWest", "West",   "NorthWest" };
   static int [] anchorValues =
               {  GridBagConstraints.CENTER,
                  GridBagConstraints.NORTH,
                  GridBagConstraints.NORTHEAST,
                  GridBagConstraints.EAST,
                  GridBagConstraints.SOUTHEAST,
                  GridBagConstraints.SOUTH,
                  GridBagConstraints.SOUTHWEST,
                  GridBagConstraints.WEST,
                  GridBagConstraints.NORTHWEST  };
```

```
static int [] anchorValues =
            {  GridBagConstraints.CENTER,
               GridBagConstraints.NORTH,
               GridBagConstraints.NORTHEAST,
               GridBagConstraints.EAST,
               GridBagConstraints.SOUTHEAST,
               GridBagConstraints.SOUTH,
               GridBagConstraints.SOUTHWEST,
               GridBagConstraints.WEST,
               GridBagConstraints.NORTHWEST  };

GBPanel(Color bgc)
{
   bgcolor = bgc;
   setBackground(bgcolor);
   JPanel p = new XYPanel(bgcolor);
   p.add(useCB = new JCheckBox("Use"));
   useCB.setBackground(bgcolor);
   add(p);
   p = new XYPanel(bgcolor);
   p.add(new JLabel("X:"));
   p.add(new JLabel("Y:"));
   add(p);
   p = new XYPanel(bgcolor);
   xChoice = new JComboBox();
   xChoice.setLightWeightPopupEnabled(false);
   xChoice.addItem("Type value");
   xChoice.addItem("Relative");
   xChoice.addItem("Remainder");
```



```
p.add(xChoice);

yChoice = new JComboBox();
yChoice.setLightWeightPopupEnabled(false);
yChoice.addItem("Type value");
yChoice.addItem("Relative");
yChoice.addItem("Remainder");
p.add(yChoice);
add(p);

p = new XYPanel(bgcolor);
p.add((xtf = new JTextField("0", 2)));
p.add((ytf = new JTextField("0", 2)));
add(p);

p = new XYPanel(bgcolor);
p.add(new JLabel("Width:"));
p.add(new JLabel("Height:"));
add(p);

p = new XYPanel(bgcolor);
p.add(wtf = new JTextField("1", 2));
p.add(htf = new JTextField("1", 2));
add(p);

p = new XYPanel(bgcolor);
p.add(new JLabel("X wt:"));
p.add(new JLabel("Y wt:"));
add(p);

p = new XYPanel(bgcolor);
p.add(xwttf = new JTextField("1.0", 3));
p.add(ywttf = new JTextField("1.0", 3));
add(p);
```

```
    p = new XYPanel(bgcolor);
    p.add(new JLabel("Fill:"));
    p.add(new JLabel("Anchor:"));
    add(p);

    p = new XYPanel(bgcolor);
    p.add(fillChoice = new ValChoice(fillStrings, fillValues));
    p.add(anchorChoice =
            new ValChoice(anchorStrings, anchorValues));
    add(p);
}

GridBagConstraints getConstraints()
{
    if (!useCB.isSelected())  return null;

    GridBagConstraints gbc = new GridBagConstraints();

    switch (xChoice.getSelectedIndex())
    {
    case 0:    gbc.gridx = Integer.parseInt(xtf.getText());
               break;
    case 1:    gbc.gridx = GridBagConstraints.RELATIVE;
               break;
    case 2:    gbc.gridx = GridBagConstraints.REMAINDER;
               break;
    }

    switch (yChoice.getSelectedIndex())
    {
    case 0:    gbc.gridy = Integer.parseInt(ytf.getText());
               break;
    case 1:    gbc.gridy = GridBagConstraints.RELATIVE;
               break;
    case 2:    gbc.gridy = GridBagConstraints.REMAINDER;
               break;
    }
```



```
        gbc.gridwidth = Integer.parseInt(wtf.getText());
        gbc.gridheight = Integer.parseInt(htf.getText());
        gbc.weightx = Double.parseDouble(xwttf.getText());
        gbc.weighty = Double.parseDouble(ywttf.getText());
        gbc.fill = fillChoice.getSelectedValue();
        gbc.anchor = anchorChoice.getSelectedValue();
        return gbc;
    }
}

class DisposableFrame extends JFrame
{
    DisposableFrame()
    {
        addWindowListener(new WindowAdapter()
        {
            public void windowClosing(WindowEvent e)
            {
                dispose();
            }
        });
    }
}
```