Class java.awt.GridBagLayout

```
java.lang.Object
   |
   +----java.awt.GridBagLayout
```

public class **GridBagLayout**
extends Object
implements LayoutManager

GridBagLayout is a flexible layout manager that aligns components vertically and horizontally, without requiring that the components be the same size. Each GridBagLayout uses a rectangular grid of cells, with each component occupying one or more cells (called its *display area*). Each component managed by a GridBagLayout is associated with a GridBagConstraints instance that specifies how the component is laid out within its display area. How a GridBagLayout places a set of components depends on each component's GridBagConstraints and minimum size, as well as the preferred size of the components' container.

To use a GridBagLayout effectively, you must customize one or more of its components' GridBagConstraints. You customize a GridBagConstraints object by setting one or more of its instance variables:

gridx, gridy
: Specifies the cell at the upper left of the component's display area, where the upper–left–most cell has address gridx=0, gridy=0. Use GridBagConstraints.RELATIVE (the default value) to specify that the component be just placed just to the right of (for gridx) or just below (for gridy) the component that was added to the container just before this component was added.

gridwidth, gridheight
: Specifies the number of cells in a row (for gridwidth) or column (for gridheight) in the component's display area. The default value is 1. Use GridBagConstraints.REMAINDER to specify that the component be the last one in its row (for gridwidth) or column (for gridheight). Use GridBagConstraints.RELATIVE to specify that the component be the next to last one in its row (for gridwidth) or column (for gridheight).

fill
: Used when the component's display area is larger than the component's requested size to determine whether (and how) to resize the component. Valid values are GridBagConstraint.NONE (the default), GridBagConstraint.HORIZONTAL (make the component wide enough to fill its display area horizontally, but don't change

its height), GridBagConstraint.VERTICAL (make the component tall enough to fill its display area vertically, but don't change its width), and GridBagConstraint.BOTH (make the component fill its display area entirely).

ipadx, ipady
: Specifies the internal padding: how much to add to the minimum size of the component. The width of the component will be at least its minimum width plus ipadx*2 pixels (since the padding applies to both sides of the component). Similarly, the height of the component will be at least the minimum height plus ipady*2 pixels.

insets
: Specifies the external padding of the component –– the minimum amount of space between the component and the edges of its display area.

anchor
: Used when the component is smaller than its display area to determine where (within the area) to place the component. Valid values are GridBagConstraints.CENTER (the default), GridBagConstraints.NORTH, GridBagConstraints.NORTHEAST, GridBagConstraints.EAST, GridBagConstraints.SOUTHEAST, GridBagConstraints.SOUTH, GridBagConstraints.SOUTHWEST, GridBagConstraints.WEST, and GridBagConstraints.NORTHWEST.

weightx, weighty
: Used to determine how to distribute space; this is important for specifying resizing behavior. Unless you specify a weight for at least one component in a row (weightx) and column (weighty), all the components clump together in the center of their container. This is because when the weight is zero (the default), the GridBagLayout puts any extra space between its grid of cells and the edges of the container.

The following figure shows ten components (all buttons) managed by a GridBagLayout:

[IMAGE]

All the components have fill=GridBagConstraints.BOTH. In addition, the components have the following non–default constraints:

- Button1, Button2, Button3: weightx=1.0
- Button4: weightx=1.0, gridwidth=GridBagConstraints.REMAINDER
- Button5: gridwidth=GridBagConstraints.REMAINDER
- Button6: gridwidth=GridBagConstraints.RELATIVE
- Button7: gridwidth=GridBagConstraints.REMAINDER
- Button8: gridheight=2, weighty=1.0,
- Button9, Button 10: gridwidth=GridBagConstraints.REMAINDER

Here is the code that implements the example shown above:

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

```
public class GridBagEx1 extends Applet {
    protected void makebutton(String name,
                              GridBagLayout gridbag,
                              GridBagConstraints c) {
        Button button = new Button(name);
        gridbag.setConstraints(button, c);
        add(button);
    }
    public void init() {
        GridBagLayout gridbag = new GridBagLayout();
        GridBagConstraints c = new GridBagConstraints();

        setFont(new Font("Helvetica", Font.PLAIN, 14));
        setLayout(gridbag);

        c.fill = GridBagConstraints.BOTH;
        c.weightx = 1.0;
        makebutton("Button1", gridbag, c);
        makebutton("Button2", gridbag, c);
        makebutton("Button3", gridbag, c);

        c.gridwidth = GridBagConstraints.REMAINDER; //end row
        makebutton("Button4", gridbag, c);

        c.weightx = 0.0;                    //reset to the default
        makebutton("Button5", gridbag, c); //another row

        c.gridwidth = GridBagConstraints.RELATIVE; //next-to-last in row
        makebutton("Button6", gridbag, c);

        c.gridwidth = GridBagConstraints.REMAINDER; //end row
        makebutton("Button7", gridbag, c);

        c.gridwidth = 1;                    //reset to the default
        c.gridheight = 2;
        c.weighty = 1.0;
        makebutton("Button8", gridbag, c);

        c.weighty = 0.0;                    //reset to the default
        c.gridwidth = GridBagConstraints.REMAINDER; //end row
        c.gridheight = 1;                   //reset to the default
        makebutton("Button9", gridbag, c);
        makebutton("Button10", gridbag, c);

        resize(300, 100);
    }

    public static void main(String args[]) {
        Frame f = new Frame("GridBag Layout Example");
        GridBagEx1 ex1 = new GridBagEx1();

        ex1.init();

        f.add("Center", ex1);
        f.pack();
        f.resize(f.preferredSize());
        f.show();
    }
}
```

Version:
1.2, 10/19/95
Author:
Doug Stein

Variable Index

o **MAXGRIDSIZE**
o **MINSIZE**
o **PREFERREDSIZE**
o **comptable**
o **defaultConstraints**

Constructor Index

o **GridBagLayout**()
Creates a gridbag layout.

Method Index

o **AdjustForGravity**(GridBagConstraints, Rectangle)
o **ArrangeGrid**(Container)
o **DumpConstraints**(GridBagConstraints)
Print the layout constraints.
o **DumpLayoutInfo**(GridBagLayoutInfo)
Print the layout information.
o **GetLayoutInfo**(Container, int)
o **GetMinSize**(Container, GridBagLayoutInfo)
o **addLayoutComponent**(String, Component)
Adds the specified component with the specified name to the layout.
o **getConstraints**(Component)
Retrieves the constraints for the specified component.
o **layoutContainer**(Container)
Lays out the container in the specified panel.
o **lookupConstraints**(Component)
Retrieves the constraints for the specified component.
o **minimumLayoutSize**(Container)
Returns the minimum dimensions needed to layout the components contained in the specified panel.
o **preferredLayoutSize**(Container)
Returns the preferred dimensions for this layout given the components in the specified panel.
o **removeLayoutComponent**(Component)
Removes the specified component from the layout.
o **setConstraints**(Component, GridBagConstraints)

Sets the constraints for the specified component.

- **toString**()
  Returns the String representation of this GridLayout's values.

Variables

- **MAXGRIDSIZE**

```
protected final static int MAXGRIDSIZE
```

- **MINSIZE**

```
protected final static int MINSIZE
```

- **PREFERREDSIZE**

```
protected final static int PREFERREDSIZE
```

- **comptable**

```
protected Hashtable comptable
```

- **defaultConstraints**

```
protected GridBagConstraints defaultConstraints
```

Constructors

- **GridBagLayout**

```
public GridBagLayout()
```

Creates a gridbag layout.

Methods

- **setConstraints**

```
public void setConstraints(Component comp,
                           GridBagConstraints constraints)
```

Sets the constraints for the specified component.
Parameters:
comp – the component to be modified
constraints – the constraints to be applied

- **getConstraints**

```
public GridBagConstraints getConstraints(Component comp)
```

Retrieves the constraints for the specified component. A copy of the constraints is returned.

Parameters:

comp – the component to be queried

- **lookupConstraints**

```
protected GridBagConstraints lookupConstraints(Component comp)
```

Retrieves the constraints for the specified component. The return value is not a copy, but is the actual constraints class used by the layout mechanism.

Parameters:

comp – the component to be queried

- **addLayoutComponent**

```
public void addLayoutComponent(String name,
                               Component comp)
```

Adds the specified component with the specified name to the layout.

Parameters:

name – the name of the component

comp – the component to be added

- **removeLayoutComponent**

```
public void removeLayoutComponent(Component comp)
```

Removes the specified component from the layout. Does not apply.

Parameters:

comp – the component to be removed

- **preferredLayoutSize**

```
public Dimension preferredLayoutSize(Container parent)
```

Returns the preferred dimensions for this layout given the components in the specified panel.

Parameters:

parent – the component which needs to be laid out

See Also:

minimumSize

- **minimumLayoutSize**

```
public Dimension minimumLayoutSize(Container parent)
```

Returns the minimum dimensions needed to layout the components contained in the specified panel.

Parameters:

parent – the component which needs to be laid out

See Also:

preferredSize

- **layoutContainer**

```
public void layoutContainer(Container parent)
```

Lays out the container in the specified panel.

Parameters:

parent – the specified component being laid out

See Also:

Container

- **toString**

```
public String toString()
```

Returns the String representation of this GridLayout's values.

Overrides:

toString in class Object

- **DumpLayoutInfo**

```
protected void DumpLayoutInfo(GridBagLayoutInfo s)
```

Print the layout information. Useful for debugging.

- **DumpConstraints**

```
protected void DumpConstraints(GridBagConstraints constraints)
```

Print the layout constraints. Useful for debugging.

- **GetLayoutInfo**

```
protected GridBagLayoutInfo GetLayoutInfo(Container parent,
                                          int sizeflag)
```

- **AdjustForGravity**

```
protected void AdjustForGravity(GridBagConstraints constraints,
                                Rectangle r)
```

- **GetMinSize**

```
protected Dimension GetMinSize(Container parent,
                               GridBagLayoutInfo info)
```

o ArrangeGrid

```
protected void ArrangeGrid(Container parent)
```
