Laying Out Components within a Container

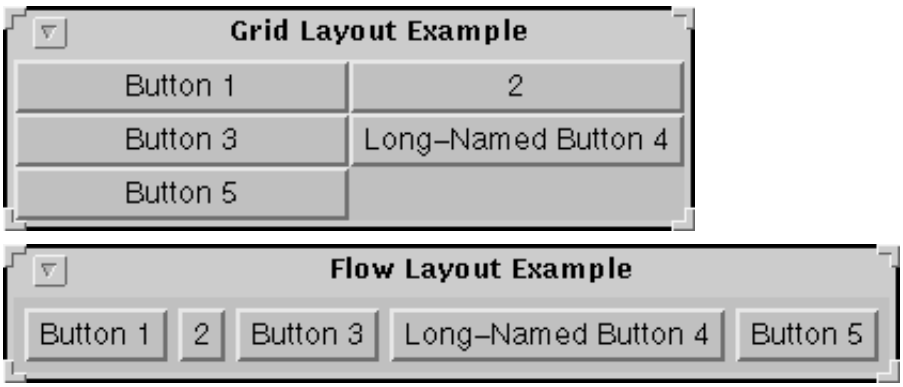
Table of Contents

- Laying Out Components within a Container
  - Using Layout Managers
    - General Rules for Using Layout Managers
    - How to Use BorderLayout
    - How to Use CardLayout
    - How to Use FlowLayout
    - How to Use GridLayout
    - How to Use GridBagLayout
  - Creating a Custom Layout Manager
  - Doing Without a Layout Manager
  - Common Layout Problems (and Their Solutions)

Laying Out Components within a Container

Table of Contents

Laying Out Components within a Container



Above are pictures of two programs, each of which displays five buttons. The Java code for both programs is almost identical. So why do they look so different? Because they use different *layout managers* to control the layout of the buttons.

This lesson tells you how to position the onscreen representations of Components. It'll show you how to use the layout managers the AWT provides. It'll also show you how to write your own layout manager. It'll even tell you how to do without a layout manager and use absolute positions. Finally, it'll discuss some common layout problems and solutions.

Using Layout Managers

Here's where to learn how to use layout managers. This section gives both general rules and detailed instructions on using each of the layout managers the AWT provides.

Creating a Custom Layout Manager

Instead of using one of the AWT's layout managers, you can write your own. Layout managers must implement the LayoutManager interface, which specifies the five methods every layout manager must define.

Doing Without a Layout Manager

You can position Components without using a layout manager. Generally, this solution is used to specify absolute positions for Components, and only for applications that are executed on only one platform. Absolute positioning is usually *un*suitable for applets and other platform–independent programs, since the size of Components can be different on different

platforms.

Common Layout Problems (and Their Solutions)

Table of Contents

Using Layout Managers

Every Container, by default, has a layout manager (an object conforming to the LayoutManager protocol). If a Container's default layout manager doesn't suit your needs, you can easily replace it with another one. We supply layout managers that range from the very simple (FlowLayout and GridLayout) to the special purpose (BorderLayout and CardLayout) to the ultra–flexible (GridBagLayout).

This lesson gives you some general rules for using layout managers, gives you an overview of the layout managers we provide, and then tells you how to use each of our layout managers.

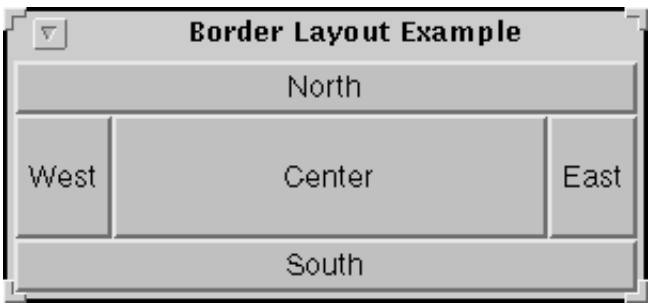
General Rules for Using Layout Managers

How do you create a layout manager, associate it with a Container, and tell it to start working? How does it know what Components it manages? This section answers these and other questions that apply whenever you use a layout manager.

How to Use BorderLayout

BorderLayout is the default layout manager for all Windows, such as Frames and Dialogs. It uses five areas to hold components: north, south, east, west, and center. All extra space is placed in the center area. Here's an applet that puts one button in each area.

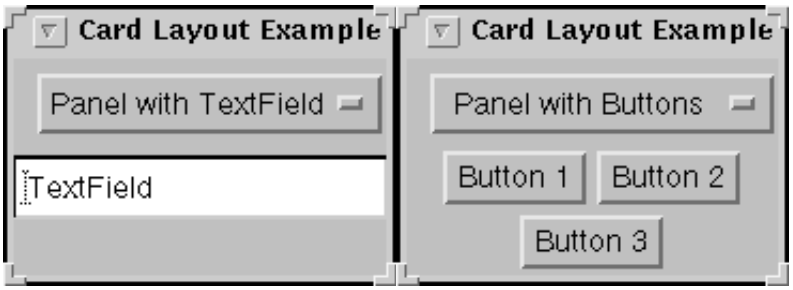
Your browser can't run 1.0 Java applets, so here's a picture of the window the program brings up:



How to Use CardLayout

Use the CardLayout class when you have an area that can contain different Components at different times. CardLayouts are commonly tied to Choices, with the state of the Choice determining which Panel (group of Components) the CardLayout displays. Here's an applet that uses a Choice and CardLayout in this way.

Your browser can't run 1.0 Java applets, so here are pictures of the window the program brings up:



How to Use FlowLayout

FlowLayout is the default layout manager for all Panels. It simply lays out components from left to right, centering them within their row and, if necessary, starting new rows. Both Panels in the CardLayout figure above use FlowLayouts. Here's another example of a program that uses a FlowLayout.

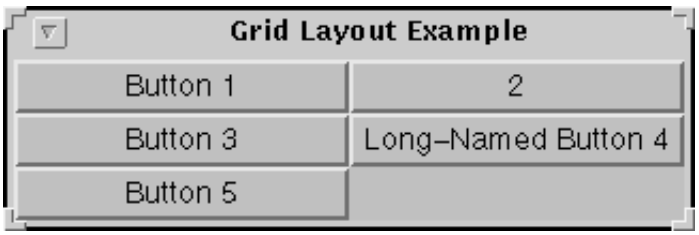
Your browser can't run 1.0 Java applets, so here's a picture of the window the program brings up:

How to Use GridLayout

GridLayouts simply make a bunch of Components have equal size, displaying them in the requested number of rows and columns. Here's a picture of a program with a Panel that uses a GridLayout to control the display of 5 buttons.

Your browser can't run 1.0 Java applets, so here's a picture of the window the program

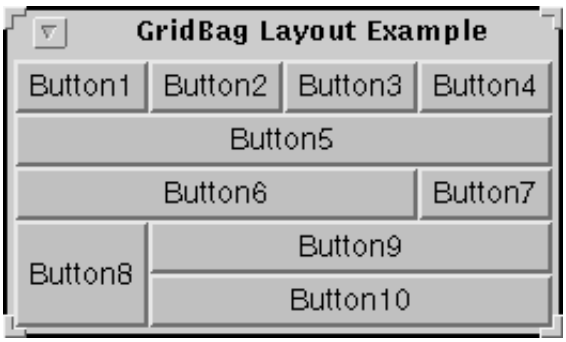
brings up:



How to Use GridBagLayout

GridBagLayout is the most sophisticated, flexible layout manager we provide. It aligns components vertically and horizontally, without requiring that the components be the same size. Here's a picture of a program that uses a GridBagLayout to manage ten buttons in a panel.

Your browser can't run 1.0 Java applets, so here's a picture of the window the program brings up:



Laying Out Components within a Container

General Rules for Using Layout Managers

Unless you explicitly tell a Container not to use a layout manager, it will be associated with its very own instance of a layout manager. This layout manager is automatically consulted every time the Container might need to change its appearance. Except for GridBagLayout, most layout managers don't require applications and applets to directly call the layout manager's methods.

How to Create a Layout Manager and Associate It with a Container

Every container has a default layout manager associated with it. If you want to use this default, you don't have to do a thing. The constructor for each Container creates a layout manager instance and initializes the Container to use it.

To use a non–default layout manager, you need to create an instance of the desired layout manager class and tell the Container to use it. Below is some typical code that does this. This code creates a CardLayout manager and sets it up as the layout manager for a Container.

```
aContainer.setLayout(new CardLayout());
```

Rules of Thumb for Using Layout Managers

The Container methods that result in calls to the Container's layout manager are `add()`, `remove()`, `removeAll()`, `layout()`, `preferredSize()`, and `minimumSize()`. The `add()`, `remove()`, and `removeAll()` methods add and remove Components from a Container; you can call them at any time. The `layout()` method, which is called as the result of any paint request to a Container, requests that the Container place and size itself and the Components it contains; you don't call it directly [CHECK]. The `preferredSize()` and `minimumSize()` methods return the Container's ideal size and minimum size, respectively. The values returned are just hints; they have no effect unless your program enforces these sizes [CHECK].

You must take special care when calling a Container's `preferredSize()` and `minimumSize()` methods. The values these methods return are meaningless unless the Container and its Components have valid peer objects. The peer object corresponding to a Component is created whenever the Component's `addNotify()` method is called, which happens:

- when the Component is a peer–less Window, or is contained in a peer–less Window, and the Window's `pack()` or `show()` method is called
- when the Component is added to a Container that already has a peer
- when your program directly calls the Component's `addNotify()` method, or the

`addNotify()` method of any Container above the Component [is this really OK?]

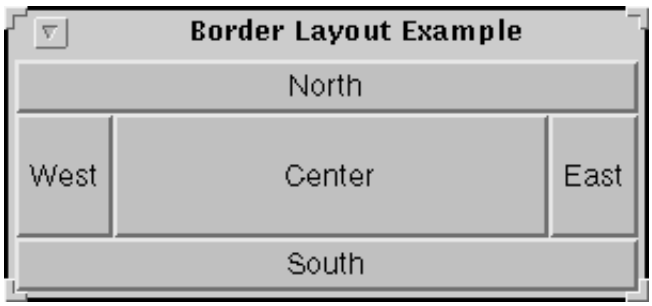
Laying Out Components within a Container

How to Use BorderLayout

Here's example code for using BorderLayout.

Here's the example program, running as an applet:

Your browser can't run 1.0 Java applets, so here's a picture of the window the program brings up:



BorderWindow.java

```
/*
 * Copyright (c) 1994 Sun Microsystems, Inc. All Rights Reserved.
 *
 * Permission to use, copy, modify, and distribute this software
 * and its documentation for NON-COMMERCIAL purposes and without
 * fee is hereby granted provided that this copyright notice
 * appears in all copies. Please refer to the file "copyright.html"
 * for further important copyright and licensing information.
 *
 * SUN MAKES NO REPRESENTATIONS OR WARRANTIES ABOUT THE SUITABILITY OF
 * THE SOFTWARE, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED
 * TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE, OR NON-INFRINGEMENT. SUN SHALL NOT BE LIABLE FOR
 * ANY DAMAGES SUFFERED BY LICENSEE AS A RESULT OF USING, MODIFYING OR
 * DISTRIBUTING THIS SOFTWARE OR ITS DERIVATIVES.
 */
import java.awt.*;

public class BorderWindow extends Frame {
    private boolean inAnApplet = true;

    public BorderWindow() {
        setLayout(new BorderLayout());
        setFont(new Font("Helvetica", Font.PLAIN, 14));

        add("North", new Button("North"));
        add("South", new Button("South"));
        add("East", new Button("East"));
        add("West", new Button("West"));
        add("Center", new Button("Center"));

        //resize(300, 150);
    }

    public synchronized boolean handleEvent(Event e) {
        if (e.id == Event.WINDOW_ICONIFY) {//DOESN'T seem to be necessary
            hide();
            return true;
        }
        if (e.id == Event.WINDOW_DESTROY) {
            if (inAnApplet) {
                dispose();
                return true;
            } else {
                System.exit(0);
            }
        }
        return super.handleEvent(e);
    }

    public static void main(String args[]) {
        BorderWindow window = new BorderWindow();
```

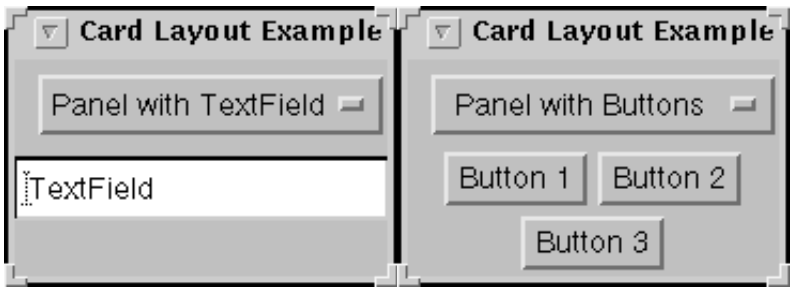
```
        window.inAnApplet = false;

        window.setTitle("BorderWindow Application");
        window.pack();
        window.resize(window.preferredSize());
        window.show();
    }
}
```

How to Use CardLayout

Here's example code for using CardLayout.

Here's the example program, running as an applet:

Your browser can't run 1.0 Java applets, so here are pictures of the window the program brings up:



CardEx1.java

```
/*
 * Copyright (c) 1994 Sun Microsystems, Inc. All Rights Reserved.
 *
 * Permission to use, copy, modify, and distribute this software
 * and its documentation for NON-COMMERCIAL purposes and without
 * fee is hereby granted provided that this copyright notice
 * appears in all copies. Please refer to the file "copyright.html"
 * for further important copyright and licensing information.
 *
 * SUN MAKES NO REPRESENTATIONS OR WARRANTIES ABOUT THE SUITABILITY OF
 * THE SOFTWARE, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED
 * TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE, OR NON-INFRINGEMENT. SUN SHALL NOT BE LIABLE FOR
 * ANY DAMAGES SUFFERED BY LICENSEE AS A RESULT OF USING, MODIFYING OR
 * DISTRIBUTING THIS SOFTWARE OR ITS DERIVATIVES.
 */
import java.awt.*;
import java.util.*;
import java.applet.Applet;

public class CardEx1 extends Applet {
    Panel cards;
    final static String BUTTONPANEL = "Panel with Buttons";
    final static String TEXTPANEL = "Panel with TextField";

    public void init() {
        setLayout(new BorderLayout());
        setFont(new Font("Helvetica", Font.PLAIN, 14));

        //Put the Choice in a Panel to get a nicer look.
        Panel cp = new Panel();
        Choice c = new Choice();
        c.addItem(BUTTONPANEL);
        c.addItem(TEXTPANEL);
        cp.add(c);
        add("North", cp);

        cards = new Panel();
        cards.setLayout(new CardLayout());

        Panel p1 = new Panel();
        p1.add(new Button("Button 1"));
        p1.add(new Button("Button 2"));
        p1.add(new Button("Button 3"));

        Panel p2 = new Panel();
        p2.add(new TextField("TextField", 20));

        cards.add(BUTTONPANEL, p1);
        cards.add(TEXTPANEL, p2);
        add("Center", cards);
        resize(200, 150);
```

```
    }

    public boolean action(Event evt, Object arg) {
        if (evt.target instanceof Choice) {
            ((CardLayout)cards.getLayout()).show(cards,(String)arg);
            return true;
        }
        return false;
    }

    public static void main(String args[]) {
        Frame f = new Frame("Card Layout Example");
        CardEx1 ex1 = new CardEx1();

        ex1.init();

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

How to Use FlowLayout

Here's example code for using FlowLayout. Since FlowLayout is the default layout manager for Panels, the example doesn't explicitly set the layout manager.

Here's the example program, running as an applet:

Your browser can't run 1.0 Java applets, so here's a picture of the window the program brings up:

FlowEx1.java

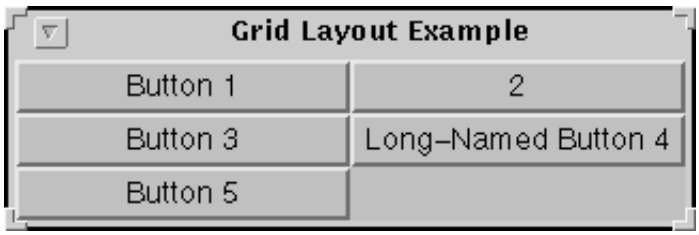
```
/*
 * Copyright (c) 1994 Sun Microsystems, Inc. All Rights Reserved.
 *
 * Permission to use, copy, modify, and distribute this software
 * and its documentation for NON-COMMERCIAL purposes and without
 * fee is hereby granted provided that this copyright notice
 * appears in all copies. Please refer to the file "copyright.html"
 * for further important copyright and licensing information.
 *
 * SUN MAKES NO REPRESENTATIONS OR WARRANTIES ABOUT THE SUITABILITY OF
 * THE SOFTWARE, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED
 * TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE, OR NON-INFRINGEMENT. SUN SHALL NOT BE LIABLE FOR
 * ANY DAMAGES SUFFERED BY LICENSEE AS A RESULT OF USING, MODIFYING OR
 * DISTRIBUTING THIS SOFTWARE OR ITS DERIVATIVES.
 */
import java.awt.*;
import java.util.*;
import java.applet.Applet;

public class FlowEx1 extends Applet {
    public void init() {
        setFont(new Font("Helvetica", Font.PLAIN, 14));

        add(new Button("Button 1"));
        add(new Button("2"));
        add(new Button("Button 3"));
        add(new Button("Long-Named Button 4"));
        add(new Button("Button 5"));

        resize(300, 100);
    }

    public static void main(String args[]) {
        Frame f = new Frame("Flow Layout Example");
        FlowEx1 ex1 = new FlowEx1();

        ex1.init();

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

How to Use GridLayout

Here's example code for using GridLayout.

Here's the same program, running as an applet:

Your browser can't run 1.0 Java applets, so here's a picture of the window the program brings up:



GridWindow.java

```
/*
 * Copyright (c) 1994 Sun Microsystems, Inc. All Rights Reserved.
 *
 * Permission to use, copy, modify, and distribute this software
 * and its documentation for NON-COMMERCIAL purposes and without
 * fee is hereby granted provided that this copyright notice
 * appears in all copies. Please refer to the file "copyright.html"
 * for further important copyright and licensing information.
 *
 * SUN MAKES NO REPRESENTATIONS OR WARRANTIES ABOUT THE SUITABILITY OF
 * THE SOFTWARE, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED
 * TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE, OR NON-INFRINGEMENT. SUN SHALL NOT BE LIABLE FOR
 * ANY DAMAGES SUFFERED BY LICENSEE AS A RESULT OF USING, MODIFYING OR
 * DISTRIBUTING THIS SOFTWARE OR ITS DERIVATIVES.
 */
import java.awt.*;

public class GridWindow extends Frame {
    private boolean inAnApplet = true;

    public GridWindow() {
        setLayout(new GridLayout(0,2));
        setFont(new Font("Helvetica", Font.PLAIN, 14));

        add(new Button("Button 1"));
        add(new Button("2"));
        add(new Button("Button 3"));
        add(new Button("Long-Named Button 4"));
        add(new Button("Button 5"));

        //resize(300, 100);
    }

    public synchronized boolean handleEvent(Event e) {
        if (e.id == Event.WINDOW_ICONIFY) {//DOESN'T seem to be necessary
            hide();
            return true;
        }
        if (e.id == Event.WINDOW_DESTROY) {
            if (inAnApplet) {
                dispose();
                return true;
            } else {
                System.exit(0);
            }
        }
        return super.handleEvent(e);
    }

    public static void main(String args[]) {
        GridWindow window = new GridWindow();
```

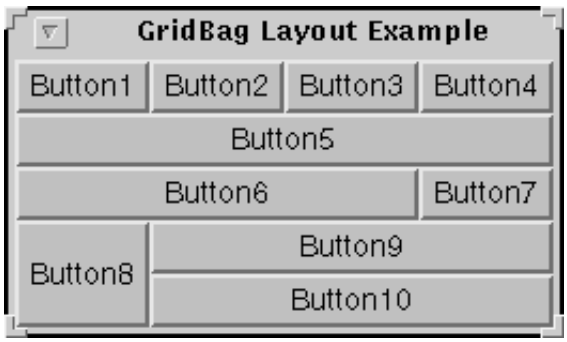
```
        window.inAnApplet = false;

        window.setTitle("GridWindow Application");
        window.pack();
        window.resize(window.preferredSize());
        window.show();
    }
}
```

How to Use GridBagLayout

Here's example code for using GridBagLayout.

Here's the same program, running as an applet:

Your browser can't run 1.0 Java applets, so here's a picture of the window the program brings up:



GridBagWindow.java

```
/*
 * Copyright (c) 1994 Sun Microsystems, Inc. All Rights Reserved.
 *
 * Permission to use, copy, modify, and distribute this software
 * and its documentation for NON-COMMERCIAL purposes and without
 * fee is hereby granted provided that this copyright notice
 * appears in all copies. Please refer to the file "copyright.html"
 * for further important copyright and licensing information.
 *
 * SUN MAKES NO REPRESENTATIONS OR WARRANTIES ABOUT THE SUITABILITY OF
 * THE SOFTWARE, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED
 * TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE, OR NON-INFRINGEMENT. SUN SHALL NOT BE LIABLE FOR
 * ANY DAMAGES SUFFERED BY LICENSEE AS A RESULT OF USING, MODIFYING OR
 * DISTRIBUTING THIS SOFTWARE OR ITS DERIVATIVES.
 */
import java.awt.*;

public class GridBagWindow extends Frame {
    private boolean inAnApplet = true;

    protected void makebutton(String name,
                              GridBagLayout gridbag,
                              GridBagConstraints c) {
        Button button = new Button(name);
        gridbag.setConstraints(button, c);
        add(button);
    }

    public GridBagWindow() {
        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 synchronized boolean handleEvent(Event e) {
        if (e.id == Event.WINDOW_ICONIFY) {//DOESN'T seem to be necessary
            hide();
            return true;
        }
        if (e.id == Event.WINDOW_DESTROY) {
            if (inAnApplet) {
                dispose();
                return true;
            } else {
                System.exit(0);
            }
        }
        return super.handleEvent(e);
    }

    public static void main(String args[]) {
        GridBagWindow window = new GridBagWindow();
        window.inAnApplet = false;

        window.setTitle("GridBagWindow Application");
        window.pack();
        window.resize(window.preferredSize());
        window.show();
    }
}
```

Creating a Custom Layout Manager

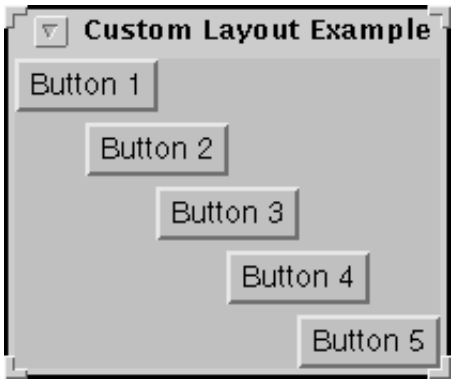
Note: Before you start creating a custom layout manager, make sure that no existing layout manager will work. In particular, GridBagLayout is flexible enough to work in most cases.

To create a custom layout manager, you must create a class that implements the LayoutManager interface. Besides implementing the five methods required by LayoutManager, layout managers generally implement at least one public constructor and the `toString()` method. Layout managers don't actually draw Components. They simply invoke each Component's `resize()`, `move()`, and/or `reshape()` to set the Component's size and position.

Here's source code for a custom layout manager named DiagonalLayout. It lays out components diagonally, from left to right, with one component per row.

Here's an example of DiagonalLayout in action:

Your browser can't run 1.0 Java applets, so here's a picture of the window the program brings up:

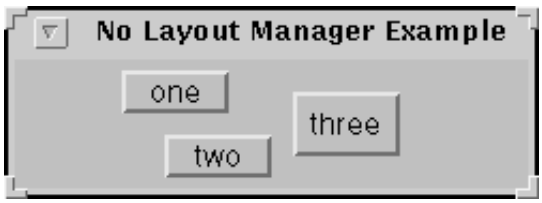


Doing Without a Layout Manager

Here's example code that uses no layout manager.

Here's the same program, running as an applet:

Your browser can't run 1.0 Java applets, so here's a picture of the window the program brings up:



NoneEx1.java

```
/*
 * Copyright (c) 1994 Sun Microsystems, Inc. All Rights Reserved.
 *
 * Permission to use, copy, modify, and distribute this software
 * and its documentation for NON-COMMERCIAL purposes and without
 * fee is hereby granted provided that this copyright notice
 * appears in all copies. Please refer to the file "copyright.html"
 * for further important copyright and licensing information.
 *
 * SUN MAKES NO REPRESENTATIONS OR WARRANTIES ABOUT THE SUITABILITY OF
 * THE SOFTWARE, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED
 * TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE, OR NON-INFRINGEMENT. SUN SHALL NOT BE LIABLE FOR
 * ANY DAMAGES SUFFERED BY LICENSEE AS A RESULT OF USING, MODIFYING OR
 * DISTRIBUTING THIS SOFTWARE OR ITS DERIVATIVES.
 */
import java.awt.*;
import java.applet.Applet;

public class NoneEx1 extends Applet {

    public void init() {
        Button b1, b2, b3;

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

        b1 = new Button("one");
        add(b1);
        b1.reshape(50, 5, 50, 20);

        b2 = new Button("two");
        add(b2);
        b2.reshape(70, 35, 50, 20);

        b3 = new Button("three");
        add(b3);
        b3.reshape(130, 15, 50, 30);
    }

    public static void main(String args[]) {
        Frame f = new Frame("No Layout Manager Example");
        NoneEx1 ex1 = new NoneEx1();

        ex1.init();
        f.add("Center", ex1);
        f.pack();
        f.resize(250, 90);
        f.show();
    }
}
```

Laying Out Components within a Container

Common Layout Problems (and Their Solutions)

This page will include answers to common questions such as:

1. How do I specify a Component's exact size?
2. How do I resize a Component (Component–initiated resizing)?
3. How do I increase or decrease the number of Components in a Container?

What else should this page cover? Send me mail!

Laying Out Components within a Container