

Graphical User Interfaces
with Swing
from Sun's Swing Tutorial

GUIs

- How do you write a program in the Java programming language with a graphical user interface?
 - The short answer — the Swing toolkit! This presentation gives you a brief overview of the graphical capabilities of the core Java platform with a special focus on Swing.

From the Java Tutorial:

<http://java.sun.com/docs/books/tutorial/uiswing/>

What is Swing?

- The Swing toolkit, shipped as part of the Java SE platform, provides a rich set of GUI components. But Swing offers much more functionality than a collection of standard widgets. This section takes a look at Swing's rich functionality.
- A rich multi-class Swing application, PasswordStore, is presented. The source code is available as a reference to the user.

What is Swing?

- Swing is far from a simple component toolkit
- It includes:
 - rich undo support
 - a highly customizable text package
 - integrated internationalization and accessibility support.
 - numerous look and feels, including the ability to create your own look and feel.
 - The ability to create a custom look and feel is made easier with Synth, a look and feel specifically designed to be customized.
 - Swing wouldn't be a component toolkit without the basic user interface primitives such as drag and drop, event handling, customizable painting, and window management.
- Swing is part of the Java Foundation Classes (JFC). The JFC also include other features important to a GUI program, such as the ability to add rich graphics functionality and the ability to create a program that can work in different languages and by users with different input devices.

What is Swing?

- The following list shows some of the features that Swing and the Java Foundation Classes provide.
 - **Swing GUI Components**
 - The Swing toolkit includes a rich array of components:
 - basic components, such as buttons and check boxes
 - simple components, such as text fields, offer sophisticated functionality, such as formatted text input or password field behavior.
 - rich and complex components, such as tables and text.
 - There are file browsers and dialogs to suit most needs, and if not, customization is possible.
 - If none of Swing's provided components are exactly what you need, you can leverage the basic Swing component functionality to create your own.

Swing and Java Foundation Classes

- **Java 2D API**
 - To make your application stand out;
 - convey information visually;
 - or add figures, images, or animation to your GUI, you'll want to use the Java 2D API.
 - Because Swing is built on the 2D package, it's trivial to make use of 2D within Swing components. Adding images, drop shadows, compositing — it's easy with Java 2D.
- **Pluggable Look-and-Feel Support**
 - Any program that uses Swing components has a choice of look and feel.
 - The JFC classes shipped by Sun and Apple provide a look and feel that matches that of the platform.
 - The Synth package allows you to create your own look and feel.
 - The GTK+ look and feel makes hundreds of existing look and feels available to Swing programs.
 - A program can specify the look and feel of the platform it is running on, or it can specify to always use the Java look and feel, and without recompiling, it will just work.
 - Or, you can ignore the issue and let the UI manager sort it out.

Swing and Java Foundation Classes

- **Data Transfer**
 - Data transfer, via cut, copy, paste, and drag and drop, is essential to almost any application.
 - Support for data transfer is built into Swing and works between Swing components
 - within an application,
 - between Java applications, and
 - between Java and native applications.
- **Internationalization**
 - This feature allows developers to build applications that can interact with users worldwide in their own languages and cultural conventions.
 - Applications can be created that accept input in languages that use thousands of different characters, such as Japanese, Chinese, or Korean.
 - Swing's layout managers make it easy to honor a particular orientation required by the UI.
 - For example, the UI will appear right to left in a locale where the text flows right to left. This support is automatic: You need only code the UI once and then it will work for left to right and right to left, as well as honor the appropriate size of components that change as you localize the text.

Swing and Java Foundation Classes

- **Accessibility API**

- People with disabilities use special software — assistive technologies — that mediates the user experience for them.
- Such software needs to obtain a wealth of information about the running application in order to represent it in alternate media:
 - for a screen reader to read the screen with synthetic speech or render it via a Braille display,
 - for a screen magnifier to track the caret and keyboard focus,
 - for on-screen keyboards to present dynamic keyboards of the menu choices and toolbar items and dialog controls, and
 - for voice control systems to know what the user can control with his or her voice.
- The accessibility API enables these assistive technologies to get the information they need, and to programmatically manipulate the elements that make up the graphical user interface.

Swing and Java Foundation Classes

- **Undo Framework API**
 - Swing's undo framework allows developers to provide support for undo and redo.
 - Undo support is built in to Swing's text component. For other components,
 - Swing supports an *unlimited* number of actions to undo and redo, and
 - is easily adapted to an application.
 - For example, you could easily enable undo to add and remove elements from a table.
- **Flexible Deployment Support**
 - If you want your program to run within a browser window, you can create it as an applet and run it using Java Plug-in, which supports a variety of browsers, such as Internet Explorer, Firefox, and Safari.
 - If you want to create a program that can be launched from a browser, you can do this with Java Web Start.
 - Of course, your application can also run outside of browser as a standard desktop application.

A Swing Demo

Here is an example of an application, PasswordStore, that illustrates some of Swing's rich feature set.

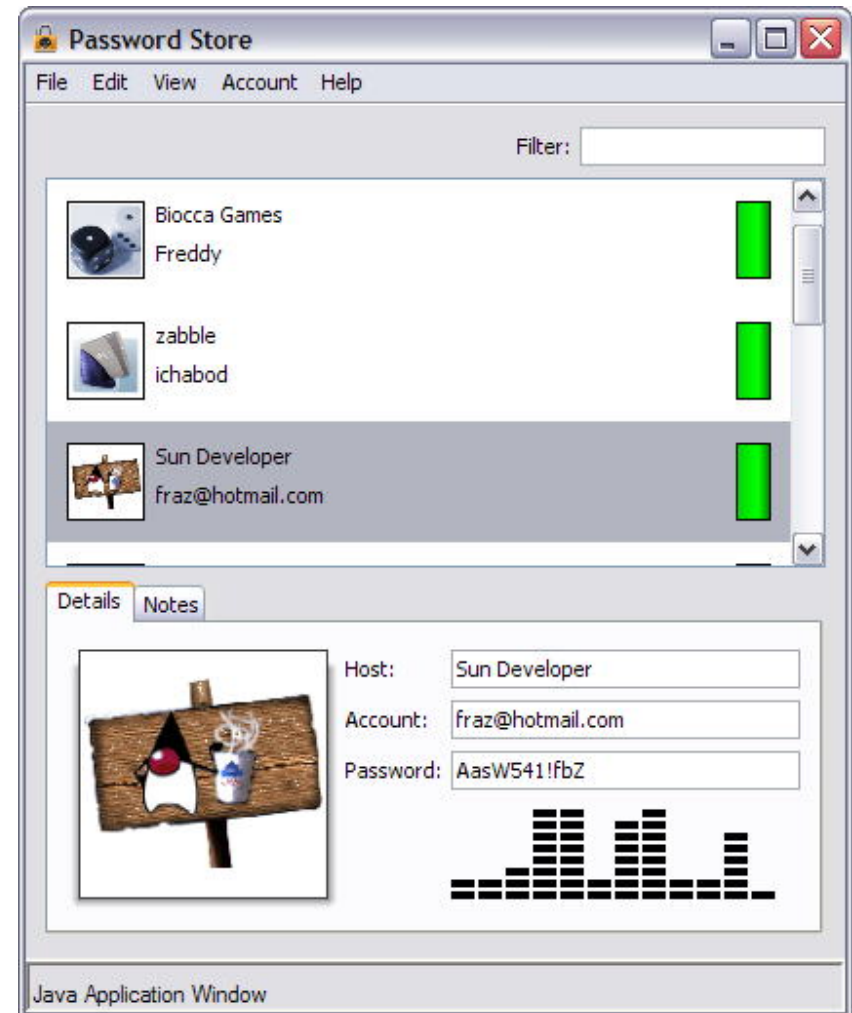
- PasswordStore allows the user to manage login information for various hosts.
- It also generates passwords, evaluates the effectiveness of a password, and allows you to store notes about a particular host or assign an icon to represent the host.

A Swing Demo

Here is an example of an application, PasswordStore

- **Host Info**

- At program launch, the list of hosts is displayed in a Swing list component. Using the View menu, the view can be toggled between the table and the list. In both views, the **Host/Account Filter** text field can be used to dynamically restrict the entries to those where the host or account name contains the typed string.

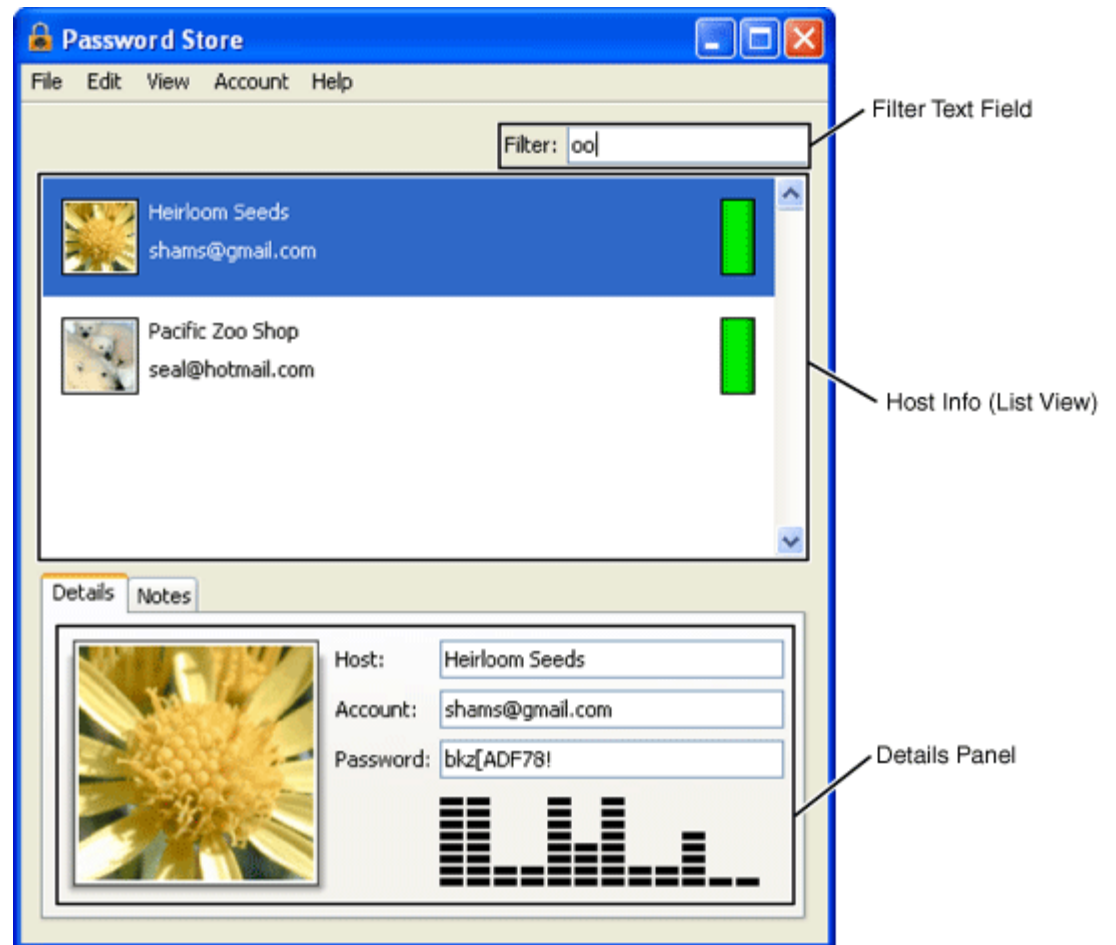


A Swing Demo

- **List View**

- The Swing list component can be customized to include visual data. As shown in the following figure, an optional miniature icon to the left of the host name represents the host. The graphic to the right uses color and proportional fill to reflect the strength of the password (Red = poor, yellow = fair, green = good). The bar changes dynamically as the user enters/modifies the password in the text field below. The user has typed the text "oo" in the filter text field, which matches two entries: Heirloom Seeds and Pacific Zoo Shop.

- Host Info (List View) and Filter Text Field →



A Swing Demo

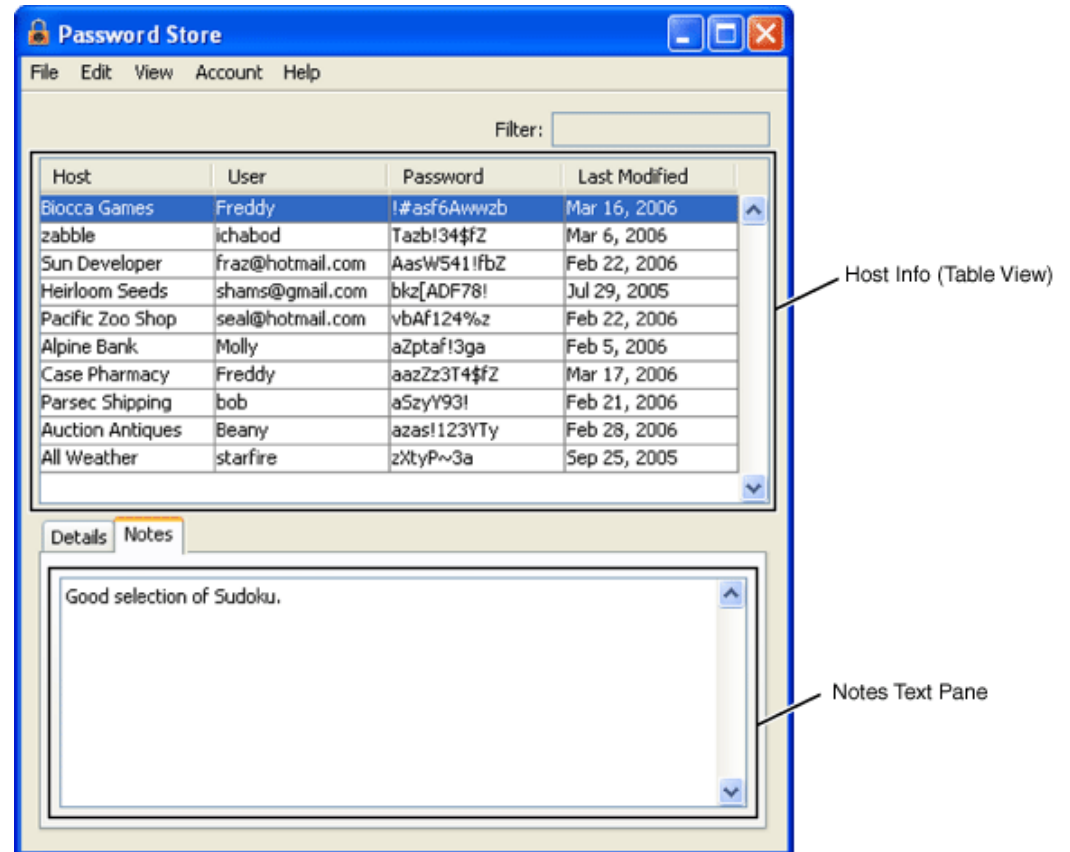
- **Table View**

- The Swing table component allows the user to rearrange the columns by dragging the column header. Also, a column can be sorted by clicking the column header. If the column you click on isn't highlighted as the primary sorted column, it will become the primary sorted column in ascending order. Clicking on the primary sorted column toggles the sort order.

For example, if column 1 isn't selected, clicking on it will make it the selected column and the data is sorted in ascending order.

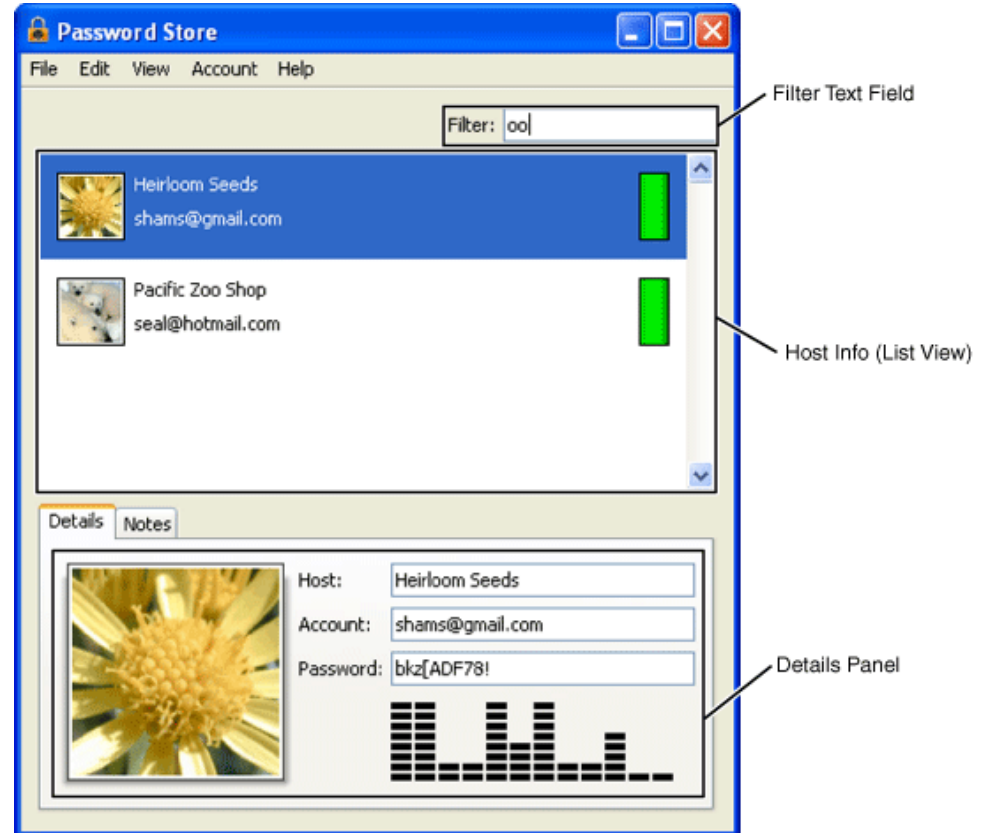
Clicking column 1 again will sort the data in descending order.

Clicking on column 2 will make column 2 the primary column in ascending order.



A Swing Demo

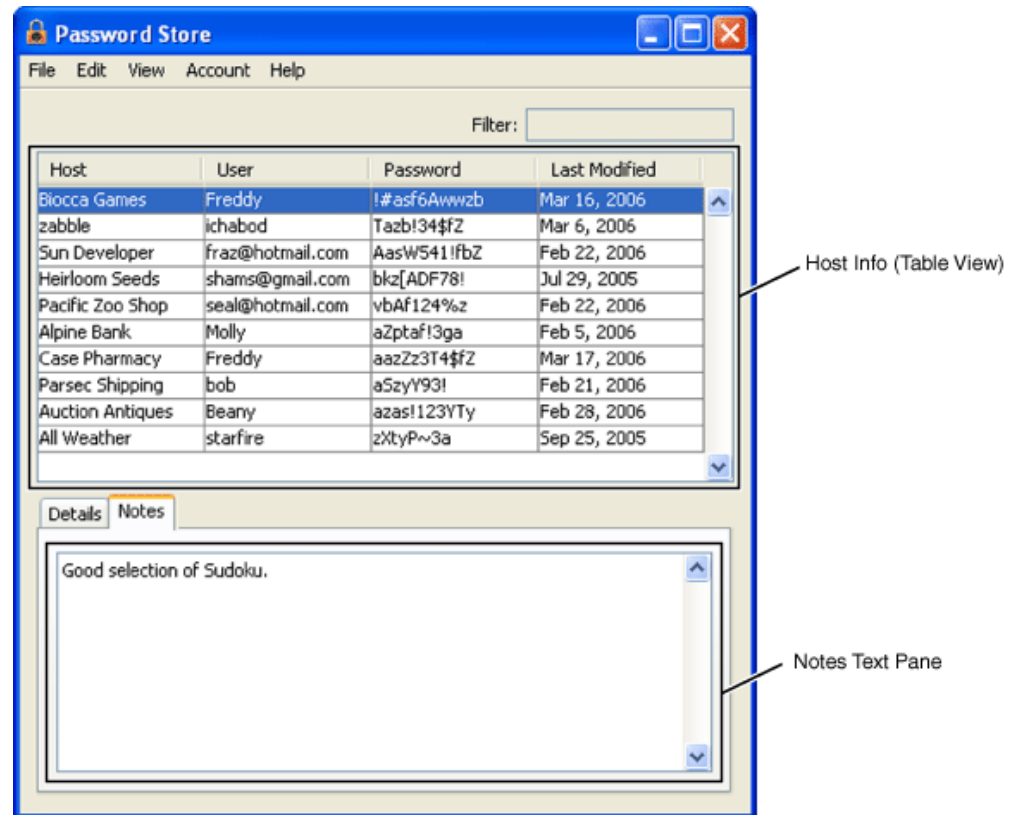
- **Details/Notes Tabbed Pane**
 - The tabbed pane below the host info allows the user to choose between the Details panel and the Notes text pane, keeping the overall footprint of the window smaller and less overwhelming.
- **Details Panel**
 - The icon area on the left can be assigned an image by either dragging an image (jpg, png, gif, or tif) to the area or by clicking the image well and bringing up a file browser. The text fields (used to enter or modify the host name, login, and password) support cut/copy, paste, drag, drop, undo, and redo.
 - As the user enters or modifies the password, the 2D bar chart dynamically displays the distribution of the password. If the list view is currently displayed, the corresponding colored bar in the list also changes dynamically.



A Swing Demo

- **Notes Text Pane**

- This is the text component where the user can save notes about the selected host. If the text pane contains a URI, Swing's text component provides the ability to click on the URI and a browser window automatically opens to that location.



A Swing Demo

- **Wizzy 2D Graphics**
 - PasswordStore uses customized graphics in several ways to enhance the UI: In the list view, images are used to represent each host; a colored bar, the *Strength Visualizer*, represents the effectiveness of a password; and a dynamic bar chart, the *Password Visualizer*, displays the distribution of a password. When you add an image, whether by dragging and dropping it into the image well (in the Details panel) or by clicking the well and bringing up the file browser, a mini-icon is automatically generated for the list view.

A Swing Demo

- **Multiple Look and Feels**

- This provides the ability to switch between three look and feels using the View menu: Java (called Metal), Motif/CDE, and the native look and feel: Windows on Microsoft Windows, Aqua on Mac OS X, and so on.

- **Undo and Redo**

- Undo and redo works on text, as you would expect, but it also works on actions. For example, you can generate a password using the Account > Generate Password menu, and if you don't like the new password you can undo it using Edit > Undo or the control-Z shortcut. Similarly, you can redo the undo using Edit > Redo, or the control-Y shortcut.

Basic Controls

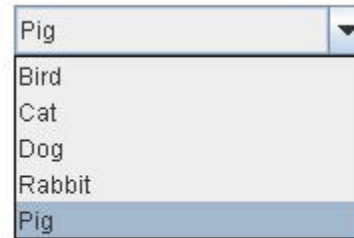
Simple components that are used primarily to get input from the user; they may also show simple state.



[JButton](#)



[JCheckBox](#)



[JComboBox](#)



[JList](#)



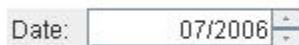
[JMenu](#)



[JRadioButton](#)



[JSlider](#)



[JSpinner](#)



[JTextField](#)



[JPasswordField](#)

Interactive Displays of Highly Formatted Information

- These components display highly formatted information that (if you choose) can be modified by the user.

Host	User	Password	Last Modified
Biocca Games	Freddy	#asf6Awwzb	Mar 16, 2006
zabble	ichabod	Tazb!34\$Iz	Mar 6, 2006
Sun Developer	fraz@hotmail.co...	AasVW541!fbZ	Feb 22, 2006
Heirloom Seeds	shams@gmail...	bkzjADF78!	Jul 29, 2005
Pacific Zoo Shop	seal@hotmail.c...	vbAf1 24%z	Feb 22, 2006

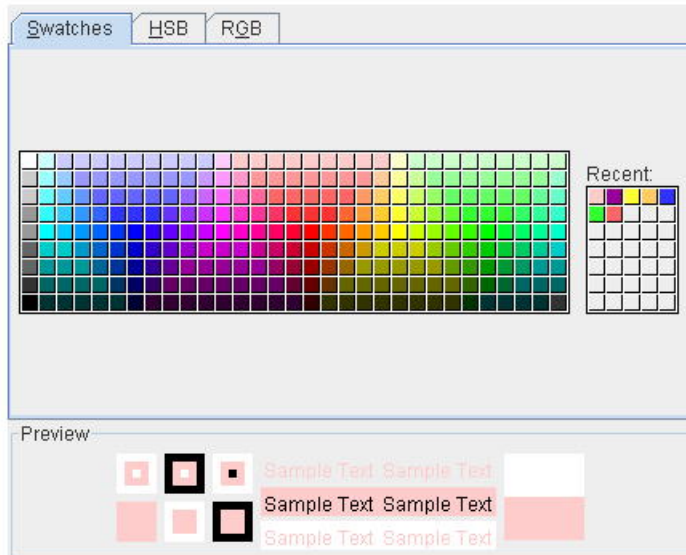
This is an editable JTextArea. A text area is a "plain" text component, which means that although it can display text in any font, all of the text is in the same font.



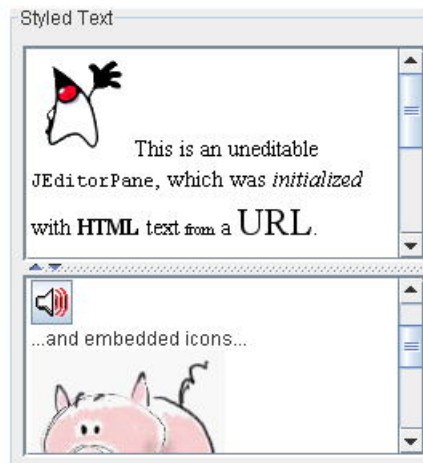
[JTable](#)

[JTextArea](#)

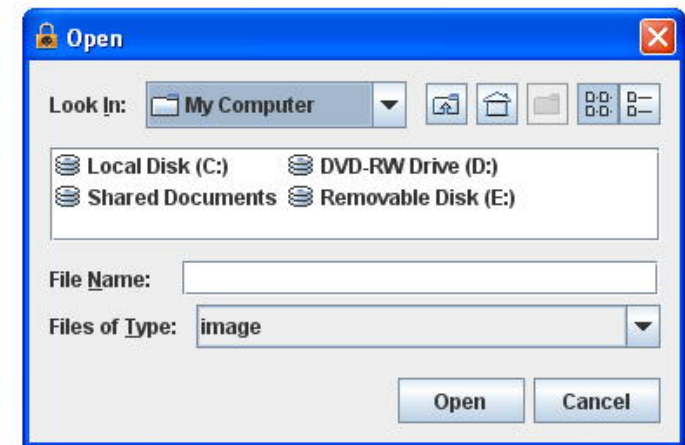
[JTree](#)



[JColorChooser](#)



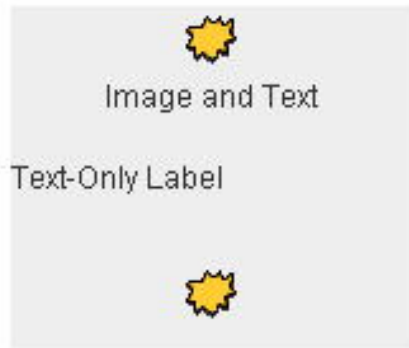
[JEditorPane](#) and [JTextPane](#)



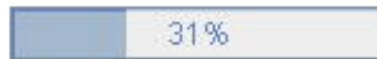
[JFileChooser](#)

Uneditable Information Displays

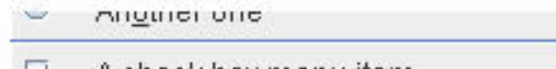
- These components exist solely to give the user information.



[JLabel](#)



[JProgressBar](#)



[JSeparator](#)



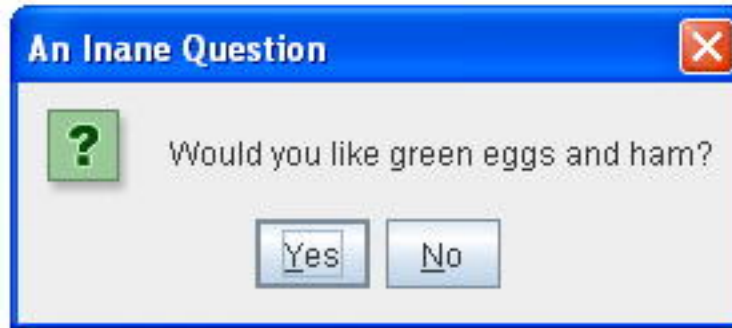
[JToolTip](#)

Top-Level Containers

- At least one of these components must be present in any Swing application.



[JApplet](#)



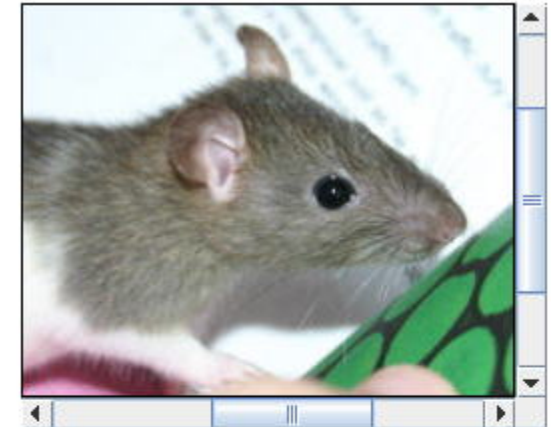
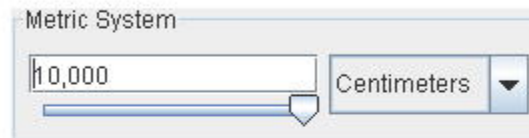
[JDialog](#)



[JFrame](#)

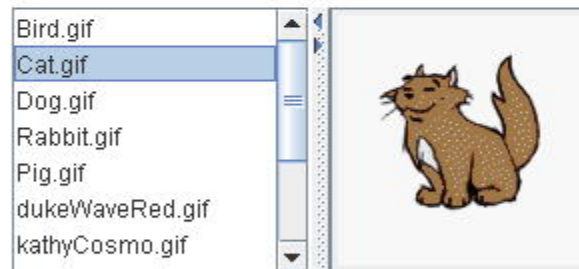
General-Purpose Containers

- These general-purpose containers are used in most Swing applications.



[JPanel](#)

[JScrollPane](#)



[JSplitPane](#)

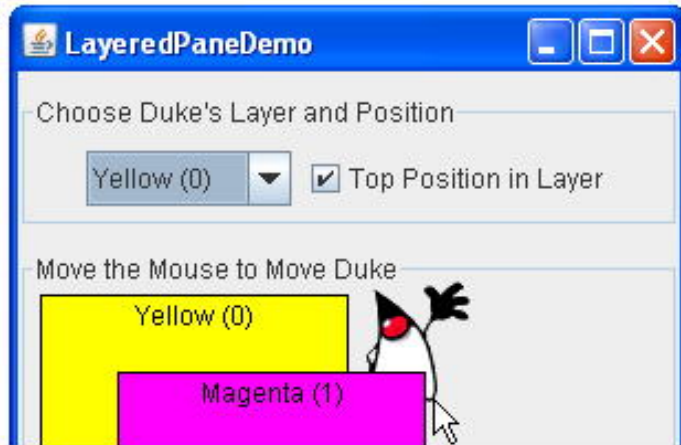
[JTabbedPane](#)



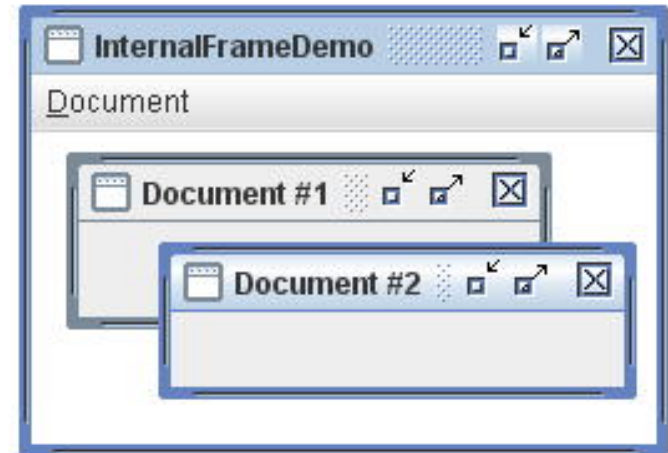
[JToolBar](#)

Special-Purpose Containers

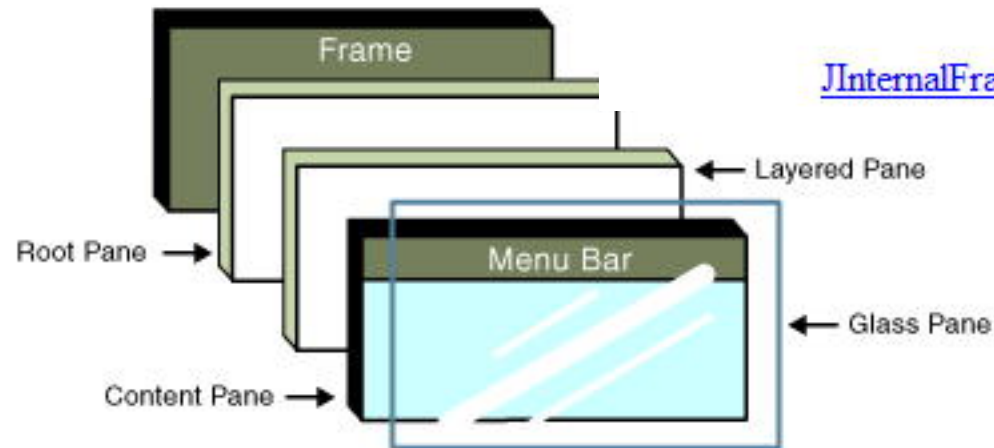
- These special-purpose containers play specific roles in the UI.



[JLayeredPane](#)



[JInternalFrame](#)



[Root pane](#)

Integrating with the Desktop

- The [Desktop API](#), introduced in version 6 of the Java Platform, Standard Edition (Java SE), enables Java applications to integrate seamlessly with the desktop. Three types of integration are supported:
- The ability to launch the host system's default browser with a specific Uniform Resource Identifier (URI).
- The ability to launch the host system's default email client.
- The ability to launch applications to open, edit, or print files associated with those applications.

System Tray Icon Support

- The desktop of some platforms, such as Microsoft Windows, includes a *system tray*, as shown in the following screenshot:



- On Microsoft Windows, it is called the "Taskbar Status Area." On Gnome, the "Notification Area", and on KDE, the "System Tray." However it may be called, the system tray is shared by all applications.
- On platforms where it is supported, an application may insert a mini-icon, called a *Tray Icon*, into the system tray. This icon can be used to notify the user of a change in the application's status, or a need to take a particular action. Clicking the tray icon can bring up the application window. A popup menu and a tooltip can also be attached to the tray icon.
- [System tray](#) support was added in version 6 of Java SE.

Getting Started with Swing

- **Which Swing Packages Should I Use?**

- The Swing API is powerful, flexible — and immense. The Swing API has 18 public packages:

javax.accessibility

javax.swing.text

javax.swing.plaf.basic

javax.swing.border

javax.swing.text.html.parser

javax.swing.plaf.multi

javax.swing.event

javax.swing.tree

javax.swing.table

javax.swing.plaf

javax.swing

javax.swing.text.html

javax.swing.plaf.metal

javax.swing.colorchooser

javax.swing.text.rtf

javax.swing.plaf.synth

javax.swing.filechooser

javax.swing.undo

- Fortunately, most programs use only a small subset of the API. The next slides sort out the API for you, giving you examples of common code and pointing you to methods and classes you're likely to need. Most of the code in this trail uses only one or two Swing packages:
- javax.swing
- javax.swing.event (not always required)

Hello World example

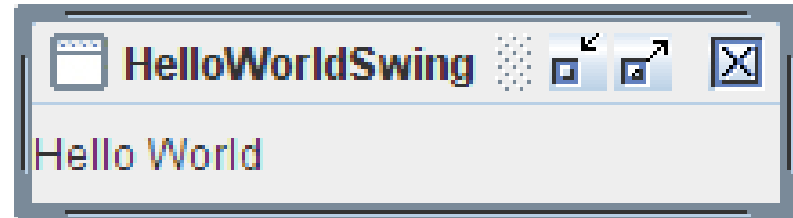
```
/* * HelloWorldSwing.java requires no other files. */
import javax.swing.*;
public class HelloWorldSwing {
/**
 * Create the GUI and show it. For thread safety,
 * this method should be invoked from the
 * event-dispatching thread. */

private static void createAndShowGUI() {
//Create and set up the window.
JFrame frame = new JFrame("HelloWorldSwing");
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

//Add the ubiquitous "Hello World" label.
JLabel label = new JLabel("Hello World");
frame.getContentPane().add(label);

//Display the window.
frame.pack();
frame.setVisible(true);
}

public static void main(String[] args) {
//Schedule a job for the event-dispatching thread:
//creating and showing this application's GUI.
javax.swing.SwingUtilities.invokeLater(new Runnable() {
public void run() {
createAndShowGUI();
}
});
}
}
```



Using Top-Level Containers

- Swing provides three generally useful top-level container classes:
 - [JFrame](#), [JDialog](#), and [JApplet](#).
 - When using these classes, you should keep these facts in mind:
 - To appear onscreen, every GUI component must be part of a *containment hierarchy*.
 - A containment hierarchy is a tree of components that has a top-level container as its root. We'll show you one in a bit.
 - Each GUI component can be contained only once.
 - If a component is already in a container and you try to add it to another container, the component will be removed from the first container and then added to the second.
 - Each top-level container has a content pane that, generally speaking, contains (directly or indirectly) the visible components in that top-level container's GUI.
 - You can optionally add a menu bar to a top-level container.
 - The menu bar is by convention positioned within the top-level container, but outside the content pane. Some look and feels, such as the Mac OS look and feel, give you the option of placing the menu bar in another place more appropriate for the look and feel, such as at the top of the screen.
- **Note:** Although [JInternalFrame](#) mimics JFrame, internal frames aren't actually top-level containers.

package components;

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

```
/* TopLevelDemo.java requires no other files. */
```

```
public class TopLevelDemo {
    /**
     * Create the GUI and show it. For thread safety,
     * this method should be invoked from the
     * event-dispatching thread.
     */
    private static void createAndShowGUI() {
        //Create and set up the window.
        JFrame frame = new JFrame("TopLevelDemo");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        //Create the menu bar. Make it have a green background.
        JMenuBar greenMenuBar = new JMenuBar();
        greenMenuBar.setOpaque(true);
        greenMenuBar.setBackground(new Color(154, 165, 127));
        greenMenuBar.setPreferredSize(new Dimension(200, 20));

        //Create a yellow label to put in the content pane.
        JLabel yellowLabel = new JLabel();
        yellowLabel.setOpaque(true);
        yellowLabel.setBackground(new Color(248, 213, 131));
        yellowLabel.setPreferredSize(new Dimension(200, 180));

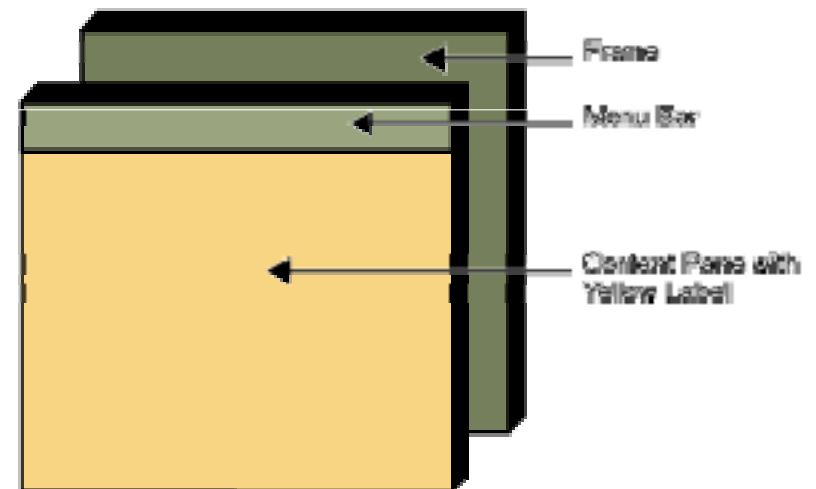
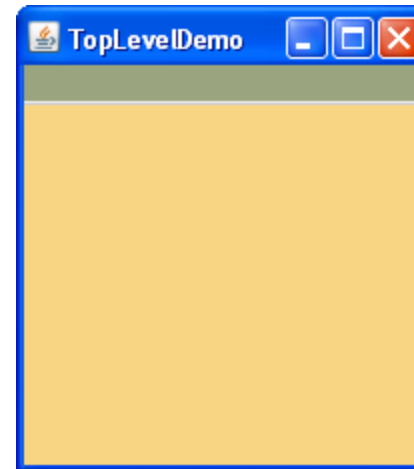
        //Set the menu bar and add the label to the content pane.
        frame.setJMenuBar(greenMenuBar);
        frame.getContentPane().add(yellowLabel, BorderLayout.CENTER);

        //Display the window.
        frame.pack();
        frame.setVisible(true);
    }

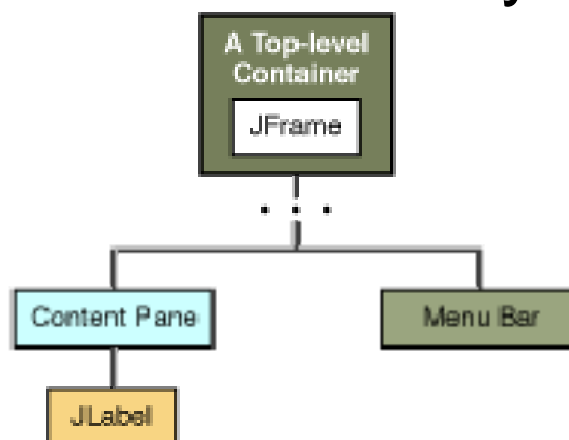
    public static void main(String[] args) {
        //Schedule a job for the event-dispatching thread:
        //creating and showing this application's GUI.
        javax.swing.SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                createAndShowGUI();
            }
        });
    }
}
```

Using Top-Level Containers

- Here's a picture of a frame created by an application. The frame contains a green menu bar (with no menus) and, in the frame's content pane, a large blank, yellow label.



- Although the example uses a JFrame in a standalone application, the same concepts apply to JApplets and JDialogs.
- Here's the containment hierarchy for this example's GUI:



- As the ellipses imply, we left some details out of this diagram. We reveal the missing details a bit later. Here are the topics this section discusses:
 - [Top-Level Containers and Containment Hierarchies](#)
 - [Adding Components to the Content Pane](#)
 - [Adding a Menu Bar](#)
 - [The Root Pane \(a.k.a. The Missing Details\)](#)

Top-Level Containers and Containment Hierarchies

- Each program that uses Swing components has at least one top-level container.
 - This top-level container is the root of a containment hierarchy —
 - the hierarchy that contains all of the Swing components that appear inside the top-level container.
 - As a rule, a standalone application with a Swing-based GUI has at least one containment hierarchy with a JFrame as its root.
 - For example, if an application has one main window and two dialogs, then the application has three containment hierarchies, and thus three top-level containers. One containment hierarchy has a JFrame as its root, and each of the other two has a JDialog object as its root.
- A Swing-based applet has at least one containment hierarchy,
 - exactly one of which is rooted by a JApplet object.
 - For example, an applet that brings up a dialog has two containment hierarchies.
 - The components in the browser window are in a containment hierarchy rooted by a JApplet object.
 - The dialog has a containment hierarchy rooted by a JDialog object.