El futuro digital es de todos

MinTIC

Universidad Pontificia Bolivariana

Vigilada Mineducación

MisiónTIC2022

Interfaces Gráficas

Introducción

Las interfaces gráficas de usuario (IGU) también se construyen a partir de objetos que interactúan, pero tienen una estructura muy especializada y es por esto que evitamos introducirlas antes de aprender la estructura de los objetos en términos más generales. Sin embargo, ahora estamos preparados para dar una mirada a la construcción de las IGU. Las IGU completan nuestras aplicaciones con una interfaz formada por ventanas, menús, botones y otros componentes gráficos, y hacen que la aplicación tenga una apariencia más similar a las típicas aplicaciones que la mayoría de la gente usa hoy en día.

Observe que estamos tropezando nuevamente con el doble significado de la palabra interfaz. Las interfaces de las que estamos hablando ahora no son las interfaces de las clases ni la construcción interface de Java. Hablamos de interfaces de usuario, la parte de una aplicación que está visible en la pantalla y que permite que un usuario interactúe con ella. Una vez que sepamos cómo crear una IGU en Java, podremos desarrollar programas que tengan una mejor presentación visual.

Componentes, gestores de disposición y captura de eventos

En Java, toda la programación de una IGU se realiza mediante el uso de bibliotecas de clases estándares especializadas. Una vez que comprendemos los principios, podemos encontrar todos los detalles necesarios en la documentación de la biblioteca estándar.

Los principios que necesitamos comprender se pueden dividir en tres áreas:

- ¿Qué clase de elementos podemos mostrar en una pantalla?

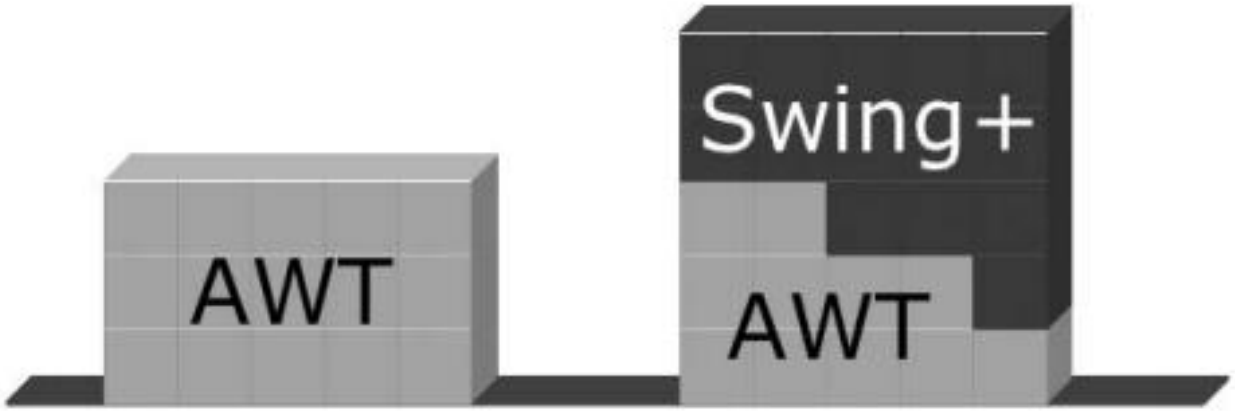
- ¿Cómo podemos acomodar estos elementos?

- ¿Cómo podemos reaccionar ante una entrada del usuario?

Trabajaremos estas cuestiones mediante los términos componentes, gestores de disposición y manejo de eventos.

AWT y Swing

Las AWT y SWING proveen componentes básicos para una GUI (Graphics User Interface - Interface Gráfica de Usuario) y son utilizados en las aplicaciones y los applets de Java. Una de las ventajas de usar AWT es que la interface es independiente de la plataforma o interface gráfica de la máquina. Esto nos asegura que lo que se vea en una computadora aparecerá igual en otra computadora.



Una estrategia para estudiar las AWT es dividirlas en : Componentes, Contenedores, Layouts (Administradores de diseño) y Eventos

Componentes: Son clases o interfaces que permiten crear los objetos gráficos que componen una GUI tales como;botones, listas desplegables, cuadros de texto, casillas de verificación, botones de opción, campos de texto, etiquetas, menús, etc.

Contenedores: Son clases o interfaces que permiten crear objetos gráficos para contener a los componentes, tales como; paneles, cuadros de diálogo, marcos, ventanas, etc.

Layouts: Son clases o interfaces que permiten crear objetos de que administren el diseño, la distribución y colocación de los objetos componentes dentro de los objetos contenedores. Por ejemplo el FlowLayout distribuye los componentes de izquierda a derecha y de arriba a abajo, el BorderLayout los distribuye en cinco áreas geográficas; norte, sur, este, oeste y centro, etc.

Eventos. Son las clases o interfaces que permiten crear objetos que capturen y manejen los eventos. Un evento es una acción sobre algún componente, por ejemplo, clic a un botón, pulsar la tecla de enter en un botón, mover un elemento con las teclas de navegación, eventos especiales como los programados por tiempo, etc. Sin los eventos las GUI serían interfaces gráficas sin vida, y por lo tanto no serían muy útiles que digamos.

Cuando se empieza a utilizar SWING, se observa que Sun ha dado un paso importante adelante respecto al AWT. Ahora los componentes de la interface gráfica son Beans y utilizan el nuevo modelo de Delegación de Eventos de Java. Swing proporciona un conjunto completo de componentes, todos ellos lightweight, es decir, ya no se usan componentes peer dependientes del sistema operativo, ademas, SWING está totalmente escrito en Java. Todo esto conlleva una mayor funcionalidad en manos del programador, y en la posibilidad de mejorar en gran médida la apariencia cosmética de las interfaces gráficas de usuario.

Hay muchas ventajas que ofrece el SWING. Por ejemplo, la navegación con el teclado es automática, cualquier aplicación SWING puede utilizarce sin ratón, sin tener que escribir una línea de código adicional. Las etiquetas de información o “tool tips”, se pueden crear con una sola línea de código. Además, con SWING la apariencia de la aplicación se adapta dinámicamente al sistema operativo y plataforma en que esté corriendo.

Los componentes Swing no soportan el modelo de Eventos de Propagación, sino solamente el modelo de eventos de Delegación incluido desde la versión JDK 1.1; por lo tanto si se van a utilizar componentes SWING, debe programar exclusivamente en el nuevo modelo, o dicho de otro forma, se recomienda al programador construir programas GUI mezclando lo menos posible SWING con AWT.

Los componentes

Los componentes son clases de objetos que permiten utilizar elementos gráficos para crear interfaces gráficas y están divididos en dos grandes grupos: los contenedores y los componentes. Un componente, también denominado componente simple o atómico, es un objeto que tiene una representación gráfica, que se puede mostrar en la pantalla y con la que puede interactuar el usuario. Ejemplos de componentes son los botones, campos de texto, etiquetas, casillas de verificación, menús, etc.,. En los lenguajes de Programación Orientada a Objetos como Java tenemos dos paquetes o conjuntos de clases principales agrupados en el paquete llamado AWT (Abstract Windows Tolkit) y en el paquete SWING, algunos de estos componentes del paquete AWT y del SWING están resumidos en la siguiente tabla

AWT	SWING	Descripción
Button	Jbutton	Es un botón usado para recibir el clic del ratón
Canvas		Unlienzo o panel usado para dibujar
Checkbox	JCheckBox	Cuadro de verificación. Es un componente que le permite seleccionar un elemento
Choice	JComboBox	Es una lista desplegable de elementos estáticos

Component		**Es el padre de todos los componentes AWT, excepto de los componentes de tipo menú**
Container		Es el padre de todos los contenedores
Dialog	JDialog	Es un cuadro de diálogo oon una ventana con titulo y bordes.
Frame	JFrame	Es un marco o ventana y es la clase base de todas las ventanas GUI con controles para ventana.
Label	Jlabel	Etiqueta. Es una cadena de texto como componente
List	Jlist	Un componente que contiene un conjunto dinámico de elementos.
Menu	Jmenu	Es un elemento dentro de la barra de menú, el cual contiene un conjunto de elementos de tipo menú.
MenuItem	JMenuItem	Un elemento dentro de un menú.
Panel	JPanel	Una clase contenedora básica usado frecuentemente para crear diseños (layouts) complejos.
Scrollbar	JScrollBar	Un componente que permite al usuario hacer una selección dentro de un rango de valores
ScrollPane	JScrollPane	Una clase contenedora que implementa un deslizador horizontal y vertical para un único componente hijo
TextArea	JTextArea	Un componente que permite al usuario introducir texto en un bloque o rectángulo.

Contenedores

Un contenedor es un componente al que se le incluirán uno o mas componentes o contenedores. Los contenedores permiten generar la estructura de una ventana y el espacio donde se muestra el resto de los componentes contenidos. En ella y que conforman la interfaz de usuario. Los contenedores de alto nivel mas utilizados se muestran en la siguiente tabla

Clase	Utilidad
javax.swing.JFrame	Proporciona una ventana principal de aplicación con su funcionalidad normal (p.ej. Borde, título, menús) y un panel de contenido
javax.swing.JDialog	Proporciona una ventana de diálogo auxiliar en una aplicación, normalmente utilizada para pedir datos al usuario o configurar elementos.
javax.swing.JApplet	Implementa una ventana que aparece dentro de otra aplicación que normalmente es un navegador de internet.

Los Frames

Se emplea para crear una ventana principal de una aplicación. Esta ventana posee un marco que incluye los controles de minimizar, restaurar/maximizar y cerrar. A este contenedor se le puede agregar una barra de menús (*javax.swing.JMenuBar*). Los componentes gráficos no se añaden directamente al JFrame sino a su panel de contenido. También el administrador de diseño (*Layout*) se le debe aplicar a este panel de contenido. Los métodos mas frecuentemente utilizados en un JFrame son:

```
f.setSize(420,450); // tamaño de la ventana
f.setLocation(50,50); // posición de la esquina superior izquierda de la ventana
f.getContentPane().setLayout(objetoLayout); // establece el administrador de diseño
f.setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE); // cerrar ventana
f.setTitle("Operadores de Java"); // pone texto en el título de la ventana
f.setJMenuBar(barraDeMenu); // agrega una barra de menú a la ventana
f.setVisible( true ); // hace visible o invisible la ventana
f.pack(); // asigna tamaño iniciales a los componentes
f.getContentPane().add(componente); // añade un componente a la ventana
f.addXXXXXListener(objetoListener ); // añade un oyente de eventos
```

JDialog

La clase *javax.swing.JDialog* es la clase raíz de las ventanas secundarias que implementan cuadros de diálogo en SWING. Estas ventanas dependen de una ventana principal (o con marco, normalmente de la clase JFrame) y si la ventana principal se cierra, se iconiza o desiconiza, las ventanas secundarias realizan la misma operación de forma automática. Estas ventanas pueden ser modales o no modales, es decir, limitan la interacción con la ventana principal si así se desea. El constructor más frecuentemente utilizado es decir, limitan la interacción con la ventana principal si así se desea. El constructor más frecuentemente utilizado es:

```
// crea un Jdialog con la ventana de la que depende, un título y si es modal o no.
JDialog dialogo = new JDialog( frame, “Titulo del Jdialog”, true );
```

La clase *JDialog* puede utilizar todos los métodos descritos para la clase JFrame.

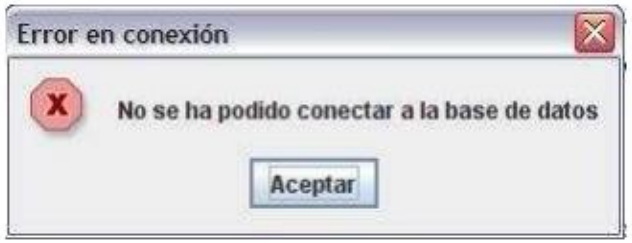
Cuadros de diálogo estándar con la clase *javax.swing.JOptionPane.*

Esta clase se utiliza para crear los tipos de cuadros de diálogo más habituales, como los que permiten pedir un valor, mostrar un mensaje de error o advertencia, solicitar una confirmación, etc. Todos los cuadros de diálogo que implementa *JOptionPane* son modales (esto es, boquean la interacción del usuario con otras ventanas). La forma habitual de uso de la clase es mediante la invocación de alguno de sus métodos estáticos para crear cuadros de diálogo.

Tienen el formato showDialog, donde el tipo puede ser:

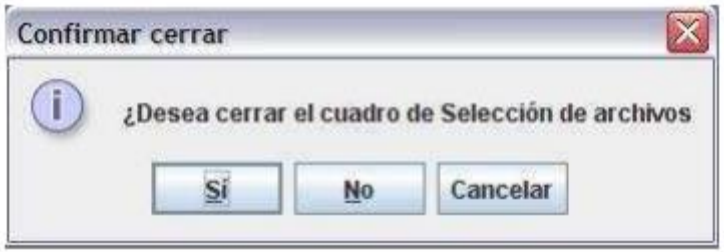
Message para informar al usuario con un mensaje

```
//cuadro de mensaje
JOptionPane.showMessageDialog(ventana,
 "No se ha podido conectar a la base de datos", //mensaje
 "Error en conexión", // título
 JOptionPane.ERROR_MESSAGE); // icono
```



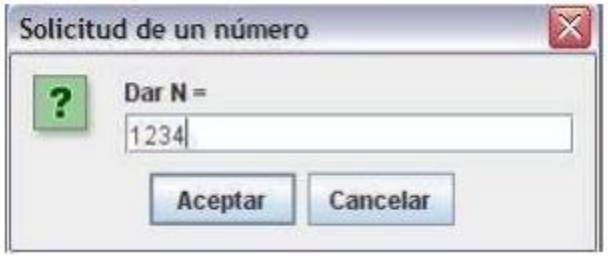
Confirm para solicitar una confirmación al usuario con las posibles respuestas de si, no o cancelar

```
// cuadro de confirmación. Devuelve YES_OPTION , NO_OPTION
int confirma = JOptionPane.showConfirmDialog(ventana,
 "¿Desea cerrar el cuadro de Selección de archivos", //mensaje
 "Confirmar cerrar", // título
 JOptionPane.YES_NO_CANCEL_OPTION, //botones
 JOptionPane.INFORMATION_MESSAGE); // icono

if( confirma == JOptionPane.YES_OPTION ) // se compara confirmación
 System.exit( 0 );
```



Input para solicitar la entrada de un datos

```
//cuadro de entrada. Devuelve la cadena introducida o null si se cancela
String cadena = JOptionPane.showInputDialog(ventana,
 "Dar N = ", // mensaje
 "Solicitud de un número", //título
 JOptionPane.QUESTION_MESSAGE); // icono
int n = Integer.parseInt(cadena); // conversion de cadena a entero
```

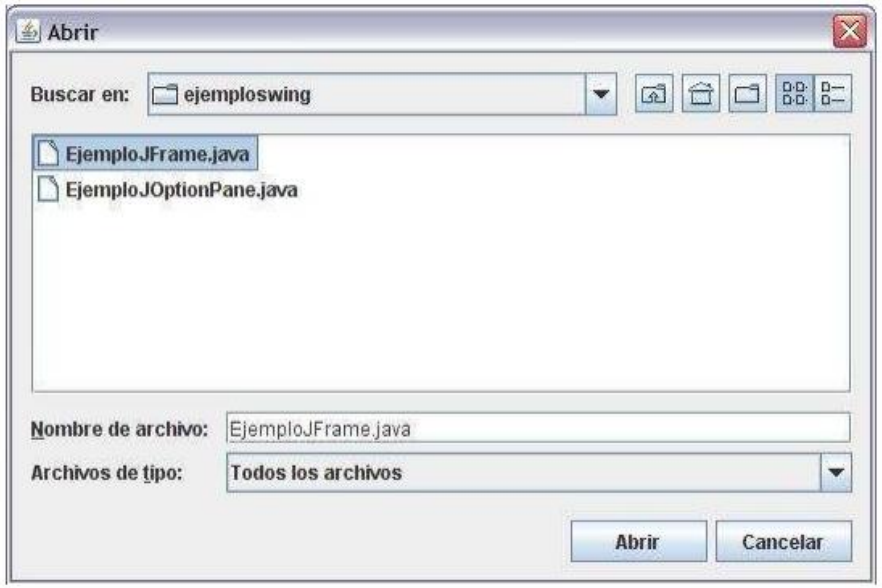


Option permite crear una ventana personalizada de cualquiera de los tipos anteriores

```
//Cuadro de opción personalizado, devuelve en n el botón pulsado
String[] perrosFamosos = { "Snoopy","Super Can","Fido","Patan","Tribilín",
 "Scoobe Doo","Pluto","Rintintín","Lasie" };

 int n = JOptionPane.showOptionDialog(null,
 "¿Cual es su perro favorito?", //mensaje
 "Mensaje de confirmación", //título
 JOptionPane.YES_NO_CANCEL_OPTION, // botones
 JOptionPane.QUESTION_MESSAGE,
 null, // utiliza icono predeterminado
 perrosFamosos,
 perrosFamosos[0]); // botón determinado
 System.out.println( "Mi perro favorito es: " + perrosFamosos[n] );
```

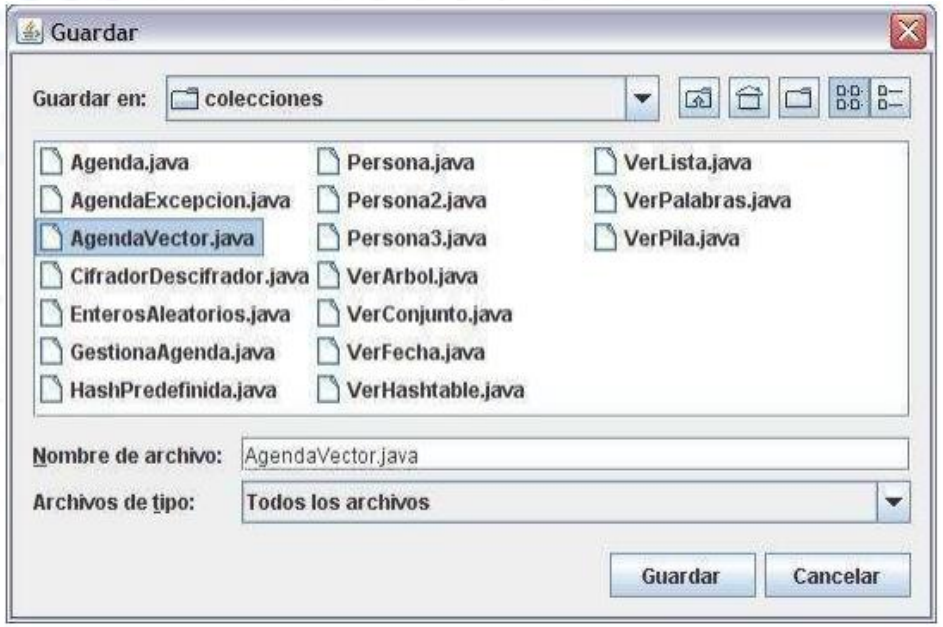
Cuadros de diálogo estándar con la clase javax.swing.JFileChooser.

La clase JFileChooser tiene métodos como el showOpenDialog(ventana) y el showSaveDialog(ventana) que permiten la elección interactiva de un archivo o directorio




```java
import java.io.File;
import javax.swing.JFileChooser;

public class EjemploFileChooserSaveDialog {

    public static void main( String[] a ) {
        File fsal= null;
        // se crea el selector de archivos
        JFileChooser selector = new JFileChooser();
        //solo posibilidad de seleccionar directorios
        selector.setFileSelectionMode( JFileChooser.FILES_AND_DIRECTORIES);
        //se muestra y se espera a que el usuario acepte o cancele la selección
        int opcion = selector.showSaveDialog(null); // cuadro de Guardar archivo
        if ( opcion == JFileChooser.APPROVE_OPTION) {
            // se obtiene el archivo o directorio seleccionado
            fsal = selector.getSelectedFile(); // devuelve un objeto File
            System.out.println("Nombre archivo:" + fsal.getName() + "\n" + "Directorio padre: " +
            fsal.getParent() + "\n" + "Ruta: " + fsal.getPath() );
        }
    }// fin del main
} // fin de la clase EjemploFileChooserSaveDialog
```

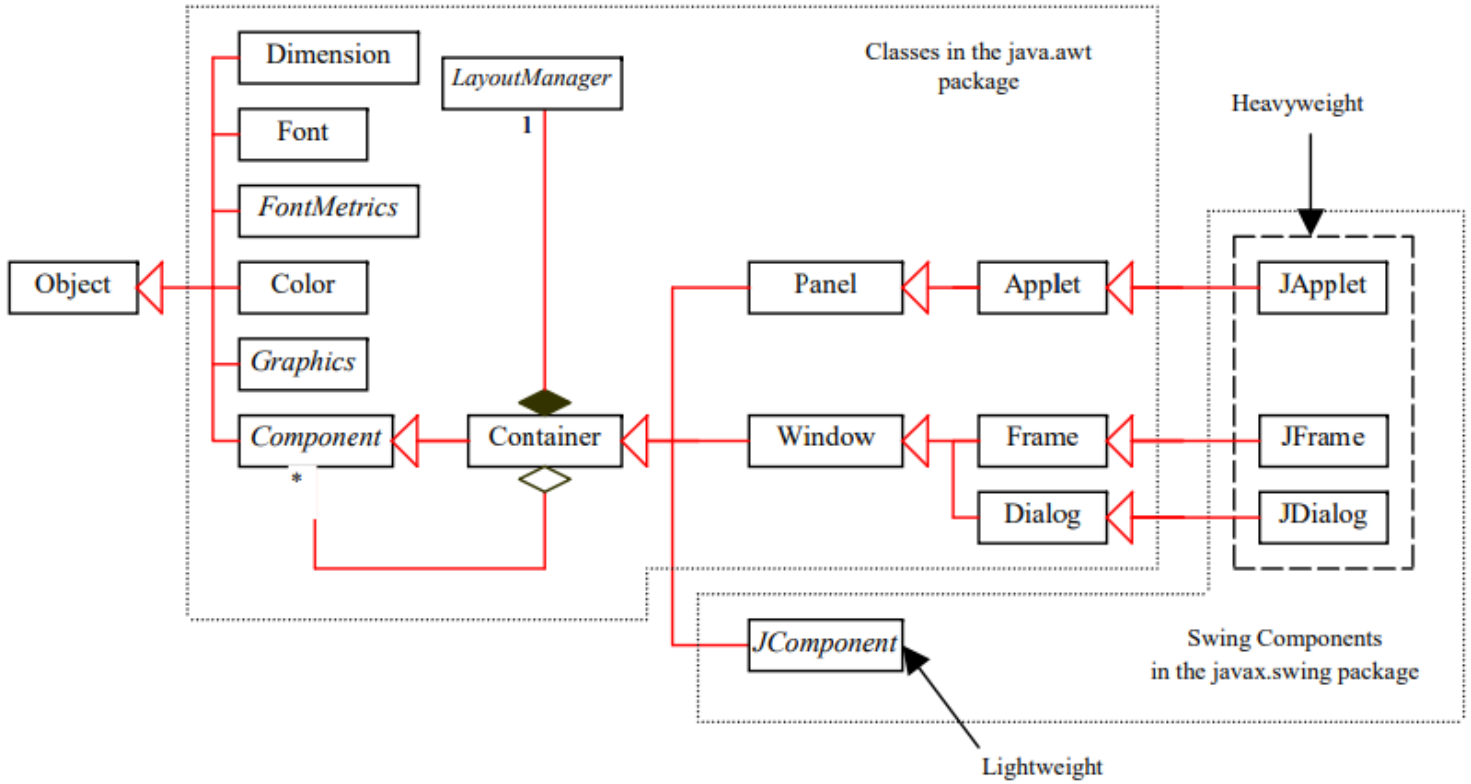
Clase EjemploFileChooserSaveDialog.java que muestra el uso de un cuadro de diálogo de Guardar archivos.



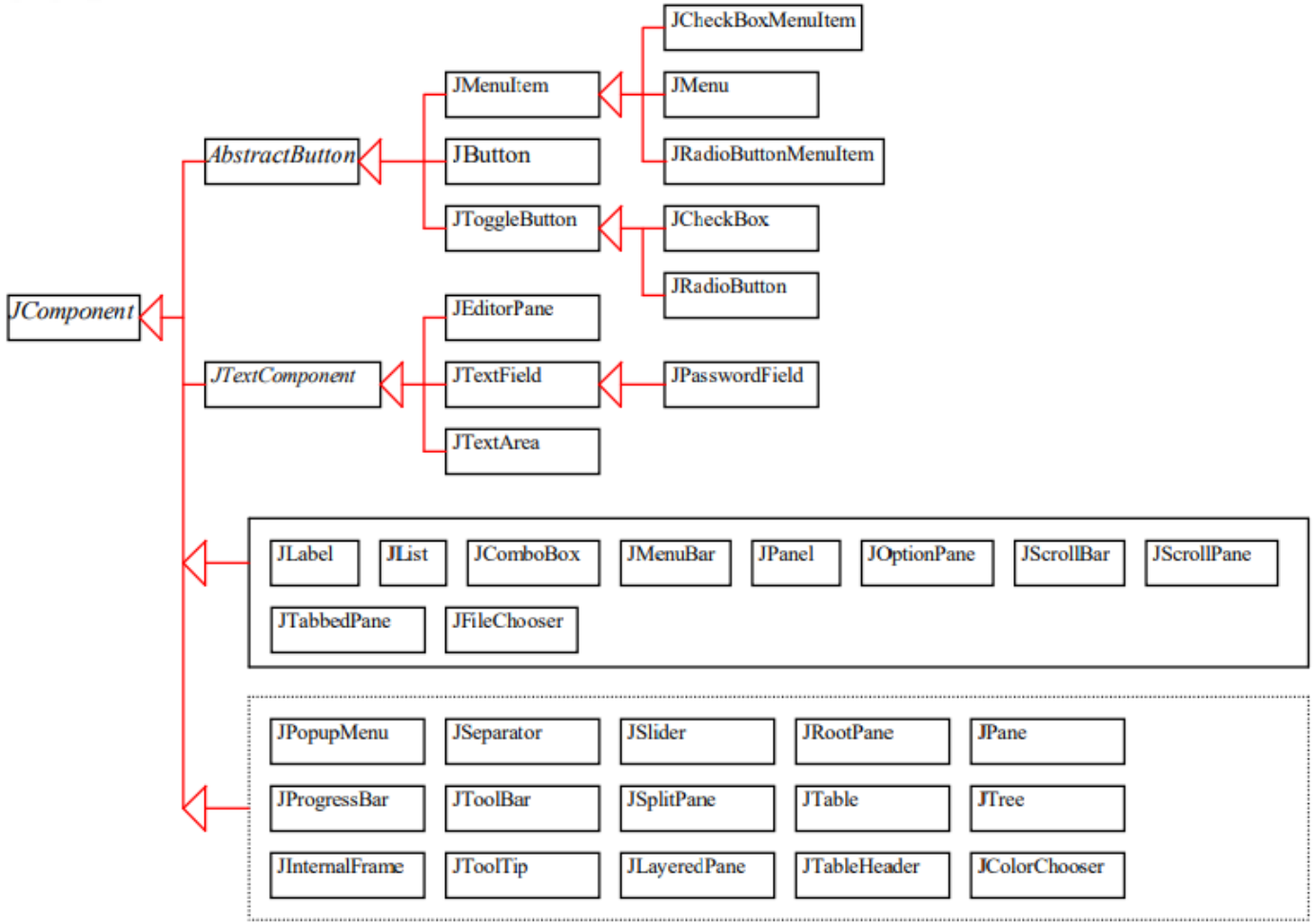

```java
import java.io.File;
import javax.swing.JFileChooser;

public class EjemploFileChooserSaveDialog {

    public static void main( String[] a ) {
        File fsal= null;
        // se crea el selector de archivos
        JFileChooser selector = new JFileChooser();
        //solo posibilidad de seleccionar directorios
        selector.setFileSelectionMode( JFileChooser.FILES_AND_DIRECTORIES);
        //se muestra y se espera a que el usuario acepte o cancele la selección
        int opcion = selector.showSaveDialog(null); // cuadro de Guardar archivo
        if ( opcion == JFileChooser.APPROVE_OPTION) {
            // se obtiene el archivo o directorio seleccionado
            fsal = selector.getSelectedFile(); // devuelve un objeto File
            System.out.println("Nombre archivo:" + fsal.getName() + "\n" +
            "Directorio padre: " + fsal.getParent() + "\n" +
            "Ruta: " + fsal.getPath() );
        }
    }
}
```

Jerarquía de clases para las GUI

Las clases de AWT reemplazadas por Swing se llaman igual pero con una J delante



Jerarquía de clases para las GUI: JComponent



Jerarquía de clases para las GUI: AWT