Universidad Politécnica de Tlaxcala Región Poniente

Ingeniería en Sistemas Computacionales

Materia: Programación Orientada a Objetos

Docente: Ing. Vanesa Tenopala Zavala

Alumnos: 22SIC008 Isaac Brandon Martínez Ramírez

Tema: Reportes programas GUI

Ciclo escolar: mayo-agosto 2024

Fecha: 16 de junio de 2024

Reporte del Programa: VentanaConTexto

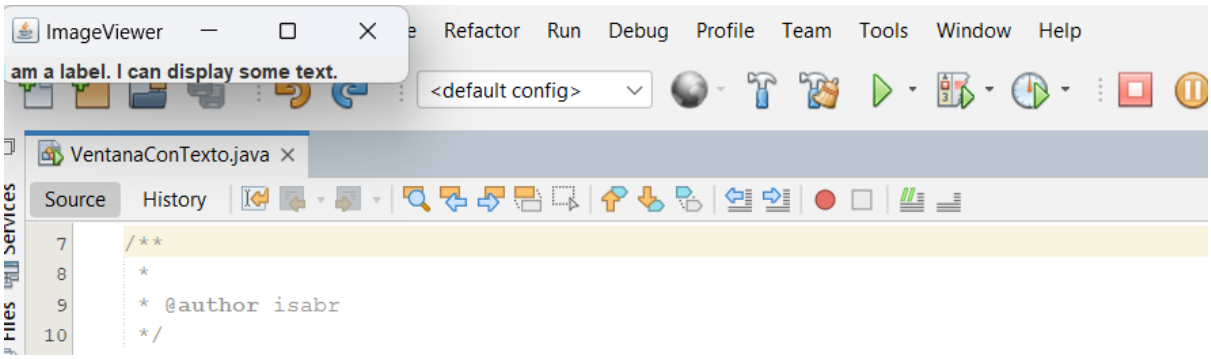
Descripción General El programa VentanaConTexto es una aplicación súper simple en Java que usa Swing para crear una interfaz gráfica (GUI). Básicamente, muestra una ventana con un texto dentro.

Detalles del Código

- **JFrame**: Utilizamos un JFrame para crear la ventana principal, que en nuestro caso se llama "ImageViewer".
- **Método makeFrame()**: Este método configura la ventana. Creamos un Container para el contenido de la ventana, añadimos un JLabel con el texto "I am a label. I can display some text.", y finalmente, ajustamos el tamaño de la ventana y la hacemos visible.
-

Resultado

Al ejecutar el programa, ves una ventana con el título "ImageViewer" y el texto "I am a label. I can display some text."



Conclusión

Este programa es un buen punto de partida para aprender sobre la creación de interfaces gráficas en Java utilizando Swing. Es simple y directo, perfecto para principiantes.

Reporte del Programa: CrearBarraMenus

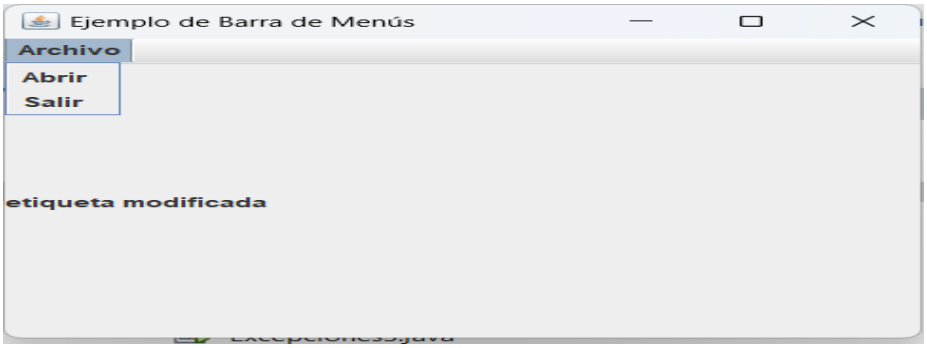
Descripción General

El programa CrearBarraMenus es una app sencilla en Java. Usa Swing para crear una GUI que incluye una barra de menús con un menú "Archivo" y dos opciones: "Abrir" y "Salir". También hay un botón y dos etiquetas en la ventana.

Detalles del Código

- **Importaciones**: Necesitamos javax.swing y java.awt.Container.
- **Clase Principal**: CrearBarraMenus tiene el método main, que es el punto de entrada.
- **Creación de la Ventana**: Creamos un JFrame, configuramos el tamaño y el comportamiento al cerrar, y lo hacemos visible.
- **Barra de Menús**: El método makeMenuBar() crea la barra de menús y añade las opciones.
- **Añadir Componentes**: El método addComponents() añade el botón y las etiquetas a la ventana.

Resultado



Conclusión

Este programa es ideal para entender cómo funcionan las barras de menú y los componentes básicos en una aplicación Java Swing.

Reporte del Programa: ProcesamientoImagenes

Descripción General

Este programa simula un procesamiento de imágenes básico como "ImageJ". Tiene botones para oscurecer, aclarar y aplicar un umbral a una imagen.

Estructura del Programa

- **Clase Principal**: ProcesamientoImagenes
- **Métodos**:
    - ProcesamientoImagenes(): Constructor que inicializa la GUI y carga la imagen.
    - main(String[] args): Inicia la aplicación.

Componentes de la GUI

- btnDarker: Botón para oscurecer la imagen.
- btnLighter: Botón para aclarar la imagen.
- btnThreshold: Botón para aplicar un umbral.
- imageLabel: Etiqueta para mostrar la imagen.

Resultado

Conclusión

Este programa es una buena base para desarrollar aplicaciones de procesamiento de imágenes en Java. Necesita más trabajo para ser funcional, pero cubre los conceptos básicos.

Reporte del Programa: EjemploDisposicion

Descripción General

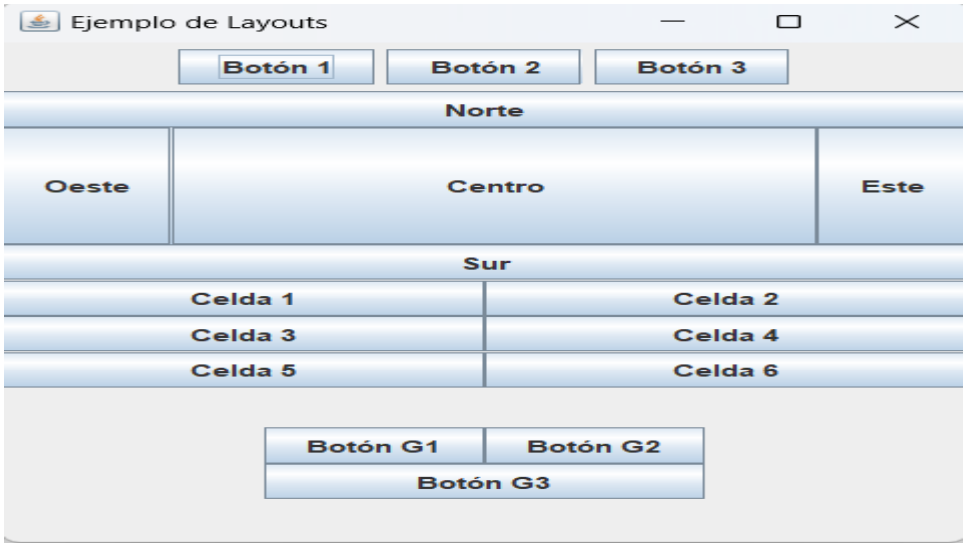
Este programa demuestra el uso de diferentes gestores de disposición en Java para organizar componentes de la GUI.

Gestores de Disposición Utilizados

- **FlowLayout**: Coloca los componentes de izquierda a derecha y luego baja a la siguiente línea.
- **BorderLayout**: Coloca los componentes en los lados (Norte, Sur, Este, Oeste) y en el centro.
- **GridLayout**: Coloca los componentes en una cuadrícula.
- **GridBagLayout**: Similar a GridLayout, pero más flexible.

Componentes Utilizados Botones (JButton).

Resultado



Conclusión

Este programa es excelente para aprender cómo organizar componentes en una interfaz gráfica en Java utilizando diferentes gestores de disposición.

Reporte del Programa de Interfaz Gráfica en Java

Descripción General

El programa es una aplicación de escritorio simple escrita en Java. Utiliza la biblioteca Swing para crear una interfaz gráfica de usuario (GUI). La GUI contiene varios tipos de botones y casillas de verificación.

Componentes de la GUI

1. **JButton**: Un botón estándar que el usuario puede hacer clic.
2. **JToggleButton**: Un botón que mantiene su estado después de ser presionado. Cambia de estado cada vez que se hace clic en él.
3. **JCheckBox**: Una casilla de verificación que el usuario puede seleccionar o deseleccionar.
4. **JRadioButton**: Un botón de opción que el usuario puede seleccionar. Los botones de opción están diseñados para ser usados en grupos, donde sólo un botón de un grupo puede estar seleccionado a la vez.

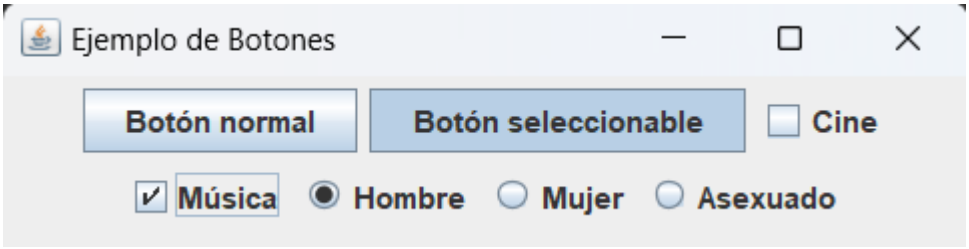
Detalles de Implementación

El programa sigue el patrón de diseño de contenedor. Crea un JFrame que actúa como la ventana principal de la aplicación. Dentro de este JFrame, añade un JPanel, que actúa como un contenedor para los otros componentes de la GUI.

Cada componente de la GUI es creado como una instancia de su respectiva clase (por ejemplo, new JButton("Botón normal")). Estos componentes se añaden al JPanel usando el método add.

Para los JRadioButton, se utiliza una instancia de ButtonGroup para asegurar que sólo un botón de opción puede estar seleccionado a la vez.

Resultado



Conclusión

Este programa es una buena manera de entender cómo funcionan diferentes tipos de botones y casillas de verificación en una interfaz gráfica de Java. Es una base sólida para desarrollar interfaces de usuario más complejas.

Reporte del Programa: EjemploMenu

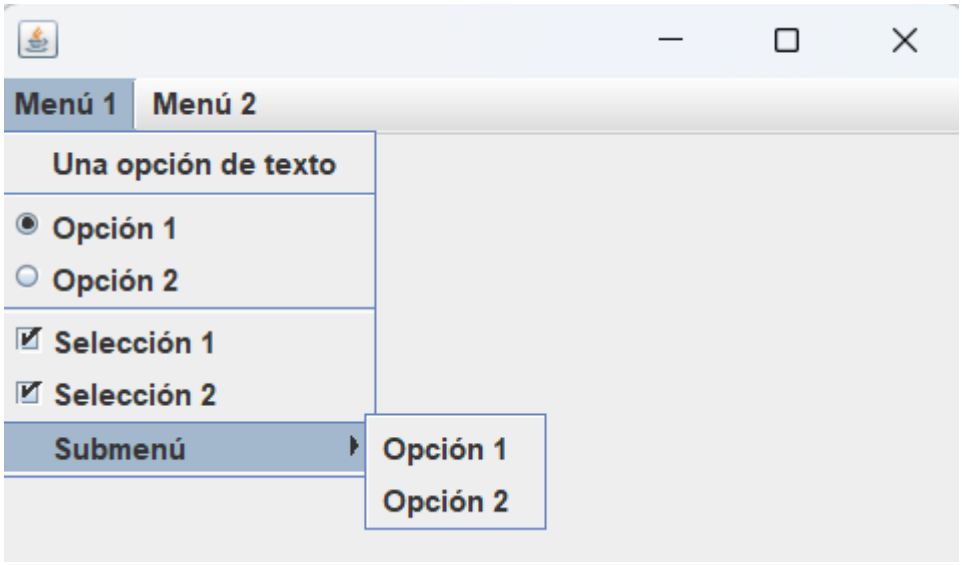
Descripción General

Este programa crea una GUI con una barra de menús que contiene dos menús principales: "Menú 1" y "Menú 2".

Detalles de Implementación

- **JMenuBar**: Barra de menús principal.
- **JMenu**: Dos menús ("Menú 1" y "Menú 2") añadidos a la barra de menús.
- **JMenuItem**: Varias opciones dentro de "Menú 1".
- **JRadioButtonMenuItem**: Opciones exclusivas dentro de "Menú 1".
- **JCheckBoxMenuItem**: Opciones seleccionables dentro de "Menú 1".
- **Submenú**: Un submenú dentro de "Menú 1" con opciones adicionales.

Resultado



Conclusión

Este programa es un buen punto de partida para entender cómo crear menús interactivos en una aplicación Java Swing.

Reporte del Programa: Diálogos Modales en Java

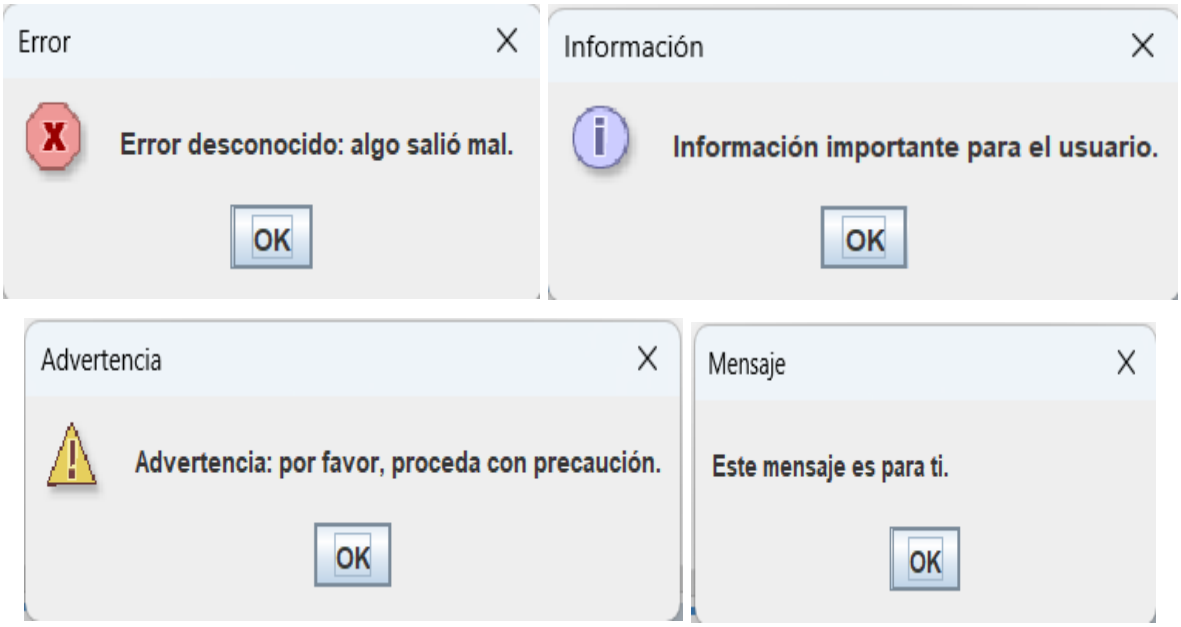
Descripción General

El programa muestra diferentes tipos de diálogos modales usando JOptionPane. Estos diálogos requieren que el usuario interactúe con ellos antes de volver a la ventana principal.

Detalles del Código

- **Clase Principal**: DialogosModales
- **Método main**: Dentro de este método, se crean cuatro tipos de diálogos modales:
  - **Diálogo de Error**: JOptionPane.showMessageDialog con JOptionPane.ERROR_MESSAGE.
  - **Diálogo de Información**: JOptionPane.showMessageDialog con JOptionPane.INFORMATION_MESSAGE.
  - **Diálogo de Advertencia**: JOptionPane.showMessageDialog con JOptionPane.WARNING_MESSAGE.
  - **Diálogo Simple**: JOptionPane.showMessageDialog con JOptionPane.PLAIN_MESSAGE.

Resultados



Conclusión

Este programa es un buen ejemplo de cómo utilizar JOptionPane para mostrar diferentes tipos de diálogos modales en Java.

Reporte del Programa: Diálogos de Confirmación en Java

Descripción General

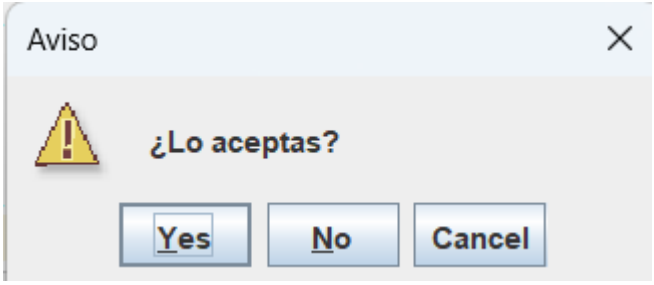
El programa muestra un diálogo de confirmación al usuario usando JOptionPane para mostrar un diálogo de confirmación al usuario. Los diálogos de confirmación son ventanas que presentan al usuario una elección, generalmente en forma de una pregunta, y requieren que el usuario seleccione una opción antes de poder continuar.

Detalles del Código

El código consta de una clase principal ConfirmacionDialogo con un método main. Dentro del método main, se crea un diálogo de confirmación utilizando el método JOptionPane.showConfirmDialog. Este método muestra un diálogo con las opciones “Sí”, “No” y “Cancelar”, y un icono de advertencia.

El valor devuelto por showConfirmDialog se almacena en la variable seleccion, y luego se utiliza en una estructura switch para determinar qué acción tomar en función de la opción seleccionada por el usuario.

Resultado



Conclusión Este programa es un buen ejemplo de cómo usar JOptionPane para crear diálogos de confirmación en Java.

Reporte del Programa: Interacciones Modales en Java

Descripción General

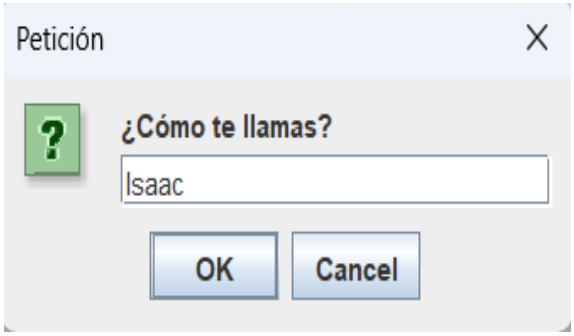
El programa utiliza JOptionPane para interactuar con el usuario a través de diálogos modales.

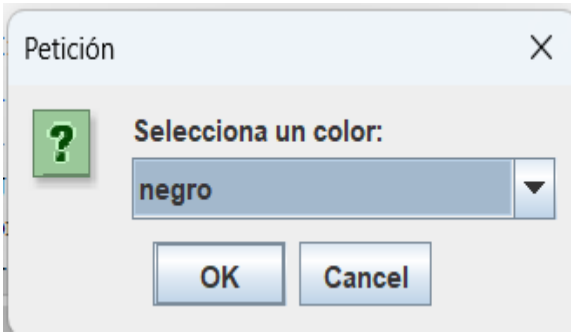
Detalles del Código

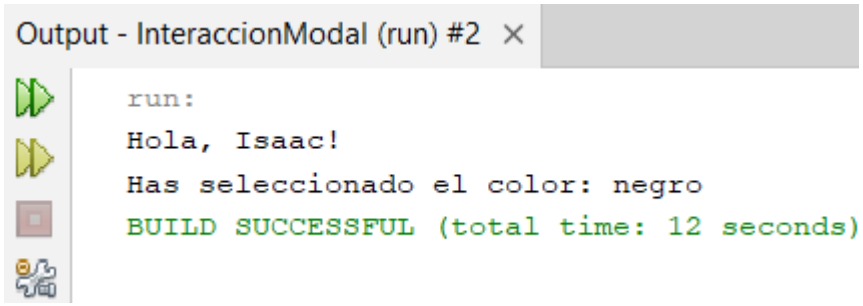
- **Clase Principal**: InteraccionModal
- **Método main**: Se crean dos diálogos modales:
    - **Diálogo de entrada de texto**: JOptionPane.showInputDialog para solicitar el nombre del usuario.
    - **Diálogo de selección de opciones**: JOptionPane.showInputDialog para ofrecer una lista de colores para elegir.

Resultado

Se muestran dos diálogos modales. El primero solicita el nombre del usuario y el segundo ofrece una lista de colores. Los resultados se imprimen en la consola.







Conclusión

Este programa muestra cómo usar JOptionPane para crear diálogos interactivos en Java de una manera sencilla y efectiva.

\

Reporte del Programa: DialogoInteractivo.java

Descripción General

El programa DialogoInteractivo.java muestra un diálogo interactivo donde el usuario puede seleccionar entre varias opciones.

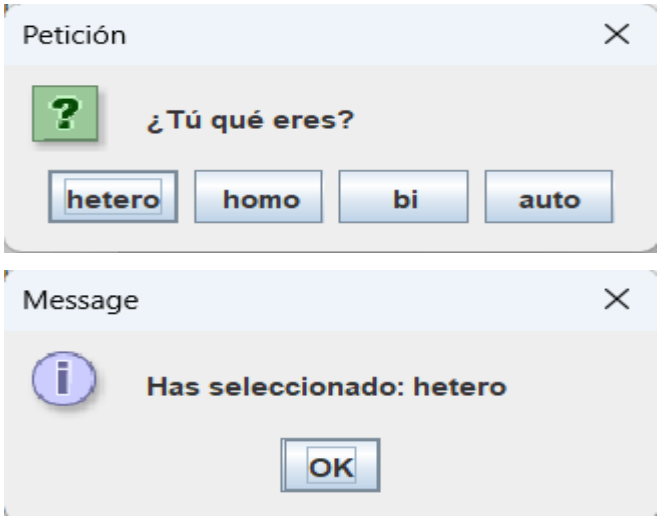
Detalles del Código

- **Clase Principal**: DialogoInteractivo
- **Método main**: Define un arreglo de cadenas sexo con opciones para el diálogo y utiliza JOptionPane para mostrar el diálogo y capturar la selección del usuario.

Funcionamiento

El diálogo pregunta "¿Tú qué eres?" con opciones como "hetero", "homo", "bi" y "auto". Dependiendo de la selección, se muestra un mensaje con la opción seleccionada.

Resultado



Conclusión Este programa es un ejemplo práctico de cómo crear diálogos interactivos personalizados en Java.

Reporte del Programa: SelectorDeArchivos.java

Descripción General El programa SelectorDeArchivos.java muestra un diálogo para seleccionar archivos, con un filtro personalizado para archivos .txt.

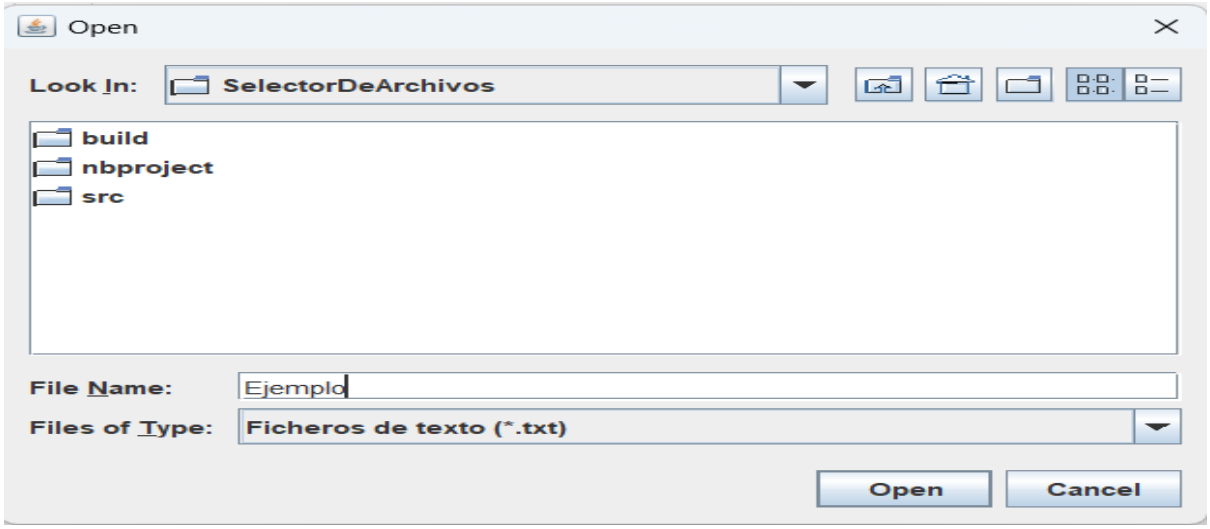
Detalles del Código

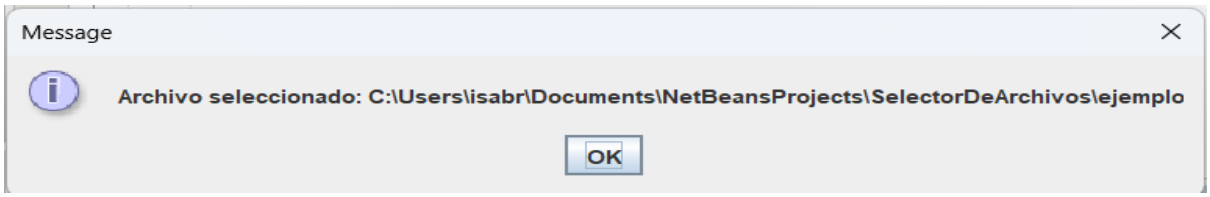
- **Clase Principal**: SelectorDeArchivos
- **Método main**: Crea un JFileChooser con un filtro personalizado para archivos de texto y muestra el diálogo para seleccionar un archivo.

Funcionamiento

Muestra un diálogo para seleccionar archivos filtrados por la extensión .txt. Si el usuario selecciona un archivo, muestra la ruta completa; si cancela, indica que no se seleccionó ningún archivo.

Resultado





Conclusión

Este programa es ideal para aprender a utilizar JFileChooser y aplicar filtros personalizados en diálogos de selección de archivos en Java.

Reporte del Programa: FileDialogExample.java

Descripción General

El programa FileDialogExample.java utiliza FileDialog para abrir y guardar archivos.

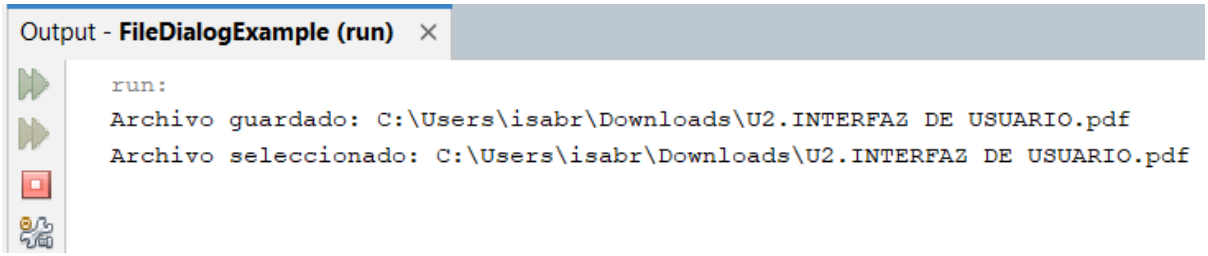
Detalles del Código

- **Clase Principal**: FileDialogExample
- **Método main**: Crea una ventana con dos botones: "Abrir Archivo" y "Guardar Archivo". Cada botón muestra un diálogo correspondiente (FileDialog).

Funcionamiento

Muestra diálogos para abrir y guardar archivos. La ruta del archivo seleccionado se imprime en la consola, o se indica si la operación fue cancelada.

Resultado



Conclusión

Este programa es un buen punto de partida para entender cómo trabajar con diálogos de archivo en Java utilizando FileDialog.

Reporte del Programa: SeleccionArchivos.java

Descripción General

El programa permite al usuario seleccionar un archivo desde un diálogo de selección de archivos.

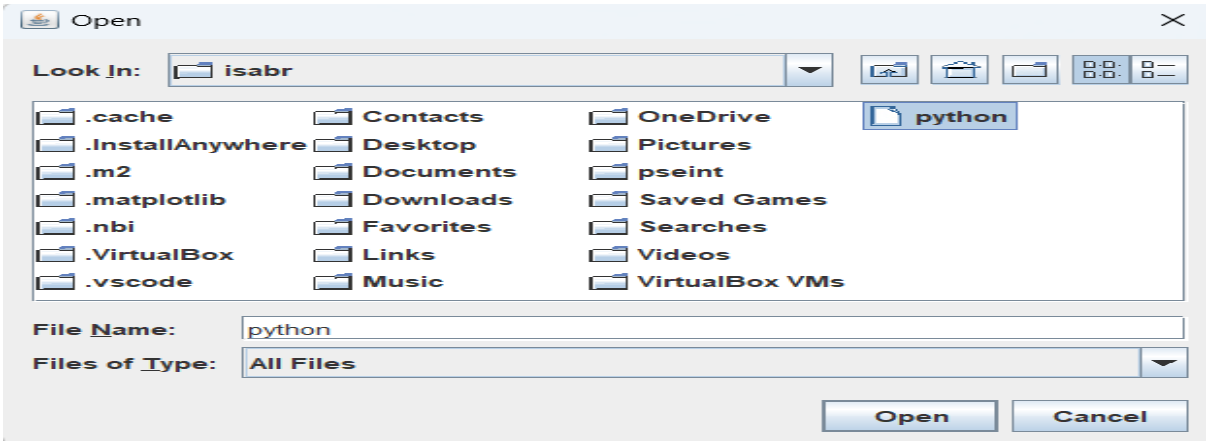
Detalles del Código

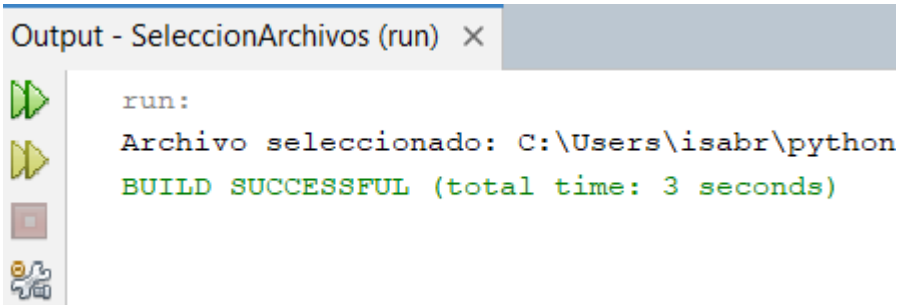
- **Clase Principal**: SeleccionArchivos
- **Método main**: Crea un JFileChooser y muestra el diálogo para seleccionar un archivo.

Funcionamiento

Muestra un diálogo de selección de archivos. La ruta del archivo seleccionado se imprime en la consola, o se indica si la operación fue cancelada.

Resultado





Conclusión

Este programa es útil para aprender a implementar diálogos de selección de archivos en Java.

Reporte del Programa: MiCanvas.java

Descripción General

El programa MiCanvas utiliza la clase Graphics para dibujar en un canvas.

Detalles del Código

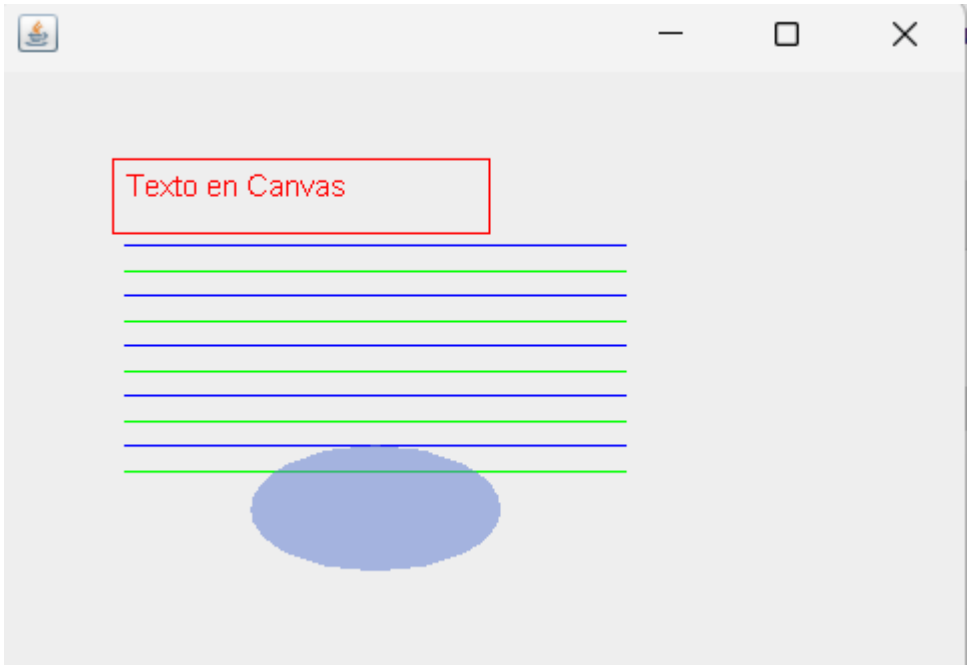
- **Clase Principal**: MiCanvas
- **Método paint(Graphics g)**: Sobrescribe este método para realizar operaciones de dibujo.

Operaciones de Dibujo

- **Texto**: Dibuja una cadena en las coordenadas (50, 50).
- **Rectángulo**: Dibuja un rectángulo alrededor del texto.
- **Líneas**: Dibuja líneas con colores alternos.
- **Óvalo**: Dibuja y rellena un óvalo con transparencia.

Resultado

Ejecutar el método main crea una ventana con los gráficos dibujados en el canvas.



Conclusión

Este programa es perfecto para aprender a dibujar en un canvas utilizando la clase Graphics en Java.

Reporte del Programa: EjemploGraphics.java

Descripción General

El programa EjemploGraphics utiliza Graphics para dibujar varios elementos en un canvas.

Detalles del Código

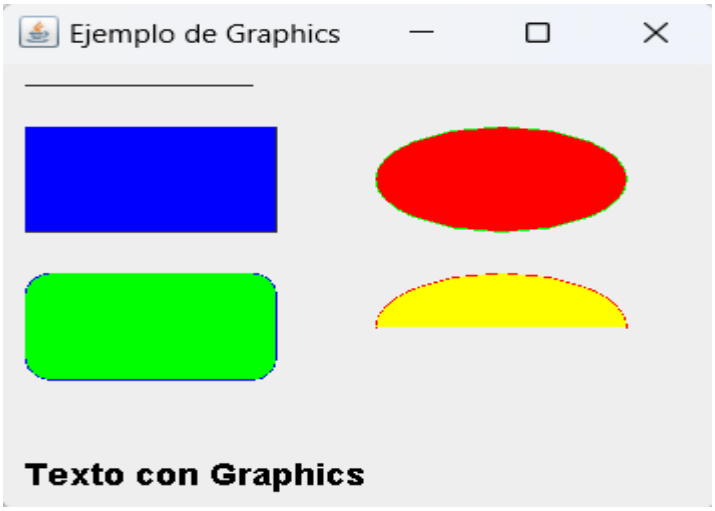
- **Clase Principal**: EjemploGraphics
- **Método paintComponent(Graphics g)**: Sobrescribe este método para realizar varias operaciones de dibujo.

Operaciones de Dibujo

- **Línea**: Dibuja una línea desde (10, 10) hasta (100, 10).
- **Rectángulos**: Dibuja y rellena rectángulos (con y sin bordes redondeados).
- **Óvalo**: Dibuja y rellena un óvalo.
- **Arco**: Dibuja y rellena un arco.
- **Texto**: Dibuja texto con una fuente específica.

Resultado

Ejecutar el método main crea una ventana con los gráficos dibujados en el canvas.



Conclusión

Este programa es excelente para aprender a dibujar diferentes formas y textos en un canvas utilizando Graphics en Java.

Reporte del Programa: Lampara.java

Descripción General

El programa Lampara dibuja una lámpara en un canvas.

Detalles del Código

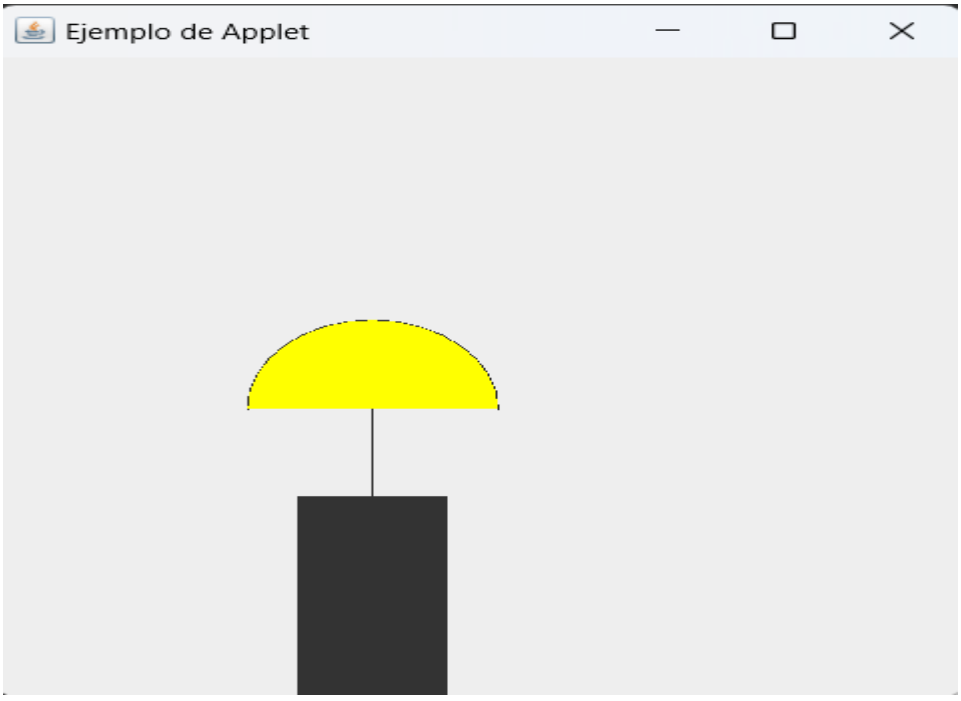
- **Clase Principal**: Lampara
- **Método paint(Graphics g)**: Sobrescribe este método para dibujar una lámpara.

Operaciones de Dibujo

- **Base de la Lámpara**: Dibuja un rectángulo en (120, 250).
- **Soporte de la Lámpara**: Dibuja una línea desde (150, 250) hasta (150, 200).
- **Pantalla de la Lámpara**: Dibuja y rellena un arco en (100, 150).

Resultado

Ejecutar el método main crea una ventana con la lámpara dibujada en el canvas.



Conclusión

Este programa es una forma divertida de aprender a combinar diferentes operaciones de dibujo para crear un objeto completo en Java.