ESCUELA DE INGENIERÍA EN CIENCIAS Y SISTEMAS
FACULTAD DE INGENIERÍA
UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

ECYS | DIDT
DESARROLLO DE TRANSFERENCIA TECNOLÓGICA

Día, Fecha:	07/09/2023
Hora de inicio:	09:00

Introducción a la Programación y Computación 1 [Sección B]

Diego Fernando Cortez Lopez

Agenda

- Clase 7
- Dudas
- Ejemplo Práctico

Introducción a AWT y SWING

Clase 7

Interfaz gráfica de usuario

Las Interfaces Gráficas de Usuario (GUI, Graphical User Interface) son la parte visible y directamente interactiva de una aplicación, permitiendo a los usuarios comunicarse de manera eficiente con el software. En lugar de depender únicamente de comandos de texto o código de programación, las GUI proporcionan una experiencia más amigable y accesible para los usuarios finales.

Crear una GUI en Java implica diseñar y programar la interfaz visual y los elementos que permiten interactuar con ella. Los elementos gráficos se llaman componentes y se agrupan en contenedores que permiten organizarlos dentro de la ventana.

Importancia de GUI en aplicaciones de Software

- Las GUI hacen que las aplicaciones sean más fáciles de usar para los usuarios de todos los niveles de experiencia, permitiendo una interacción intuitiva a través de elementos visuales.
- La aplicaciones con interfaces gráficas atractivas y bien diseñadas a menudo atraen a más usuarios y generan una impresión positiva, influyendo en la percepción de calidad de la aplicación.
- Las GUI pueden mejorar significativamente la productividad al simplificar tareas complejas.
- Las GUI garantizan que una aplicación sea inclusiva y cumpla con estándares de accesibilidad.
- Las GUI pueden proporcionar retroalimentación instantánea a los usuarios, como confirmaciones visuales o mensajes de error, lo que mejora la experiencia del usuario al brindar información relevante en el momento adecuado.

Herramientas en JAVA

Java es conocido por proporcionar herramientas efectivas para crear interfaces gráficas en aplicaciones. Algunas son:

1. **Plataforma Independiente:** Las aplicaciones Java con GUI son altamente portátiles, ya que pueden ejecutarse en diferentes sistemas operativos sin modificaciones significativas.
2. **Librerías AWT y Swing:** Java ofrece dos conjuntos de librerías para crear interfaces Gráficas: AWT (Abstract Window Toolkit) y Swing.
3. **Event Handling:** Java proporciona un sistema de manejo de eventos que simplifica la gestión de interacciones de usuario.
4. **Herramientas de Diseño:** Java cuenta con herramientas y entornos de desarrollo integrados (IDE) que facilitan la recreación y el diseño de Interfaces Gráficas de manera eficiente

Librerías de interfaz gráfica

- Swing: Es una de las librerías de UI más antiguas y utilizadas en Java.
- JavaFX: Es la librería de UI recomendada por Oracle para el desarrollo de aplicaciones de escritorio en Java.
- AWT: Es una librería de UI más antigua que Swing, y se utiliza principalmente para desarrollar aplicaciones simples.
- SWT: Es una librería utilizada para desarrollar aplicaciones de alta velocidad.
- JGoodies: Se enfoca en mejorar la apariencia y la usabilidad de las aplicaciones de escritorio.

AWT (Abstract Widow Toolkit)

AWT es la librería de Interfaz Gráfica original en Java. Fue introducida en las primeras versiones de Java y está diseñada para aprovechar los componentes nativos del sistema operativo en el que se ejecuta la aplicación.

Ventajas y Desventajas de AWT

Ventajas

- **Acceso a Componentes Nativos:** AWT utiliza los componentes de interfaz gráfica nativos del sistema operativo, lo que puede dar como resultado una apariencia y comportamiento más coherente con la plataforma en la que se ejecuta.
- **Rendimiento:** Dado que AWT se integra directamente con los componentes nativos, puede ofrecer un rendimiento más rápido en comparación con otras librerías.
- **Sencillez:** AWT tiene una API más sencilla y más limitada en comparación con Swing, lo que puede ser beneficioso para aplicaciones simples y de bajo consumo de recursos.

Ventajas y Desventajas de AWT

Desventajas

- **Limitaciones en la Personalización:** AWT ofrece menos opciones de personalización y estilo en comparación con otras librerias. Esto puede hacer que sea difícil crear interfaces altamente personalizadas o modernas
- **Portabilidad Limitada:** Debido a su fuerte dependencias de componentes nativos, las aplicaciones AWT pueden no ser tan portables, ya que pueden verse afectadas por diferencias en la apariencia y el comportamiento en diferentes sistemas operativos.

Cuando utilizar AWT

AWT puede ser más apropiado en situaciones donde se requiere un rendimiento excepcional y la apariencia nativa del sistema operativo es fundamental.

Por ejemplo, aplicaciones que utilizan gráficos intensivos o que pueden integrarse de manera perfecta en la plataforma host pueden beneficiarse de AWT.

Swing

Swing es una libreria mas moderna y flexible para crear interfaces Gráficas en Java. Fue desarrollada como una extensión de AWT y está diseñada para ser independiente de la plataforma, lo que significa que las aplicaciones Swing se ven y se comportan de manera más consistente en diferentes sistemas operativos.

Ventajas y Desventajas de Swing

Ventajas

- **Independencia de Plataforma:** Swing ofrece una apariencia y comportamiento consistentes en todas las plataformas, lo que garantiza la portabilidad de las aplicaciones.
- **Personalización:** Swing permite una personalización extensa de la apariencia y el comportamiento de los componentes de la GUI. Esto es útil para crear interfaces altamente personalizadas y modernas.
- **Amplia Variedad de Componentes:** Swing ofrece una amplia gama de componentes gráficos, lo que facilita la creación de interfaces ricas y complejas.

Ventajas y Desventajas de Swing

Desventajas

- **Limitaciones en la Personalización:** Debido a su flexibilidad y personalización, Swing puede consumir más recursos en comparación con AWT, lo que puede afectar el rendimiento en sistemas con recursos limitados.

Cuando utilizar Swing

Swing es más apropiado cuando la portabilidad y la personalización son fundamentales. Es ideal para aplicaciones empresariales, aplicaciones de escritorio multiplataforma y proyectos que requieren interfaces de usuario modernas y atractivas.

Eventos

Los eventos se refiere a las acciones o sucesos que ocurren en una interfaz de usuario y que pueden ser detectados y gestionados por la aplicación. Estos eventos son esenciales para permitir que los usuarios interactúen con la aplicación de manera efectiva. Algunos de ventos en una interfaz gráfica incluyen hacer clic en un botón, presionar una tecla en el teclado, mover el mouse sobre un componente y seleccionar un elemento de una lista.

Los pasos para trabajar con eventos en Java son:

1. Crear el objeto que genera el evento.
2. Registrar el objeto con un listener que manejara el evento.
3. Implementar el listener para definir la acción que se tomará cuando se produzca el evento.

Gestión de Eventos en Java

La gestión de eventos en Interfaces Gráficas se realiza a través del uso de oyentes de eventos y manejadores de eventos.

- **Oyentes de Evento:** Los oyentes de eventos son objetivos que escuchan o vigilan ciertos tipos de eventos en un componente de la interfaz gráfica. Cada tipo de evento tiene su propio oyente correspondiente.
- **Registro de Oyentes:** Para que un componente pueda recibir eventos, se debe registrar un oyente apropiado en ese componente. Esto se hace utilizando métodos específicos proporcionados por la API de Java
- **Manejo de Eventos:** Cuando ocurre un evento en un componente, el oyente correspondiente detecta ese veneto y llama a un método específico del manejador de eventos asociado. El manejador de eventos es una clase que implementa la lógica para responder al evento en particular.

Tipos de Eventos Comunes

- **Clic de Botón:** Este evento ocurre cuando un usuario hace clic en un botón. Puede utilizarse para activar una acción específica, como guardar un archivo o enviar un formulario.
- **Teclas Presionadas:** Los eventos de teclas presionadas se producen cuando el usuario presiona una tecla en el teclado. Se pueden usar para capturar la entrada de texto o realizar acciones basadas en las teclas presionadas.
- **Selección de Elementos de Lista:** Cuando un usuario selecciona un elemento de una lista desplegable o una lista de selección múltiple, se genera un evento de selección. Este evento puede usarse para actualizar la interfaz en función de la selección del usuario.

Creación de Manejadores de Eventos

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

public class MiVentana extends Frame implements ActionListener {
    Button miBoton;

    public MiVentana() {
        miBoton = new Button("Haz clic");
        miBoton.addActionListener(this);
        add(miBoton);
    }

    public void actionPerformed(ActionEvent e) {
        if (e.getSource() == miBoton) {
            System.out.println("Se hizo clic en el botón");
        }
    }

    public static void main(String[] args) {
        MiVentana ventana = new MiVentana();
        ventana.setSize(200, 200);
        ventana.setVisible(true);
    }
}
```

Tarea 2

Describir con sus palabras cada pilar de POO y dar un ejemplo de herencia y polimorfismo.

- Entrega en formato PDF
- Fecha de entrega: 10/09/2023 23:59

¿Dudas?

Recursos

- Librerias JFreeChart: https://sourceforge.net/projects/jfreechart/
- Documentacion JFreeChart: https://www.jfree.org/jfreechart/api/javadoc/index.html