
Politécnico
Internacional

Politécnico Internacional

Java

Programación I

Politécnico Internacional

Java

“Celebra cada pequeña victoria.”

AGENDA

- Conceptos Java
- BetPlay
- Conclusiones

AWT

AWT, significa Abstract Window Toolkit, es una biblioteca de Java utilizada para crear interfaces de usuario gráficas (GUI, por sus siglas en inglés). AWT proporciona una serie de clases y métodos que permiten a los programadores crear ventanas, cuadros de diálogo, botones, campos de texto y otros elementos de interfaz gráfica de usuario en aplicaciones Java.

AWT

AWT es parte del núcleo de Java y proporciona una forma de interactuar con el sistema operativo subyacente para crear elementos de la interfaz de usuario que se integran con el entorno de ejecución nativo. Esto significa que las aplicaciones AWT pueden verse y comportarse de manera similar a las aplicaciones nativas de la plataforma en la que se ejecutan.

AWT - Componentes

Frame: El marco es una ventana principal de una aplicación AWT. Puede contener otros componentes, como botones, campos de texto, paneles, etc.

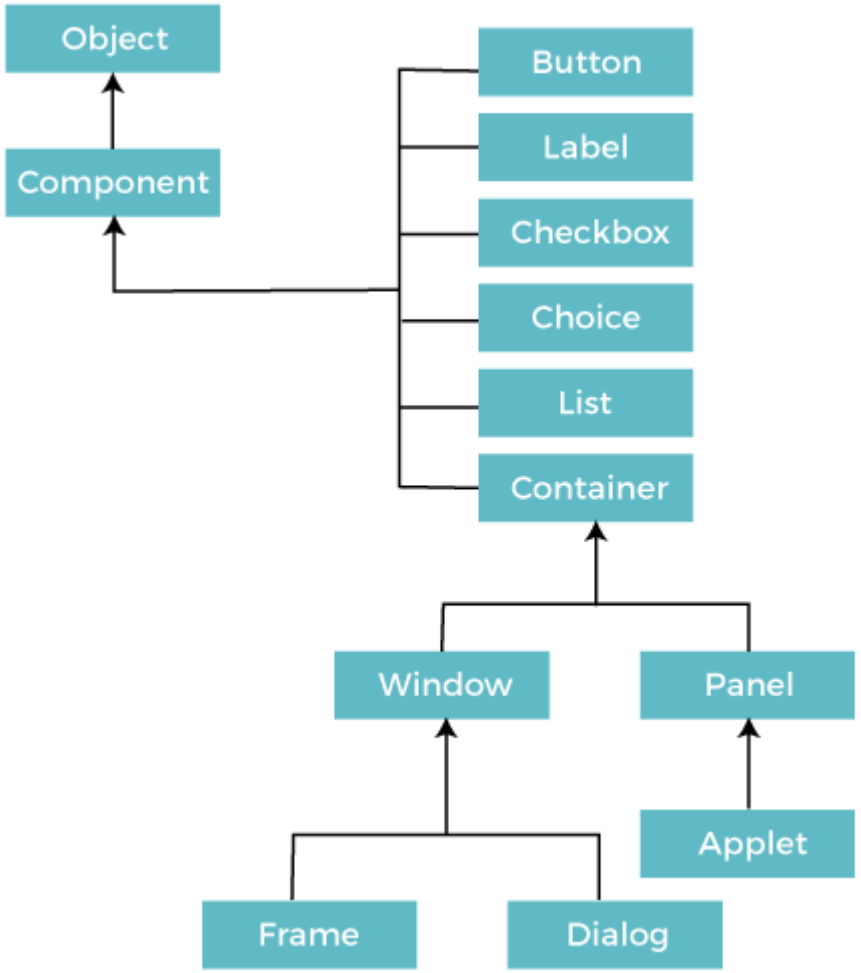
Button: Los botones son componentes que permiten a los usuarios realizar acciones cuando se hace clic en ellos.

TextField: Los campos de texto permiten al usuario ingresar y editar texto.

Label: Las etiquetas se utilizan para mostrar texto o información descriptiva en la interfaz de usuario.

Panel: Los paneles son contenedores que se utilizan para agrupar otros componentes.

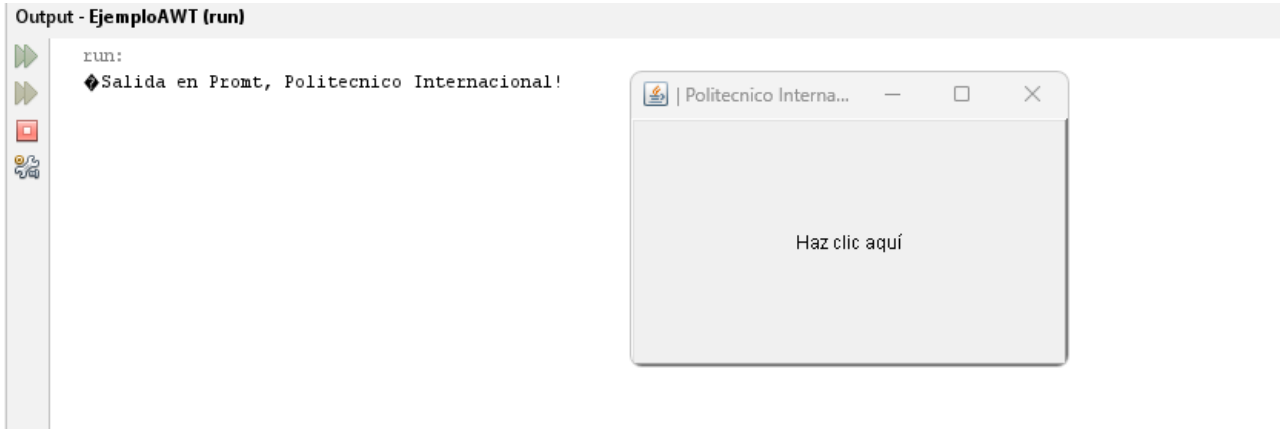
AWT - Componentes



AWT - Componentes

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

public class EjemploAWT {

    public static void main(String[] args) {
        // Crear una instancia de Frame (ventana)
        Frame frame = new Frame(title: "| Politecnico Internacional | Ejemplo AWT");

        // Crear un botón con etiqueta
        Button button = new Button(label: "Haz clic aquí");

        // Crear un manejador de eventos para el botón
        button.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                // Este código se ejecuta cuando se hace clic en el botón
                System.out.println(x: "¡Salida en Promt, Politecnico Internacional!");
            }
        });

        // Agregar el botón al Frame
        frame.add(comp: button);

        // Configurar el cierre de la ventana
        frame.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent e) {
                // Este código se ejecuta cuando se cierra la ventana
                System.exit(status: 0); // Salir de la aplicación
            }
        });

        // Configurar el tamaño de la ventana
        frame.setSize(width: 300, height: 200);

        // Hacer visible la ventana
        frame.setVisible(b: true);
    }
}
```

AWT - Componentes



Java Event Handling

El manejo de eventos en Java se refiere a la forma en que una aplicación Java **responde a las interacciones del usuario o a otros eventos que ocurren durante la ejecución de un programa**. Estos eventos pueden ser acciones del usuario, como hacer clic en un botón, mover el mouse, escribir en un campo de texto, presionar una tecla del teclado, etc., o eventos generados por el sistema o por otros componentes de la aplicación.

Java Event Handling

El manejo de eventos es fundamental en las aplicaciones de interfaz de usuario, ya que permite que la aplicación reaccione de manera apropiada a las acciones del usuario y a los cambios en el entorno de ejecución.

Java Event Handling

Interfaces de usuario gráficas (GUI): En las aplicaciones de escritorio, como las aplicaciones Swing o JavaFX, se utilizan eventos para responder a acciones del usuario, como hacer clic en botones, arrastrar elementos, escribir en campos de texto, etc.

Manejo de teclado y ratón: El manejo de eventos permite detectar y responder a eventos de teclado y mouse, como presionar teclas, mover el puntero del mouse y hacer clic en elementos de la interfaz de usuario.

.

Java Event Handling

Comunicación entre componentes: Las aplicaciones a menudo tienen múltiples componentes que necesitan comunicarse entre sí. Los eventos proporcionan un mecanismo para que los componentes notifiquen y respondan a cambios o acciones de otros componentes.

Temporización: Los eventos también se utilizan para implementar temporizadores y programación de tareas en un momento específico.

Java Event Handling

El manejo de eventos en Java se logra mediante el uso de clases y mecanismos específicos, como escuchadores de eventos (event listeners) y clases de eventos. Los escuchadores de eventos son interfaces que definen métodos para manejar eventos específicos, mientras que las clases de eventos encapsulan la información sobre el evento en sí.

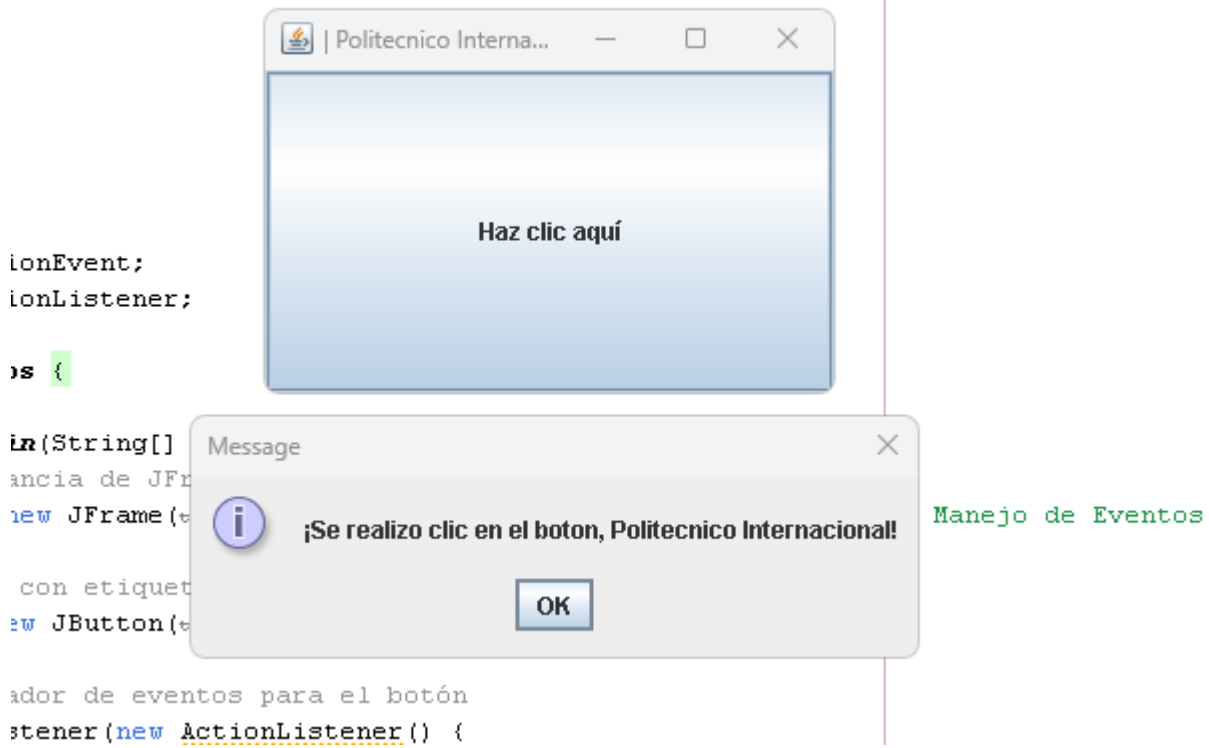
Java Event Handling

El manejo de eventos en Java es esencial para crear aplicaciones interactivas y receptivas. Permite que una aplicación responda a las acciones del usuario y a otros eventos, lo que hace que las aplicaciones sean más dinámicas y funcionales.

Java Event Handling

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

public class EjemploEventos {

    public static void main(String[] args) {
        // Crear una instancia de JFrame (ventana)
        JFrame ventana = new JFrame(title: "| Politecnico Internacional | Ejemplo de Manejo de Eventos |");

        // Crear un botón con etiqueta
        JButton boton = new JButton(text: "Haz clic aquí");

        // Crear un manejador de eventos para el botón
        boton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                // Este código se ejecuta cuando se hace clic en el botón
                JOptionPane.showMessageDialog(parentComponent: null, message: "¡Se realizo clic en el boton, Politecnico Internacional!");
            }
        });

        // Configurar el cierre de la ventana
        ventana.setDefaultCloseOperation(operation: JFrame.EXIT_ON_CLOSE);

        // Agregar el botón a la ventana
        ventana.add(comp: boton);

        // Configurar el tamaño de la ventana
        ventana.setSize(width: 300, height: 200);

        // Hacer visible la ventana
        ventana.setVisible(b: true);
    }
}
```

Java Event Handling



Java AWT Button

java.awt.Button es una clase proporcionada por la Biblioteca de Clases Abstractas de Java (AWT) que se utiliza para crear botones en aplicaciones de interfaz gráfica de usuario (GUI) en Java. Estos botones permiten a los usuarios interactuar con la aplicación haciendo clic en ellos. Cada botón puede tener una etiqueta de texto que describe su función, y se puede asociar un manejador de eventos **para responder a los clics de los usuarios.**

Java AWT Button

java.awt.Button se utiliza para crear botones que representan acciones o comandos que los usuarios pueden realizar en una aplicación Java con una interfaz gráfica. Los botones son componentes esenciales en la construcción de interfaces de usuario interactivas.

Java AWT Button

java.awt.Button se utiliza para crear botones que representan acciones o comandos que los usuarios pueden realizar en una aplicación Java con una interfaz gráfica. Los botones son componentes esenciales en la construcción de interfaces de usuario interactivas.

Java AWT Button

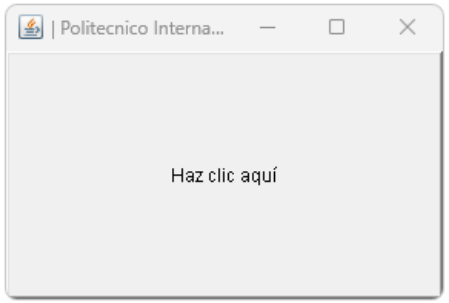
java.awt.Button sirve para permitir que los usuarios interactúen con una aplicación Java haciendo clic en botones. Cuando un usuario hace clic en un botón, se dispara un evento que puede ser manejado por el programa para realizar una acción específica. Por ejemplo, puedes usar un botón para guardar un archivo, enviar un formulario, cerrar una ventana o cualquier otra acción que sea relevante para tu aplicación

Java AWT Button

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

public class EjemploBotonAWT {

    public static void main(String[] args) {
        Frame frame = new Frame(title: "| Politecnico Internacional | Ejemplo de Botón AWT");
        Button boton = new Button(label: "Haz clic aquí");

        // Agregar un manejador de eventos al botón
        boton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                System.out.println(x: "¡Se realizo clic en el boton, Politecnico Internacional! ");
            }
        });

        frame.add(comp: boton);
        frame.setSize(width: 300, height: 100);
        frame.setVisible(b: true);
    }
}
```

Java AWT Button

```
Output - EjemploAWT (run)
run:
Salida en Promt, Politecnico Internacional!
```



Java AWT Label

java.awt.Label es una clase proporcionada por la Biblioteca de Clases Abstractas de Java (AWT) que se utiliza para crear etiquetas de texto en aplicaciones de interfaz gráfica de usuario (GUI) en Java. Estas etiquetas se utilizan principalmente para mostrar información descriptiva o texto estático en una interfaz de usuario. No son interactivas, es decir, los usuarios no pueden interactuar directamente con ellas.

Java AWT Label

java.awt.Label se utiliza para mostrar texto o información descriptiva en una interfaz gráfica de usuario en Java. Puede utilizarse para etiquetar otros componentes, como campos de texto, botones u otros elementos de la GUI, para proporcionar información sobre su función o contenido.

Java AWT Label

java.awt.Label sirve para proporcionar etiquetas descriptivas en una interfaz de usuario. Estas etiquetas ayudan a los usuarios a comprender la función de los elementos cercanos o a proporcionar información adicional. A menudo se usan en formularios, paneles u otras ventanas para mejorar la usabilidad y la comprensión de la interfaz de usuario.

Java AWT Label

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

public class EjemploLabelAWT {

    public static void main(String[] args) {
        Frame frame = new Frame(title: "Ejemplo de Label AWT");
        Label etiqueta = new Label(text: "Esta es una etiqueta de texto.");

        frame.add(comp: etiqueta);
        frame.setSize(width: 300, height: 100);
        frame.setVisible(b: true);

          // Centrar Pantalla
        frame.setLocationRelativeTo(c: null);


        // Agregar un manejador de eventos para cerrar la ventana
        frame.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent e) {
                System.exit(status: 0); // Salir de la aplicación cuando se cierre la ventana
            }
        });
    }
}
```

Java AWT Label

Java AWT TextField

Java AWT TextField es un componente gráfico proporcionado por la biblioteca AWT (Abstract Window Toolkit) de Java. Este componente se utiliza para permitir a los usuarios ingresar texto de una sola línea en una aplicación de interfaz gráfica de usuario (GUI

Java AWT TextField

Entrada de texto: Permite a los usuarios ingresar texto, como nombres de usuario, contraseñas, números u otra información de texto en una ventana de la aplicación Java.

Recopilación de datos: Facilita la recopilación de datos de entrada del usuario, que luego se pueden utilizar en la lógica de la aplicación para realizar diversas acciones, como cálculos, consultas a bases de datos o manipulación de datos.

Java AWT TextField

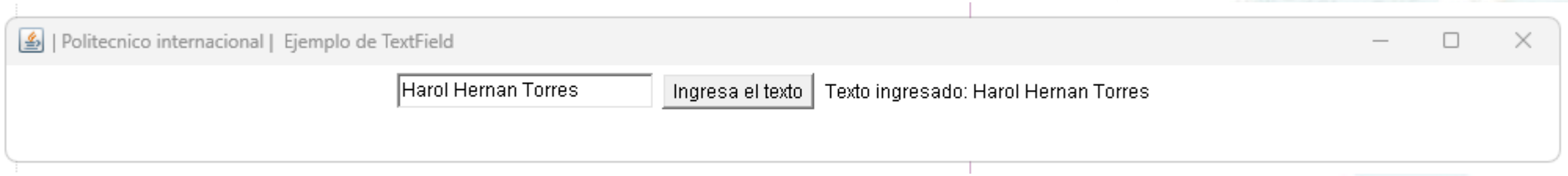
Interacción con el usuario: Proporciona una forma interactiva para que los usuarios ingresen datos en una aplicación, lo que mejora la experiencia del usuario y hace que la aplicación sea más amigable.

Captura de eventos: Puede asociarse con escuchadores de eventos para capturar y responder a eventos de usuario, como cuando el usuario presiona la tecla "Enter" o hace clic en otro componente después de ingresar texto.

Java AWT TextField

```java
public class EjemploTextField {
    public static void main(String[] args) {
        // Crear una ventana
        Frame ventana = new Frame(title: "| Politecnico internacional |  Ejemplo de TextField");

        // Crear un TextField
        TextField textField = new TextField(text: "Texto por defecto", columns:20);

        // Crear un botón para obtener el texto del TextField
        Button boton = new Button(label: "Ingresa el texto");

        // Crear un Label para mostrar el resultado
        Label resultadoLabel = new Label(text: "");

        // Configurar el administrador de diseño de la ventana
        ventana.setLayout(new FlowLayout());

        // Agregar el TextField, el botón y el Label a la ventana
        ventana.add(comp: textField);
        ventana.add(comp: boton);
        ventana.add(comp: resultadoLabel);

        // Configurar un ActionListener para el botón
        boton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                // Obtener el texto del TextField y mostrarlo en el Label
                String texto = textField.getText();
                resultadoLabel.setText("Texto ingresado: " + texto);
            }
        });

        // Configurar el cierre de la ventana
        ventana.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent e) {
                System.exit(status: 0);
            }
        });

        // Configurar el tamaño de la ventana
        ventana.setSize(width: 300, height: 100);

        // Centrar la ventana en la pantalla
        ventana.setLocationRelativeTo(c: null);

        // Hacer visible la ventana
        ventana.setVisible(b: true);
    }
}
```

Java AWT TextField



Java AWT TextArea

Java AWT TextArea es otro componente gráfico proporcionado por la biblioteca AWT (Abstract Window Toolkit) de Java. A diferencia de TextField, que permite la entrada de texto de una sola línea, TextArea se utiliza para permitir a los usuarios ingresar y mostrar texto de múltiples líneas en una aplicación de interfaz gráfica de usuario (GUI)

Java AWT TextArea

Entrada y visualización de texto multilineal: Permite a los usuarios ingresar texto en varios párrafos o líneas de texto, lo que es útil para capturar texto más extenso, como comentarios, descripciones o notas.

Edición de texto: Proporciona una interfaz de edición de texto enriquecido que incluye capacidades de copiar, cortar, pegar, deshacer y rehacer, lo que permite a los usuarios interactuar y modificar el texto de manera eficiente.

Java AWT TextArea

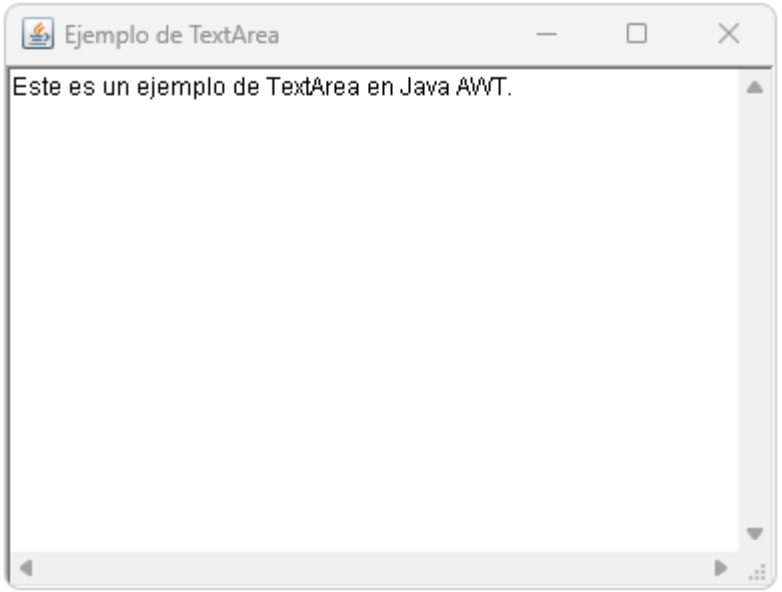
Visualización de datos: Puede usarse para mostrar grandes cantidades de texto que no cabrían en un solo TextField, lo que es especialmente útil cuando se necesita mostrar información detallada o registros extensos.

Captura de eventos: Al igual que con otros componentes AWT, TextArea se puede configurar para capturar eventos, como cuando el usuario presiona una tecla o realiza una acción específica en el área de texto.

Java AWT TextArea

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

public class EjemploTextArea {
    public static void main(String[] args) {
        // Crear una ventana
        Frame ventana = new Frame(title: "Ejemplo de TextArea");

        // Crear un TextArea con un texto predeterminado
        TextArea areaTexto = new TextArea(text: "Este es un ejemplo de TextArea en Java AWT.", rows: 10, columns: 30);

        // Configurar el administrador de diseño de la ventana
        ventana.setLayout(new BorderLayout());

        // Agregar el TextArea al centro de la ventana
        ventana.add(comp: areaTexto, constraints: BorderLayout.CENTER);

        // Configurar el cierre de la ventana
        ventana.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent e) {
                System.exit(status: 0);
            }
        });

        // Configurar el tamaño de la ventana
        ventana.setSize(width: 400, height: 300);

        // Centrar la ventana en la pantalla
        Dimension pantalla = Toolkit.getDefaultToolkit().getScreenSize();
        int x = (pantalla.width - ventana.getWidth()) / 2;
        int y = (pantalla.height - ventana.getHeight()) / 2;
        ventana.setLocation(x, y);

        // Hacer visible la ventana
        ventana.setVisible(b: true);
    }
}
```

Java AWT TextArea



Java AWT CheckBox

Java AWT Checkbox es un componente gráfico proporcionado por la biblioteca AWT (Abstract Window Toolkit) de Java. Este componente se utiliza para permitir a los usuarios seleccionar una o varias opciones de un conjunto de opciones disponibles.

Java AWT CheckBox

Selección de opciones: Proporciona una forma de presentar múltiples opciones al usuario en forma de casillas de verificación (checkboxes), donde el usuario puede marcar o desmarcar una o varias de ellas según su elección.

Selección múltiple: A diferencia de un componente como RadioButton, que permite al usuario seleccionar una sola opción de un grupo, Checkbox permite al usuario seleccionar múltiples opciones independientes entre sí.

Java AWT CheckBox

Configuración de preferencias o ajustes: Se utiliza comúnmente en aplicaciones de configuración o preferencias de usuario, donde el usuario puede habilitar o deshabilitar características específicas o elegir entre diferentes opciones de configuración.

Interacción con el usuario: Mejora la interacción del usuario al proporcionar una forma visualmente clara de indicar las preferencias o selecciones, lo que facilita la comunicación de la elección del usuario a la aplicación.

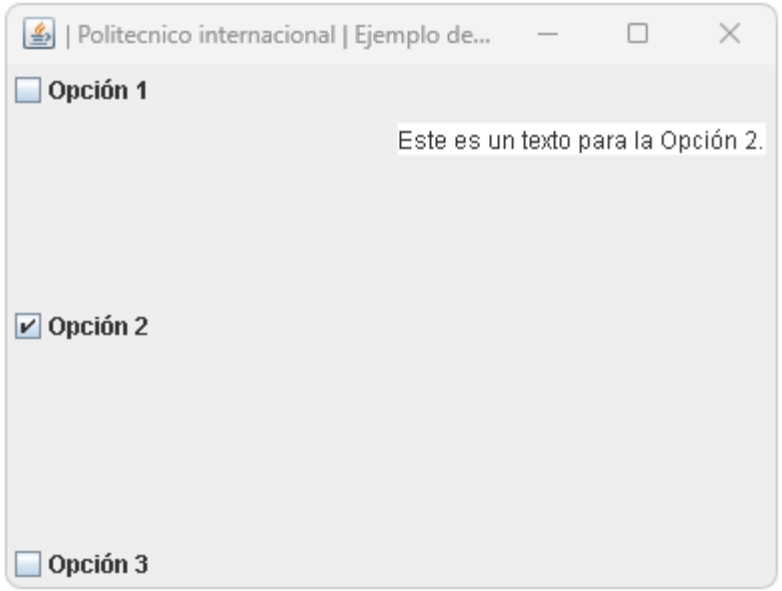
Java AWT CheckBox

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

public class EjemploCheckbox {
    public static void main(String[] args) {
        // Crear una ventana
        JFrame ventana = new JFrame(title: "| Politecnico internacional | Ejemplo de Checkbox");

        // Crear Checkbox para diferentes opciones
        JCheckBox checkBox1 = new JCheckBox(text: "Opción 1");
        JCheckBox checkBox2 = new JCheckBox(text: "Opción 2");
        JCheckBox checkBox3 = new JCheckBox(text: "Opción 3");

        // Crear un Panel para mostrar el texto de la "Opción 2"
        JPanel panelTexto = new JPanel();
        JTextArea textoArea = new JTextArea(text: "Este es un texto para la Opción 2.");
        panelTexto.add(comp: textoArea);

        // Configurar el cierre de la ventana
        ventana.setDefaultCloseOperation(operation: JFrame.EXIT_ON_CLOSE);

        // Configurar el tamaño de la ventana
        ventana.setSize(width: 400, height: 300);

        // Centrar la ventana en la pantalla
        Dimension pantalla = Toolkit.getDefaultToolkit().getScreenSize();
        int x = (pantalla.width - ventana.getWidth()) / 2;
        int y = (pantalla.height - ventana.getHeight()) / 2;
        ventana.setLocation(x, y);

        // Agregar ActionListener para los checkboxes
        checkBox1.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                if (checkBox1.isSelected()) {
                    // Aquí puedes agregar el código para mostrar la tabla
                }
            }
        });
```

Java AWT CheckBox

```
        checkBox2.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                if (checkBox2.isSelected()) {
                    // Mostrar el panel de texto para la "Opción 2"
                    ventana.getContentPane().removeAll();
                    ventana.getContentPane().add(comp: checkBox1, constraints: BorderLayout.NORTH);
                    ventana.getContentPane().add(comp: checkBox2, constraints: BorderLayout.CENTER);
                    ventana.getContentPane().add(comp: checkBox3, constraints: BorderLayout.SOUTH);
                    ventana.getContentPane().add(comp: panelTexto, constraints: BorderLayout.EAST);
                    ventana.revalidate();
                    ventana.repaint();
                }
            }
        });

        checkBox3.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                if (checkBox3.isSelected()) {
                    // Aquí puedes agregar el código para mostrar el botón de créditos
                }
            }
        });

        // Agregar los checkboxes a la ventana
        ventana.add(comp: checkBox1, constraints: BorderLayout.NORTH);
        ventana.add(comp: checkBox2, constraints: BorderLayout.CENTER);
        ventana.add(comp: checkBox3, constraints: BorderLayout.SOUTH);

        // Hacer visible la ventana
        ventana.setVisible(b: true);
    }
}
```

Java AWT CheckBox



Java AWT CheckBoxGroup

Java AWT CheckboxGroup es una clase proporcionada por la biblioteca AWT (Abstract Window Toolkit) de Java que se utiliza para agrupar varios Checkbox en una ventana de tal manera que solo uno de ellos pueda estar seleccionado en un momento dado

Java AWT CheckBoxGroup

Agrupar Checkbox: Permite agrupar varios Checkbox relacionados en una ventana de manera que, al seleccionar uno de ellos, se deseleccione automáticamente cualquier otro Checkbox en el mismo grupo. Esto garantiza que solo un elemento del grupo esté seleccionado en un momento dado.

Selección exclusiva: Proporciona una forma sencilla de implementar la selección exclusiva en conjuntos de opciones relacionadas. Por ejemplo, en un grupo de opciones de género (masculino, femenino, otro), solo se puede seleccionar una opción a la vez.

Simplificar el código: Al utilizar CheckboxGroup, se reduce la necesidad de escribir código adicional para gestionar la exclusión mutua entre los Checkbox del mismo grupo.

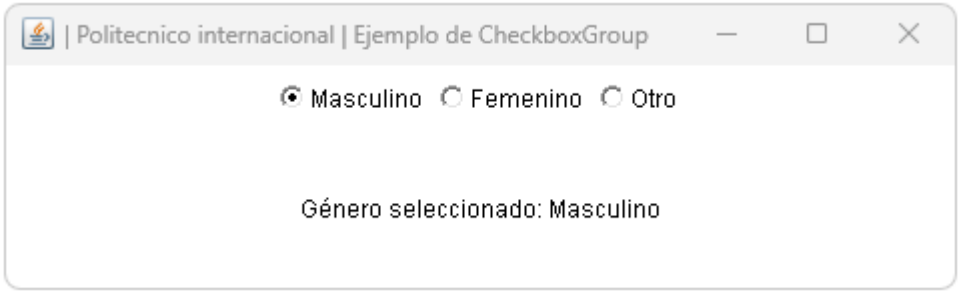
Java AWT CheckBoxGroup

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

public class EjemploCheckboxGroup {
    public static void main(String[] args) {
        // Crear una ventana
        Frame ventana = new Frame(title: "| Politecnico internacional | Ejemplo de CheckboxGroup");

        // Crear un grupo de checkboxes exclusivos
        CheckboxGroup grupoGenero = new CheckboxGroup();

        // Crear checkboxes relacionados en el mismo grupo
        Checkbox masculinoCheckbox = new Checkbox(label: "Masculino", group: grupoGenero, state: false);
        Checkbox femeninoCheckbox = new Checkbox(label: "Femenino", group: grupoGenero, state: false);
        Checkbox otroCheckbox = new Checkbox(label: "Otro", group: grupoGenero, state: false);

        // Crear un Label para mostrar la información
        Label informacionLabel = new Label(text: "", alignment: Label.CENTER);

        // Configurar el administrador de diseño de la ventana
        ventana.setLayout(new BorderLayout());

        // Agregar los checkboxes y el Label a la ventana
        Panel panelCheckboxes = new Panel();
        panelCheckboxes.setLayout(new FlowLayout());
        panelCheckboxes.add(comp: masculinoCheckbox);
        panelCheckboxes.add(comp: femeninoCheckbox);
        panelCheckboxes.add(comp: otroCheckbox);
        ventana.add(comp: panelCheckboxes, constraints: BorderLayout.NORTH);
        ventana.add(comp: informacionLabel, constraints: BorderLayout.CENTER);

        // Configurar el cierre de la ventana
        ventana.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent e) {
                System.exit(status: 0);
            }
        });
```

Java AWT CheckBoxGroup

```
        // Configurar el ItemListener para los checkboxes
        ItemListener checkBoxListener = new ItemListener() {
            public void itemStateChanged(ItemEvent e) {
                Checkbox checkboxSeleccionado = grupoGenero.getSelectedCheckbox();
                if (checkboxSeleccionado != null) {
                    String generoSeleccionado = checkboxSeleccionado.getLabel();
                    // Actualizar el Label con la información correspondiente al género seleccionado
                    informacionLabel.setText("Género seleccionado: " + generoSeleccionado);
                }
            }
        };

        masculinoCheckbox.addItemListener(1: checkBoxListener);
        femeninoCheckbox.addItemListener(1: checkBoxListener);
        otroCheckbox.addItemListener(1: checkBoxListener);

        // Configurar el tamaño de la ventana
        ventana.setSize(width: 300, height: 150);

        // Centrar la ventana en la pantalla
        Dimension pantalla = Toolkit.getDefaultToolkit().getScreenSize();
        int x = (pantalla.width - ventana.getWidth()) / 2;
        int y = (pantalla.height - ventana.getHeight()) / 2;
        ventana.setLocation(x, y);

        // Hacer visible la ventana
        ventana.setVisible(b: true);
    }
}
```

Java AWT CheckBoxGroup



Java AWT Choice

Java AWT Choice es un componente gráfico proporcionado por la biblioteca AWT (Abstract Window Toolkit) de Java. Este componente se utiliza para crear una lista desplegable de opciones que permite al usuario seleccionar una única opción de un conjunto predefinido.

Java AWT Choice

Selección de una opción: Permite al usuario seleccionar una opción de una lista de elementos desplegados. Solo se puede seleccionar una opción a la vez.

Menús desplegables: Se utiliza comúnmente para crear menús desplegables en aplicaciones de interfaz gráfica de usuario (GUI), donde el usuario puede seleccionar una opción de una lista predefinida.

Conservar espacio en la interfaz de usuario: Es útil cuando se desea proporcionar un conjunto de opciones, pero se quiere ahorrar espacio en la ventana de la aplicación hasta que el usuario interactúe con él.

Interfaz de usuario amigable: Ayuda a crear interfaces de usuario más amigables al agrupar opciones relacionadas en una lista desplegable, lo que facilita la elección para el usuario.

Java AWT Choice

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

public class EjemploChoice {
    public static void main(String[] args) {
        // Crear una ventana
        Frame ventana = new Frame(title: "| Politecnico internacional | Ejemplo de Choice");

        // Crear un componente Choice
        Choice opcionesChoice = new Choice();

        // Agregar opciones al Choice
        opcionesChoice.add(item: "Opción 1");
        opcionesChoice.add(item: "Opción 2");
        opcionesChoice.add(item: "Opción 3");
        opcionesChoice.add(item: "Opción 4");

        // Crear un Label para mostrar la selección
        Label resultadoLabel = new Label(text: "Selección: ");

        // Configurar el administrador de diseño de la ventana
        ventana.setLayout(new FlowLayout());

        // Agregar el Choice y el Label a la ventana
        ventana.add(comp: opcionesChoice);
        ventana.add(comp: resultadoLabel);

        // Configurar el cierre de la ventana
        ventana.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent e) {
                System.exit(status: 0);
            }
        });

        // Configurar ActionListener para el Choice
        opcionesChoice.addItemListener(new ItemListener() {
            public void itemStateChanged(ItemEvent e) {
                // Obtener la opción seleccionada y mostrarla en el Label
                String seleccion = opcionesChoice.getSelectedItem();
                resultadoLabel.setText("Selección: " + seleccion);
            }
        });
```

Java AWT Choice

```
        // Configurar el tamaño de la ventana
        ventana.setSize(width: 300, height: 100);

        // Centrar la ventana en la pantalla
        Dimension pantalla = Toolkit.getDefaultToolkit().getScreenSize();
        int x = (pantalla.width - ventana.getWidth()) / 2;
        int y = (pantalla.height - ventana.getHeight()) / 2;
        ventana.setLocation(x, y);

        // Hacer visible la ventana
        ventana.setVisible(b: true);
    }
}
```

Java AWT Choice

Java AWT List

Java AWT List es un componente gráfico proporcionado por la biblioteca AWT (Abstract Window Toolkit) de Java. Este componente se utiliza para crear listas de elementos, donde el usuario puede seleccionar uno o varios elementos de una lista predefinida

Java AWT List

Selección de elementos: Permite al usuario seleccionar uno o varios elementos de una lista predefinida de opciones. A diferencia de Choice, que permite seleccionar una sola opción, List permite selección múltiple.

Listas de opciones: Se utiliza comúnmente para crear listas de opciones en aplicaciones de interfaz gráfica de usuario (GUI), donde el usuario puede seleccionar una o varias opciones de una lista**.**

Gestión de selecciones múltiples: Es especialmente útil cuando se necesita permitir al usuario seleccionar varios elementos de una lista, como en la selección de elementos en una lista de reproducción de música o la selección de varios archivos en un explorador de archivos.

Interfaz de usuario versátil: Puede adaptarse a una variedad de situaciones en las que se requiere una selección de elementos de una lista predefinida en una interfaz gráfica de usuario.

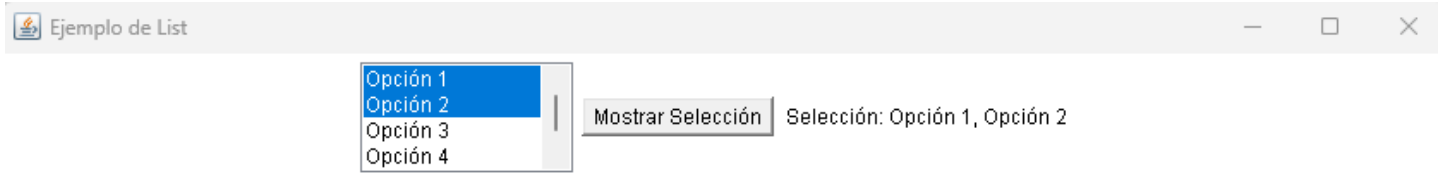
Java AWT List

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

public class EjemploList {
    public static void main(String[] args) {
        // Crear una ventana
        Frame ventana = new Frame(title: "Ejemplo de List");

        // Crear un componente List
        List opcionesList = new List(rows: 4, multipleMode: true); // El segundo parámetro permite selección múltiple

        // Agregar elementos a la lista
        opcionesList.add(item: "Opción 1");
        opcionesList.add(item: "Opción 2");
        opcionesList.add(item: "Opción 3");
        opcionesList.add(item: "Opción 4");
        opcionesList.add(item: "Opción 5");

        // Crear un botón para mostrar selecciones
        Button mostrarSeleccionButton = new Button(label: "Mostrar Selección");

        // Crear un Label para mostrar la selección
        Label resultadoLabel = new Label(text: "Selección: ");

        // Configurar el administrador de diseño de la ventana
        ventana.setLayout(new FlowLayout());

        // Agregar la lista, el botón y el Label a la ventana
        ventana.add(comp: opcionesList);
        ventana.add(comp: mostrarSeleccionButton);
        ventana.add(comp: resultadoLabel);

        // Configurar el cierre de la ventana
        ventana.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent e) {
                System.exit(status: 0);
            }
        });

        // Configurar ActionListener para el botón
        mostrarSeleccionButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                // Obtener las selecciones y mostrarlas en el Label
                String[] selecciones = opcionesList.getSelectedItems();
                resultadoLabel.setText("Selección: " + String.join(delimiter: ", ", elements: selecciones));
            }
        });
```

Java AWT List

```
58
59          // Configurar el tamaño de la ventana
60          ventana.setSize(width: 300, height: 200);
61
62          // Centrar la ventana en la pantalla
63          Dimension pantalla = Toolkit.getDefaultToolkit().getScreenSize();
64          int x = (pantalla.width - ventana.getWidth()) / 2;
65          int y = (pantalla.height - ventana.getHeight()) / 2;
66          ventana.setLocation(x, y);
67
68          // Hacer visible la ventana
69          ventana.setVisible(b: true);
70      }
71  }
```

Java AWT List



Java AWT Canvas

Java AWT Canvas es un componente gráfico proporcionado por la biblioteca AWT (Abstract Window Toolkit) de Java. Se utiliza para proporcionar un área de dibujo en una ventana de aplicación donde los desarrolladores pueden realizar representaciones personalizadas y gráficos a través de la manipulación directa de gráficos en 2D

Java AWT Canvas

Dibujo personalizado: Proporciona un lienzo en blanco donde los desarrolladores pueden realizar dibujos personalizados, gráficos, gráficos vectoriales y representaciones visuales específicas para su aplicación.

Animaciones: Puede ser utilizado para crear animaciones personalizadas al redibujar el contenido del lienzo a intervalos regulares, lo que permite crear efectos de movimiento y transiciones suaves.

Java AWT Canvas

Gráficos 2D: Es especialmente útil cuando se necesita dibujar gráficos 2D personalizados, como gráficos de datos, diagramas, gráficos de juego y otras representaciones visuales.

Interacción con el usuario: Permite interactuar con el usuario mediante la captura de eventos del mouse y del teclado en el lienzo para realizar acciones específicas en respuesta a la interacción del usuario.

Personalización de componentes: Puede usarse para personalizar la apariencia y el comportamiento de otros componentes gráficos, como botones personalizados o controles deslizantes personalizados.

Java AWT Canvas

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

public class EjemploCanvas extends Canvas {
    public void paint(Graphics g) {
        // Dibujar un rectángulo rojo en el lienzo
        g.setColor(c: Color.RED);
        g.fillRect(x: 50, y: 50, width: 200, height: 100);
    }

    public static void main(String[] args) {
        // Crear una ventana
        Frame ventana = new Frame(title: "| Politecnico internacional | Ejemplo de Canvas");

        // Crear un lienzo personalizado
        Canvas lienzo = new EjemploCanvas();

        // Configurar el administrador de diseño de la ventana
        ventana.setLayout(new BorderLayout());

        // Agregar el lienzo al centro de la ventana
        ventana.add(comp: lienzo, constraints: BorderLayout.CENTER);

        // Configurar el cierre de la ventana
        ventana.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent e) {
                System.exit(status: 0);
            }
        });

        // Configurar el tamaño de la ventana
        ventana.setSize(width: 400, height: 300);

        // Centrar la ventana en la pantalla
        Dimension pantalla = Toolkit.getDefaultToolkit().getScreenSize();
        int x = (pantalla.width - ventana.getWidth()) / 2;
        int y = (pantalla.height - ventana.getHeight()) / 2;
        ventana.setLocation(x, y);

        // Hacer visible la ventana
        ventana.setVisible(b: true);
    }
}
```

Java AWT Canvas

Java AWT Scrollbar

Java AWT Scrollbar es un componente gráfico proporcionado por la biblioteca AWT (Abstract Window Toolkit) de Java. Se utiliza para permitir la interacción del usuario con áreas desplazables, como ventanas de visualización, paneles o componentes que muestran una cantidad mayor de contenido que lo que cabe en la pantalla en un momento dado.

Java AWT Scrollbar

Desplazamiento de contenido: Proporciona una forma de desplazar el contenido dentro de un área o componente que no cabe completamente en la pantalla. Esto es útil para navegar por listas largas, documentos extensos, imágenes grandes y otras áreas de contenido.

Control de navegación: Permite al usuario controlar la posición visible del contenido desplazando la barra de desplazamiento hacia arriba o hacia abajo (en el caso de una barra de desplazamiento vertical) o hacia la izquierda y hacia la derecha (en el caso de una barra de desplazamiento horizontal).

Java AWT Scrollbar

Eventos de desplazamiento: Puede utilizarse para capturar eventos de desplazamiento, lo que permite a la aplicación responder a los cambios en la posición de la barra de desplazamiento y actualizar el contenido en consecuencia.

Personalización: Puede personalizarse para ajustarse a las necesidades específicas de la interfaz de usuario, como la personalización de la apariencia y el comportamiento de la barra de desplazamiento.

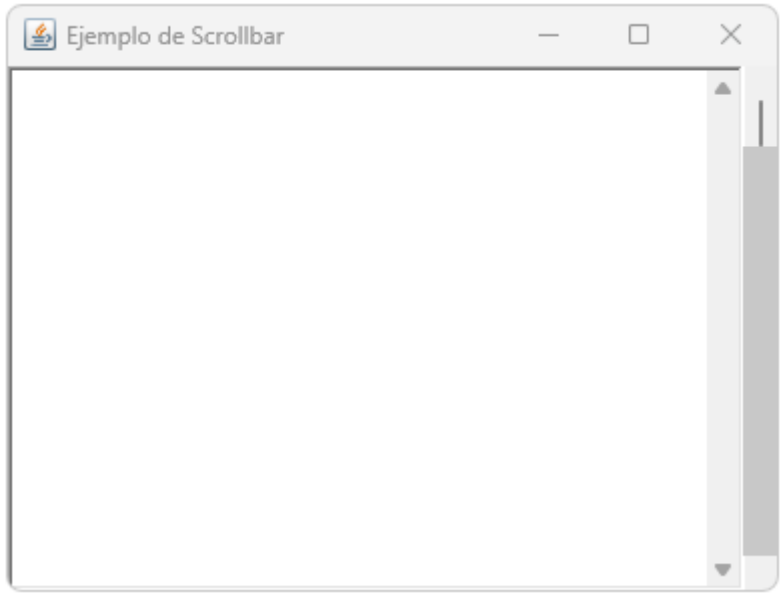
Java AWT Scrollbar

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

public class EjemploScrollbar {
    public static void main(String[] args) {
        // Crear una ventana
        Frame ventana = new Frame(title: "Ejemplo de Scrollbar");

        // Crear una barra de desplazamiento vertical
        Scrollbar barraVertical = new Scrollbar(orientation:Scrollbar.VERTICAL, value: 0, visible:10, minimum:0, maximum:100);

        // Crear un área de texto
        TextArea areaTexto = new TextArea(rows: 10, columns:30);

        // Configurar el administrador de diseño de la ventana
        ventana.setLayout(new BorderLayout());

        // Agregar la barra de desplazamiento y el área de texto a la ventana
        ventana.add(comp: barraVertical, constraints:BorderLayout.EAST);
        ventana.add(comp: areaTexto, constraints:BorderLayout.CENTER);

        // Configurar el cierre de la ventana
        ventana.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent e) {
                System.exit(status: 0);
            }
        });

        // Configurar el listener para la barra de desplazamiento
        barraVertical.addAdjustmentListener(new AdjustmentListener() {
            public void adjustmentValueChanged(AdjustmentEvent e) {
                // Obtener el valor actual de la barra de desplazamiento
                int valor = barraVertical.getValue();
                // Establecer la posición del área de texto en función del valor de la barra
                areaTexto.setScrollPosition(valor);
            }
        });
```

Java AWT Scrollbar

```
48
49          // Configurar el tamaño de la ventana
50          ventana.setSize(width: 400, height: 300);
51
52          // Centrar la ventana en la pantalla
53          Dimension pantalla = Toolkit.getDefaultToolkit().getScreenSize();
54          int x = (pantalla.width - ventana.getWidth()) / 2;
55          int y = (pantalla.height - ventana.getHeight()) / 2;
56          ventana.setLocation(x, y);
57
58          // Hacer visible la ventana
59          ventana.setVisible(b: true);
60      }
61  }
```

Java AWT Scrollbar



Java AWT MenuItem y Menu

Java AWT MenuItem y Menu son componentes gráficos proporcionados por la biblioteca AWT (Abstract Window Toolkit) de Java que se utilizan para crear menús y elementos de menú en una interfaz de usuario

Java AWT MenuItem y Menu

Menu (java.awt.Menu)

Definición: Un objeto Menu representa un menú desplegable en una aplicación de interfaz de usuario. Es un contenedor de elementos de menú (MenuItem) que se pueden seleccionar.

Función: El componente Menu se utiliza para agrupar y organizar elementos de menú relacionados en una estructura de menú jerárquica. Puede contener submenús (Menu) y elementos de menú (MenuItem) que pueden ser seleccionados por el usuario.

Java AWT MenuItem y Menu

MenuItem (java.awt.MenuItem)

Definición: Un objeto MenuItem representa un elemento de menú individual en una aplicación de interfaz de usuario. Puede estar contenido dentro de un menú (Menu) o aparecer directamente en la barra de menús de una ventana.

Función: Los componentes MenuItem se utilizan para proporcionar opciones o comandos que el usuario puede seleccionar desde un menú. Estos elementos de menú pueden desencadenar acciones cuando se seleccionan, como abrir una ventana, realizar una operación o mostrar un cuadro de diálogo.

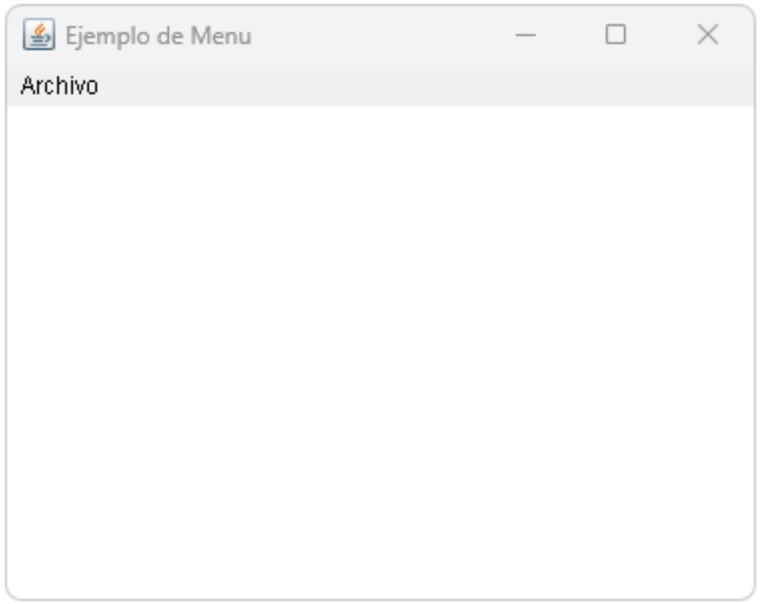
Java AWT MenuItem y Menu

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

public class EjemploMenu {
    public static void main(String[] args) {
        // Crear una ventana
        Frame ventana = new Frame(title: "Ejemplo de Menu");

        // Crear una barra de menús
        MenuBar barraMenus = new MenuBar();

        // Crear un menú "Archivo"
        Menu menuArchivo = new Menu(label: "Archivo");

        // Crear un elemento de menú "Abrir"
        MenuItem abrirItem = new MenuItem(label: "Abrir");

        // Crear un elemento de menú "Salir"
        MenuItem salirItem = new MenuItem(label: "Salir");

        // Agregar elementos de menú al menú "Archivo"
        menuArchivo.add(mi: abrirItem);
        menuArchivo.addSeparator(); // Separador entre elementos de menú
        menuArchivo.add(mi: salirItem);

        // Agregar el menú "Archivo" a la barra de menús
        barraMenus.add(m: menuArchivo);

        // Configurar la barra de menús en la ventana
        ventana.setMenuBar(mb: barraMenus);

        // Configurar el cierre de la ventana
        ventana.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent e) {
                System.exit(status: 0);
            }
        });

        // Configurar ActionListener para el elemento "Salir"
        salirItem.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                System.exit(status: 0); // Salir de la aplicación al hacer clic en "Salir"
            }
        });
```

Java AWT MenuItem y Menu

```
55
56          // Configurar el tamaño de la ventana
57          ventana.setSize(width: 300, height: 200);
58
59          // Centrar la ventana en la pantalla
60          Dimension pantalla = Toolkit.getDefaultToolkit().getScreenSize();
61          int x = (pantalla.width - ventana.getWidth()) / 2;
62          int y = (pantalla.height - ventana.getHeight()) / 2;
63          ventana.setLocation(x, y);
64
65          // Hacer visible la ventana
66          ventana.setVisible(b: true);
67      }
68  }
```

Java AWT MenuItem y Menu



Java AWT PopupMenu

Java AWT PopupMenu es un componente gráfico proporcionado por la biblioteca AWT (Abstract Window Toolkit) de Java. Este componente se utiliza para crear menús contextuales o emergentes que aparecen en respuesta a una acción específica del usuario, generalmente un clic derecho del mouse en una ventana o componente de la interfaz de usuario.

Java AWT PopupMenu

Menús contextuales: Permite crear menús que aparecen en respuesta a eventos del mouse, como un clic derecho en un área específica de una ventana o componente.

Acciones contextualmente relevantes: Se utiliza para proporcionar opciones de menú que son relevantes para el contexto en el que se activa el menú emergente. Esto puede incluir acciones relacionadas con elementos seleccionados o el área en la que se hizo clic.

Java AWT PopupMenu

Interfaz de usuario receptiva: Mejora la experiencia del usuario al proporcionar acceso rápido a acciones comunes o funciones específicas en función de la posición del cursor en la interfaz de usuario.

Personalización: Puede personalizarse para incluir una variedad de elementos de menú, como elementos de menú de acción, elementos de menú de submenú y elementos de menú de verificación.

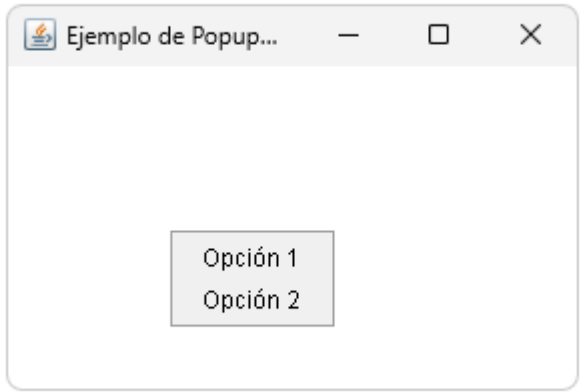
Java AWT PopupMenu

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

public class EjemploPopupMenu {
    public static void main(String[] args) {
        // Crear una ventana
        Frame ventana = new Frame(title: "Ejemplo de PopupMenu");

        // Crear un PopupMenu
        PopupMenu popupMenu = new PopupMenu(label: "Menú Emergente");

        // Crear elementos de menú para el PopupMenu
        MenuItem opcion1 = new MenuItem(label: "Opción 1");
        MenuItem opcion2 = new MenuItem(label: "Opción 2");

        // Agregar elementos de menú al PopupMenu
        popupMenu.add(mi: opcion1);
        popupMenu.add(mi: opcion2);

        // Configurar el administrador de diseño de la ventana
        ventana.setLayout(new BorderLayout());

        // Agregar el PopupMenu a la ventana
        ventana.add(popup: popupMenu);

        // Configurar el cierre de la ventana
        ventana.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent e) {
                System.exit(status: 0);
            }
        });

        // Configurar ActionListener para las opciones del PopupMenu
        opcion1.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                System.out.println(x: "Opción 1 seleccionada");
            }
        });

        opcion2.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                System.out.println(x: "Opción 2 seleccionada");
            }
        });
```

Java AWT PopupMenu

```
55
56          // Configurar el tamaño de la ventana
57          ventana.setSize(width: 300, height: 200);
58
59          // Centrar la ventana en la pantalla
60          Dimension pantalla = Toolkit.getDefaultToolkit().getScreenSize();
61          int x = (pantalla.width - ventana.getWidth()) / 2;
62          int y = (pantalla.height - ventana.getHeight()) / 2;
63          ventana.setLocation(x, y);
64
65          // Configurar el listener para mostrar el PopupMenu en clic derecho
66          ventana.addMouseListener(new MouseAdapter() {
67              public void mouseClicked(MouseEvent e) {
68                  if (e.getButton() == MouseEvent.BUTTON3) { // Clic derecho
69                      popupMenu.show(origin: ventana, x: e.getX(), y: e.getY());
70                  }
71              }
72          });
73
74          // Hacer visible la ventana
75          ventana.setVisible(b: true);
76      }
77  }
```

Java AWT PopupMenu



Java AWT Panel

Java AWT Panel es un componente gráfico proporcionado por la biblioteca AWT (Abstract Window Toolkit) de Java. Sirve para actuar como un contenedor de componentes secundarios (como botones, etiquetas, campos de texto, etc.) dentro de una ventana o un contenedor principal.

Java AWT Panel

Organización de componentes: Proporciona una manera de agrupar y organizar otros componentes gráficos de la interfaz de usuario en un área definida dentro de una ventana o un contenedor principal.

Mejora de la estructura: Ayuda a mejorar la estructura de la interfaz de usuario al dividir una ventana en áreas o secciones lógicas, cada una de las cuales puede contener un conjunto específico de componentes.

Java AWT Panel

Aislamiento de funcionalidad: Permite agrupar componentes que realizan una función similar o están relacionados en una unidad coherente, lo que facilita la administración y el mantenimiento del código.

Personalización de diseño: Puede personalizarse con un administrador de diseño para controlar la disposición y la apariencia de los componentes secundarios dentro del panel.

Reutilización de componentes: Los paneles se pueden reutilizar en diferentes partes de la aplicación para mantener la consistencia en la apariencia y la funcionalidad.

Java AWT Panel

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

public class EjemploPanel {
    public static void main(String[] args) {
        // Crear una ventana
        Frame ventana = new Frame(title: "| Politecnico internacional | Ejemplo de Panel");

        // Crear un Panel 1
        Panel panel1 = new Panel();
        // Configurar un botón en el Panel 1
        Button boton1 = new Button(label: "Botón 1");
        panel1.add(comp: boton1);

        // Crear un Panel 2
        Panel panel2 = new Panel();
        // Configurar una etiqueta en el Panel 2
        Label etiqueta = new Label(text: "Etiqueta en Panel 2");
        panel2.add(comp: etiqueta);

        // Configurar el administrador de diseño de la ventana
        ventana.setLayout(new GridLayout(rows: 2, cols: 1)); // Dos filas, una columna

        // Agregar los paneles a la ventana
        ventana.add(comp: panel1);
        ventana.add(comp: panel2);

        // Configurar el cierre de la ventana
        ventana.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent e) {
                System.exit(status: 0);
            }
        });

        // Configurar el tamaño de la ventana
        ventana.setSize(width: 300, height: 200);

        // Centrar la ventana en la pantalla
        Dimension pantalla = Toolkit.getDefaultToolkit().getScreenSize();
        int x = (pantalla.width - ventana.getWidth()) / 2;
        int y = (pantalla.height - ventana.getHeight()) / 2;
        ventana.setLocation(x, y);

        // Hacer visible la ventana
        ventana.setVisible(b: true);
    }
}
```

Java AWT Panel

Java AWT Dialog

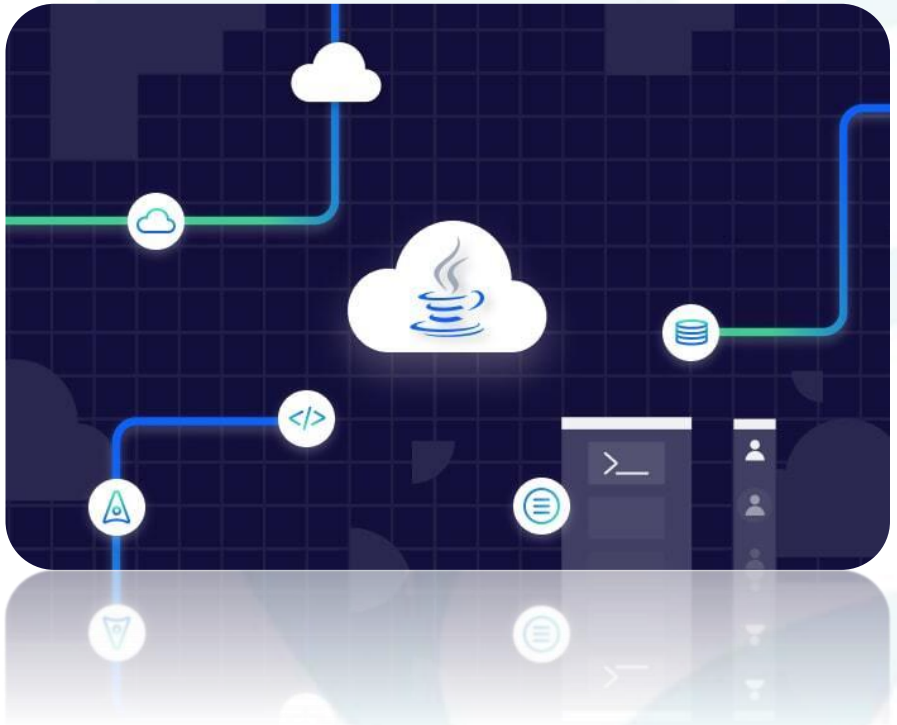
Java AWT Dialog es un componente gráfico proporcionado por la biblioteca AWT (Abstract Window Toolkit) de Java. Sirve para crear ventanas de diálogo modales o no modales que se utilizan para interactuar con el usuario y recopilar información o confirmaciones específicas durante la ejecución de una aplicación.

Java AWT Dialog

Interacción de usuario: Proporciona una forma de interactuar con el usuario a través de ventanas emergentes que pueden mostrar información importante, recopilar datos o confirmar acciones.

Recopilación de datos: Puede utilizarse para recopilar datos del usuario, como formularios de entrada, selecciones de opciones o cualquier otro tipo de entrada.

Mensajes de alerta o confirmación: Se utiliza para mostrar mensajes de alerta o confirmación que requieren la atención o la acción del usuario antes de continuar con la aplicación.

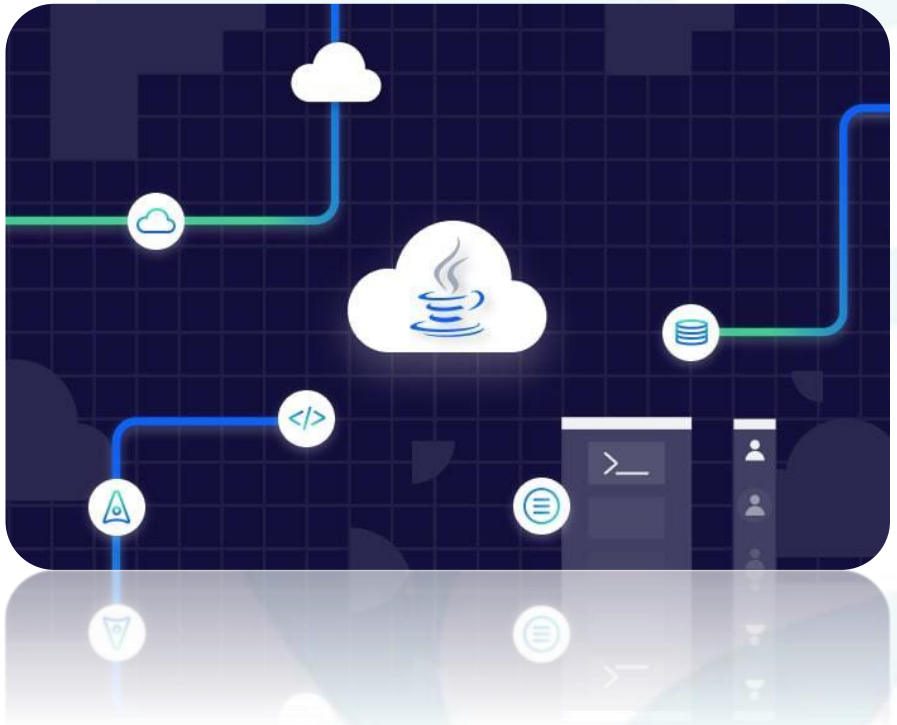


Java AWT Dialog

Ventanas modales: Los diálogos pueden ser modales o no modales. Los diálogos modales bloquean la interacción con la ventana principal hasta que se cierre el diálogo, lo que garantiza que el usuario atienda el diálogo antes de continuar.

Personalización: Los diálogos se pueden personalizar con componentes como etiquetas, botones, campos de texto y otros elementos para adaptarse a las necesidades específicas de la aplicación.

Flujo de trabajo de la aplicación: Ayuda a controlar el flujo de trabajo de la aplicación al proporcionar una manera de pausar o detener temporalmente la ejecución hasta que se complete la interacción con el diálogo.



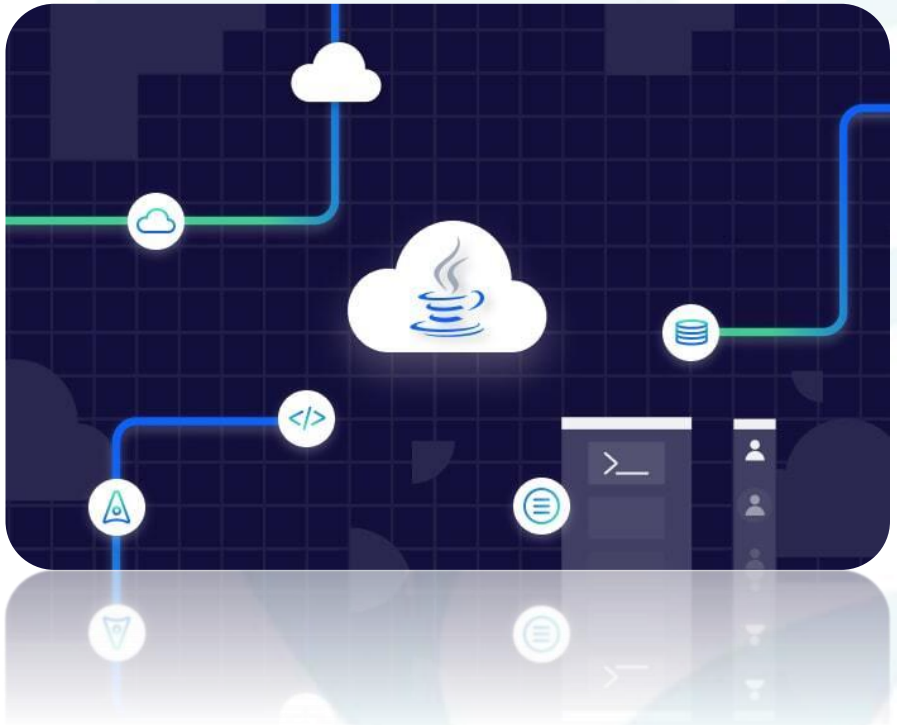
Java AWT Dialog

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

public class EjemploDialogo {
    public static void main(String[] args) {
        // Crear una ventana
        Frame ventana = new Frame(title: "| Politecnico internacional | Ejemplo de Dialogo");

        // Crear un botón en la ventana
        Button botonAbrirDialogo = new Button(label: "Abrir Diálogo");
        ventana.add(comp: botonAbrirDialogo, constraints: BorderLayout.CENTER);

        // Configurar el cierre de la ventana principal
        ventana.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent e) {
                System.exit(status: 0);
            }
        });

        // Configurar el ActionListener para el botón
        botonAbrirDialogo.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                // Crear un diálogo modal
                Dialog dialogo = new Dialog(owner: ventana, title: "Ingrese su nombre", modal: true);

                // Configurar un layout para el diálogo
                dialogo.setLayout(new FlowLayout());

                // Crear un campo de texto en el diálogo
                TextField campoTexto = new TextField(columns: 20);
                dialogo.add(comp: campoTexto);

                // Crear un botón "Aceptar" en el diálogo
                Button botonAceptar = new Button(label: "Aceptar");
                dialogo.add(comp: botonAceptar);

                // Configurar el cierre del diálogo
                dialogo.addWindowListener(new WindowAdapter() {
                    public void windowClosing(WindowEvent e) {
                        dialogo.dispose(); // Cerrar el diálogo al hacer clic en el botón de cierre
                    }
                });
```

Java AWT Dialog

```
53
54                  // Configurar ActionListener para el botón "Aceptar"
55                  botonAceptar.addActionListener(new ActionListener() {
56                      public void actionPerformed(ActionEvent e) {
57                          String nombre = campoTexto.getText();
58                          ventana.setTitle("Hola, " + nombre); // Mostrar el nombre en la ventana principal
59                          dialogo.dispose(); // Cerrar el diálogo
60                      }
61                  });
62
63                  // Configurar el tamaño y mostrar el diálogo
64                  dialogo.setSize(width: 300, height: 100);
65                  dialogo.setLocationRelativeTo(c: null); // Centrar el diálogo
66                  dialogo.setVisible(b: true);
67              }
68          });
69
70          // Configurar el tamaño de la ventana principal
71          ventana.setSize(width: 300, height: 100);
72
73          // Centrar la ventana principal en la pantalla
74          Dimension pantalla = Toolkit.getDefaultToolkit().getScreenSize();
75          int x = (pantalla.width - ventana.getWidth()) / 2;
76          int y = (pantalla.height - ventana.getHeight()) / 2;
77          ventana.setLocation(x, y);
78
79          // Hacer visible la ventana principal
80          ventana.setVisible(b: true);
81      }
82  }
```

Java AWT Dialog



¿Preguntas?

CONCLUSIONES