



**Tecnológico Nacional de México  
Campus Orizaba**

**Tópicos Avanzados de Programación**

**Ingeniería en Sistemas Computacionales**

**Tema 1:  
Interfaz gráfica de Usuario**

**Integrantes:**

**Muñoz Hernández Vania Lizeth – 21011009  
Rodríguez Pulido Jesús Raymundo – 21011032  
Romero Ovando Karyme Michelle – 21011037**

**Grupo:  
4g2A**

**Fecha de entrega: 17/Febrero/2023**

## **1. Introducción**

Hoy en día, las interfaces de usuario son esenciales en cualquier proyecto de desarrollo de software. Estas interfaces son la forma en que las personas interactúan con las computadoras y otros dispositivos. Una interfaz de usuario bien diseñada puede marcar la diferencia entre un programa difícil de usar y uno que es intuitivo y fácil de manejar. Es por eso que en este informe se tratará el tema de las interfaces de usuario y se describirán las prácticas y herramientas necesarias para crear interfaces efectivas en Java. También se presentarán algunos ejemplos de interfaces de usuario funcionales que ilustrarán los conceptos teóricos relacionados con este tema. El objetivo principal de este informe es proporcionar una visión general completa de las interfaces de usuario y su importancia en el desarrollo de software, además de presentar cada uno de los GUIs elaborados y su funcionamiento.

## **2. Competencia específica**

Desarrolla soluciones de software para resolver problemas en diversos contextos utilizando programación concurrente, acceso a datos, que soporten interfaz gráfica de usuario y consideren dispositivos móviles.

## **3. Marco Teórico**

Las interfaces de usuario son un componente fundamental de la interacción humano-computadora, ya que son el medio a través del cual los usuarios interactúan con los sistemas informáticos. Una interfaz de usuario bien diseñada puede hacer que un sistema sea fácil de usar y comprender, mediante la representación del código del backend de un sistema de la forma más clara posible para el usuario para simplificarle las tareas diarias, mientras que una interfaz pobre puede dificultar el uso y frustrar al usuario.

La creación de una interfaz gráfica de usuario (GUI, por sus siglas en inglés) implica diseñar y desarrollar una interfaz visual que permita a los usuarios interactuar con un software de manera intuitiva y eficiente. A continuación, se describen algunos de los aspectos clave a tener en cuenta durante el proceso de creación de una GUI:

1. **Análisis de requisitos:** antes de comenzar a diseñar la GUI, es importante comprender los requisitos del usuario y del software. Esto puede implicar la recopilación de información sobre el público objetivo, los objetivos del software y las tareas que los usuarios necesitan realizar.
2. **Diseño de la interfaz:** una vez que se han identificado los requisitos del usuario y del software, se puede comenzar a diseñar la GUI. Esto implica crear un boceto o prototipo que muestre cómo se verá y funcionará la interfaz. Es importante tener en cuenta la facilidad de uso, la estética y la accesibilidad durante el proceso de diseño.
3. **Implementación:** después de que se haya diseñado la interfaz, se debe desarrollar el software que la respalde. Esto implica escribir el código que permitirá que la interfaz se comunique con el software subyacente y realizar pruebas para asegurarse de que todo funciona correctamente.
4. **Pruebas de usuario:** una vez que se ha implementado la interfaz, es importante realizar pruebas de usuario para asegurarse de que es fácil de usar y comprender. Esto puede implicar la realización de pruebas con usuarios reales, la recopilación de comentarios y la realización de cambios en función de la retroalimentación recibida.
5. **Mantenimiento:** después de que se haya implementado y probado la interfaz, es importante realizar el mantenimiento regular para garantizar que siga funcionando correctamente. Esto puede implicar la realización de actualizaciones de software, la solución de errores y la realización de mejoras en función de la retroalimentación del usuario.

Dentro de nuestras interfaces, hay diversas acciones o sucesos que son generados en respuesta a una interacción del usuario con la interfaz, esto es a lo que se conoce como evento. A continuación, se describen algunos de los eventos más comunes en las interfaces gráficas de usuario:

- **Eventos de ratón:** Estos eventos se generan cuando el usuario realiza una acción con el ratón, como hacer clic en un botón, mover el puntero o arrastrar y soltar elementos.

- Eventos de teclado: Estos eventos se generan cuando el usuario presiona una tecla del teclado, como escribir texto en un campo de entrada o presionar una tecla de acceso directo.
- Eventos de ventana: Estos eventos se producen cuando se abre, cierra, mueve o cambia el tamaño de una ventana de la interfaz.
- Eventos de tiempo: Estos eventos se producen después de un período de tiempo determinado, como un temporizador que activa una función después de un tiempo preestablecido.
- Eventos de acción: Estos eventos se producen cuando se realiza una acción específica, como hacer clic en un botón o seleccionar un elemento de una lista desplegable.
- Eventos de cambio de estado: Estos eventos se generan cuando cambia el estado de un elemento de la interfaz, como cuando se selecciona un elemento de una lista o se cambia un valor en una barra de desplazamiento.
- Eventos de validación: Estos eventos se producen cuando se valida la entrada del usuario, como cuando se verifica que los datos ingresados en un formulario sean válidos.

El modelo de eventos de AWT en Java se descompone en dos grupos de elementos: las fuentes y los oyentes de eventos. Las fuentes son los elementos que generan los eventos (un botón, un cuadro de texto, etc), mientras que los oyentes son elementos que están a la espera de que se produzcan determinados tipos de eventos para emitir determinadas respuestas.

Para poder gestionar eventos, necesitamos definir el manejador de eventos correspondiente, un elemento que actúe de oyente sobre las fuentes de eventos que necesitamos considerar. Cada tipo de evento tiene asignada una interfaz, de modo que para poder gestionar dicho evento, el manejador deberá implementar la interfaz asociada. Los oyentes más comunes son:

<b>ActionListener</b>	<b>Para eventos de acción (pulsar un <i>Button</i> , por ejemplo)</b>
<b>ItemListener</b>	Cuando un elemento ( <i>Checkbox</i> , <i>Choice</i> , etc), cambia su estado
<b>KeyListener</b>	Indican una acción sobre el teclado: pulsar una tecla, soltarla, etc.
<b>MouseListener</b>	Indican una acción con el ratón que no implique movimiento del mismo: hacer click, presionar un botón, soltarlo, entrar / salir...
<b>MouseMotionListener</b>	Indican una acción con el ratón relacionada con su movimiento: moverlo por una zona determinada, o arrastrar el ratón.
<b>WindowListener</b>	Indican el estado de una ventana

Los componentes gráficos de control más comunes en las GUI (interfaces gráficas de usuario) incluyen:

1. Component: La clase padre *Component* no se puede utilizar directamente. Es una clase abstracta, que proporciona algunos métodos útiles para sus subclases.
2. Botón: Un botón es un componente que el usuario puede presionar para ejecutar una acción específica.
3. Campo de texto: Un campo de texto permite al usuario ingresar datos de texto, como un nombre o una dirección de correo electrónico.
4. Etiqueta: Una etiqueta es un componente que se utiliza para mostrar texto en la interfaz de usuario, como un título o una descripción de un elemento de la interfaz.

5. Lista desplegable: Una lista desplegable muestra una lista de opciones para que el usuario seleccione una opción específica.
6. Casilla de verificación: Una casilla de verificación permite al usuario seleccionar una opción, como una preferencia de configuración.
7. Barra de progreso: Una barra de progreso indica visualmente el progreso de una tarea en curso.
8. Barra de desplazamiento: Una barra de desplazamiento permite al usuario desplazarse por un contenido que es demasiado grande para mostrarse en su totalidad en la pantalla.
9. Cuadro de diálogo: Un cuadro de diálogo es una ventana emergente que muestra información o solicita una respuesta del usuario.
10. Imágenes: Las imágenes se utilizan a menudo en la interfaz de usuario para agregar elementos visuales y ayudar a transmitir información.
11. Menús: Los menús se utilizan para organizar y presentar opciones y comandos al usuario de manera estructurada.

Los desarrolladores pueden utilizar diferentes lenguajes de programación y herramientas de desarrollo para manejar los componentes gráficos de control en las GUI. Su manejo implica varios pasos, como son:

1. Creación de componentes: Los desarrolladores crean los componentes de la interfaz de usuario utilizando herramientas de diseño o código.
2. Asignación de identificadores: Cada componente debe tener un identificador único para poder ser identificado y manejado en el código.
3. Definición de eventos: Los desarrolladores definen qué eventos deben activarse cuando el usuario interactúa con un componente específico. Por ejemplo, un botón puede activar un evento cuando se hace clic en él.
4. Implementación de manejadores de eventos: Los manejadores de eventos son los fragmentos de código que se ejecutan cuando se activa un evento.

Los desarrolladores escriben los manejadores de eventos para realizar las acciones deseadas en respuesta a la interacción del usuario.

5. Pruebas y depuración: Una vez que se ha creado el código para manejar los componentes gráficos de control, se debe probar y depurar para asegurarse de que la interfaz de usuario funciona como se espera.

Algunas de las herramientas y lenguajes más comunes incluyen Java Swing, C# con Windows Forms, Python con Tkinter y JavaScript con React o Vue.

Con motivo de desarrollo de nuestra práctica, usaremos el lenguaje Java, asegurándonos de aplicar conocimiento teórico en el desarrollo de cada uno de nuestros programas.

## 4. Material y Equipo

El material y equipo que se necesita para llevar a cabo la práctica son:

- ✓ Computadora
- ✓ Software y versión usados
- ✓ Materiales de apoyo para el desarrollo de la práctica

## 5. Desarrollo de la Práctica

Clase	Método	Descripción
JCheckBox	JCheckBox()	Constructor de la clase JCheckBox.
	addItemListener()	Se conecta el componente con un objeto de la clase que maneja los sucesos originados en dicho componente.
	add()	Agrega el menú emergente especificado al componente.
	isSelected()	Devuelve true si el componente está seleccionado, en caso contrario devuelve false

Clase	Método	Descripción
<b>JRadioButton</b>	JRadioButton()	Constructor de la clase JRadioButton.
	setBounds()	Mueve y cambia el tamaño del componente.
	addChangeListener()	Agrega un ChangeListener al botón.
	add()	Agrega el menú emergente especificado al componente.

Clase	Método	Descripción
<b>JTextArea</b>	JTextArea ()	Constructor de la clase JTextArea.
	setText()	Se utiliza para establecer el texto del JTextArea.
	append()	
	setVisible()	Hace que el componente sea visible o invisible.

Clase	Método	Descripción
<b>ButtonGroup</b>	ButtonGroup()	Constructor de la clase ButtonGroup.
		Se utiliza para agrupar varios CheckBox.
	add()	Agrega el menú emergente especificado al componente.

Clase	Método	Descripción
-------	--------	-------------



<b>FlowLayout</b>	FlowLayout()	Constructor de la clase FlowLayout.
	setLayout()	Establece el administrador de diseño para este contenedor.
	pack()	Presenta una ventana con el tamaño necesario para mostrar los componentes que hay en ella.
<b>BorderLayout</b>		Ubica los componentes en cualquiera de sus 5 regiones que tiene.
<b>GridLayout</b>		Coloca los componentes en forma de matriz, haciendo que todos ocupen el mismo tamaño

Clase	Método	Descripción
<b>ChangeListener</b>	ChangeListener	
	stateChanged(ChangeEvent e)	Se invoca cuando el destino del listener ha cambiado de estado.

Clase	Método	Descripción
<b>ItemListener</b>	ItemListener	Recibe una notificación cada vez que hace clic en la casilla de verificación
	itemStateChanged(ItemEvent event)	Se invoca automáticamente cada vez que haces clic o dejas de hacer clic en el componente de casilla de

	verificación registrado.
addItemListener()	Agrega un ItemListener al checkbox.

Clase	Método	Descripción
<b>JFrame</b>	Frame()	Constructor de la clase JFrame.
	void setTitle(String titulo)	Establece el titulo de la ventana con el String especificado.
	void setSize(int x, int y)	Cambia el tamaño del componente para que tenga una anchura x y una altura y.
	void setLocationRelativeTo (Component c)	Establece la ubicación de la ventana en relación con el componente especificado.
	void setDefaultCloseOperation (opciones)	Usado para especificar una de las siguientes opciones del botón de cierre EXIT_ON_CLOSE.
	void setResizable (boolean resizable)	Para evitar que se cambie el tamaño de la ventana.
	void setVisible (boolean b)	Muestra u oculta la ventana según el valor del parámetro b.
<b>ActionListener</b>	ActionListener	Interface que debe ser

		implementada para gestionar eventos.
	void actionPerformed (actionEvent e)	se invoca cuando ocurre un evento.
<b>Container</b>	Container getContentPane()	Retorna un objeto ContentPane de la ventana.
	void setLayout (LayoutManager mgr)	Establece el layout de la ventana.
	Component add (Component comp)	Añade el componente especificado al final del contenedor.
<b>Component</b>	void addActionListener (this)	Añade un oyente de eventos al componente actual.
	void setBounds (int x, int y, int ancho, int alto)	Mueve y cambia el tamaño del componente.
<b>JLabel</b>	JLabel()	Constructor de la clase JLabel.
	void setText(String txt)	Define una línea de texto que mostrará el componente.
<b>TextField</b>	TextField()	Constructor de la clase TextField.
	String getText()	Retorna el texto contenido en el componente de texto.
<b>Button</b>	Button()	Constructor de la clase Button.
	void setText(String txt)	Define una línea de texto que mostrará el componente.

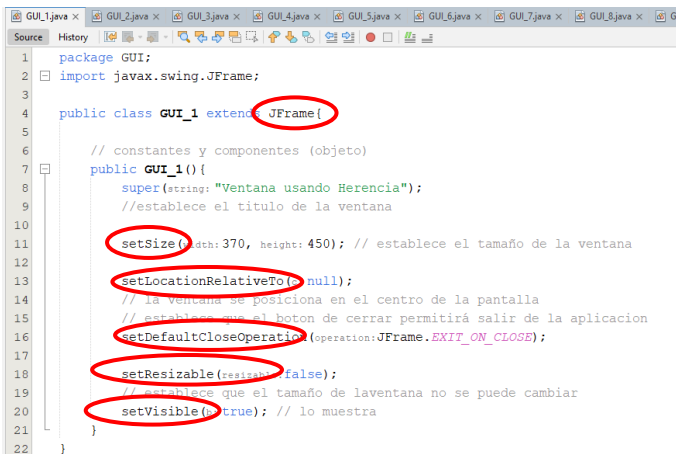
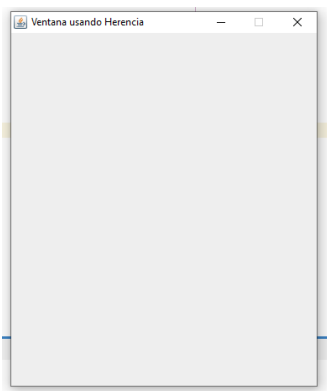
<b>JList</b>	JList()	Constructor de la clase JList.
	void setText(String txt)	Define una línea de texto que mostrará el componente.
	void setSelectionMode(int modoSeleccion)	Establece el modo de selección de la lista.
	void setModel(ListModel <E> model)	Establece el modelo que representa el contenido de la lista.
	int getSelectedIndex()	Devuelve el índice seleccionado.
<b>DefaultListModel</b>	DefaultListModel()	Contructor de la clase DefaultListModel.
	void addElement (Element e)	Añade el componente especificado al final de la lista.
	void removeElementAt(int índice)	Elimina el componente en el índice especificado.
	void removeAllElements()	Elimina todos los componentes de la lista y coloca su tamaño en cero.
	void clear()	Elimina todos los elementos de la lista.
<b>JScrollPane</b>	JScrollPane()	Descripcion de la clase JScrollPane.
	void setViewportView(component view)	Crea una ventana grafica si es necesario y luego establece su vista.
<b>Event</b>	Object getSource()	El objeto sobre el cual el evento inicialmente ha

		ocurrido.
<b>JOptionPane</b>	void showMessageDialog(Component componentePadre,      Object mensaje)	Crea un cuadro de dialogo.

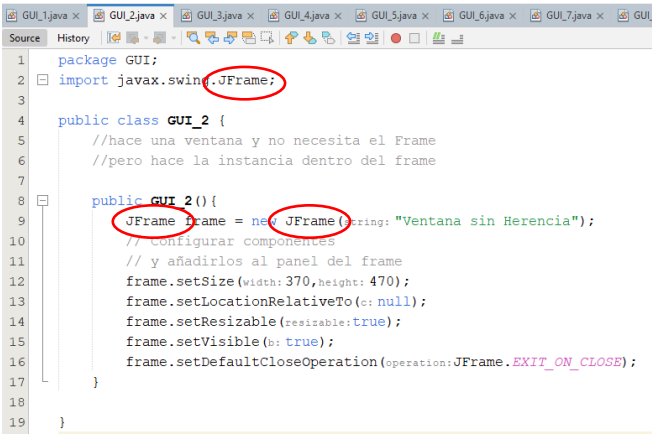
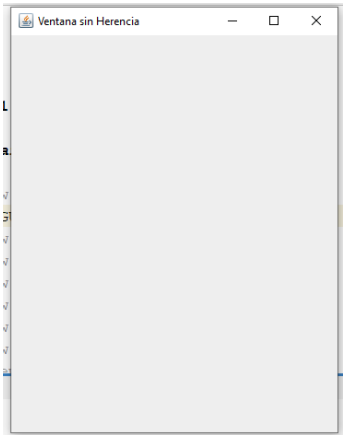
## 6. Resultados

Presentar la clase que genera el GUI e imágenes que señalen los componentes usados e imagen con breves descripciones del funcionamiento de este.

 GUI\_1

Clase	Ejecución
 <pre> 1 package GUI; 2 import javax.swing.JFrame; 3 4 public class GUI_1 extends JFrame{ 5 6     // constantes y componentes (objeto) 7     public GUI_1(){ 8         super(string: "Ventana usando Herencia"); 9         //establece el titulo de la ventana 10 11         setSize(370, height: 450); // establece el tamaño de la ventana 12 13         setLocationRelativeTo(null); 14         // la ventana se posiciona en el centro de la pantalla 15         // establece que el boton de cerrar permitirá salir de la aplicacion 16         setDefaultCloseOperation(operation:JFrame.EXIT_ON_CLOSE); 17 18         setResizable(false); 19         // establece que el tamaño de la ventana no se puede cambiar 20         setVisible(true); // lo muestra 21     } 22 } </pre>	
Componentes usados: Herencia de la clase JFrame con “extends JFrame”, métodos(setSize(), setLocationRelativeTo(), setDefaultCloseOperation(), setResizable(), setVisible())	
Pasos y funcionamiento: Primero creamos una clase usando herencia de la clase JFrame, posteriormente le asignaremos un nombre a la ventana creada en la línea 8, posteriormente definiremos las dimensiones de la ventana con el método setSize(), usamos el setDefaultCloseOperation() para que la ejecución se termine cuando se cierre el JFrame, y se uso un setVisible(true) para mostrar el JFrame creado.	

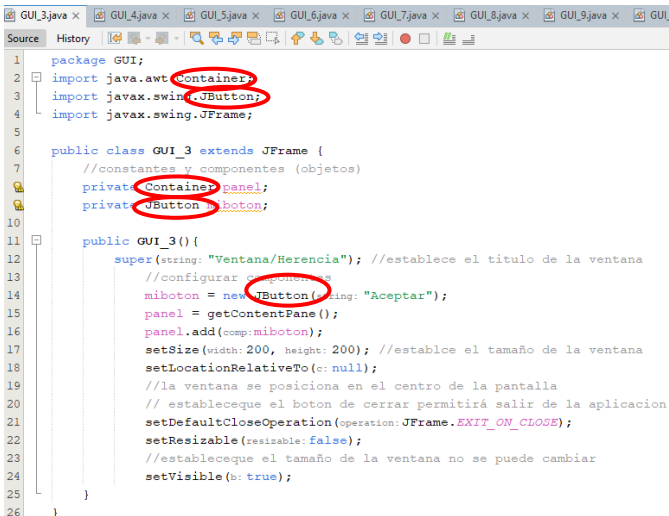
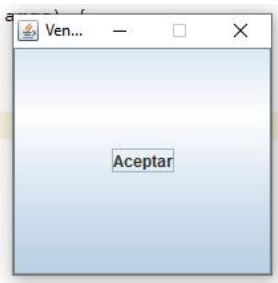
## GUI\_2.

Clase	Ejecución
 <pre> 1 package GUI; 2 import javax.swing.JFrame; 3 4 public class GUI_2 { 5     //hace una ventana y no necesita el Frame 6     //pero hace la instancia dentro del frame 7 8     public GUI_2() { 9         JFrame frame = new JFrame("Ventana sin Herencia"); 10        // configurar componentes 11        // y añadirlos al panel del frame 12        frame.setSize(width: 370, height: 470); 13        frame.setLocationRelativeTo(null); 14        frame.setResizable(resizable: true); 15        frame.setVisible(b: true); 16        frame.setDefaultCloseOperation(operation: JFrame.EXIT_ON_CLOSE); 17    } 18 } 19 </pre>	

Componentes usados: importación de la clase javax.swing.JFrame, instanciación del objeto de tipo JFrame.

Pasos: Este código se realizaron los mismos pasos que el GUI\_1 sin embargo este no usó herencia, por lo que primero importamos el package de JFrame, luego creamos el objeto de la clase JFrame con el nombre de la ventana, posteriormente se realizan los pasos del GUI\_1, Funcionamiento: Esta clase generará una ventana con un nombre.

## GUI\_3.

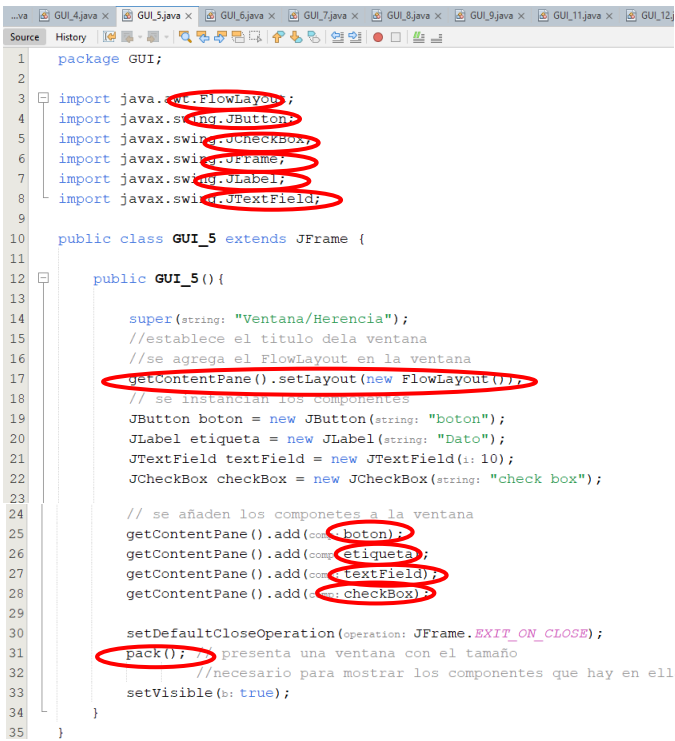
Clase	Ejecución
 <pre> 1 package GUI; 2 import java.awt.Container; 3 import javax.swing.JButton; 4 import javax.swing.JFrame; 5 6 public class GUI_3 extends JFrame { 7     //constantes y componentes (objetos) 8     private Container panel; 9     private JButton miboton; 10 11    public GUI_3() { 12        super("Ventana/Herencia"); //establece el título de la ventana 13        //configurar componentes 14        miboton = new JButton("Aceptar"); 15        panel = getContentPane(); 16        panel.add(miboton); 17        setSize(width: 200, height: 200); //establece el tamaño de la ventana 18        setLocationRelativeTo(null); 19        //la ventana se posiciona en el centro de la pantalla 20        // establece el botón de cerrar permitiendo salir de la aplicación 21        setDefaultCloseOperation(operation: JFrame.EXIT_ON_CLOSE); 22        setResizable(resizable: false); 23        //establece que el tamaño de la ventana no se puede cambiar 24        setVisible(b: true); 25    } 26 } </pre>	

Componentes usados: importación de la clase javax.swing.JButton y javax.awt.Container, variables para un JButton y Container, instanciación de un objeto de clase JButton(), se declara la variable del Container con el método getContentPane(),

Pasos: Para la elaboración de este GUI se importaron 3 package, el de Container, JButton y JFrame. Posteriormente se crea la clase usando herencia para usar JFrame, luego se declararon 2 variables, "panel" de tipo Container y "miboton" de tipo JButton, después se instanciará la variable "miboton" que tendrá el texto "Aceptar", luego la variable "panel" sera el valor que retorne el método getContentPane(), y a ese panel creado se añadirá el botón llamado "miboton", luego realizaremos los mismos pasos que seguimos en el GUI\_1.

Funcionamiento: Esta clase generara una ventana con un botón que dice "Aceptar" del tamaño de la ventana

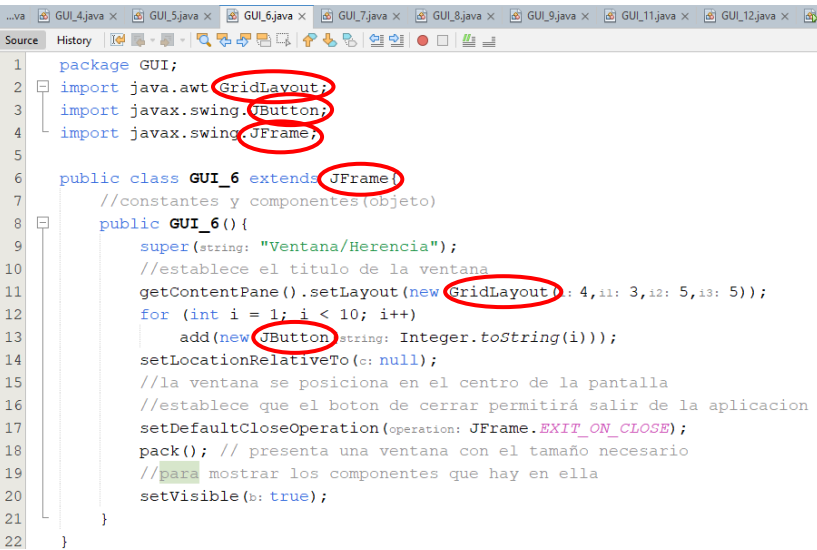
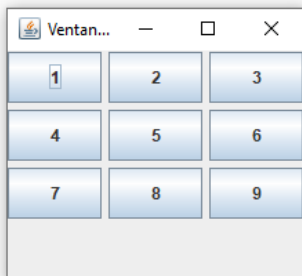
 GUI\_5.

Clase	Ejecución
	
<p>Componentes usados: importación de las clases (FlowLayout, JButton, JCheckBox, JFrame, JLabel, JTextField.), método getContentPane().setLayout(new FlowLayout()), y método .add() y pack()</p>	

Pasos: Para crear este GUI se importaron las clases antes mencionadas, luego se crea la clase con el título que tendrá la ventana, posteriormente con el método `getContentPane().setLayout(new FlowLayout())` es un `FlowLayout` que se agregara a la ventana, después de esto se declararan 4 variables con su respectiva instanciación de lo que llevara cada uno, primero la variable sera un botón llamado “botón” de tipo `JButton` que dirá “botón”, como segunda variable sera una etiqueta llamada “etiqueta” de tipo `JLabel` que dirá “Dato”, como tercer variable sera cuadro de texto llamado “textField” de tipo `JTextField`, este tendrá un tamaño de 10 de largo y la cuarta variable sera una casilla de selección llamada “checkBox” de tipo `JCheckBox` que dirá “check Box”, una vez inicializadas las variables procederemos a usar el método `.getContentPane. Add()` para agregar estos elementos al panel y después realizaremos los mismos pasos del GUI\_1.

Funcionamiento: Esta clase generara una ventana con 4 elementos , un botón que dice “botón”, una etiqueta que dice “Dato”, un cuadro de texto para escribir y una casilla de selección que dice “check box”.

#### GUI\_6.

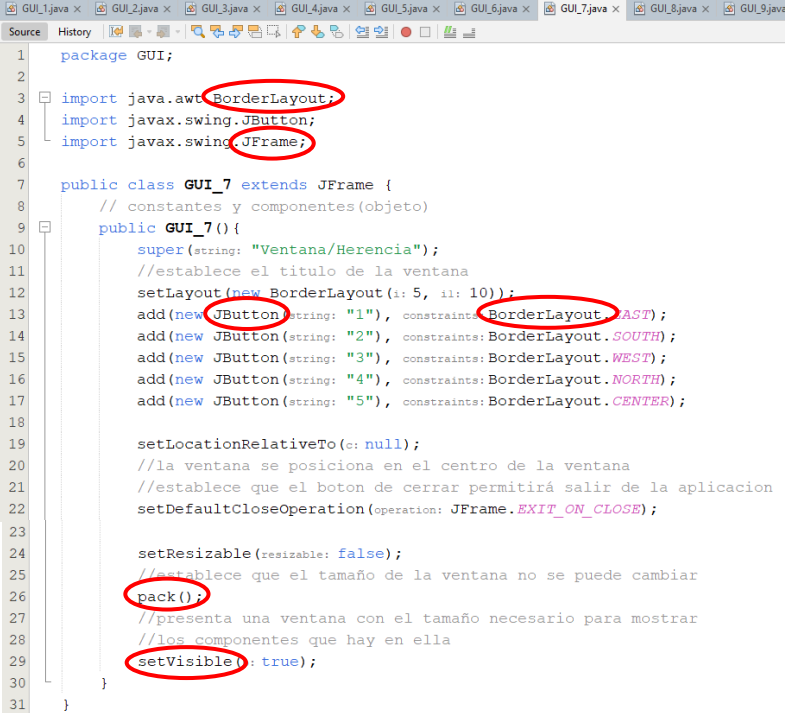
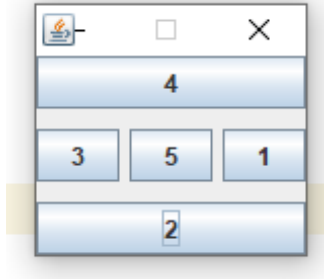
Clase	Ejecución
 <pre> 1 package GUI; 2 import java.awt.GridLayout; 3 import javax.swing.JButton; 4 import javax.swing.JFrame; 5 6 public class GUI_6 extends JFrame { 7     //constantes y componentes(objeto) 8     public GUI_6() { 9         super("Ventana/Herencia"); 10        //establece el titulo de la ventana 11        getContentPane().setLayout(new GridLayout(4,11: 3,12: 5,13: 5)); 12        for (int i = 1; i &lt; 10; i++) 13            add(new JButton(Integer.toString(i))); 14        setLocationRelativeTo(null); 15        //la ventana se posiciona en el centro de la pantalla 16        //establece que el boton de cerrar permitirá salir de la aplicacion 17        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); 18        pack(); // presenta una ventana con el tamaño necesario 19        //para mostrar los componentes que hay en ella 20        setVisible(true); 21    } 22 } </pre>	
<p>Componentes usados: importación de las clases (<code>GridLayout</code>, <code>JButton</code> y <code>JFrame</code>), Herencia de la clase <code>JFrame</code>, Elemento <code>GridLayout</code> y <code>JButton</code>.</p>	
<p>Pasos: Para este GUI, se importaran las clases <code>GridLayout</code>, <code>JButton</code> y <code>JFrame</code>, posteriormente se crea la clase usando herencia para la clase <code>JFrame</code>, posteriormente con ayuda del método</p>	



getContentPane().setLayout(new GridLayout) podemos realizar una distribución ordenada en la ventana para cada botón, después realizaremos un ciclo for donde la variable “i” comienza en 1 y terminara hasta que “i” sea menor que 10, donde cada ciclo “i” se aumentara en 1, y en cada ciclo se creara un botón que tendrá escrito el valor de la variable “i” y se agregara al panel creado, y usaremos el método pack() para presentar una ventana con el tamaño necesario.

Funcionamiento: Esta clase generara una ventana con una serie de botones, para ser precisos los números del ciclo creado que son del 1 al 9, y se ordenaran en forma de tablero gracias al método GridLayout.

## GUI\_7.

Clase	Ejecución
 <pre> 1 package GUI; 2 3 import java.awt.BorderLayout; 4 import javax.swing.JButton; 5 import javax.swing.JFrame; 6 7 public class GUI_7 extends JFrame { 8     // constantes y componentes (objeto) 9     public GUI_7() { 10         super(string: "Ventana/Herencia"); 11         //establece el titulo de la ventana 12         setLayout(new BorderLayout(5, 10)); 13         add(new JButton(string: "1"), constraints: BorderLayout.EAST); 14         add(new JButton(string: "2"), constraints: BorderLayout.SOUTH); 15         add(new JButton(string: "3"), constraints: BorderLayout.WEST); 16         add(new JButton(string: "4"), constraints: BorderLayout.NORTH); 17         add(new JButton(string: "5"), constraints: BorderLayout.CENTER); 18 19         setLocationRelativeTo(c: null); 20         //la ventana se posiciona en el centro de la ventana 21         //establece que el boton de cerrar permitirá salir de la aplicacion 22         setDefaultCloseOperation(operation: JFrame.EXIT_ON_CLOSE); 23 24         setResizable(resizable: false); 25         //establece que el tamaño de la ventana no se puede cambiar 26         pack(); 27         //presenta una ventana con el tamaño necesario para mostrar 28         //los componentes que hay en ella 29         setVisible(true); 30     } 31 } </pre>	

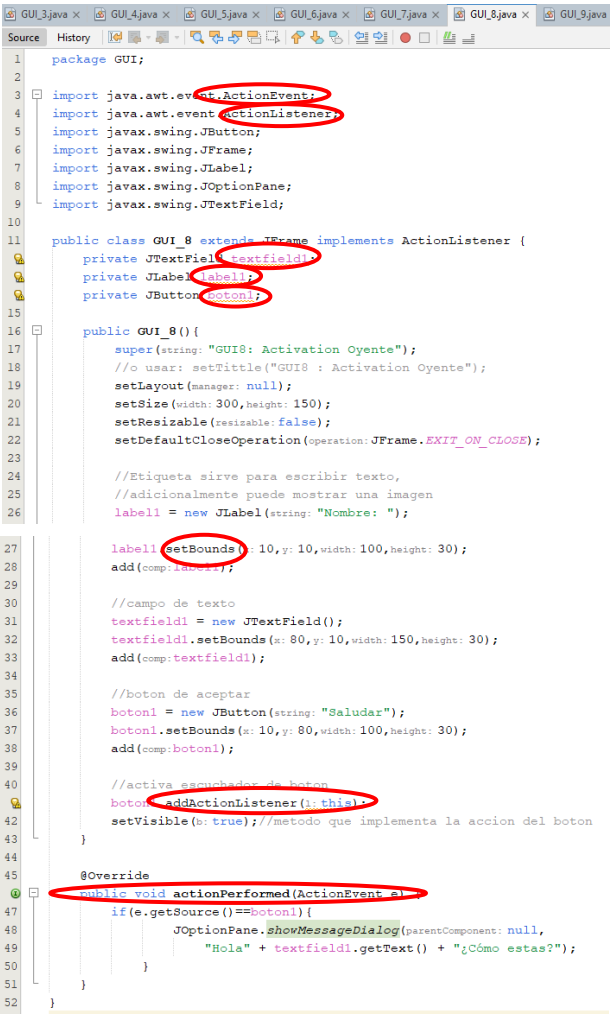
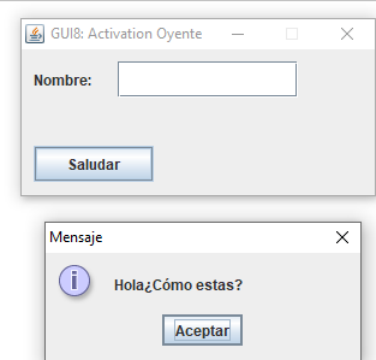
Componentes usados: Importación de las clases (BorderLayout, JButton y JFrame), elemento BorderLayout.(EAST,SOUTH,WEST,NORTH,CENTER), método pack() y set Visible().

Pasos: Para la elaboración de este GUI se importaron las clases mencionadas anteriormente en los componentes, luego se inicia la clase usando Herencia para la clase JFrame, luego con ayuda del método setLayout(BorderLayout(5,10)) se especificara que los espacios entre cada boton,

posteriormente se añadirán al JFrame 5 diferentes botones, y con ayuda del BorderLayout especificaremos en que lugar estarán situados ya sea Norte, Sur, Este, Oeste o en el centro, después con los mismos pasos del GUI\_1 se termina de realizar la clase.

Funcionamiento: Esta clase generara una ventana con 5 botones que llevaran escritos su numero correspondiente (1,2,3,4,5) y se distribuirán con ayuda del BorderLayout dependiendo a donde posicionemos

## GUI\_8.

Clase	Ejecución
 <pre> 1 package GUI; 2 3 import java.awt.event.ActionEvent; 4 import java.awt.event.ActionListener; 5 import javax.swing.JButton; 6 import javax.swing.JFrame; 7 import javax.swing.JLabel; 8 import javax.swing.JOptionPane; 9 import javax.swing.JTextField; 10 11 public class GUI_8 extends JFrame implements ActionListener { 12     private JTextField textField1; 13     private JLabel label1; 14     private JButton boton1; 15 16     public GUI_8() { 17         super("GUI8: Activation Oyente"); 18         //o user: setTitle("GUI8 : Activation Oyente"); 19         setLayout(manager: null); 20         setSize(width: 300,height: 150); 21         setResizable(resizable: false); 22         setDefaultCloseOperation(operation: JFrame.EXIT_ON_CLOSE); 23 24         //Etiqueta sirve para escribir texto, 25         //adicionalmente puede mostrar una imagen 26         label1 = new JLabel(string: "Nombre: "); 27 28         label1.setBounds(x: 10,y: 10,width: 100,height: 30); 29         add(comp: label1); 30 31         //campo de texto 32         textField1 = new JTextField(); 33         textField1.setBounds(x: 80,y: 10,width: 150,height: 30); 34         add(comp: textField1); 35 36         //boton de aceptar 37         boton1 = new JButton(string: "Saludar"); 38         boton1.setBounds(x: 10,y: 80,width: 100,height: 30); 39         add(comp: boton1); 40 41         //activa escuchador de boton 42         boton1.addActionListener(this); 43         setVisible(b: true); //metodo que implementa la accion del boton 44     } 45 46     @Override 47     public void actionPerformed(ActionEvent e) { 48         if (e.getSource() == boton1) { 49             JOptionPane.showMessageDialog(parentComponent: null, 50                 "Hola" + textField1.getText() + "¿Cómo estas?"); 51         } 52     } 53 } </pre>	

Componentes usados: Importación de las clases (ActionEvent, ActionListener, JButton, JFrame, JLabel, JOptionPane, JTextField), 3 variables(textField, label, botón), elemento .setBounds, .addActionListener(this) y un método llamado actionPerformed(ActionEvent e),

Pasos: Para la elaboración de este GUI se importaron las clases mencionadas en el apartado anterior, posteriormente crearemos 3 variables, las que serán un JTextField, un JButton y un JLabel, que usaremos en este GUI, realizaremos los pasos del GUI\_1 para crear la ventana con dimensiones escritas, después utilizaremos las 3 variables, comenzando por el JLabel la cual instanciaremos de la clase JLabel y tendrá escrita esta etiqueta la palabra “Nombre” y con ayuda del método .setBounds() lo posicionaremos en unas coordenadas específicas del JFrame, después se agregara esta etiqueta al JFrame, después repetiremos el mismo proceso con el JTextField, este únicamente lo posicionaremos guiándonos de las coordenadas para poder situarlo junto a la etiqueta de “Nombre”, y al final repetiremos este paso con la variable de JButton, que tendrá el mensaje de “Saludar” y lo añadiremos al JFrame.

Después de esto tendremos que usar el método .addActionListener(this), lo que nos ayudara para mas adelante con el método actionPerformed(ActionEvent e) le demos una acción específica para realizar.

Con ayuda de una decisión de if, recuperaremos la acción del evento que recibió el método actionPerformed, y comparara si la acción recuperada fue la del botón que asignamos como “Saludar”, si esta es la misma, se realizara lo que hay dentro de la condicional, que en este caso es una impresión con ayuda del JOptionPane donde imprimirá un saludo recuperando el JTextField con el método .getText().

Funcionamiento: Esta clase genera una pequeña interfaz con una etiqueta que dice “Nombre”, junto tendrá un cuadro de texto donde solicitaremos el nombre y junto a este se colocó un botón que dice “Saludar”, la clase creada generara un mensaje de saludo a la persona que escriba su nombre en el cuadro de texto

## GUI\_9.

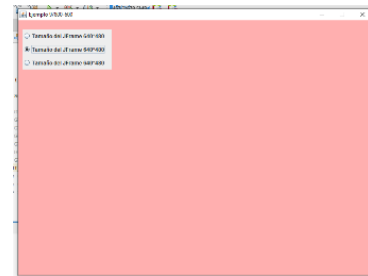
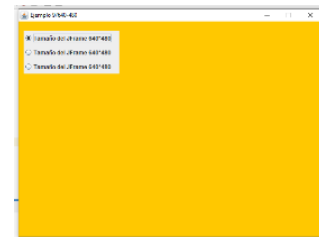
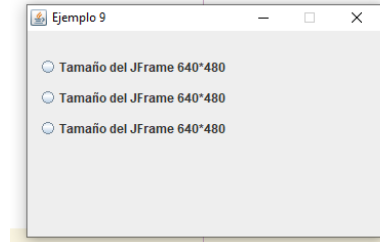
### Clase

```

1 package GUI;
2 import java.awt.Button;
3 import java.awt.Color;
4 import javax.swing.ButtonGroup;
5 import javax.swing.JFrame;
6 import javax.swing.JRadioButton;
7 import javax.swing.event.ChangeEvent;
8 import javax.swing.event.ChangeListener;
9
10 public class GUI_9 extends JFrame implements ChangeListener {
11     private JRadioButton radio1, radio2, radio3;
12     private ButtonGroup grupoBotones;
13
14     public GUI_9() {
15         super(string: "GUI9: Activation Oyente");
16         setLayout(manager:null); // layout absoluto
17         setTitle(title: "Ejemplo 9"); // titulo del JFrame
18         setSize(width: 350, height: 230); // dimensiones del JFrame
19         setResizable(resizable: false); // no redimensionable
20         setDefaultCloseOperation(operation: JFrame.EXIT_ON_CLOSE);
21         setVisible(b: true); // mostrar JFrame
22         //crear nuevo grupo de botones
23         grupoBotones = new ButtonGroup();
24
25         //agregar los 3 radioButton con su ChangeListener
26         radio1 = new JRadioButton(string: "Tamaño del JFrame 640*480");
27         radio1.setBounds(x: 10, y: 20, width: 200, height: 30);
28         radio1.addChangeListener(this);
29         add(comp: radio1);
30         grupoBotones.add(b: radio1);
31
32         radio2 = new JRadioButton(string: "Tamaño del JFrame 640*480");
33
34         radio2.setBounds(x: 10, y: 20, width: 200, height: 30);
35         radio2.addChangeListener(this);
36         add(comp: radio2);
37         grupoBotones.add(b: radio2);
38
39         radio3 = new JRadioButton(string: "Tamaño del JFrame 640*480");
40         radio3.setBounds(x: 10, y: 20, width: 200, height: 30);
41         radio3.addChangeListener(this);
42         add(comp: radio3);
43         grupoBotones.add(b: radio3);
44     }
45
46     @Override
47     public void stateChanged(ChangeEvent e) {
48         if(radio1.isSelected()) {
49             setSize(width: 640, height: 480);
50             setTitle(title: "Ejemplo 9/640-480");
51             this.getContentPane().setBackground(c: Color.orange);
52         }
53         if(radio2.isSelected()) {
54             setSize(width: 800, height: 600);
55             setTitle(title: "Ejemplo 9/800-600");
56             this.getContentPane().setBackground(c: Color.PINK);
57         }
58         if(radio3.isSelected()) {
59             setSize(width: 1024, height: 768);
60             setTitle(title: "Ejemplo 9/1024-768");
61             this.getContentPane().setBackground(c: Color.darkGray);
62         }
63     }
64 }

```

### Ejecución

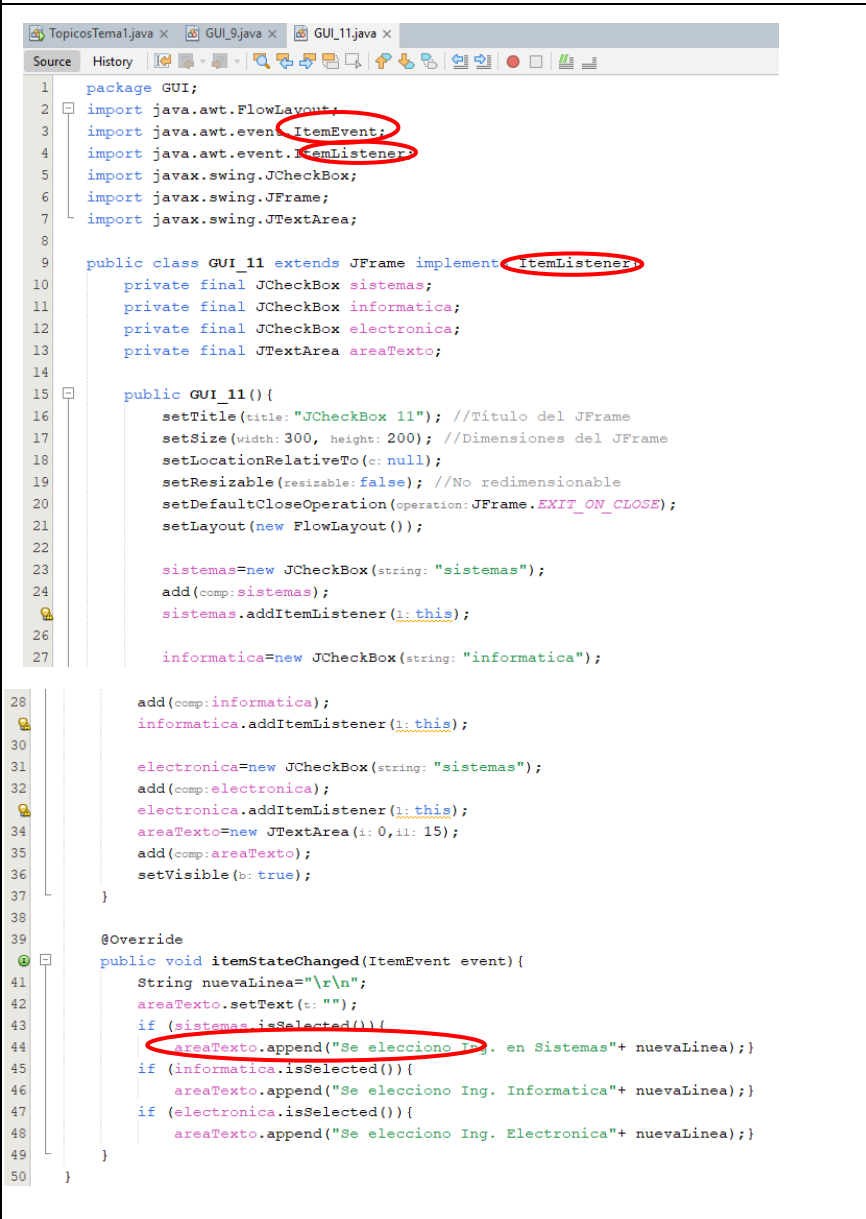



Componentes usados: Importacion de las clases(JRadioButton, Color, ButtonGroup, JFrame, ChangeEvent, ChangeListener) , un implements de ChangeListener y un método nuevo llamado stateChanged(ChangeEvent e).

Pasos: Para elaborar este GUI, comenzamos importando las clases mencionadas anteriormente, después se inicia la clase usando Herencia para la clase JFrame y se usa implements para la clase ChangeListener, comenzamos la clase creando 4 variables que usaremos, 3 variables de tipo JRadioButton y 1 variable de tipo ButtonGroup, primero se creara una ventana con ciertas características en donde vamos a trabajar, después instanciaremos la variable ButtonGroup, donde se añadirán los botones para después asignarles instrucciones, una vez instanciada la variable del ButtonGroup, inicializaremos las 3 variables de tipo JRadioButton asignándole un nombre, y unas coordenadas en donde se situaran en el JFrame, después de crear cada JRadioButton utilizaremos el método .add() para añadirlo al JFrame, y usaremos ese mismo método pero desde la variable ButtonGroup, para que de esta manera los botones pertenezcan solo a un grupo de botones y solo se pueda seleccionar 1 botón a la vez. Y para finalizar esta clase, usaremos el método stateChanged(ChangeEvent e) añadiéndole un override, en este método realizaremos una decisión para cada JRadioButton que tengamos en nuestra interfaz, usaremos el método .isSelected() el cual nos devolverá un true si es que ese JRadioButton fue seleccionado, de lo contrario un false, y con ayuda de la condición if asignaremos una serie de instrucciones si ese JRadioButton fue seleccionado, en este caso cambiaremos el tamaño del frame y le cambiaremos el color con ayuda del método this.getContentPane().setBackground(Color. "Color sugerido")

Funcionamiento: Esta clase realizara una ventana que contendrá 3 casillas de botones de tipo JRadioButton, la cual al seleccionar alguna de estas realizara unas acciones distintas, la cual modificara su tamaño y cambiara de color a la ventana ya que fueron las acciones que nosotros le asignamos

## GUI\_11.

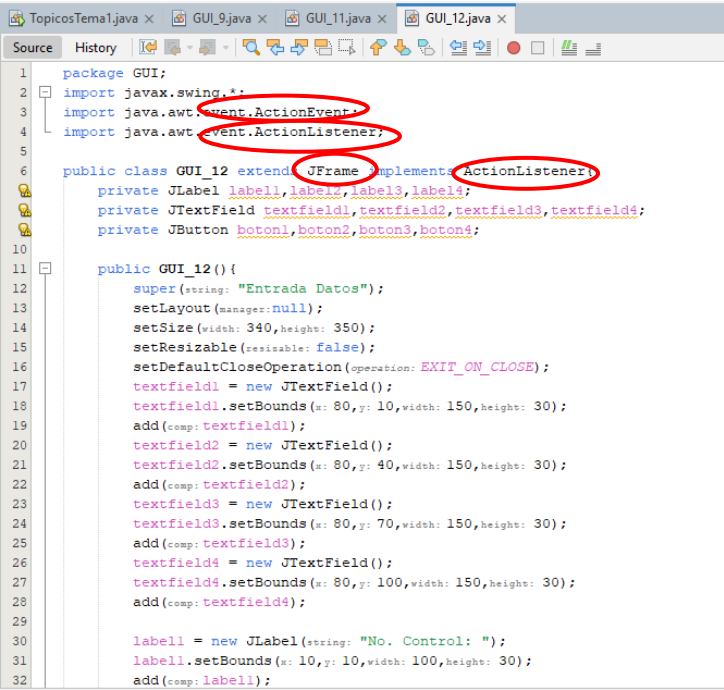
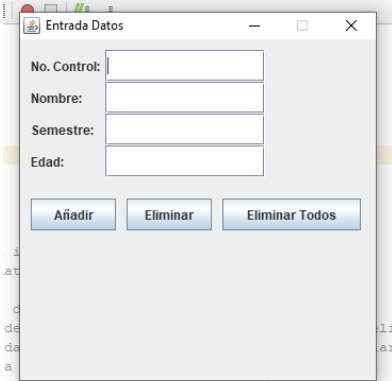
Clase	Ejecución
 <pre> 1 package GUI; 2 import java.awt.FlowLayout; 3 import java.awt.event.ItemEvent; 4 import java.awt.event.ItemListener; 5 import javax.swing.JCheckBox; 6 import javax.swing.JFrame; 7 import javax.swing.JTextArea; 8 9 public class GUI_11 extends JFrame implements ItemListener { 10     private final JCheckBox sistemas; 11     private final JCheckBox informatica; 12     private final JCheckBox electronica; 13     private final JTextArea areaTexto; 14 15     public GUI_11() { 16         setTitle("JCheckBox 11"); //Título del JFrame 17         setSize(width: 300, height: 200); //Dimensiones del JFrame 18         setLocationRelativeTo(c: null); 19         setResizable(resizable: false); //No redimensionable 20         setDefaultCloseOperation(operation: JFrame.EXIT_ON_CLOSE); 21         setLayout(new FlowLayout()); 22 23         sistemas=new JCheckBox(string: "sistemas"); 24         add(comp:sistemas); 25         sistemas.addItemListener(this); 26 27         informatica=new JCheckBox(string: "informatica"); 28 29         add(comp:informatica); 30         informatica.addItemListener(this); 31 32         electronica=new JCheckBox(string: "sistemas"); 33         add(comp:electronica); 34         electronica.addItemListener(this); 35         areaTexto=new JTextArea(i: 0, l: 15); 36         add(comp:areaTexto); 37         setVisible(b: true); 38     } 39 40     @Override 41     public void itemStateChanged(ItemEvent event){ 42         String nuevaLinea="\r\n"; 43         areaTexto.setText(c: ""); 44         if (sistemas.isSelected()){ 45             areaTexto.append("Se eleccion Ing. en Sistemas"+ nuevaLinea); 46         } 47         if (informatica.isSelected()){ 48             areaTexto.append("Se eleccion Ing. Informatica"+ nuevaLinea); 49         } 50         if (electronica.isSelected()){ 51             areaTexto.append("Se eleccion Ing. Electronica"+ nuevaLinea); 52         } 53     } 54 } </pre>	
<p>Componentes usados: Importacion de las clases (FlowLayout, ItemEvent, ItemListener, JCheckBox, JFrame, JTextArea) , implementación de la clase ItemListener, nuevo método .append</p>	
<p>Pasos: Para la elaboración de este GUI se comienza importando las clases mencionadas anteriormente, después crearemos 4 variables, 3 de tipo JCheckBox y 1 de tipo JTextArea, ahora se realizaran los mismos pasos del GUI_1 para asignarle dimensiones y nombre a la ventana que crearemos, para continuar instanciaremos cada variable de tipo JCheckBox con sus distintas</p>	

características, con ayuda del método `.add` los añadiremos al `JFrame` y con ayuda del método `.addChangeListener(this)` mas adelante le podremos asignar acciones a realizar.

Usaremos un nuevo método llamado `itemStateChanged(ItemEvent evento)` el cual recibirá el evento de tipo `ItemEvent`, y con ayuda de la condición `if` iremos asignándole acciones a cada evento que suceda, con cada variable de `JCheckBox` creada y con el método `.isSelected()`, en este caso las acciones que realizarán los `JCheckBox` serán añadir al `JTextArea` que creamos al inicio una `String` diferente en cada uno.

Funcionamiento: Esta clase genera una ventana con 3 distintas casillas para seleccionar, usamos los títulos de “Sistemas”, “informática” y “Electrónica”, y las acciones que realizara estas casillas son imprimir una cadena que dirá “Se selecciono la Ing. En” + la casilla que se seleccione.

## GUI\_12.

Clase	Ejecución
 <pre> 1 package GUI; 2 import javax.swing.*; 3 import java.awt.event.ActionEvent; 4 import java.awt.event.ActionListener; 5 6 public class GUI_12 extends JFrame implements ActionListener { 7     private JLabel label1, label2, label3, label4; 8     private JTextField textfield1, textfield2, textfield3, textfield4; 9     private JButton boton1, boton2, boton3, boton4; 10 11     public GUI_12() { 12         super("Entrada Datos"); 13         setLayout(manager: null); 14         setSize(width: 340, height: 350); 15         setResizable(resizable: false); 16         setDefaultCloseOperation(operation: EXIT_ON_CLOSE); 17         textfield1 = new JTextField(); 18         textfield1.setBounds(x: 80, y: 10, width: 150, height: 30); 19         add(comp: textfield1); 20         textfield2 = new JTextField(); 21         textfield2.setBounds(x: 80, y: 40, width: 150, height: 30); 22         add(comp: textfield2); 23         textfield3 = new JTextField(); 24         textfield3.setBounds(x: 80, y: 70, width: 150, height: 30); 25         add(comp: textfield3); 26         textfield4 = new JTextField(); 27         textfield4.setBounds(x: 80, y: 100, width: 150, height: 30); 28         add(comp: textfield4); 29 30         label1 = new JLabel(string: "No. Control: "); 31         label1.setBounds(x: 10, y: 10, width: 100, height: 30); 32         add(comp: label1); </pre>	

```

33     label2 = new JLabel(string: "Nombre: ");
34     label2.setBounds(x: 10, y: 40, width: 100, height: 30);
35     add(comp: label2);
36     label3 = new JLabel(string: "Semestre: ");
37     label3.setBounds(x: 10, y: 70, width: 100, height: 30);
38     add(comp: label3);
39     label4 = new JLabel(string: "Edad: ");
40     label4.setBounds(x: 10, y: 100, width: 100, height: 30);
41     add(comp: label4);
42
43     boton1 = new JButton(string: "Añadir");
44     boton1.setBounds(x: 10, y: 150, width: 80, height: 30);
45     add(comp: boton1);
46     boton2 = new JButton(string: "Eliminar");
47     boton2.setBounds(x: 100, y: 150, width: 80, height: 30);
48     add(comp: boton2);
49     boton3 = new JButton(string: "Eliminar Todos");
50     boton3.setBounds(x: 190, y: 150, width: 130, height: 30);
51     add(comp: boton3);
52     boton1.addActionListener(this);
53     boton2.addActionListener(this);
54     boton3.addActionListener(this);
55     setVisible(b: true);
56 }
57 @Override
58 public void actionPerformed(ActionEvent e) {
59     if(e.getSource() == boton3) {
60         JOptionPane.showMessageDialog(parentComponent: null,
61                                     message: "Se eliminó la lista.");
62     }
63 }
64 }

```

Componentes usados: Importación de las clases(Swing, ActionListener, ActionEvent), implementación de la clase ActionListener

Pasos: Para este GUI importaremos las clases mencionadas anteriormente, luego crearemos las variables que necesitaremos para este ejercicio, que serán 4 de tipo JLabel, 4 de tipo JTextField y 3 de tipo JButton, ahora con los pasos del GUI\_1 crearemos una ventana con las dimensiones específicas para la elaboración de esta interfaz, posterior a esto inicializaremos cada variable que creamos asignándole sus datos que necesitamos y las coordenadas donde los situaremos, a los botones les añadiremos una acción con el método .addActionListener(this), para posteriormente darles acciones, en este GUI solamente nos solicito que le asignaremos una acción al botón 3 creado que imprimiera una ventana que diga “Se elimino la lista”, por lo que después de la clase, se añade el método actionPerformed(ActionEvent e) donde con ayuda de la condición if, recibiremos el evento.getSource() y lo compararemos con el boton3, si es true por lo tanto si fue apretado ese botón e imprimirá lo solicitado

Funcionamiento: Esta clase creara una ventana con los elementos solicitados para este GUI, las cuales son 4 etiquetas que dirán “Nombre”, “Numero de control”, “Edad”, y “Semestre”, junto a estas etiquetas estarán 4 cuadros de textos donde se introducirá cada dato solicitado, y debajo de estos elementos estarán los 3 botones solicitados, que dirán “Añadir”, “Eliminar”, “Eliminar Todo”, y como en este ejercicio solo nos pidieron darle acción a un botón, este botón de “Eliminar Todo” imprimirá un mensaje diciendo “Se elimino la lista”.



## 7. Conclusiones

A partir de este reporte, se puede concluir que la interfaz gráfica de usuario es un medio el cual ayuda a la comunicación con un sistema. En la actualidad el contacto con las computadoras es una actividad muy necesaria, importante y común, con esto surge la necesidad de sistemas que faciliten el uso de las aplicaciones del ordenador.

La interfaz de usuario viene de la mano con la visualización de la información, es decir, que esta es importante para el éxito de los servicios o productos a la hora en que los usuarios toman decisiones. Con este reporte nos damos una idea general sobre cómo se empiezan las interfaces de usuario y como poco a poco podemos ir complementando cada una de ellas con respecto a la necesidad que se requiera.

## 8. Bibliografía

(Nieva, 2016)

### Bibliografía

Nieva, G. (25 de Abril de 2016). *d.Codingames*. Obtenido de [dcodingames.com: https://dcodingames.com/jcheckbox-y-jradiobutton/](https://dcodingames.com/jcheckbox-y-jradiobutton/)

Dix, A., Finlay, J., & Abowd, G. &. (2004). *Interacciones ser humano-computadora*. Pearson Education Limited.

Microsoft. (2021). *Control Events*. Disponible en: <https://docs.microsoft.com/en-us/dotnet/desktop/winforms/controls/events?view=netdesktop-7.0>

Microsoft. (2021). *Windows Forms Controls*. Disponible en: <https://docs.microsoft.com/en-us/dotnet/desktop/winforms/controls/>

Myers, B. A. (1998). *Una breve historia de la tecnología de interacción humano-computadora*. ACM interactions.

Shneiderman, B. &. (2010). *Diseñando la interfaz de usuario: estrategias para la interacción efectiva humano-computadora*. Pearson Education, Inc