

**TECNOLOGICO NACIONAL DE MEXICO
INSTITUTO TECNOLOGICO DE OAXACA
DEPARTAMENTO DE SISTEMAS Y COMPUTACION**

**PLAN DE TRABAJO SOBRE LAS ACTIVIDADES DE CLUB DE
PROGRAMACION**

RESPONSABLE	MC. MARICELA MORALES HERNANDEZ
DIA Y HORA	VIERNES DE 3:00 A 5:00
AULA	LABORATORIO DE REDES
SEMESTRE	ENERO- JUNIO 2020

INDICE

PRESENTACION	3
INTENCION.....	3
JUSTIFICACION	3
COMPETENCIAS ESPECÍFICAS:.....	3
COMPETENCIAS GENÉRICAS:.....	4
MATERIAL Y EQUIPO NECESARIO PARA LAS CLASES	4
RUBRICA DE EVALUACION PARA LAS ACTIVIDADES SOLICITADAS	5
CUADRO DE ACTIVIDADES REALIZADAS A LO LARGO DEL CURSO	6
TAREAS QUE SE DEJARON	11
EJERCICIOS VISTOS EN CLASE	14
Evaluación Diagnóstica.....	14
FlowLayout + Eventos 1	15
ExampleGridLayout.java+ Eventos2	16
GridBagLayoutExample.java	17
BorderLayoutExample.java	18
EjemploSetBounds.java+Eventos_3	19
Componente Lista (JList).....	20
Más Componentes y Otros Eventos.....	21
Clase para manejar archivos	23
Ejemplo JTable	23
Clases POO.....	24
Implementación con una interfaz gráfica de usuario.....	24
SOLUCIÓN EJERCICIOS DE TAREA	25
ColoresRGB.java.....	25

Solucion de ExampleGridLayout.java	26
DESCRIPCIÓN DEL PROYECTO FINAL	27
REFERENCIAS DEL CURSO	28

PRESENTACION

Esta actividad extraescolar apoya a los alumnos de la carrera de ingeniería en sistemas computacionales (siendo estos en su totalidad en la actividad), en la implementación de aplicaciones computacionales para solucionar problemas de diversos contextos, integrando diferentes tecnologías, plataformas, por medio del desarrollo de software utilizando programación que soporte interfaz gráfica de usuario.

Para el logro de los objetivos es necesario que el estudiante tenga competencias previas en cuanto a paradigmas de programación, el uso de metodologías para la solución de problemas mediante la construcción de algoritmos utilizando un lenguaje de programación orientada a objetos.

INTENCION

Se organizan las actividades a lo largo del semestre, para el estudio de los componentes de una interfaz gráfica de usuario (GUI) y con ayuda de un entorno de desarrollo integrado (IDE) lograr que el estudiante aprenda a utilizar los componentes gráficos de una manera más ágil.

JUSTIFICACION

Al observar que los estudiantes que están inscritos en la actividad son en su totalidad de la carrera de sistemas computacionales, siendo en su totalidad alumnos con conocimientos básicos en programación, entonces se le sugirió algunos temas que podían aprender, siendo el de interfaces gráficas de usuario el que escogieron, es por esta razón que el curso se centró en esto.

COMPETENCIAS ESPECÍFICAS:

- Desarrolla programas para interactuar con el usuario de una manera amigable, utilizando GUI (Interfaz Gráfica de Usuario) manipuladas a través de eventos.
- Analiza e implementa algoritmos orientado a la resolución de problemas, con el fin de hacer funcionales sus aplicaciones.
- Conoce y aplica el paradigma orientado a objetos y secuencial para resolver los problemas planteados

COMPETENCIAS GENÉRICAS:

- Capacidad de análisis y síntesis.
- Capacidad de aplicar los conocimientos en la práctica
- Habilidades en el uso de las tecnologías de la información y de la comunicación.

MATERIAL Y EQUIPO NECESARIO PARA LAS CLASES

- Equipo de cómputo: Laptop o PC.
- Proyector Multimedia.
- Grupo en WhatsApp.
- Servicios como Google Drive para subir sus evidencias

RUBRICA DE EVALUACION PARA LAS ACTIVIDADES SOLICITADAS

	100%	75%	50%	0%
Paradigma	<p>Aplica herramientas de programación. Tipos de datos, definiciones literales, constantes, variables, identificadoras, parámetros, operadores y salida de datos.</p> <p>El paradigma resuelve correctamente el problema planteado, y aplicando los datos de prueba dados, obtener los resultados esperados.</p>	<p>Aplica las dos terceras partes de las herramientas de programación planteadas y el programa resuelve correctamente el problema planteado, y aplicando los datos de prueba dados,</p>	<p>Aplica la mitad de las herramientas de programación planteadas en la unidad.</p> <p>Y el programa resuelve el problema planteado.</p>	<p>El programa no aplica el paradigma.</p>
Documentación (15%)	<p>El archivo fuente (.java) contiene una explicación con lo que realiza el código. Así como su nombre, numero de control y carrera.</p>	<p>Solo el archivo (.java) esta escasamente explicado y con los datos del estudiante</p>	<p>El repositorio solo Presenta el código con la información del estudiante.</p>	<p>El archivo fuente (.java) está sin ningún comentario.</p>

CUADRO DE ACTIVIDADES REALIZADAS A LO LARGO DEL CURSO

Clase	Fecha	Tema	Actividades Realizadas	Tarea
01 y 02	06/Mar/2020	Presentación e Introducción	Realicé una plática de bienvenida a los alumnos. Y los lineamientos de la actividad. Explique con qué herramientas se iban a trabajar. Aplique una evaluación Diagnóstica sobre algoritmos de programación.	Instalar un framework para programar en java, se sugirió Netbeans IDE 8.2
03	13/Mar/20	Introducción a las interfaces graficas de usuario. Y uso de la API de Java.	Por medio de una presentación electrónica, les explique acerca de: ¿Qué es una interfaz Gráfica de Usuario? ¿Cómo se compone una GUI?	
04	13/Mar/20	Componentes de una GUI en Java	Mediante una presentación electrónica y el Framework Netbeans IDE les presente los componentes básicos de una GUI, así de como insertarlos ya en una pequeña aplicación para observar su comportamiento. (ver aplicación)	Realizar un bosquejo en dibujo o buscar una imagen de una GUI básica en java que quisieran replicar.
05	20/ Mar/20	Administradores de Distribuciones (FlowLayout)	Mediante el uso del IDE Netbeans les explique a los estudiantes la primera forma de	Realizar su aplicación solicitada en la sesión anterior, usando el gestor

			como acomodar los componentes gráficos de una GUI dentro de un Contenedor. Con una aplicación sencilla.	de distribución visto en la clase.
06	20/Mar/20	Eventos Aplicaciones Graficas I	Use el IDE Netbeans para mostrarles cómo hacer que las interfaces graficas que creen tengan funcionalidad. Usando los temas que se ven en la asignatura de Programación Orientada a Objetos (uso de interfaces).	
			<ul style="list-style-type: none"> • Debido a la emergencia sanitaria que se presentó, establecí una nueva forma de trabajar, la cual consistió en preparar el material de los temas que se iban a ver en clase, con su respectivo código fuente comentado lo más posible para su entendimiento. • Se Implementó la herramienta de google drive para subir sus evidencias y un chat para atender sus dudas en base a los temas que se van desarrollando. 	
07	27/Mar/20	Administradores de Distribuciones II (GridLayout)	Mediante el uso del IDE Netbeans les explique a los estudiantes una segunda forma de como acomodar los componentes gráficos de una GUI dentro de un Contenedor. Con una aplicación sencilla, la cual consiste en una pseudo calculadora.	<p>Se les entrego una aplicación completa con el gestor de distribución visto en clase, el cual implementa arreglos de componentes.</p> <p>Se les solicito que implementaran ese código sin arreglos, y a su vez implementando la manera de realizar eventos como los vistos en clase.</p>
08	27/Mar/20	Eventos Aplicaciones Graficas II	Junto con la aplicación vista en la misma clase les explique otra forma de llamar a eventos para hacer funcionales sus	

			interfaces gráficas, la cual consiste en hacer uso de clases anónimas. Para lo cual les explique a grandes rasgos que son, como se implementan.	
09	03/Abr/20	Administradores de Distribuciones III (BorderLayout)	Elabore una aplicación sencilla en java donde explique lo más detallado posible, acerca de esta nueva forma de acomodar componentes de una GUI en un contenedor.	Se les mando un documento donde vienen indicaciones acerca de la actividad, la cual consistió en implementar en una interfaz gráfica que pinte un panel con los colores RGB para lo cual deben implementar los gestores de distribución que se les mando en la clase.
10	03/Abr/20	Administradores de Distribuciones IV (GridBagLayout)	Elabore una aplicación sencilla en java donde explique lo más detallado posible, acerca de esta nueva forma de acomodar componentes de una GUI en un contenedor.	
11	24/Abr/20	Administradores de Distribuciones V (Layout Nulo)	Elabore un pequeño formulario de datos de una persona donde se ocupe este gestor, de igual forma se comentó lo más posible la aplicación para que su entendimiento sea el ideal.	
12	24/Abr/20	Eventos con otros componentes gráficos	Elabore un código con los componentes: JSlider, JSpinner, JList, JComboBox Los cuales involucran otros	

				tipos de eventos para su funcionamiento. Y se les documento para su entendimiento	
13	08/May/20	Programación Orientada Objetos	a	Mediante el uso del IDE Netbeans les di una clase sobre conceptos básicos de Programación orientada a objetos, como: Diseño de clases Encapsulamiento Herencia, Polimorfismo	
14	08/May/20	Programacion Orientada Objetos Interfaces Graficas usuario	e de	Usando el IDE Netbeans, les explique cómo unir sus clases que diseñen en POO con sus Interfaces gráficas. Con el fin de hacer más optimas sus aplicaciones que programen.	Diseñar al menos 2 clases usando los conceptos de programación orientada a objetos vistos en esta sesión.
15	15/May/20	Componente JTable		Con ayuda de Netbeans, les explique cómo funciona el componente JTable y sus métodos para poderla usar en sus interfaces graficas	Investigar más implementaciones que deseen para este componente gráfico. <u>(ver ejemplo JTable)</u>
16	15/May/20	La clase ArrayList		Con ayuda de la API de java revise el uso de la clase ArrayList y sus métodos para el manejo de datos y su implementación en sus interfaces graficas de usuario. Y se les mostro un <u>ejemplo</u> .	Investigar más ejemplos en la API de java sobre la clase ArrayList.

17	22/May/20	Manejo Archivos Excepciones de y	Con ayuda de la API de java y Netbeans se revisaron dos clases para el manejo de archivos con el fin de cuando diseñen sus interfaces la información que almacenen se quede guardada. Y se les mostro un <u>ejemplo.</u>	Pensar un método que cree un archivo en base a la información que recupere de los componentes gráficos.
18	22/May/20	Proyecto Integración de	Se les mando un archivo con indicaciones de como diseñar su proyecto integrador. Y el uso de sus clases que diseñaron en la sesión anterior.	Realizar un proyecto integrador, con lo visto en todo el curso. Manejo de interfaces graficas con el uso de POO y con el manejo de archivos para almacenar la información ingresen.
19	29/May/20	Revisión Proyecto de	Se hizo revisión de proyectos y trabajos atrasados que tuvieran para que se pusieran al corriente con sus actividades	
20	29/May/20	Liberación	Mediante el uso de la plataforma Google Drive puse una liga con la que los que acreditaron pusieran la palabra “correcto” si es que sus datos son correctos para la tramitación de constancias.	

TAREAS QUE SE DEJARON

Clase 01 y 02

Fecha: 06/Marzo/2020

Actividad: Descargar un marco de Trabajo (FrameWork) para programar interfaces graficas en Java, se sugirió NetBeans IDE 8.2

Clase 03 y 04

Fecha: 13/Marzo/2020

Actividad: Buscar un ejemplo de interfaz gráfica de usuario en java, sencilla para poder realizarla en el lenguaje.

Clase 05 y 06

Fecha: 20/Marzo/2020

Actividad: Codificar su interfaz gráfica que se les dejo en la sesión anterior y con lo aprendido esta clase.

Clase 07 y 08

Fecha: 27/Marzo/2020

Actividad: El siguiente código: **ExampleGridLayout.java** maneja un arreglo de JButton's, deberán modificar el código sin arreglos y usando clases anónimas para los eventos, cuando se presione el botón debe mostrar en un JOptionPane el número que se esta presionando. (Ver solución)

Clase 09 y 10

Fecha: 03/Abril/2020

Actividad: Realizar la siguiente interfaz gráfica ([ir a imagen](#)) la cual consiste en un pequeño visor de los colores RGB, se deberá replicar la interfaz gráfica usando los dos administradores de distribución (BorderLayout y GridBagLayout).

Ver Solucion.

Características y ayuda:

- El Frame tendrá distribución BorderLayout Tendrá dos Paneles
- El panel de lado izquierdo tendrá distribución GridBagLayout
- El panel derecho será normal pueden dejarlo sin distribución
- El panel con gridbaglayout ira en la posición EAST
- Y el panel derecho será en la posición CENTER
- Crearan una etiqueta con el titulo de paleta de colores y ese ira en la parte norte del frame
- Para el Evento del botón calcular: en el método actionPerformed, puedes usar una clase anónima o usando la interface de ActionListener

- Dentro del método actionPerformed irá lo siguiente:

```

calcular.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        int r = Integer.parseInt(dRojo.getText().trim());
        int g = Integer.parseInt(dVerde.getText().trim());
        int b = Integer.parseInt(dAzul.getText().trim());
        System.out.println(dRojo + " " + dVerde + " " + dAzul);
        if ((r <= 255 && g <= 255 && b <= 255) && (r >= 0 && g >= 0 && b >= 0)) {
            Color color = new Color(r, g, b);
            panel_center.setBackground(color);
        } else {
            JOptionPane.showMessageDialog(null, "Fuera de Rango");
        }
    }
});
```

Interfaz a Replicar:



Clase 13 y 14

Fecha: 27/Abril/2020

Actividad: Diseñar un conjunto de clases en la cual ocupen al menos dos clases (una madre y una hija) para la siguiente sesión, con sus respectivos métodos set y get

Clase 15 y 16

Fecha: 27/Abril/2020

Actividad: Investigar más implementaciones que deseen para este componente gráfico. Y más ejemplos en la API de java sobre la clase ArrayList.

Clase 17 y 18

Fecha: 27/Abril/2020

Actividad: Realizar un proyecto integrador, con lo visto en todo el curso. Manejo de interfaces graficas con el uso de POO y con el manejo de archivos para almacenar la información ingresen. ([Ir a especificaciones del proyecto](#))

EJERCICIOS VISTOS EN CLASE

Evaluación Diagnóstica

1. Diseñar un algoritmo que sea capaz de saber si un número es primo o no.

Sol:

```
public class Ejercicio1{
    public static void main(String args[]){
        int numero=9;
        int contador=0;
        for (int i=1;i<=numero;i++){
            if(numero%2==0){
                contador++;
            }
        }
        if(contador==2){
            System.out.println(numero+" es primo");
        }else{
            System.out.println(numero+" no es primo");
        }
    }
}
```

2. Diseñar una clase persona con sus métodos set y get respectivos.

```
public class Persona {
    int edad;
    String nombre;
    char sexo;
    public int getEdad() {
        return edad;
    }

    public void setEdad(int edad) {
        this.edad = edad;
    }

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public char getSexo() {
        return sexo;
    }

    public void setSexo(char sexo) {
        this.sexo = sexo;
    }
}
```

Aplicación en Java para la sesión 4

FlowLayout + Eventos 1 Esta es una aplicación donde se usan los componentes más básicos de java

```
* La clase IMC (Indice de Masa Corporal)
* consiste en una pequeña aplicación que usa
* distribución FlowLayout para acomodar los componentes
* que están dentro de ella.
* Clases que se usaron:
*     JFrame
*     JLabel
*     JButton
*     JSpinner
*
*/
/** *COSAS NUEVAS
* Hacemos uso de un tema que se ve en P00 que son interfaces:
* Una interfaz es una clase
*/

```

```
import java.awt.FlowLayout; //Importamos la librería de java para usar la distribución
import java.awt.event.ActionEvent; //Nueva librería para manejo de eventos de Acción
import java.awt.event.ActionListener; //Nueva librería a usar, para escuchador de eventos
import javax.swing.*; //Importamos todos los componentes de javax.swing
```

```
public class IMC extends JFrame implements ActionListener //Activamos la interfaz con implements actionPerformed y tenemos que reescribir los métodos
//que vamos a usar de esa interfaz, los métodos se sobreescreiben con @Override
```

```
{
    //Creamos la clase y le indicamos que hereda de la clase JFrame
```

```
//Declaramos las variables a usar, las variables de tipo JLabel(etiqueta)
//Inician con una e minuscula, pero pueden ponerle
```

```
private JLabel encabezado;
private JLabel eEdad;
private JLabel eCalculo;
private JLabel eEstatura;
private JLabel ePeso;
private JLabel eMensaje;
private JLabel eMensaje2;
private JLabel eCinCad;
private JLabel eCintura;
private JLabel eCadera;
private JLabel eSexo;
private JLabel foto;
```

```
/*
 * @CajadeTexto
 */

```

```
private JTextField imagen;
private JTextField dEdad;
private JTextField dPeso;
private JTextField dCadera;
private JTextField dCintura;
/*
 * @RadioButton
 */

```

```
private JRadioButton sMas, sFem;
/*
 * @Botones
 */

```

```
private JButton calcularIMC;
private JButton calcularICC;
private JButton borrar;
//Spinner para la edad
private JSpinner dEstatura;
```

```
//Administrador del grupo de botones
private ButtonGroup grupoBotones;
```

```
public IMC() { //Constructor
    init(); //Iniciamos el método init en el constructor
}
```

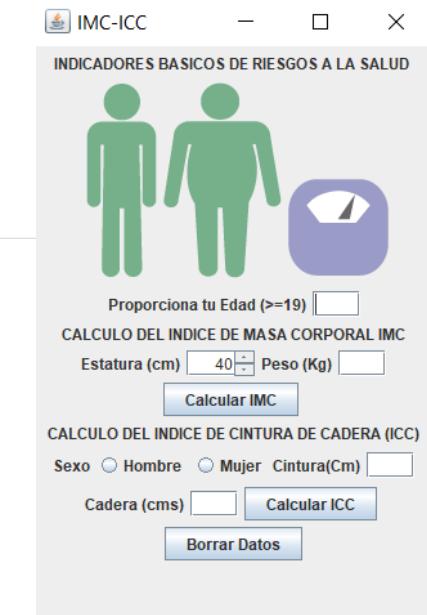
```
private void init() {
    /**
     * Como heredamos de la clase JFrame, podemos usar sus métodos en esta
     * clase entonces lo indicamos con t1 this, si no lo ponemos no hay
     * mucho problema
     */
    this.setTitle("IMC-ICC"); //Establecer el título de la ventana
    this.setSize(330, 500); //Establecer sus dimensiones
    this.setLayout(new FlowLayout()); //Establecemos una distribución FlowLayout
    this.setLayout(new FlowLayout()); //Vamos a guardar los elementos en un this
    this.setSize(330, 500);
    //Y ese this tendrá también distribución FlowLayout

    //Inicializamos los componentes

    encabezado = new JLabel("INDICADORES BASICOS DE RIESGOS A LA SALUD ");
    foto = new JLabel(); //Inicializamos una etiqueta vacía, accederá a la imagen
    foto.setIcon(new ImageIcon("bascula.png")); //Establecemos la imagen, mediante su nombre y la extensión
    eEdad = new JLabel("Proporciona tu Edad (>=19)");
    dEdad = new JTextField(3); //Inicializamos la caja de texto que pide la edad
    eCalculo = new JLabel("CALCULO DEL INDICE DE MASA CORPORAL IMC");
    eEstatura = new JLabel("Estatura (cm)");
    //Instanciamos el JSpinner para la edad
    //JSpinner maneja varios formatos para la presentación de datos, uno de ellos es el numérico
    //entonces como parámetro del constructor de JSpinner instanciamos un objeto de
    //SpinnerNumberModel, el cual pide como parámetros:
    //SpinnerNumberModel, el cual pide como parámetros:
    //Valor donde inicia cuando se inicia la interfaz
    //Valor mínimo, valor máximo, y de cuánto en cuánto va a ir incrementando
    dEstatura = new JSpinner(new SpinnerNumberModel(40, 40, 200, 1));
    ePeso = new JLabel("Peso (Kg)");
    dPeso = new JTextField(3); //Caja de texto para el peso
    calcularIMC = new JButton("Calcular IMC");
    eMensaje = new JLabel(); //Esta etiqueta la vamos a usar la siguiente clase, con un evento
    eCinCad = new JLabel("CALCULO DEL INDICE DE CINTURA DE CADERA (ICC)");
    eCintura = new JLabel("Cintura(Cm)");
    dCintura = new JTextField(3); //Dato de la cintura
    eCadera = new JLabel("Cadera (cms)");
    dCadera = new JTextField(3); //Dato de la cadera
    eSexo = new JLabel("Sexo");

    grupoBotones = new ButtonGroup();
    /**
     * Crea un grupo de botones
     */
    sMas = new JRadioButton("Hombre"); //Iniciamos los botones para el sexo
    grupoBotones.add(sMas); //Agregamos al administrador de botones los radiobutton
    sFem = new JRadioButton("Mujer"); //Radiobutton para el sexo femenino
    grupoBotones.add(sFem); //Añadimos al administrador de botones el radiobutton
    calcularICC = new JButton("Calcular ICC");
    eMensaje2 = new JLabel(); //Etiqueta que se usará con eventos después
    borrar = new JButton("Borrar Datos"); //Iniciamos el botón

    //Como estamos usando distribución flowlayout, ingresamos todo en el orden
```



ExampleGridLayout.java+ Eventos2

```
*****GRIDLAYOUT*****
*Gestor de distribución de contenido que organiza los componentes de un
*contenedor en una cuadrícula rectangular de tamaño nxm. El contenedor se divide en
*rectángulos de igual tamaño y se coloca un componente en cada rectángulo.
* puede aplicarse a un JFrame o JPanel en ambos casos el metodo es el mismo
* setLayout(new GridLayout(filas,columnas));
*****CLASES ANONIMAS*****
* En pocas palabras, el uso de una clase anónima te permite crear un objeto que
* implementa una interfaz en particular y poder usarlo libremente sin la molestia
* de tener que definir explícitamente una clase.
* Esta es una alternativa a la implementación de la interface ActionListener en
* en nuestra clase y lo implementamos con el siguiente código
* el asistente de netBeans te puede generar el código
* componente.addActionListener(new ActionListener(){
*     @Override el override indica que es un método sobreescrito
*     public void actionPerformed(ActionEvent e) {
*         aca va todo nuestro código que deseemos poner todo el código que deseemos
*         una buena práctica es poner únicamente el método que haga lo que le indiquemos
*         para no sobrecargar esta parte, por ejemplo el método limpiar(); y ya
*     }
* });
*****este aplicacion consiste consiste en un Frame con 9 botones y al oprimirlos
* despliega un JOptionPane con el valor que tiene como msj
* Para esta interfaz se usaron solo JButton's entonces para no declarar
* tantas variables usamos un arreglo de JButton's
* Y en conjunto con la distribución GridLayout los ponemos en esa forma
* de cuadrícula
*
*****Actividad*****
* En vez de usar un arreglo de JButton's deben hacer 9 variables de JButton
* y agregarlos al JFrame además usando las clases anónimas para los eventos y
* esto hacer que cuando se presione un botón aparezca en un JOptionPane el
* número que se presiona.
* tendrán que hacer 9 veces uso de una clase anónima
* bt1.addActionListener(...);
* todo es dentro del método init();
* y ese se pondrá en el constructor de la clase como se ha venido trabajando
* si tienen una duda pregúnten.
* Pueden Ponerle colores a los botones como quieran o si gustan dejarlo así
*****import java.awt.Color;//importamos la clase Color
import java.awt.GridLayout;//Importamos la clase GridLayout
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.*;
```

```
public class EjemploGridLayout extends JFrame {//Clase hereda de JFrame

    private GridLayout gl;//Declaramos un objeto de tipo GridLayout

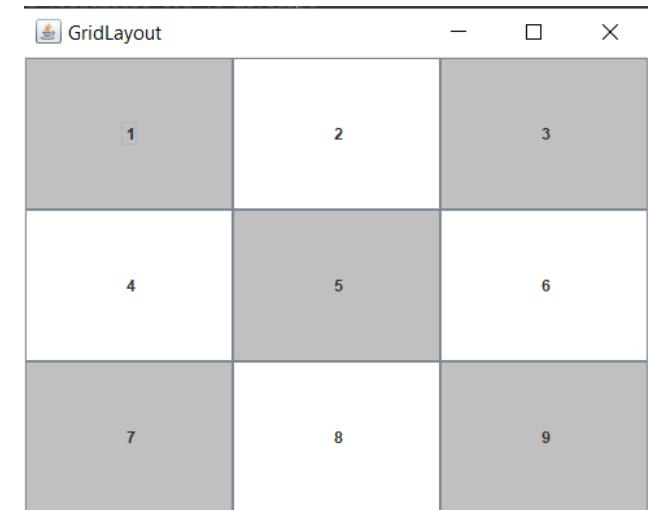
    public EjemploGridLayout() {//constructor
        init();//iniciamos el método
    }

    private void init() {
        int filas = 3;
        int columnas = 3;
        //GridLayout nos pide como parámetros filas y columnas que tendrá
        gl = new GridLayout(filas, columnas);

        this.setLayout(gl);//establecemos la distribución GridLayout al frame

        JButton arreglo_botones[] = new JButton[9];//Declaramos un arreglo de botones
        for (int i = 0; i < arreglo_botones.length; i++) {
            //Creamos un for para recorrer nuestro arreglo de botones
            arreglo_botones[i] = new JButton(String.format("%d", i + 1));
            //le decimos que coloque como texto el valor de i+1 para que muestre del 1 al 9
            this.add(arreglo_botones[i]);
            //agregamos el botón al frame
            arreglo_botones[i].setHorizontalAlignment(SwingConstants.CENTER);//pone el texto de los botones en el centro
            arreglo_botones[i].addActionListener(new ActionListener() {
                @Override
                public void actionPerformed(ActionEvent e) {
                    JButton boton = (JButton) e.getSource();
                    if (boton == arreglo_botones[0]) {
                        JOptionPane.showMessageDialog(null, arreglo_botones[0].getText());
                    }if (boton == arreglo_botones[1]) {
                        JOptionPane.showMessageDialog(null, arreglo_botones[1].getText());
                    }if (boton == arreglo_botones[2]) {
                        JOptionPane.showMessageDialog(null, arreglo_botones[2].getText());
                    }if (boton == arreglo_botones[3]) {
                        JOptionPane.showMessageDialog(null, arreglo_botones[3].getText());
                    }if (boton == arreglo_botones[4]) {
                        JOptionPane.showMessageDialog(null, arreglo_botones[4].getText());
                    }if (boton == arreglo_botones[5]) {
                        JOptionPane.showMessageDialog(null, arreglo_botones[5].getText());
                    }if (boton == arreglo_botones[6]) {
                        JOptionPane.showMessageDialog(null, arreglo_botones[6].getText());
                    }if (boton == arreglo_botones[7]) {
                        JOptionPane.showMessageDialog(null, arreglo_botones[7].getText());
                    }if (boton == arreglo_botones[8]) {
                        JOptionPane.showMessageDialog(null, arreglo_botones[8].getText());
                    }
                }
            });
        }
        //hace que el texto se aline al centro, para otros lados vean la API de Java
        //Esto es opcional, para darle un color.
        //Si el módulo de i/2 es 0 pondrá los botones Gris Claro si no BLANCOS
        if (i % 2 == 0) {
            arreglo_botones[i].setBackground(Color.LIGHT_GRAY);
            //el método setBackground se usa para establecer color a un componente
            //Pide como parámetros un objeto de la clase Color, con el color que
            //se desea. La clase Color tiene variables estáticas que pone colores por defecto
            //Debemos importarla, import java.awt.Color;
        } else {
            arreglo_botones[i].setBackground(Color.WHITE);//pone de blanco los botones
        }
    }
    this.setTitle("GridLayout");
    this.setSize(500, 400);//establece el tamaño del frame
    this.setVisible(true);//lo hace visible
    this.setLocationRelativeTo(null);//método que hace que aparezca centrado en la pantalla
    this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);//cierra el programa
}

public static void main(String[] args) {
    new EjemploGridLayout();//prueba
}
```



GridLayoutExample.java

```
*****  
*GridLayout coloca los componentes en una cuadrícula de filas y columnas,  
*lo que permite que los componentes especificados abarquen varias filas o columnas.  
*No todas las filas necesariamente tienen la misma altura. Del mismo modo, no todas  
*las columnas tienen necesariamente el mismo ancho. Esencialmente, GridLayout coloca  
*los componentes en rectángulos (celadas) en una cuadrícula y luego usa los tamaños  
*preferidos de los componentes para determinar qué tan grandes deberían ser las celdas.  
*El detalle para esta distribución es necesario que se declare un objeto de  
*GridBagConstraints el cual tiene la función de que este hará posible que coloquemos  
*los componentes en la cuadrícula. En esta distribución no es necesaria declarar el tamaño  
*de la cuadrícula como en gridlayout, aca conforme necesitemos vamos metiendo  
*la numeración empieza en la coordenada (0,0), lo basico que tenemos que declarar  
*al momento de colocar nuestro componente es la siguiente  
* gbc.gridx=coordenada x  
* gbc.gridy=coordenada y  
* panel o frame.add(componente(boton,etiqueta,etc),gbc);  
* eso es lo basico que necesitamos para colocar nuestros componentes  
* conforme necesitemos algo en especifico necesitamos los metodos  
* para mas informacion consultar  
*  
* @http://dar10comyр.blogspot.com/2014/11/gestores-de-diseno-gridlayout.html  
* aca vienen los metodos basicos que se ocupan para dar formato  
*****  
*/  
  
/* Algunos metodos  
*gbc.fill=Este campo se usa cuando el área de visualización del componente es mayor  
*el tamaño requerido del componente.  
* gbc.fill=No necesita una variable entera, GridBagConstraints ofrece variables estáticas como  
* GridBagConstraints.WEST,EAST,NORTH,SOUTH,CENTER,HORIZONTAL,VERTICAL  
* GridBagConstraints.BOTH Hacer que el componente llene su área de visualización por completo.  
* gbc.weightx=Especifica cómo distribuir el espacio horizontal extra. Reciben dobles como parametros  
* gbc.weighty=Especifica cómo distribuir el espacio vertical extra.  
* gbc.gridwidth=sirve para indicar cuantas celdas ocupara si mi componente ocupa mas celdas hacia lo ancho  
* gbc.gridheight= indica cuantas celdas ocupo si exiendo mi componente hacia lo alto  
* estos dos ultimos metodos tienen el valor por default de 1  
* si llamamos estos metodos si ya no necesitamos mas sus datos tenemos que volverlos a dejar  
* en su valor original, ya que puede que los otros componentes no se ajusten |como se desea  
*****  
import java.awt.Color;  
import java.awt.Dimension;  
import java.awt.GridLayout;  
import java.awt.GridBagConstraints;  
import java.awt.Image;  
import java.awt.Insets;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
import java.io.InputStream;  
import java.util.logging.Level;  
import java.util.logging.Logger;  
import javax.imageio.ImageIO;  
import javax.swing.*;  
  
/*En esta aplicación lo importante es como se acomodan los componentes, adicionalmente  
* viene como ingresar una imagen y ponerla en un label la cual hace una escala para que  
* se ajuste al tamaño del label.  
*/  
  
public class GridLayoutExample extends JFrame implements ActionListener {  
  
    GridBagConstraints gbc;  
    GridLayout gbl;  
    private JLabel eTitulo;  
    private JLabel eNombre;  
    private JLabel eApellidos;  
    private JLabel eDireccion;  
    private JPanel imagen;  
    private JButton guardar;  
    private JTextField dNombre;  
    private JTextField dApellido;  
    private JTextField dDireccion;  
    private JLabel eImagen;  
    private JButton abrir;  
    private FileInputStream fis; //Aca se almacena el flujo del archivo  
    int longitud_bytes;//longitud se bytes del archivo leido
```

```
public GridLayoutExample() {  
    init(); //Constructor  
}  
  
private void init() {  
    gbl = new GridLayout(); //Instanciamos el objeto para asignar la distribución  
    gbc = new GridBagConstraints(); //Instanciamos para poder poner nuestros componentes  
    this.setLayout(gbl); //Establecemos la distribución  
    this.setSize(new Dimension(500, 400)); //Establecemos el tamaño del frame  
    this.setLocationRelativeTo(null); //Que se ponga a media pantalla  
    gbc.insets = new Insets(10, 10, 10, 10); //Esto hace que los componentes no estén tan pegadas  
    //Sirve para especificar cuantos pixeles arriba abajo izquierda y derecha va a estar  
  
    eTitulo = new JLabel("FORMULARIO PARA DAR DE ALTA A PERSONAS");  
    gbc.gridx = 0; //Se pone en la fila 0  
    gbc.gridy = 0; //columna 0  
    gbc.gridwidth = 4; //que abarque 4 celdas  
    gbc.weightx = 1.0; //  
    gbc.fill = GridBagConstraints.CENTER; //que el label se ponga al centro  
    this.add(eTitulo, gbc);  
  
    eNombre = new JLabel("Nombre");  
    gbc.gridx = 0; //fila 0  
    gbc.gridy = 1; //columna 1  
    gbc.gridwidth = 1; //ajustamos la etiqueta a su valor original para que no se descuadre  
    gbc.weightx = 0.0;  
    this.add(eNombre, gbc);  
    dNombre = new JTextField(5);  
    gbc.gridx = 1;  
    gbc.gridy = 1;  
    gbc.weightx = 1.0;  
    gbc.fill = GridBagConstraints.HORIZONTAL; //hace que la cajita de texto se alargue  
    gbc.gridwidth=1; //que ocupe una celda mas a lo ancho  
    this.add(dNombre, gbc);  
  
    eApellidos = new JLabel("Apellidos");  
    gbc.gridx = 0;  
    gbc.gridy = 2;  
    gbc.gridwidth = 1;  
    gbc.weightx = 0.0;  
    this.add(eApellidos, gbc);  
  
    //Esto es lo basico que se hace para posicionar el componente  
    //y hasta aca se les va a pedir  
    dApellido = new JTextField(5);  
    gbc.gridx = 1;  
    gbc.gridy = 2;  
    this.add(dApellido, gbc);  
  
    eDireccion = new JLabel("Dirección");  
    gbc.gridx = 0;  
    gbc.gridy = 3;  
    this.add(eDireccion, gbc);  
    dDireccion = new JTextField(5);  
    gbc.gridx = 1;  
    gbc.gridy = 3;  
    this.add(dDireccion, gbc);  
  
    guardar = new JButton("Guardar");  
    gbc.gridx = 2;  
    gbc.gridy = 2;  
    gbc.fill = GridBagConstraints.WEST;  
    this.add(guardar, gbc);  
  
    abrir = new JButton("Inserta Imagen");  
    gbc.gridx = 2;  
    gbc.gridy = 1;  
    gbc.fill = GridBagConstraints.CENTER;  
    this.add(abrir, gbc);
```

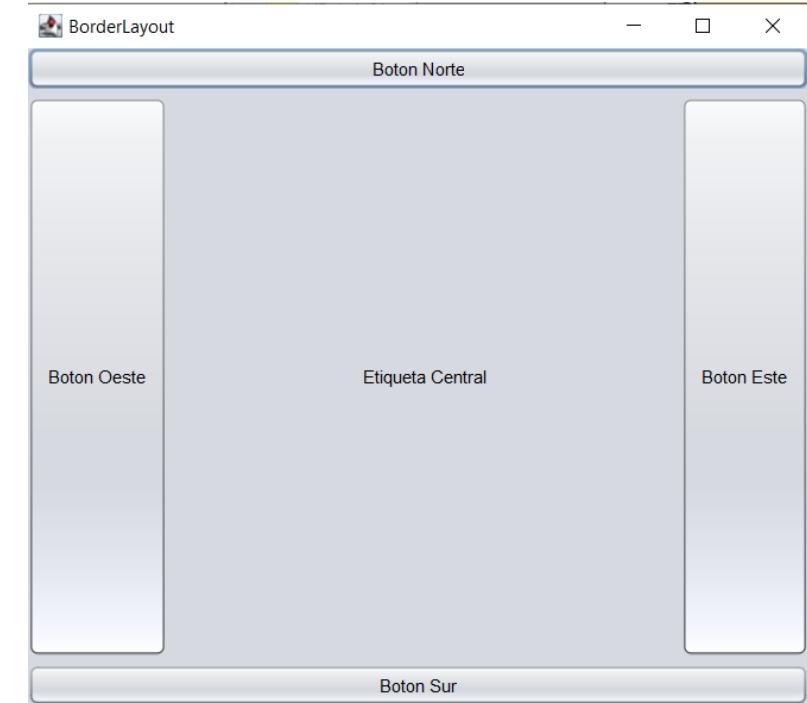
```
//Esto es opcional para la imagen, para ponerla se necesita un label o un button  
eImagen = new JLabel();  
gbc.gridx = 1; //4  
gbc.gridy = 4; //2  
gbc.weightx = 1.0;  
gbc.weighty = 1.0;  
gbc.gridwidth = 1; //va a abarcar una celda mas a lo ancho  
gbc.gridheight = 1; //una celda mas a lo alto  
gbc.fill = GridBagConstraints.BOTH; //hazlo proporcional  
this.add(eImagen, gbc);  
  
this.setVisible(true);  
this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
abrir.addActionListener(this);  
}  
  
public static void main(String[] args) {  
    //Esto hace que al abrir el JFileChooser nos aparezca con la apariencia de windows  
    try {  
        UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());  
    } catch (ClassNotFoundException ex) {  
        Logger.getLogger(GridLayoutExample.class.getName()).log(Level.SEVERE, null, ex);  
    } catch (InstantiationException ex) {  
        Logger.getLogger(GridLayoutExample.class.getName()).log(Level.SEVERE, null, ex);  
    } catch (IllegalAccessException ex) {  
        Logger.getLogger(GridLayoutExample.class.getName()).log(Level.SEVERE, null, ex);  
    } catch (UnsupportedLookAndFeelException ex) {  
        Logger.getLogger(GridLayoutExample.class.getName()).log(Level.SEVERE, null, ex);  
    }  
    new GridLayoutExample();  
}  
  
@Override  
public void actionPerformed(ActionEvent e) {  
    //aca hara los ajustes a la imagen que se obtenga  
    JButton evento = (JButton) e.getSource();  
    if (evento == abrir) {  
        JFileChooser jfc = new JFileChooser(); //el objeto filechooser abre una biblioteca de archivos  
        jfc.setFileSelectionMode(JFileChooser.FILES_ONLY); //hace que solo se seleccione uno  
        int estado = jfc.showOpenDialog(null);  
        if (estado == JFileChooser.APPROVE_OPTION) {  
            try {  
                fis = new FileInputStream(fis.getSelectedFile()); //obten la ruta del archivo  
                longitud_bytes = (int) fis.getSelectedFile().length(); //obten el tamaño  
                Image icono = ImageIO.read(fis.getSelectedFile());  
                eImagen.setIcon(new ImageIcon(icono)); //establece al label la imagen con el metodo SetIcon()  
                eImagen.updateUI(); //haz una actualización  
                longitud_bytes = longitud_bytes;  
            } catch (Exception ee) {  
            }  
        }  
    }  
}
```



BorderLayoutExample.java

```
Cerrar *****BORDERLAYOUT*****  
*Distribucion BorderLayout:  
*Organiza los componentes en uno de los 5 paneles que corresponde a una de las  
*coordenadas(Norte, Sur, Este, Oeste) y el centro.  
*Basicamente luego de crear el objeto del gestor borde y configurarlo para un contenedor,  
*Como un JPanel a la hora de agregar componentes, el metodo  
*add necesitará 2 parametros:  
* add(Componente, Coordenada)  
-estas coordenadas tienen que ser Strings de clase de BorderLayout:  
NORTH, para el panel superior (Norte)  
WEST, para el panel izquierdo  
CENTER, para el panel central  
EAST, para el panel derecho  
SOUTH, para el panel inferior (Sur)  
*Por ejemplo: miPanel.add(button, BorderLayout.NORTH);  
*de esta manera agregamos el componente al contenedor  
*y en el lugar que el gestor lo divido.  
* Dentro de estas 5 secciones podemos meter cuando componente necesitemos  
* obviamente dentro de un panel, es decir puedo meter en la posicion central  
* un panel con alguna otra distribucion, por ejemplo la GridLayout  
* La FlowLayout, etc, solo es cuestion de tener orden  
* una buena practica es crear metodos que devuelvan JPaneles  
* public JPanel panel(){  
*     JPanel panel=new JPanel();  
*     return panel;  
* }  
*****  
* Esta clase tiene dos constructores para inicializar  
* Constructor vacio BorderLayout():  
* Construye un gestor BorderLayout y sin espacios entre los componentes.  
* Constructor BorderLayout(int hgap, int vgap)  
*     Construye un diseño BorderLayout con los espacios entre los componentes  
*     especificados.  
*     La distancia horizontal se especifica mediante hgap y  
*     la distancia vertical se especifica por vgap.  
*****  
import java.awt.BorderLayout;  
import java.awt.Dimension;  
import java.util.logging.Level;  
import java.util.logging.Logger;  
import javax.swing.*;  
  
public class BorderLayoutExample extends JFrame {  
  
    private JButton boton_NORTE;  
    private JButton boton_SUR;  
    private JButton boton_ESTE;  
    private JButton boton_OESTE;  
    private JLabel etiqueta_Central;  
  
    public BorderLayoutExample(){  
        init();  
    }
```

```
private void init() {  
    this.setSize(new Dimension(540, 480)); //tamaño de la ventana  
    this.setTitle("BorderLayout"); //Titulo de la ventana  
    this.setLocationRelativeTo(null); //para que se ponga al centro de la pantalla  
    this.setLayout(new BorderLayout(5, 5)); //separacion de componentes  
    //Inicializamos nuestros componentes  
    boton_NORTE = new JButton("Boton Norte");  
    boton_SUR = new JButton("Boton Sur");  
    boton_ESTE = new JButton("Boton Este");  
    boton_OESTE = new JButton("Boton Oeste");  
    etiqueta_Central=new JLabel("Etiqueta Central");  
    etiqueta_Central.setHorizontalTextPosition(SwingConstants.CENTER);  
    //ponemos el texto de la etiqueta en el centro  
    //Los añadimos al frame con la posicion de donde los queremos  
    this.add(boton_NORTE, BorderLayout.NORTH);  
    this.add(boton_SUR, BorderLayout.SOUTH);  
    this.add(boton_ESTE, BorderLayout.EAST);  
    this.add(boton_OESTE, BorderLayout.WEST);  
    this.add(etiqueta_Central,BorderLayout.CENTER);  
  
    this.setVisible(true);  
    this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
}  
  
public static void main(String[] args) {  
    //Este Tramo de codigo ocupa un tema que se ve en POO que son las  
    //Excepciones, la cual consiste en realizar una determinada actividad pero si  
    //no se cumple todo ese proceso se disparan las excepciones por medio del catch  
    //por ejemplo se ocupan en division sobre 0 o cuando manejamos archivos  
    //o manejando Hilos (Thread's), entonces el catch atrapa ese error  
    //y hace algo  
    //este codigo permite que tenga la apariencia que tiene con botones mas  
    //redondos etc... se puede quitar solo instanciar su clase  
    try {  
        for (UIManager.LookAndFeelInfo info : UIManager.getInstalledLookAndFeels()) {  
            if ("Nimbus".equals(info.getName())) {  
                UIManager.setLookAndFeel(info.getClassName());  
                break;  
            }  
        }  
    } catch (ClassNotFoundException ex) {  
        Logger.getLogger(BorderLayoutExample.class.getName()).log(Level.SEVERE, null, ex);  
    } catch (InstantiationException ex) {  
        Logger.getLogger(BorderLayoutExample.class.getName()).log(Level.SEVERE, null, ex);  
    } catch (IllegalAccessException ex) {  
        Logger.getLogger(BorderLayoutExample.class.getName()).log(Level.SEVERE, null, ex);  
    } catch (javax.swing.UnsupportedLookAndFeelException ex) {  
        Logger.getLogger(BorderLayoutExample.class.getName()).log(Level.SEVERE, null, ex);  
    }  
    new BorderLayoutExample();  
}
```



EjemploSetBounds.java+Eventos_3

```
/** *****DISTRINUCION NULA, o POSICION ABSOLUTA***** */
* Consiste en dejar en null el metodo setLayout de un frame o un panel
* al hacer esto ningun componente se podra poner en el contenedor en el cual
* estemos trabajando, para poner los componentes, requerimos del metodo
* setBounds(), el cual requiere como parametros, valores enteros y debemos ver
* a la pantalla como un plano cartesiano con coordenadas x y Y solo que no existen
* coordenadas negativas, los parametros que pide el metodo setBounds son:
* posicion en X: viene siendo como en que columna lo pondras, 1 Columna=1 unidad
* posicion en Y: es la fila donde pondre el componente 1 fila= 1Unidad
* largo del componente
* ancho del componente
* las coordenadas son en base al frame o panel donde estan estableciendo
* si tienen una ventana de tamano 600x600 su limite dentro del area visible sera
* hasta la coordenada 600,600 y asi sucesivamente.
* Esta distribucion es algo lenta debido a que hay que calcularle en que parte
* vamos a poner cada componente, y que no queden encimados,
* es util para poner cosas que sabemos que no se van a mover, incluso para los que
* no dominan los layouts encuentran facil usar esta ya que donde digamos que va el
* componente ya no se va a mover, ni haciendo grande la ventana,
* si quieren que no se haga grande la ventana solo al tamaño deseado usar el metodo
* setResizable(false);
***** */

import java.awt.Color;
import java.awt.Dimension;
import java.awt.Image;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.InputStream;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.imageio.ImageIO;
import javax.swing.*;
import javax.swing.border.Border;
import javax.swing.border.EtchedBorder;
import javax.swing.border.TitledBorder;

/**
 * Este es el mismo formulario que el del ejemplo de GridBagLayout, solo cambia
 * la imagen de lugar y los botones, las coordenadas 0,0 empiezan en la esquina
 * superior izquierda del frame o panel donde trabajen, para que no se cofundan,
 * esto son bases para entender el comportamiento de los graficos en java ya que
 * bajo este estilo funcionan. Si les causa Curiosidad chequen la clase Graphics
 * y Graphics2D
 */
public class SetBoundsExample extends JFrame {

    private JLabel eNombre;
    private JLabel eApellidos;
    private JLabel eDireccion;
    private JLabel eImagen;
    private JPanel eEncabezado;
    private JTextField dNombre;
    private JTextField dApellido;
    private JTextField dDireccion;
    private JButton inserta;
    private JButton guardar;
    private JPanel pImagen;

    private FileInputStream fis; //Aca se almacena el flujo del archivo
    int longitud_bytes;//longitud se bytes del archivo leido
}

public SetBoundsExample() {
    init();
}

private void init() {
    this.setLayout(null); //aca le indicamos que es nula la distribucion
    this.setSize(new Dimension(600, 400)); //tamano de la ventana
    //Instanciamos nuestros Objetos

    eEncabezado = new JLabel("Formulario para dar de alta una persona");
    eNombre = new JLabel("Nombre");
    dNombre = new JTextField();
    eApellidos = new JLabel("Apellidos");
    dApellido = new JTextField();
    eDireccion = new JLabel("Direccion");
    dDireccion = new JTextField();
    inserta = new JButton("Insertar Imagen");
    guardar = new JButton("Guardar");

    //Aca le indicamos la posicion de donde vamos a poner cada componente
    eEncabezado.setBounds(100, 0, 300, 20);
    //s.setBounds(200,300,100,30);
    //La etiqueta encabezado se pondra en la posicion 100,0 y tendra un largo de 300 y un ancho de 20
    eNombre.setBounds(20, 40, 100, 20);
    dNombre.setBounds(120, 40, 200, 25);
    eApellidos.setBounds(20, 70, 100, 20);
    dApellido.setBounds(120, 70, 200, 25);
    eDireccion.setBounds(20, 100, 100, 20);
    dDireccion.setBounds(120, 100, 200, 25);
    inserta.setBounds(170, 150, 150, 30);
    guardar.setBounds(220, 190, 100, 30);

    // una buena practica para no sobrecargar de informacion es hacer el programa modular
    //de acuerdo a los paneles principales que trabajemos, entonces
    //Para trabajar el panel donde ira la foto hice un metodo get que nos regresa
    //el panel donde ira la foto, el cual tiene a su vez varias instrucciones
    //a realizar
    //para llamarlo lo igualamos a la variable pImagen ya que son del mismo tipo JPanel
    pImagen = getpanel_foto();
    pImagen.setBounds(350, 40, 200, 200);

    //aca solo agregamos al frame lo que vamos a usar
    //el orden para agregarlos no es muy importante ya que cada uno tiene ya su posicion especificada
    this.add(eEncabezado);
    this.add(pImagen); //aca agregamos el panel
    this.add(eNombre);
    this.add(dNombre);
    this.add(eApellidos);
    this.add(dApellido);
    this.add(eDireccion);
    this.add(dDireccion);
    this.add(inserta);
    this.add(guardar);
    //Nota: de llamar a eventos es mediante el uso de clases internas
    //las cuales las señalamos con el modificador de acceso private,
    //es decir solo se puede acceder dentro de la misma clase y ya
    //Cabe señalar que esta clase privada debe implementar la interface ActionListener para
    //que se reconozca para operar ese evento.
    Eventos_Botones evento = new Eventos_Botones();
    //le asignamos a nuestro metodo que en la clase Eventos_Botones estan los escuchadores para
    //los eventos
    inserta.addActionListener(evento);
    guardar.addActionListener(evento);

    this.setVisible(true);
    this.setLocationRelativeTo(null);
    this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
}

public static void main(String[] args) {
    try {
        UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
    } catch (ClassNotFoundException ex) {
        Logger.getLogger(SetBoundsExample.class.getName()).log(Level.SEVERE, null, ex);
    } catch (InstantiationException ex) {
        Logger.getLogger(SetBoundsExample.class.getName()).log(Level.SEVERE, null, ex);
    } catch (IllegalAccessException ex) {
        Logger.getLogger(SetBoundsExample.class.getName()).log(Level.SEVERE, null, ex);
    } catch (UnsupportedLookAndFeelException ex) {
        Logger.getLogger(SetBoundsExample.class.getName()).log(Level.SEVERE, null, ex);
    }
}
new SetBoundsExample();
```

```
public SetBoundsExample() {
    init();
}

private void init() {
    this.setLayout(null); //aca le indicamos que es nula la distribucion
    this.setSize(new Dimension(600, 400)); //tamano de la ventana
    //Instanciamos nuestros Objetos

    eEncabezado = new JLabel("Formulario para dar de alta una persona");
    eNombre = new JLabel("Nombre");
    dNombre = new JTextField();
    eApellidos = new JLabel("Apellidos");
    dApellido = new JTextField();
    eDireccion = new JLabel("Direccion");
    dDireccion = new JTextField();
    inserta = new JButton("Insertar Imagen");
    guardar = new JButton("Guardar");

    //Aca le indicamos la posicion de donde vamos a poner cada componente
    eEncabezado.setBounds(100, 0, 300, 20);
    //s.setBounds(200,300,100,30);
    //La etiqueta encabezado se pondra en la posicion 100,0 y tendra un largo de 300 y un ancho de 20
    eNombre.setBounds(20, 40, 100, 20);
    dNombre.setBounds(120, 40, 200, 25);
    eApellidos.setBounds(20, 70, 100, 20);
    dApellido.setBounds(120, 70, 200, 25);
    eDireccion.setBounds(20, 100, 100, 20);
    dDireccion.setBounds(120, 100, 200, 25);
    inserta.setBounds(170, 150, 150, 30);
    guardar.setBounds(220, 190, 100, 30);

    // una buena practica para no sobrecargar de informacion es hacer el programa modular
    //de acuerdo a los paneles principales que trabajemos, entonces
    //Para trabajar el panel donde ira la foto hice un metodo get que nos regresa
    //el panel donde ira la foto, el cual tiene a su vez varias instrucciones
    //a realizar
    //para llamarlo lo igualamos a la variable pImagen ya que son del mismo tipo JPanel
    pImagen = getpanel_foto();
    pImagen.setBounds(350, 40, 200, 200);

    //aca solo agregamos al frame lo que vamos a usar
    //el orden para agregarlos no es muy importante ya que cada uno tiene ya su posicion especificada
    this.add(eEncabezado);
    this.add(pImagen); //aca agregamos el panel
    this.add(eNombre);
    this.add(dNombre);
    this.add(eApellidos);
    this.add(dApellido);
    this.add(eDireccion);
    this.add(dDireccion);
    this.add(inserta);
    this.add(guardar);
    //Nota: de llamar a eventos es mediante el uso de clases internas
    //las cuales las señalamos con el modificador de acceso private,
    //es decir solo se puede acceder dentro de la misma clase y ya
    //Cabe señalar que esta clase privada debe implementar la interface ActionListener para
    //que se reconozca para operar ese evento.
    Eventos_Botones evento = new Eventos_Botones();
    //le asignamos a nuestro metodo que en la clase Eventos_Botones estan los escuchadores para
    //los eventos
    inserta.addActionListener(evento);
    guardar.addActionListener(evento);

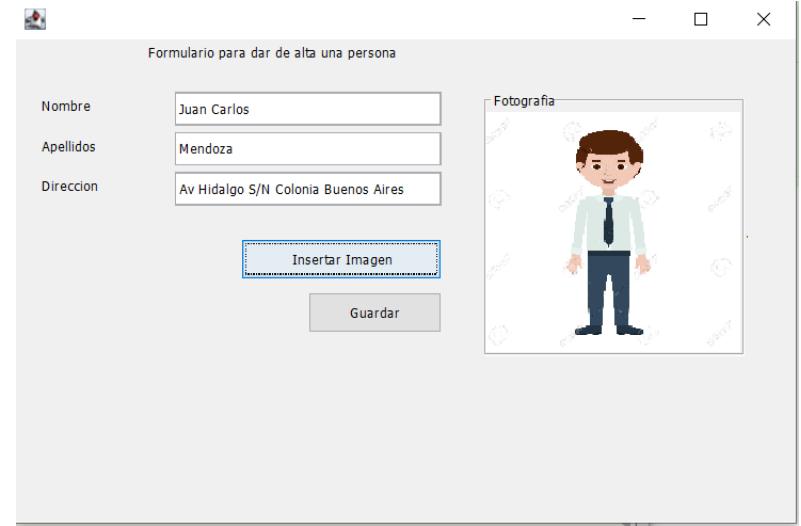
    this.setVisible(true);
    this.setLocationRelativeTo(null);
    this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
}

public static void main(String[] args) {
    try {
        UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
    } catch (ClassNotFoundException ex) {
        Logger.getLogger(SetBoundsExample.class.getName()).log(Level.SEVERE, null, ex);
    } catch (InstantiationException ex) {
        Logger.getLogger(SetBoundsExample.class.getName()).log(Level.SEVERE, null, ex);
    } catch (IllegalAccessException ex) {
        Logger.getLogger(SetBoundsExample.class.getName()).log(Level.SEVERE, null, ex);
    } catch (UnsupportedLookAndFeelException ex) {
        Logger.getLogger(SetBoundsExample.class.getName()).log(Level.SEVERE, null, ex);
    }
}
new SetBoundsExample();
```

```
public JPanel getpanel_foto() {
    JPanel foto
    pImagen = new JPanel();
    pImagen.setLayout(null); //indicamos disteubucion nula
    eImagen = new JLabel("Fotografia");
    eImagen.setHorizontalTextPosition(SwingConstants.CENTER); //texto centrado
    eImagen.setBounds(4, 15, 197, 180); //dentro de este panel empezamos en 0 nuevamente
    // y todo lo que pase en panel se queda ahí
    Border bordepanel = new TitledBorder(new EtchedBorder(), "Fotografia"); //Borde para paneles que permite crear
    //una linea a un panel para darle mas estetica
    pImagen.setBorder(bordepanel); //establecemos el borde con los metodos set's
    pImagen.add(eImagen); //al panel agregale la etiqueta
    return pImagen; //regresa el panel
}

//clase privada para el manejo de eventos tiene que implementar la interface ActionListener
private class Eventos_Botones implements ActionListener {
    @Override
    public void actionPerformed(ActionEvent e) {
        //aca hace los ajustes a la imagen que se obtenga
        JButton evento = (JButton) e.getSource();
        if (evento == inserta) {
            JFileChooser jfc = new JFileChooser(); //el objeto fileChooser abre una biblioteca de archivos
            if (evento == inserta) {
                JFileChooser jfc = new JFileChooser(); //el objeto fileChooser abre una biblioteca de archivos
                jfc.setFileSelectionMode(JFileChooser.FILES_ONLY); //hace que solo se seleccione uno
                int estado = jfc.showOpenDialog(null);
                if (estado == JFileChooser.APPROVE_OPTION) //si seleccionamos un archivo se hace lo siguiente
                    try {
                        fis = new FileInputStream(jfc.getSelectedFile()); //obten la ruta del archivo
                        longitud_bytes = (int) jfc.getSelectedFile().length(); //obten el tamano
                        Image icono = ImageIO.read(jfc.getSelectedFile()); //realiza la escala de acuerdo al tamano del label
                        eImagen.setIcon(new ImageIcon(icono)); //establece al label la imagen con el metodo SetIcon()
                        eImagen.updateUI(); //haz una actualizacion
                        longitud_bytes = longitud_bytes;
                    } catch (Exception ee) {
                    }
            }
        }
        //para hacer una validacion que los campos no esten vacios y el programa no truene usamos el metodo isEmpty()
        //el metodo regresa true si estan vacios o un false si tienen algo, son usados muy frecuentes
        else if (evento == guardar) {
            //si el campo del nombre esta vacio o el del apellido esta vacio o el de la direccion esta vacio manda un msj de error
            if (dNombre.getText().isEmpty() || dApellido.getText().isEmpty() || dDireccion.getText().isEmpty()) {
                JOptionPane.showMessageDialog(null, "Un campo esta vacio revise por favor",
                    "ERROR", JOptionPane.ERROR_MESSAGE);
            } else { // si no muestra un msj de registro exitoso
                JOptionPane.showMessageDialog(null, "Registro Exitoso");
            }
        }
    }
}
```

Salida:

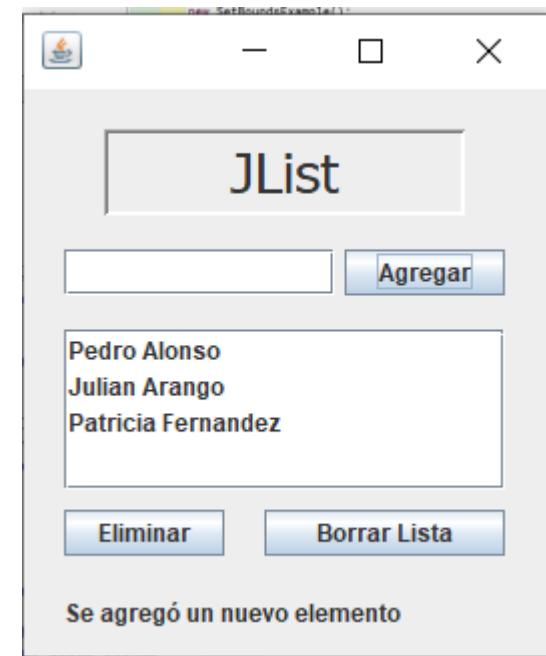


Componente Lista (JList)

```
Cerrar ****JLIST****  
* Este componente nos permite presentar una lista de selección donde podemos  
* escoger uno o varios elementos.  
* Agregar Elementos...  
* Se puede agregar elementos a un JList, la primera mediante un arreglo y  
* la segunda usando la clase DefaultListModel...  
* Forma 1.  
* Para agregar elementos usando un arreglo es muy simple, tan solo tenemos que  
* declarar nuestro arreglo y agregarselo al constructor del objeto JList con el  
* que estemos trabajando..... esta forma es muy usada cuando traemos datos  
* directamente de una BD y tenemos que mostrarla en pantalla....  
* JList listaNombres;  
* String nombres[] = { "Cristian", "Julian", "Milena"};  
* listaNombres = new JList( nombres );  
* Forma 2.  
* La segunda forma también es muy simple, tenemos que declarar un objeto de tipo  
* DefaultListModel y por medio del método addElement(elemento),  
* vamos agregando elementos a nuestro modelo, posteriormente dicho modelo  
* se agrega al JList con el que trabajemos..... esta forma es muy usada cuando  
* tenemos que agregar y mostrar directamente los datos ingresados en pantalla.  
* Jlist listaNombres=new Jlist();  
* DefaultListModel modelo = new DefaultListModel();  
* modelo.addElement("Elemento1");  
* modelo.addElement("Elemento2");  
* modelo.addElement("Elemento3");  
* listaNombres.setModel(modelo);  
*****  
import java.awt.Container;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
import javax.swing.DefaultListModel;  
import javax.swing.JButton;  
import javax.swing.JFrame;  
import javax.swing.JLabel;  
import javax.swing.JList;  
import javax.swing.JOptionPane;  
import javax.swing.JScrollPane;  
import javax.swing.JTextField;  
import javax.swing.ListSelectionModel;  
import javax.swing.JSpinner;  
public class Jlist_Example extends JFrame implements ActionListener {  
  
    private JButton agregar, eliminar, borrar; /*declaramos el objeto Botón*/  
    private JLabel labelTitulo, mensaje; /*declaramos el objeto Label*/  
    private JTextField campo;  
    private JList listaNombres; /*declaramos La Lista*/  
    private DefaultListModel modelo; /*declaramos el Modelo*/  
    private JScrollPane scrollLista; /*permite que se desplace de arriba a abajo como un navegador*/  
  
    public Jlist_Example() {  
        /*permite iniciar las propiedades de los componentes*/  
        iniciarComponentes();  
        /*Asigna un título a la barra de título*/  
        /*tamaño de la ventana*/  
        this.setVisible(true);  
        setSize(280, 330);  
  
        setSize(280, 330);  
        /*pone la ventana en el Centro de la pantalla*/  
        setLocationRelativeTo(null);  
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
    }  
  
    /*con esto definimos nosotros mismos los tamaños y posición  
     * de los componentes*/  
  
    setLayout(null);  
  
    campo = new JTextField();  
    campo.setBounds(20, 80, 135, 23);  
    /*Propiedades del botón, lo instanciamos, posicionamos* activamos los eventos*/  
    agregar = new JButton();  
    agregar.setText("Agregar");  
    agregar.setBounds(160, 80, 80, 23);  
    agregar.addActionListener(this);  
  
    eliminar = new JButton();  
    eliminar.setText("Eliminar");  
    eliminar.setBounds(20, 210, 80, 23);  
    eliminar.addActionListener(this);  
  
    borrar = new JButton();  
    borrar.setText("Borrar Lista");  
    borrar.setBounds(120, 210, 120, 23);  
    borrar.addActionListener(this);  
  
    /*Propiedades del Label, lo instanciamos, posicionamos y  
     * activamos los eventos*/  
    labelTitulo = new JLabel();  
    labelTitulo.setFont(new java.awt.Font("Tahoma", 0, 28));  
  
    labelTitulo.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);  
    labelTitulo.setText("Jlist");  
    labelTitulo.setBorder(javax.swing.BorderFactory.createBevelBorder(javax.swing.border.BevelBorder.LOWERED));  
    labelTitulo.setBounds(40, 20, 180, 43);  
  
    mensaje = new JLabel();  
    mensaje.setBounds(20, 250, 280, 23);  
  
    //instanciamos la lista  
    listaNombres = new JList();  
    listaNombres.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);  
  
    //instanciamos el modelo  
    modelo = new DefaultListModel();  
  
    //instanciamos el Scroll que tendrá la lista  
    scrollLista = new JScrollPane();  
    scrollLista.setBounds(20, 120, 220, 80);  
    scrollLista.setViewportView(listaNombres);  
  
    /*Agregamos los componentes al Contenedor*/  
    this.add(labelTitulo);  
    this.add(campo);  
    this.add(agregar);  
    this.add(eliminar);  
    this.add(borrar);  
    this.add(mensaje);  
    this.add(scrollLista);  
    //this.add(botonCam);  
}  
  
@Override  
public void actionPerformed(ActionEvent evento) {  
    if (evento.getSource() == agregar) {  
        agregarNombre();  
        mensaje.setText("Se agregó un nuevo elemento");  
    }  
    if (evento.getSource() == eliminar) {  
        eliminarNombre(listaNombres.getSelectedIndex());  
    }  
    if (evento.getSource() == borrar) {  
        borrarLista();  
        mensaje.setText("Se borró toda la lista");  
    }  
}  
  
private void agregarNombre() {  
    String nombre = campo.getText();  
    modelo.addElement(nombre);  
    listaNombres.setModel(modelo);  
    campo.setText("");  
}
```

```
private void iniciarComponentes() {  
    /*con esto definimos nosotros mismos los tamaños y posición  
     * de los componentes*/  
  
    setLayout(null);  
  
    campo = new JTextField();  
    campo.setBounds(20, 80, 135, 23);  
    /*Propiedades del botón, lo instanciamos, posicionamos* activamos los eventos*/  
    agregar = new JButton();  
    agregar.setText("Agregar");  
    agregar.setBounds(160, 80, 80, 23);  
    agregar.addActionListener(this);  
  
    eliminar = new JButton();  
    eliminar.setText("Eliminar");  
    eliminar.setBounds(20, 210, 80, 23);  
    eliminar.addActionListener(this);  
  
    borrar = new JButton();  
    borrar.setText("Borrar Lista");  
    borrar.setBounds(120, 210, 120, 23);  
    borrar.addActionListener(this);  
  
    /*Propiedades del Label, lo instanciamos, posicionamos y  
     * activamos los eventos*/  
    labelTitulo = new JLabel();  
    labelTitulo.setFont(new java.awt.Font("Tahoma", 0, 28));  
  
    labelTitulo.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);  
    labelTitulo.setText("Jlist");  
    labelTitulo.setBorder(javax.swing.BorderFactory.createBevelBorder(javax.swing.border.BevelBorder.LOWERED));  
    labelTitulo.setBounds(40, 20, 180, 43);  
  
    mensaje = new JLabel();  
    mensaje.setBounds(20, 250, 280, 23);  
  
    //instanciamos la lista  
    listaNombres = new JList();  
    listaNombres.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);  
  
    //instanciamos el modelo  
    modelo = new DefaultListModel();  
  
    //instanciamos el Scroll que tendrá la lista  
    scrollLista = new JScrollPane();  
    scrollLista.setBounds(20, 120, 220, 80);  
    scrollLista.setViewportView(listaNombres);  
  
    /*Agregamos los componentes al Contenedor*/  
    this.add(labelTitulo);  
    this.add(campo);  
    this.add(agregar);  
    this.add(eliminar);  
    this.add(borrar);  
    this.add(mensaje);  
    this.add(scrollLista);  
    //this.add(botonCam);  
}  
  
@Override  
public void actionPerformed(ActionEvent evento) {  
    if (evento.getSource() == agregar) {  
        agregarNombre();  
        mensaje.setText("Se agregó un nuevo elemento");  
    }  
    if (evento.getSource() == eliminar) {  
        eliminarNombre(listaNombres.getSelectedIndex());  
    }  
    if (evento.getSource() == borrar) {  
        borrarLista();  
        mensaje.setText("Se borró toda la lista");  
    }  
}  
  
private void agregarNombre() {  
    String nombre = campo.getText();  
    modelo.addElement(nombre);  
    listaNombres.setModel(modelo);  
    campo.setText("");  
}
```

```
private void eliminarNombre(int indice) {  
    if (indice >= 0) {  
        modelo.removeElementAt(indice);  
        mensaje.setText("Se eliminó un elemento en la posición " + indice);  
    } else {  
        JOptionPane.showMessageDialog(null, "Debe seleccionar un índice",  
            "Error", JOptionPane.ERROR_MESSAGE);  
  
        mensaje.setText("NO se seleccionó ningún elemento");  
    }  
}  
  
private void borrarLista() {  
    modelo.clear(); // borra todo lo que tiene la lista  
}  
  
public static void main(String[] args) {  
    new JList_Example();  
}
```



Más Componentes y Otros Eventos

```
import java.awt.Container;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.*;
import javax.swing.JToggleButton;
import javax.swing.border.TitledBorder;
import javax.swing.event.ChangeEvent;
import javax.swing.event.ChangeListener;

public class VentanaPrincipal extends JFrame implements ActionListener, ChangeListener {
```

```
    private Container contenedor;
    //declaramos el contenedor
    JLabel labelTitulo;
    //declaramos el objeto Label
    // declaramos los objetos JLabels
    JLabel etiquetaLabel;
    JLabel etiquetaBoton;
    JLabel etiquetaToogleButton;
    JLabel etiquetaCheckbox;
    JLabel etiquetaRadioButton;
    JLabel etiquetaCombo;
    JLabel etiquetaSeparator;
    JLabel etiquetaSpinner;
    JLabel etiquetaDeslizador;
    JLabel etiquetaBarra;
    JButton boton;
    /* declaramos el objeto Boton*/
    JCheckBox checkbox1, checkbox2;
    /** declaramos los objetos checkbox*/
    ButtonGroup grupoRadios;
    /**declaramos un grupo de radioButtons*/
    JRadioButton radio1, radio2;
    /** declaramos los objetos radioButtons*/
    JToggleButton toggleButton;
    /**declaramos el objeto ToggleButton*/
    JComboBox combo;
    /**declaramos el objeto Combo*/
    JSeparator separadorVertical, separadorHorizontal;
    /** declaramos los separadores*/
    JSpinner spinner;
    /** declaramos el objeto spinner*/
    JSlider deslizador;
    /**declaramos el objeto deslizador*/
    JProgressBar barra;
    /** declaramos el objeto barra*/
    public VentanaPrincipal() {
```

```
        /*permite iniciar las propiedades de los componentes*/
        iniciarComponentes();
        /*Asigna un titulo a la barra de titulo*/
        setTitle("JFrame Componentes");
        /*tamaño de la ventana*/
        setSize(630, 250);
        this.setVisible(true);
        /*pone la ventana en el Centro de la pantalla*/
        setLocationRelativeTo(null);
        setResizable(false);
```

```
    private void iniciarComponentes() {
        contenedor = getContentPane();/*instanciamos el contenedor*/
        /*con esto definimos nosotros mismos los tamaños y posicion
         * de los componentes*/
        contenedor.setLayout(null);
        /*Definimos las propiedades de los componentes*/
        labelTitulo = new JLabel();
        labelTitulo.setText("Componentes");
        //pone formato a la letra
        labelTitulo.setFont(new java.awt.Font("Tahoma", 1, 20));
        labelTitulo.setBounds(180, 5, 380, 40);
        etiquetaLabel = new JLabel();
        etiquetaLabel.setText("JLabel : Esto es un Label o Etiqueta");
        etiquetaLabel.setBounds(20, 50, 280, 23);

        etiquetaBoton = new JLabel();
        etiquetaBoton.setText("JButton : ");
        etiquetaBoton.setBounds(20, 80, 80, 23);

        boton = new JButton();
        boton.setText("Boton");
        boton.setBounds(80, 80, 80, 23);
        boton.addActionListener(this);

        etiquetaCheckbox = new JLabel();
        etiquetaCheckbox.setText("JCheckBox : ");
        etiquetaCheckbox.setBounds(20, 110, 80, 23);
        checkbox1 = new JCheckBox();
        checkbox1.setText("Check1");
        checkbox1.setBounds(100, 110, 80, 23);

        checkbox2 = new JCheckBox();
        checkbox2.setText("Check2");
        checkbox2.setBounds(180, 110, 80, 23);

        etiquetaRadioButton = new JLabel();
        etiquetaRadioButton.setText("JRadioButton : ");
        etiquetaRadioButton.setBounds(20, 140, 100, 23);

        grupoRadios = new ButtonGroup();
        radio1 = new JRadioButton();
        radio1.setText("Radio1");
        radio1.setBounds(110, 140, 80, 23);

        radio2 = new JRadioButton();
        radio2.setText("Radio2");
        radio2.setBounds(190, 140, 80, 23);

        grupoRadios.add(radio1);
        grupoRadios.add(radio2);

        etiquetaToogleButton = new JLabel();
        etiquetaToogleButton.setText("JToogleButton : ");
        etiquetaToogleButton.setBounds(20, 180, 100, 23);
```

```
        toggleButton = new JToggleButton();
        toggleButton.setText("Activar");
        toggleButton.setBounds(120, 180, 80, 23);

        etiquetaCombo = new JLabel();
        etiquetaCombo.setText("JComboBox : ");
        etiquetaCombo.setBounds(350, 50, 100, 23);

        combo = new JComboBox();
        combo.addItem("Opciones");
        combo.addItem("Opcion1");
        combo.addItem("Opcion2");
        combo.addItem("Opcion3");
        combo.addItem("Opcion4");
        combo.setBounds(430, 50, 100, 23);
        combo.setSelectedIndex(0);

        separadorVertical = new JSeparator();
        separadorVertical.setOrientation(javax.swing.SwingConstants.VERTICAL);
        separadorVertical.setBounds(300, 60, 10, 200);

        etiquetaSeparator = new JLabel();
        etiquetaSeparator.setText("JSeparator : ");
        etiquetaSeparator.setBounds(350, 80, 100, 23);

        separadorHorizontal = new JSeparator();
        separadorHorizontal.setBounds(430, 92, 100, 5);

        etiquetaSpinner = new JLabel();
        etiquetaSpinner.setText("JSpinner : ");
        etiquetaSpinner.setBounds(350, 110, 100, 23);

        spinner = new JSpinner();
        spinner.setBounds(430, 110, 50, 23);
        spinner.addChangeListener(this);

        etiquetaDeslizador = new JLabel();
        etiquetaDeslizador.setText("JSlider : ");
        etiquetaDeslizador.setBounds(350, 140, 100, 23);

        deslizador = new JSlider(JSlider.HORIZONTAL, 0, 100, 30);
        deslizador.setBounds(430, 140, 100, 30);
        deslizador.setPaintTicks(true);
        deslizador.setMajorTickSpacing(50);
        deslizador.setMinorTickSpacing(5);
        deslizador.setBorder(new TitledBorder(""));
        deslizador.setValue(0);
        deslizador.addChangeListener(this);

        etiquetaBarra = new JLabel();
        etiquetaBarra.setText("JProgressBar : ");
        etiquetaBarra.setBounds(350, 180, 100, 23);

        barra = new JProgressBar();
        barra.setBounds(430, 180, 100, 20);
```

Parte 2

```
/*Agregamos los componentes al Contenedor*/
contenedor.add(labelTitulo);
contenedor.add(etiquetaLabel);
contenedor.add(etiquetaBoton);
contenedor.add(etiquetaCheckbox);
contenedor.add(checkbox1);
contenedor.add(checkbox2);
contenedor.add(etiquetaRadioButton);
contenedor.add(radio1);
contenedor.add(radio2);
contenedor.add(etiquetaToogleButton);
contenedor.add(toggleButton);
contenedor.add(etiquetaCombo);
contenedor.add(separadorVertical);
contenedor.add(etiquetaSeparador);
contenedor.add(separadorHorizontal);
contenedor.add(etiquetaSpinner);
contenedor.add(spinner);
contenedor.add(etiquetaDeslizador);
contenedor.add(deslizador);
contenedor.add(etiquetaBarra);
contenedor.add(barra);
contenedor.add(combo);
contenedor.add(botón);
```

```
Cerrar /**
 * Agregamos los eventos de presionar
 */
@Override
public void actionPerformed(ActionEvent evento) {
    /**
     * Evento cuando presiona el botón
     */
    if (evento.getSource() == botón) {
        String salida = "";
        salida = validaEventos();
        JOptionPane.showMessageDialog(null, salida);
    }
}

/**
 * permite validar cuando se selecciona un check un radioButton, una opción
 * del combo o se presiona el ToogleButton y se
 */
private String validaEventos() {
    String cad = "Selecciona : \n";
    if (checkbox1.isSelected()) {
        cad += "checkbox1\n";
    }
    if (checkbox2.isSelected()) {
        cad += "checkbox2\n";
    }
}
```

```
if (radio1.isSelected()) {
    cad += "radio1\n";
}
if (radio2.isSelected()) {
    cad += "radio2\n";
}
if (toggleButton.isSelected()) {
    cad += "Toogle Activo\n";
} else {
    cad += "Toogle InActivo\n";
}
cad += combo.getSelectedItem() + "\n";
return cad;
}

/**
 * Permite definir los eventos del deslizador y el spinner
 */
@Override
public void stateChanged(ChangeEvent evento) {
    int valor;
    if (evento.getSource() == deslizador) {
        valor = deslizador.getValue();
        barra.setValue(valor);
        spinner.setValue(valor);
    }

    if (evento.getSource() == spinner) {
        valor = (Integer) spinner.getValue();
        deslizador.setValue(valor);
        barra.setValue(valor);
    }
}

public static void main(String[] args) {
    new VentanaPrincipal();
}
```

Salida:



Clase ArrayList

```
import java.util.ArrayList;
public class Test_ArrayList {
    public static void main(String[] args) {
        // Creando una ArrayList genérica
        ArrayList <String> arlTest = new ArrayList<String>();
        // Tamaño de arrayList
        System.out.println("Tamaño de ArrayList en la creación:" + arlTest.size());
        // Permite agregar algunos elementos
        arlTest.add("D");
        arlTest.add("U");
        arlTest.add("K");
        arlTest.add("E");
        // Vuelve a verificar el tamaño después de agregar elementos
        System.out.println("Tamaño de ArrayList después de agregar elementos:" + arlTest.size());
        // Mostrar todos los contenidos de ArrayList
        System.out.println("Lista de todos los elementos:" + arlTest);
        // Eliminar algunos elementos de la lista arlTest.remove ("D");
        System.out.println("Ver contenido después de eliminar un elemento:" + arlTest);
        // Eliminar elemento por índice arlTest.remove (2);
        System.out.println("Ver contenido después de eliminar elemento por índice:" + arlTest);
        // Verifique el tamaño después de eliminar los elementos
        System.out.println("Tamaño de arrayList después de eliminar elementos:" + arlTest.size());
        System.out.println("Lista de todos los elementos después de eliminar elementos:" + arlTest);
        // Verifica si la lista contiene "K"
        System.out.println(arlTest.contains("K"));
        // Recorrer un ArrayList
        for(int i=0;i<arlTest.size();i++){
            System.out.println(arlTest.get(i));
        }
    }
}
```

Salida:

```
Tamaño de ArrayList en la creación:0
Tamaño de ArrayList después de agregar elementos:4
Lista de todos los elementos:[D, U, K, E]
Ver contenido después de eliminar un elemento:[D, U, K, E]
Ver contenido después de eliminar elemento por índice:[D, U, K, E]
Tamaño de arrayList después de eliminar elementos:4
Lista de todos los elementos después de eliminar elementos:[D, U, K, E]
true
D
U
K
E
```

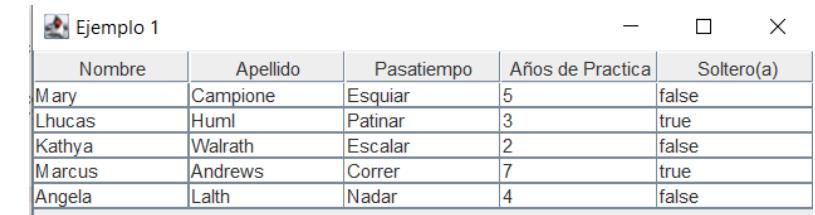
Clase para manejar archivos

```
import java.io.*; //Importamos todas las clases de java.io.
public class FicheroTextoBufferedApp {
    public static void main(String[] args) {
        try(BufferedReader br=new BufferedReader(new FileReader("D:\\fichero1.txt"));
            BufferedWriter bw=new BufferedWriter(new FileWriter("D:\\fichero1.txt"))){
            escribeFichero(bw);
            //Guardamos los cambios del fichero
            bw.flush();
            leeFichero(br);
        }catch(IOException e){
            System.out.println("Error E/S: "+e);
        }
    }
    public static void escribeFichero(BufferedWriter bw) throws IOException{
        //Escribimos en el fichero
        bw.write("Esto es una prueba usando Buffered");
        bw.newLine();
        bw.write("Seguimos usando Buffered");
    }
    public static void leeFichero(BufferedReader br) throws IOException{
        //Leemos el fichero y lo mostramos por pantalla
        String linea=br.readLine();
        while(linea!=null){
            System.out.println(linea);
            linea=br.readLine();
        }
    }
}
```

Ejemplo JTable

```
import javax.swing.JTable;
import javax.swing.JScrollPane;
import javax.swing.JFrame;
import java.awt.*;
import java.awt.event.*;
public class SimpleTable1 extends JFrame {
    public SimpleTable1() {
        super("Ejemplo 1");
        //Array bidimensional de objetos con los datos de la tabla
        Object[][] data = {
            {"Mary", "Campione", "Esquier", new Integer(5), new Boolean(false)},
            {"Lhucas", "Huml", "Patinar", new Integer(3), new Boolean(true)},
            {"Kathy", "Walrath", "Escalar", new Integer(2), new Boolean(false)},
            {"Marcus", "Andrews", "Correr", new Integer(7), new Boolean(true)},
            {"Angela", "Lalth", "Nadar", new Integer(4), new Boolean(false)}
        };
        //Array de 'String' con los títulos de las columnas
        String[] columnNames = {"Nombre", "Apellido", "Pasatiempo", "Años de Práctica", "Soltero(a)"};
        //Creacion de la tabla
        final JTable table = new JTable(data, columnNames);
        table.setPreferredScrollableViewportSize(new Dimension(500, 80));
        //Creamos un scrollpanel y se lo agregamos a la tabla
        JScrollPane scrollpane = new JScrollPane(table);
        //Agregamos el scrollpanel al contenedor
        getContentPane().add(scrollpane, BorderLayout.CENTER);
        //manejamos la salida
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}
```

Salida:



Nombre	Apellido	Pasatiempo	Años de Práctica	Soltero(a)
Mary	Campione	Esquier	5	false
Lhucas	Huml	Patinar	3	true
Kathy	Walrath	Escalar	2	false
Marcus	Andrews	Correr	7	true
Angela	Lalth	Nadar	4	false

clases poo

Clase Padre

```
public class Jugador {  
  
    private int numUniforme, minJugados, golesAnotados;  
    private String nombre;  
  
    public Jugador(String nombre, int numUniforme, int minJugados, int golesAnotados) {  
        this.nombre = nombre;  
        this.numUniforme = numUniforme;  
        this.minJugados = minJugados;  
        this.golesAnotados = golesAnotados;  
    }  
  
    public int getNumUniforme() {  
        return numUniforme;  
    }  
  
    public int getMinJugados() {  
        return minJugados;  
    }  
  
    public int getGolesAnotados() {  
        return golesAnotados;  
    }  
  
    public String getNombre() {  
        return nombre;  
    }  
  
    public String toString() {/*Este metodo nos regresa informacion del objeto, pero al sobre escribirlo  
    nosotros escogemos como queremos esa informacion.*/  
        return "Nombre:" + getNombre() + ".\nNúmero de Uniforme:" + getNumUniforme()  
            + ".\nMinutos Jugados:" + getMinJugados() + ".\nGoles Anotados"  
            + getGolesAnotados();  
    }  
}
```

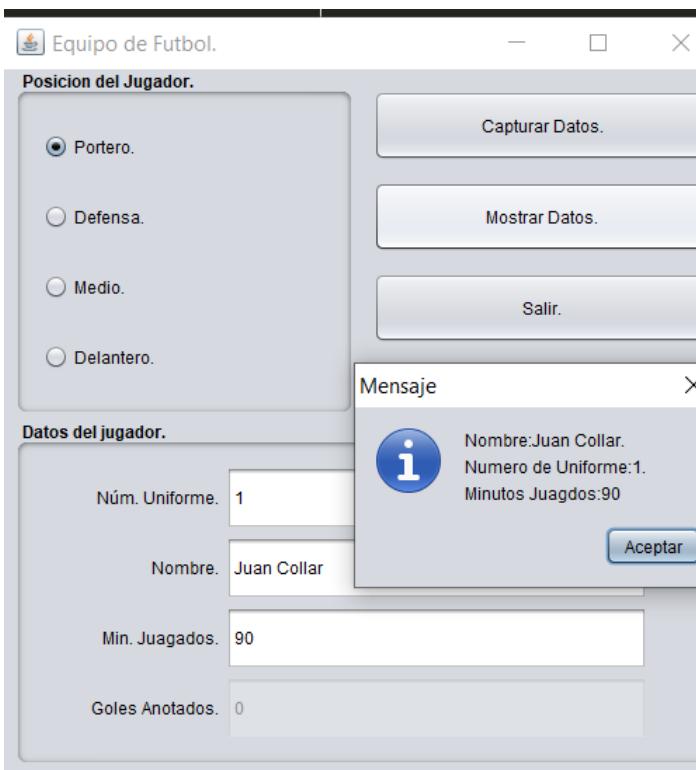
Clase Hija

```
public class Portero extends Jugador{  
  
    public Portero(String nombre, int numUniforme, int minJugados) {  
        super(nombre, numUniforme, minJugados, -1);/*super-> crea al objeto apartir de la super clase  
        en este cao Jugador*/  
    }  
    public String toString(){/*Este metodo nos regresa informacion del objeto, pero al sobre escribirlo  
    nosotros escogemos como queremos esa informacion.*/  
        return "Nombre:" + getNombre() + ".\nNúmero de Uniforme:" + getNumUniforme() + ".\nMinutos Jugados:" + getMinJugados();  
    }  
}
```

implementacion con una interfaz grafica de usuario

metodo donde unimos nuestros objetos con la interfaz grafica

```
private void botonAP(ActionEvent e) {  
  
    //isSelected() -> nos regresa el estado (booleano true-false) del boton si esta o no activo.  
    if (radio_P.isSelected()) {  
        /*texto_Nom.getText() -> Nos regresa le texto que escribimos sobre la caja */  
        /*Al pasarle los parametros estamos llenanando los atributos del objeto 'miPortero' */  
        miPortero = new Portero(texto_Nom.getText(), Integer.parseInt(texto_NU.getText()), Integer.parseInt(texto_MJ.getText()));  
        texto_GA.setText("0");  
    } else if (radio_D.isSelected()) {  
        /*El primer objeto del arreglo equibaldr a el defensa del equipo.  
        jugadores[0] = new Jugador(texto_Nom.getText(), Integer.parseInt(texto_NU.getText()), Integer.parseInt(texto_MJ.getText()),  
        Integer.parseInt(texto_GA.getText()));  
    } else if (radio_M.isSelected()) {  
        /*El segundo objeto del arreglo equibaldr a el Medio campista del equipo.  
        jugadores[1] = new Jugador(texto_Nom.getText(), Integer.parseInt(texto_NU.getText()), Integer.parseInt(texto_MJ.getText()),  
        Integer.parseInt(texto_GA.getText()));  
    } else if (radio_Del.isSelected()) {  
        /*El tercer objeto del arreglo equibaldr a el Delantero del equipo.  
        jugadores[2] = new Jugador(texto_Nom.getText(), Integer.parseInt(texto_NU.getText()), Integer.parseInt(texto_MJ.getText()),  
        Integer.parseInt(texto_GA.getText()));  
    } else {  
        JOptionPane.showMessageDialog(null, "No seleccionaste algun jugador.");  
    }  
    generar_archivo();  
}
```



SOLUCION EJERCICIOS DE TAREA

ColoresRGB.java

```
import java.awt.BorderLayout;
import java.awt.Dimension;
import javax.swing.*;
import java.awt.GridBagLayout;
import java.awt.GridBagConstraints;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.Color;
import java.awt.Insets;
import java.util.logging.Level;
import java.util.logging.Logger;

public class ColoresRGB extends JFrame {

    private JPanel panel_este, panel_center, panel_sur, panel_oeste;
    private JLabel eRojo, eVerde, eAzul, titulo;
    private JTextField dRojo, dVerde, dAzul;
    private JButton calcular;

    public ColoresRGB() {
        init();
    }

    private void init() {
        this.setSize(new Dimension(600, 380));
        this.setTitle("Paleta de Colores");
        this.setLocationRelativeTo(null);
        this.setLayout(new BorderLayout(10, 10));
        GridBagConstraints gbc = new GridBagConstraints();
        gbc.insets = new Insets(10, 10, 10, 10);

        panel_oeste = new JPanel();
        panel_oeste.setLayout(new GridBagLayout());

        eRojo = new JLabel("Rojo");
        gbc.gridx = 0;
        gbc.gridy = 0;
        panel_oeste.add(eRojo, gbc);

        dRojo = new JTextField(10);
        gbc.gridx = 1;
        gbc.gridy = 0;
        panel_oeste.add(dRojo, gbc);

        eVerde = new JLabel("Verde");
        gbc.gridx = 0;
        gbc.gridy = 1;
        panel_oeste.add(eVerde, gbc);

        dVerde = new JTextField(10);
        gbc.gridx = 1;
        gbc.gridy = 1;
        panel_oeste.add(dVerde, gbc);

        eAzul = new JLabel("Azul");
        gbc.gridx = 0;
        gbc.gridy = 2;
        panel_oeste.add(eAzul, gbc);

        dAzul = new JTextField(10);
        gbc.gridx = 1;
        gbc.gridy = 2;
        panel_oeste.add(dAzul, gbc);

        calcular = new JButton("Mostrar Color");
        gbc.gridx = 2;
        gbc.gridy = 1;
        gbc.fill = GridBagConstraints.WEST;
        panel_oeste.add(calcular, gbc);

        panel_center = new JPanel();
        panel_sur = new JPanel();

        panel_este = new JPanel();
        this.add(panel_oeste, BorderLayout.WEST);
        this.add(panel_center, BorderLayout.CENTER);
        this.add(panel_sur, BorderLayout.SOUTH);
        this.add(panel_este, BorderLayout.EAST);
        titulo = new JLabel("Paleta de Colores");
        titulo.setHorizontalAlignment(SwingConstants.CENTER);
        this.add(titulo, BorderLayout.NORTH);
        panel_center.setBackground(Color.WHITE);

        calcular.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                int r = Integer.parseInt(dRojo.getText().trim());
                int g = Integer.parseInt(dVerde.getText().trim());
                int b = Integer.parseInt(dAzul.getText().trim());
                System.out.println(dRojo + " " + dVerde + " " + dAzul);
                if ((r <= 255 & g <= 255 & b <= 255) && (r >= 0 && g >= 0 && b >= 0)) {
                    Color color = new Color(r, g, b);
                    panel_center.setBackground(color);
                } else {
                    JOptionPane.showMessageDialog(null, "Fuera de Rango");
                }
            }
        });
        this.setVisible(true);
        this.setLocationRelativeTo(null);
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}
```

```
dVerde = new JTextField(10);
gbc.gridx = 1;
gbc.gridy = 1;
panel_oeste.add(dVerde, gbc);

eAzul = new JLabel("Azul");
gbc.gridx = 0;
gbc.gridy = 2;
panel_oeste.add(eAzul, gbc);

dAzul = new JTextField(10);
gbc.gridx = 1;
gbc.gridy = 2;
panel_oeste.add(dAzul, gbc);

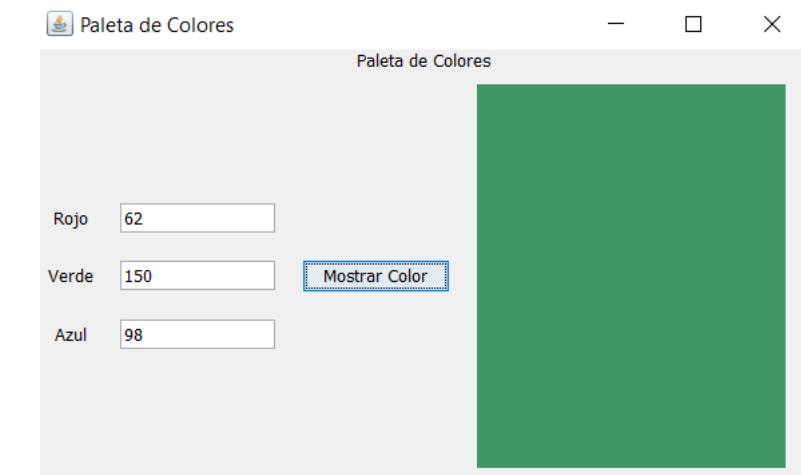
calcular = new JButton("Mostrar Color");
gbc.gridx = 2;
gbc.gridy = 1;
gbc.fill = GridBagConstraints.WEST;
panel_oeste.add(calcular, gbc);

panel_center = new JPanel();
panel_sur = new JPanel();

panel_este = new JPanel();
this.add(panel_oeste, BorderLayout.WEST);
this.add(panel_center, BorderLayout.CENTER);
this.add(panel_sur, BorderLayout.SOUTH);
this.add(panel_este, BorderLayout.EAST);
titulo = new JLabel("Paleta de Colores");
titulo.setHorizontalAlignment(SwingConstants.CENTER);
this.add(titulo, BorderLayout.NORTH);
panel_center.setBackground(Color.WHITE);

calcular.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        int r = Integer.parseInt(dRojo.getText().trim());
        int g = Integer.parseInt(dVerde.getText().trim());
        int b = Integer.parseInt(dAzul.getText().trim());
        System.out.println(dRojo + " " + dVerde + " " + dAzul);
        if ((r <= 255 & g <= 255 & b <= 255) && (r >= 0 && g >= 0 && b >= 0)) {
            Color color = new Color(r, g, b);
            panel_center.setBackground(color);
        } else {
            JOptionPane.showMessageDialog(null, "Fuera de Rango");
        }
    }
});
this.setVisible(true);
this.setLocationRelativeTo(null);
this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
}
```

```
public static void main(String[] args) {
    try {
        UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
    } catch (ClassNotFoundException ex) {
        Logger.getLogger(ColoresRGB.class.getName()).log(Level.SEVERE, null, ex);
    } catch (InstantiationException ex) {
        Logger.getLogger(ColoresRGB.class.getName()).log(Level.SEVERE, null, ex);
    } catch (IllegalAccessException ex) {
        Logger.getLogger(ColoresRGB.class.getName()).log(Level.SEVERE, null, ex);
    } catch (UnsupportedLookAndFeelException ex) {
        Logger.getLogger(ColoresRGB.class.getName()).log(Level.SEVERE, null, ex);
    }
    new ColoresRGB();
}
```



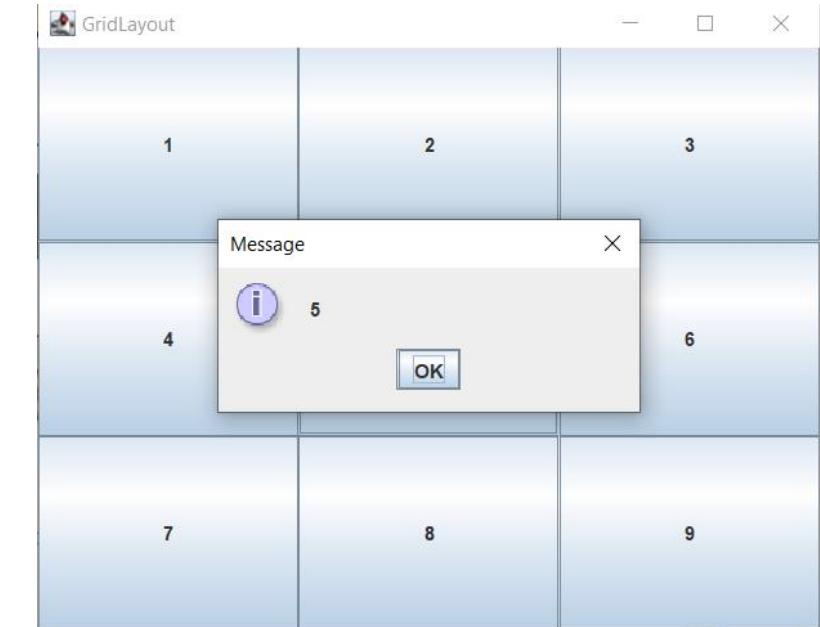
Solucion de ExampleGridLayout.java

```
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.Color;
import java.awt.GridLayout;
import javax.swing.*;  
**/  
*  
* @author  
*/  
  
public class EjercicioDelClub extends JFrame{  
    private GridLayout gl;  
    private JButton btn1,btn2,btn3,btn4,btn5,btn6,btn7,btn8,btn9;  
  
    public EjercicioDelClub() {  
        init();  
    }  
  
    private void init() {  
        int filas = 3;  
        int columnas = 3;  
        gl = new GridLayout(filas, columnas);  
        this.setLayout(gl);  
        btn1 = new JButton("1");  
        btn2 = new JButton("2");  
        btn3 = new JButton("3");  
        btn4 = new JButton("4");  
        btn5 = new JButton("5");  
        btn6 = new JButton("6");  
        btn7 = new JButton("7");  
        btn8 = new JButton("8");  
        btn9 = new JButton("9");  
  
        this.add(btn1);  
        this.add(btn2);  
        this.add(btn3);  
        this.add(btn4);  
        this.add(btn5);  
        this.add(btn6);  
        this.add(btn7);  
        this.add(btn8);  
        this.add(btn9);  
  
        btn1.addActionListener(new ActionListener() {  
            @Override  
            public void actionPerformed(ActionEvent e) {  
                JOptionPane.showMessageDialog(null, btn1.getText());  
            }  
        });  
  
        btn2.addActionListener(new ActionListener() {  
            @Override  
            public void actionPerformed(ActionEvent e) {  
                JOptionPane.showMessageDialog(null, btn2.getText());  
            }  
        });  
    }  
}
```

```
btn3.addActionListener(new ActionListener() {  
    @Override  
    public void actionPerformed(ActionEvent e) {  
        JOptionPane.showMessageDialog(null, btn3.getText());  
    }  
});  
  
btn4.addActionListener(new ActionListener() {  
    @Override  
    public void actionPerformed(ActionEvent e) {  
        JOptionPane.showMessageDialog(null, btn4.getText());  
    }  
});  
  
btn5.addActionListener(new ActionListener() {  
    @Override  
    public void actionPerformed(ActionEvent e) {  
        JOptionPane.showMessageDialog(null, btn5.getText());  
    }  
});  
  
btn6.addActionListener(new ActionListener() {  
    @Override  
    public void actionPerformed(ActionEvent e) {  
        JOptionPane.showMessageDialog(null, btn6.getText());  
    }  
});  
  
btn7.addActionListener(new ActionListener() {  
    @Override  
    public void actionPerformed(ActionEvent e) {  
        JOptionPane.showMessageDialog(null, btn7.getText());  
    }  
});  
  
btn8.addActionListener(new ActionListener() {  
    @Override  
    public void actionPerformed(ActionEvent e) {  
        JOptionPane.showMessageDialog(null, btn8.getText());  
    }  
});  
  
btn9.addActionListener(new ActionListener() {  
    @Override  
    public void actionPerformed(ActionEvent e) {  
        JOptionPane.showMessageDialog(null, btn9.getText());  
    }  
});  
  
this.setTitle("GridLayout");  
this.setSize(500, 400);  
this.setVisible(true);  
this.setLocationRelativeTo(null);  
this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
}
```

```
public static void main(String[] args) {  
    new EjercicioDelClub();  
}  
}
```

Demostracion



DESCRIPCIÓN DEL PROYECTO FINAL

Elaborar una Aplicación la cual esté basada en sus diseños de clases que hicieron en la sesión anterior, esta aplicación debe ser capaz de guardar toda aquella información de los atributos de sus clases que consideren necesarios en un archivo con extensión .txt separado por comas.

Especificaciones:

- Deben ser al menos 2 clases que diseñen una donde al menos usen herencia.
- Pueden usar cualquier componente grafico visto en clase y algún otro que hayan investigado (opcional).
- Pueden usar cualquiera de los administradores de distribuciones vistos durante el curso, incluso combinarlos.
- Deben tener un método el cual cree su archivo de texto y otro donde guarde la información capturada
- Si utiliza el componente JTable (Tabla) asegúrese de implementar un objeto de la clase ArrayList <Object> la cual debe ser de objetos de sus clases que diseñaron.
- La aplicación debe ser capaz de borrar registros del archivo de texto y de la tabla si es que se implementa
- La documentación del proyecto debe contener:
 - Portada (nombre, numero de control, semestre, carrera)
 - Planteamiento del problema
 - Diseño de la solución, en este punto deben estar una breve descripción de como diseñaron sus clases.
 - Herramientas que se implementaron
 - Capturas del código (opcional) o una breve explicación de cómo funciona su aplicación
 - Capturas de la misma funcionando.
 - Conclusiones.
- En un archivo.zip debe venir su documentación, así como sus archivos .java que implementaron para la creación del mismo.
- El archivo .zip debe tener su nombre.

REFERENCIAS DEL CURSO

DEITEL, H. (2008). *Como Programar en Java* (7ma ed.). Mexico: Pearson.