



## Carátula para entrega de prácticas

Facultad de Ingeniería

Laboratorio de docencia

# Laboratorios de computación salas A y B

*Profesor:* M.C. Jorge Rommel Santiago Arce

*Asignatura:* Programación Orientada a Objetos

*Grupo:* 2

*No de Práctica(s):* Z

*Integrante(s):* Cruz Rangel Leonardo Said  
Ibañez Guzman Osvaldo  
Maya Ortega Maria Fernanda  
van der Werff Vargas Pieter Alexander

*No. de Equipo de  
cómputo empleado:* 45-48

*No. de Lista o Brigada:* 7

*Semestre:* 2020-1

*Fecha de entrega:* 15/10/2019

*Observaciones:*

**CALIFICACIÓN:** \_\_\_\_\_

## Práctica complementaria: “GUI”

### Objetivos:

Conocer otras características del lenguaje de programación Java.

### Actividad 1 - La clase JOptionPane

#### Clase Bienvenido.

Ésta clase muestra en un panel gráfico el mensaje “Hola Mundo!!”.

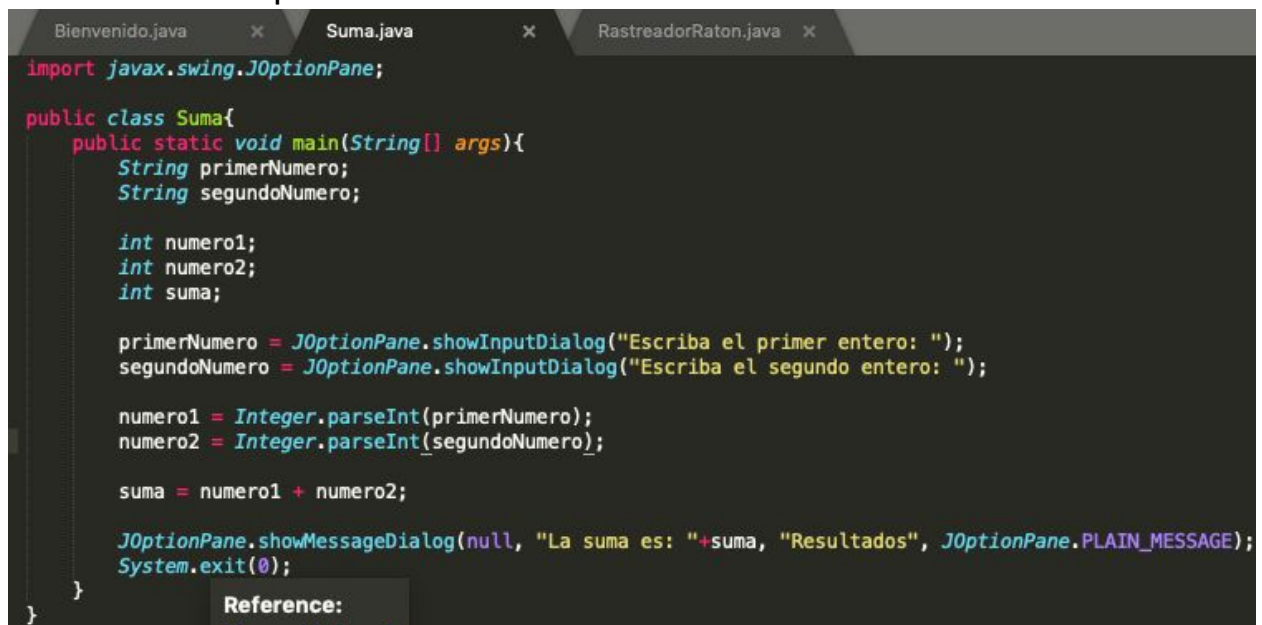


```
import javax.swing.JOptionPane;

public class Bienvenido{
    public static void main(String[] args){
        JOptionPane.showMessageDialog(null, "Hola Mundo!!");
        System.exit(0);
    }
}
```

#### Clase Suma.

Se muestra en pantalla un panel en donde se piden dos números enteros, el programa recibe la entrada y la convierte en entero, se realiza la suma y se muestra otro panel con el resultado.



```
import javax.swing.JOptionPane;

public class Suma{
    public static void main(String[] args){
        String primerNumero;
        String segundoNumero;

        int numero1;
        int numero2;
        int suma;

        primerNumero = JOptionPane.showInputDialog("Escriba el primer entero: ");
        segundoNumero = JOptionPane.showInputDialog("Escriba el segundo entero: ");

        numero1 = Integer.parseInt(primerNumero);
        numero2 = Integer.parseInt(segundoNumero);

        suma = numero1 + numero2;

        JOptionPane.showMessageDialog(null, "La suma es: "+suma, "Resultados", JOptionPane.PLAIN_MESSAGE);
        System.exit(0);
    }
}
```

### Actividad 2 - GUI

#### 2.1

Con esta implementación se observa una interfaz gráfica en la que el

usuario puede escoger entre hacer click en “Botoncito simple” o en “Botoncito elegante”. Para lo anterior, en la clase *Boton* que hereda de *JFrame* se tienen como atributos los dos botones como objetos de la clase  *JButton*. Se configura la GUI en el método *Boton*, empezando con el nombre del panel que se da con ayuda de la palabra *super* y entre paréntesis el nombre de la misma. Y con ayuda del constructor *Container* se obtiene el panel con el nombre dado.

Una vez creado el panel para crear los botones se utiliza la clase *JButton* y se agrega al panel con el método *.add()*. En el caso de *Botoncito elegante*, para que se agregue la imagen se utiliza la misma clase pero como parámetro además del nombre del botón la imagen que se quiere. Y para que cambie al pasar el mouse por ahí se utiliza el método *.setRolloverIcon()* y como parámetro se le da la segunda imagen.

Finalizando con el manejo de eventos, se utiliza un objeto de la clase *ManejadorBoton* y el método de los botones *.addActionListener()*.

Para la clase *ManejadorBoton* que utiliza *ActionListener* se indica lo que sucederá al hacer click a ese botón. En este caso se muestra un mensaje con el método *.showMessageDialog* de la clase *JOptionPane*.





## 2.2

```
[Grecia48:Desktop poo02alu24$ javac Lista.java
Note: Lista.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
Grecia48:Desktop poo02alu24$
```



Con esta clase se despliega una lista en donde la ventana de fondo cambia al color correspondiente al que se haya seleccionado por el usuario.

Esta clase solamente hereda de JFrame, se hace uso de objetos de tipo *Color*, para mediante una lista de nombres de cada color, poder hacer una Lista que despliega los diferentes colores y cambiar al color correspondiente en base al que está seleccionado.

Para esto es necesario implementar los métodos que tienen Container y JList mediante un objeto de cada clase, que se instancian mediante sus respectivos constructores.

*setLayout* se utiliza para agregar un panel de contenido, *setVisibleRowCount* se utiliza para crear una lista de elementos de *Color*. Posteriormente se utiliza *setSelectionMode*, para poder marcar un solo elemento de la lista. Por último se agrega la lista al objeto de tipo *Container*.

### Actividad 3 - Eventos de Mouse

La actividad consistía en hacer la corrección del código para que pudiera funcionar sin ningún tipo de problema, se encontraron tres errores en el código, los cuales son:

Jframe ---> JFrame

```
public class RastreadorRaton extends JFrame implements MouseListener, MouseMotionListener {  
    private JLabel barraEstado;
```

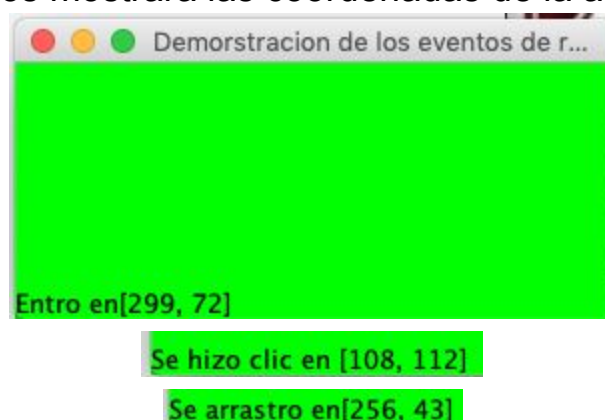
mouseCliked ---> mouseClicked

```
public void mouseClicked(MouseEvent evento){  
    barraEstado.setText("Se hizo clic en [" + evento.getX() + ", " + evento.getY() + "]);  
}
```

Demorstracion ---> Demostracion

```
super("Demostracion de los eventos de raton");  
  
barraEstado = new JLabel();  
add(barraEstado, BorderLayout.SOUTH);
```

Cuando se compilo el programa se abre una ventana nueva, donde puede mostrar las coordenadas desde donde ingresa el mouse, además de donde se hace un click o si se mantiene presionado o si solo se presiona, en todos estos casos mostrará las coordenadas de la acción.



Actividad 4 - ¿Qué temas del curso revisados hasta ahora se ocupan

## **en esta práctica?**

Para realizar las clases en esta práctica, se implementan conceptos que vimos en los temas de herencia, polimorfismo, mutadores, constructores, el uso de campos y atributos dentro de la clase.

Principalmente los conceptos de herencia y polimorfismo, se requieren, ya que se requiere que las clases con las que se esté trabajando implementen sobreescritura de métodos de distintas clases, para trabajar con una interfaz gráfica y modificar el comportamiento, por ejemplo de las acciones que se realizan después de revisar la entrada de los periféricos del usuario, de esperar que presione algún botón o seleccione algún objeto de una lista, etc.

## **Conclusiones:**

### ***Cruz Rangel Leonardo Said:***

Estuvo entretenida la práctica, descubrí que me gusta más trabajar con entornos gráficos que con la terminal, esto porque en la terminal no ves un resultado que pueda ser usado y con los entornos gráficos puedes ver lo que haz hecho con un mayor nivel de satisfacción.

### ***Ibañez Guzman Osvaldo:***

En esta práctica se vio la importancia, además de la utilidades de las GUI para el mejor manejo y vista de un programa, ya que muchas algo muy importante es la vista que se le dé al usuario.

### ***Maya Ortega María Fernanda:***

Es importante recordar que no todos los usuarios que utilicen algún programa serán programadores o personas expertas en el manejo de estos, es por eso que la interfaz gráfica es muy importante para que estos usuarios puedan hacer un mejor uso del programa, sin importar la complejidad. Me parece muy interesante que java tenga clases para implementar estas interfaces de manera sencilla y con muchas utilidades, para mejorar aspectos como los botones, colores, tamaños, etc.

### ***van der Werff Vargas Pieter Alexander:***

Esta práctica es un poco más visual y de analizar, más que requiera de pensamiento para realizar algún programa, sin embargo es interesante analizar cómo funcionan las clases y métodos que tiene java para implementar una interfaz gráfica. Es importante además de que funcione bien un programa, que sea sencillo e intuitivo de manejar para el usuario, dependiendo de su aplicación, ya que si resulta muy complicado no se tendrá todo el aprovechamiento que pueda tener un backend bien estructurado.