Índice:

- Interfaces visuales (componentes Swing)

Hasta ahora hemos resuelto todos los algoritmos haciendo las salidas a través de una consola en modo texto. La realidad que es muy común la necesidad de hacer la entrada y salida de datos mediante una interfaz más amigables con el usuario.

En Java existen varias librerías de clase para implementar interfaces visuales. Utilizaremos las componentes Swing.

Problema 1:

Confeccionar el programa "Hola Mundo" utilizando una interfaz gráfica de usuario.

Programa:

```
import javax.swing.*;
public class Formulario extends JFrame{
    private JLabel label1;
    public Formulario() {
        setLayout(null);
        label1=new JLabel("Hola Mundo.");
        label1.setBounds(10,20,200,30);
        add(label1);
    }

    public static void main(String[] ar) {
        Formulario formulario1=new Formulario();
        formulario1.setBounds(10,10,400,300);
        formulario1.setVisible(true);
    }
}
```

Hasta ahora habíamos utilizado la clase Scanner para hacer la entrada de datos por teclado. Dicha clase debemos importarla en nuestro programa con la sintaxis:

```
import java.util.Scanner;
```

Otra sintaxis para importarla es:

```
import java.util.*;
```

Si disponemos un * indicamos que importe todas las clases del paquete java.util.

Ahora bien las componentes Swing hay que importarlas del paquete javax.swing. Cuando debemos importar varias componentes de un paquete es más conveniente utilizar el asterisco que indicar cada clase a importar:

```
import javax.swing.JFrame;
import javax.swing.JLabel;
```

En lugar de las dos líneas anteriores es mejor utilizar la sintaxis:

```
import javax.swing.*;
```

La clase JFrame encapsula el concepto de una ventana. Luego para implementar una aplicación que muestre una ventana debemos plantear una clase que herede de la clase JFrame:

```
public class Formulario extends JFrame{
```

Con la sintaxis anterior estamos indicando que que la clase Formulario hereda todos los métodos y propiedades de la clase JFrame.

Para mostrar un texto dentro de una ventana necesitamos requerir la colaboración de la clase JLabel (que tiene por objetivo mostrar un texto dentro de un JFrame)

Definimos luego como atributo de la clase un objeto de la clase JLabel:

```
    private JLabel label1;
```

En el constructor de la clase llamamos al método heredado de la clase JFrame llamado setLayout y le pasamos como parámetro un valor null, con esto estamos informándole a la clase JFrame que utilizaremos posicionamiento absoluto para los controles visuales dentros del JFrame.

```
    public Formulario() {
        setLayout(null);
```

Luego tenemos que crear el objeto de la clase JLabel y pasarle como parámetro al constructor el texto a mostrar:

```
        label1=new JLabel("Hola Mundo.");
```

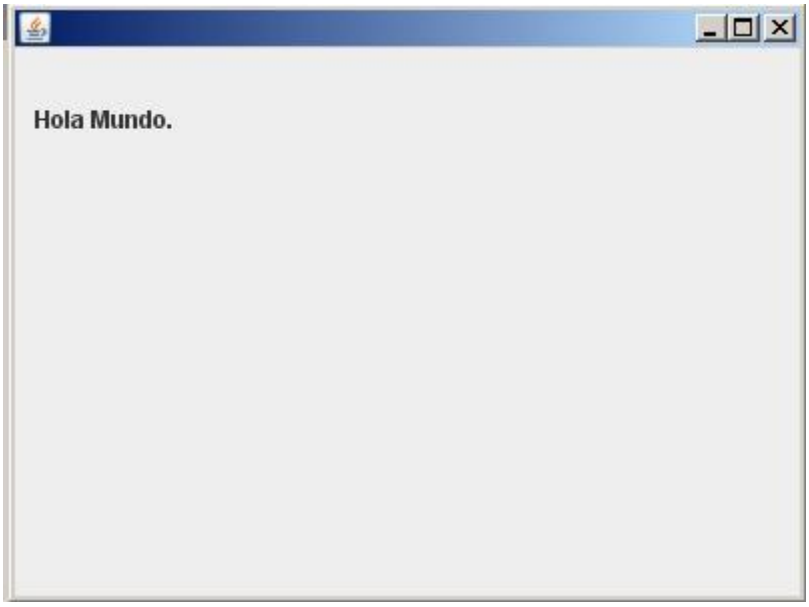
Ubicamos al objeto de la clase JLabel llamando al método setBounds, este requiere como parámetros la columna, fila, ancho y alto del JLabel. Finalmente llamamos al método add (metodo heredado de la clase JFrame) que tiene como objetivo añadir el control JLabel al control JFrame.

```
        label1=new JLabel("Hola Mundo.");
        label1.setBounds(10,20,200,30);
        add(label1);
    }
```

Finalmente debemos codificar la main donde creamos un objeto de la clase Formulario, llamamos al método setBounds para ubicar y dar tamaño al control y mediante el método setVisible hacemos visible el JFrame:

```
    public static void main(String[] ar) {
        Formulario formulario1=new Formulario();
        formulario1.setBounds(10,10,400,300);
        formulario1.setVisible(true);
    }
```

Cuando ejecutamos nuestro proyecto tenemos como resultado una ventana similar a esta:


Hola Mundo.

- Swing - JFrame

La componente básica que requerimos cada vez que implementamos una interfaz visual con la libraría Swing es la clase JFrame. Esta clase encapsula una Ventana clásica de cualquier sistema operativo con entorno gráfico (Windows, OS X, Linux etc.)

Hemos dicho que esta clase se encuentra en el paquete javax.swing y como generalmente utilizamos varias clases de este paquete luego para importarla utilizamos la sintaxis:

```
import javax.swing.*;
```

Podemos hacer una aplicación mínima con la clase JFrame:

```
import javax.swing.JFrame;
public class Formulario {
    public static void main(String[] ar) {
        JFrame f=new JFrame();
        f.setBounds(10,10,300,200);
        f.setVisible(true);
    }
}
```

Como vemos importamos la clase JFrame del paquete javax.swing:

```
import javax.swing.JFrame;
```

y luego en la main definimos y creamos un objeto de la clase JFrame (llamando luego a los métodos setBounds y setVisible):

```
    public static void main(String[] ar) {
        JFrame f=new JFrame();
        f.setBounds(10,10,300,200);
        f.setVisible(true);
    }
```

Pero esta forma de trabajar con la clase JFrame es de poca utilidad ya que rara vez necesitemos implementar una aplicación que muestre una ventana vacía.

Lo más correcto es plantear una clase que herede de la clase JFrame y extienda sus responsabilidades agregando botones, etiquetas, editores de línea etc.

Entonces la estructura básica que emplearemos para crear una interfaz visual será:

Programa:

```
import javax.swing.*;
public class Formulario extends JFrame{
    public Formulario() {
        setLayout(null);
    }
```

```
    public static void main(String[] ar) {
        Formulario formulario1=new Formulario();
        formulario1.setBounds(10,20,400,300);
        formulario1.setVisible(true);
    }
}
```

Importamos el paquete donde se encuentra la clase JFrame:

```
import javax.swing.*;
```

Planteamos una clase que herede de la clase JFrame:

```
public class Formulario extends JFrame{
```

En el constructor indicamos que ubicaremos los controles visuales con coordenadas absolutas mediante la desactivación del Layout heredado (más adelante veremos otras formas de ubicar los controles visuales dentro del JFrame):

```
    public Formulario() {
        setLayout(null);
    }
```

En la main creamos un objeto de la clase y llamamos a los métodos setBounds y setVisible:

```
    public static void main(String[] ar) {
        Formulario formulario1=new Formulario();
        formulario1.setBounds(10,20,400,300);
        formulario1.setVisible(true);
    }
```

El método setBounds ubica el JFrame (la ventana) en la columna 10, fila 20 con un ancho de 400 píxeles y un alto de 300.

Debemos llamar al método setVisible y pasarle el valor true para que se haga visible la ventana.

Problemas propuestos

1. Crear una ventana de 1024 píxeles por 800 píxeles. Luego no permitir que el operador modifique el tamaño de la ventana. Sabiendo que hacemos visible al JFrame llamando la método setVisible pasando el valor true, existe otro método llamado setResizable que también requiere como parámetro un valor true o false.

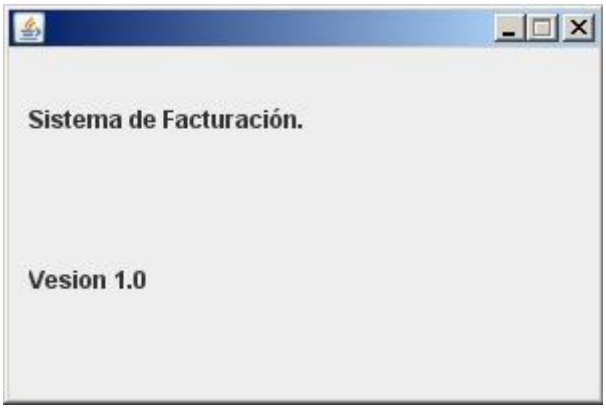
- Swing - JLabel

Veamos la segunda componente de la librería swing llamada JLabel.

La clase JLabel nos permite mostrar básicamente un texto.

Problema 1:

Confeccionar una ventana que muestre el nombre de un programa en la parte superior y su número de versión en la parte inferior. No permitir modificar el tamaño de la ventana en tiempo de ejecución.



Programa:

```
import javax.swing.*;
public class Formulario extends JFrame {
    private JLabel label1,label2;
    public Formulario() {
        setLayout(null);
        label1=new JLabel("Sistema de Facturación.");
        label1.setBounds(10,20,300,30);
        add(label1);
        label2=new JLabel("Vesion 1.0");
        label2.setBounds(10,100,100,30);
        add(label2);
    }

    public static void main(String[] ar) {
        Formulario formulario1=new Formulario();
        formulario1.setBounds(0,0,300,200);
        formulario1.setResizable(false);
        formulario1.setVisible(true);
    }
}
```

Importamos el paquete javax.swing donde se encuentran definidas las clase JFrame y JLabel:

```
import javax.swing.*;
```

Heredamos de la clase de JFrame:

```
public class Formulario extends JFrame {
```

Definimos dos atributos de la clase JLabel:

```
    private JLabel label1,label2;
```

En el constructor creamos las dos JLabel y las ubicamos llamando al método setBounds, no hay que olvidar de llamar al método add que añade la JLabel al JFrame:

```
    public Formulario() {
        setLayout(null);
        label1=new JLabel("Sistema de Facturación.");
        label1.setBounds(10,20,300,30);
        add(label1);
        label2=new JLabel("Vesion 1.0");
        label2.setBounds(10,100,100,30);
        add(label2);
    }
```

Por último en la main creamos un objeto de la clase que acabamos de codificar, llamamos al setBounds para ubicar y dar tamaño al JFrame, llamamos al método setResizable pasando el valor false para no permitir modificar el tamaño del JFrame en tiempo de ejecución y finalmente llamamos al método setVisible para que se visualice el JFrame:

```
    public static void main(String[] ar) {
        Formulario formulario1=new Formulario();
        formulario1.setBounds(0,0,300,200);
        formulario1.setResizable(false);
        formulario1.setVisible(true);
    }
```

Problemas propuestos

1. Crear tres objetos de la clase JLabel, ubicarlos uno debajo de otro y mostrar nombres de colores.

- Swing - JButton

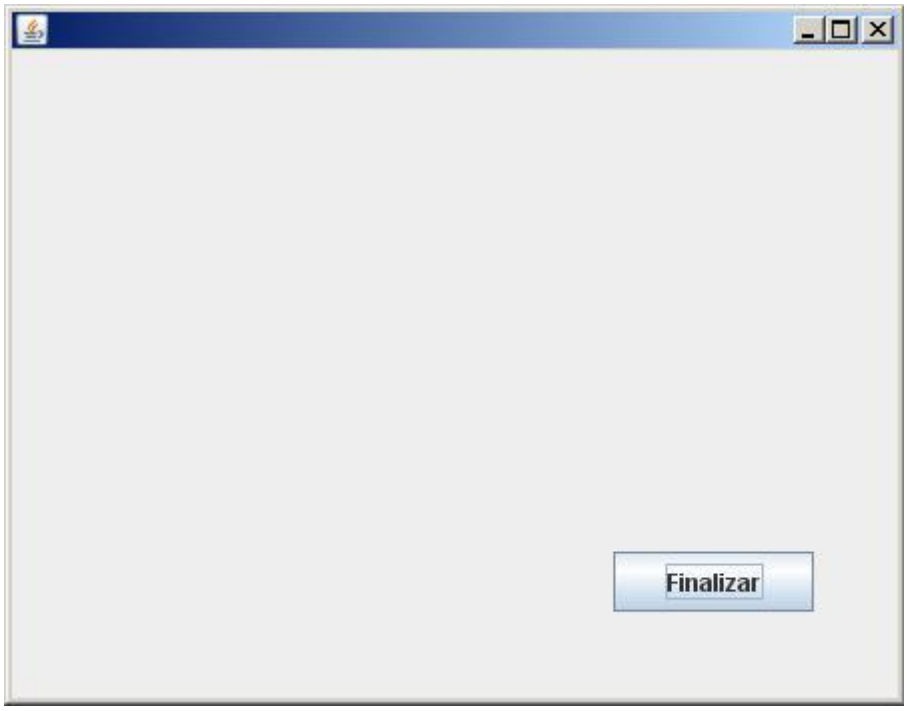
El tercer control visual de uso muy común es el que provee la clase JButton. Este control visual muestra un botón.

El proceso para añadir botones a un control JFrame es similar a añadir controles de tipo JLabel.

Ahora veremos la captura de eventos con los controles visuales. Uno de los eventos más comunes es cuando hacemos clic sobre un botón.
Java implementa el concepto de interfaces para poder llamar a métodos de una clase existente a una clase desarrollada por nosotros.

Problema 1:

Confeccionar una ventana que muestre un botón. Cuando se presione finalizar la ejecución del programa Java.



Programa:

```
import javax.swing.*;
import java.awt.event.*;
public class Formulario extends JFrame implements ActionListener {
    JButton boton1;
    public Formulario() {
        setLayout(null);
        boton1=new JButton("Finalizar");
        boton1.setBounds(300,250,100,30);
        add(boton1);
```

```
        boton1.addActionListener(this);
    }

    public void actionPerformed(ActionEvent e) {
        if (e.getSource()==boton1) {
            System.exit(0);
        }
    }

    public static void main(String[] ar) {
        Formulario formulario1=new Formulario();
        formulario1.setBounds(0,0,450,350);
        formulario1.setVisible(true);
    }
}
```

La mecánica para atrapar el clic del objeto de la clase JButton se hace mediante la implementación de una interface. Una interface es un protocolo que permite la comunicación entre dos clases. Una interface contiene uno o más cabecera de métodos, pero no su implementación. Por ejemplo la interface ActionListener tiene la siguiente estructura:

```
interface ActionListener {
    public void actionPerformed(ActionEvent e) {
}
```

Luego las clases que implementen la interface ActionListener deberán especificar el algorítmo del método actionPerformed.

Mediante el concepto de interfaces podemos hacer que desde la clase JButton se puede llamar a un método que implementamos en nuestra clase.

Para indicar que una clase implementará una interface lo hacemos en la declaración de la clase con la sintaxis:

```
public class Formulario extends JFrame implements ActionListener {
```

Con esto estamos diciendo que nuestra clase implementa la interface ActionListener, luego estamos obligados a codificar el método actionPerformed.

Definimos un objeto de la clase JButton:

```
    JButton boton1;
```

En el constructor creamos el objeto de la clase JButton y mediante la llamada del método addActionListener le pasamos la referencia del objeto de la clase JButton utilizando la palabra clave this (this almacena la dirección de memoria donde se almacena el objeto de la clase JFrame, luego mediante dicha dirección podemos llamar al método actionPerformed):

```
    public Formulario() {
        setLayout(null);
        boton1=new JButton("Finalizar");
        boton1.setBounds(300,250,100,30);
```

```
        add(boton1);
        boton1.addActionListener(this);
    }
```

El método actionPerformed (este método de la interface ActionListener debe implementarse obligatoriamente, en caso de no codificarlo o equivocarnos en su nombre aparecerá un mensaje de error en tiempo de compilación de nuestro programa.

El método actionPerformed se ejecutará cada vez que hagamos clic sobre el objeto de la clase JButton.

La interface ActionListener y el objeto de la clase ActionEvent que llega como parámetro están definidos en el paquete:

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

Es decir que cada vez que se presiona el botón desde la clase JButton se llama al método actionPerformed y recibe como parámetro un objeto de la clase ActionEvent.

En el método actionPerformed mediante el acceso al método getSource() del objeto que llega como parámetro podemos analizar que botón se presionó:

```
    public void actionPerformed(ActionEvent e) {
        if (e.getSource()==boton1) {
            System.exit(0);
        }
    }
```

Si se presionón el boton1 luego el if se verifica verdadero y por lo tanto llamando al método exit de la clase System se finaliza la ejecución del programa.

La main no varía en nada con respecto a problemas anteriores.

Problema 2:

Confeccionar una ventana que contenga tres objetos de la clase JButton con las etiquetas "1", "2" y "3". Al presionarse cambiar el título del JFrame indicando cuál botón se presionó.

Programa:

```
import javax.swing.*;
import java.awt.event.*;
public class Formulario extends JFrame implements ActionListener{
    private JButton boton1,boton2,boton3;
    public Formulario() {
        setLayout(null);
        boton1=new JButton("1");
        boton1.setBounds(10,100,90,30);
        add(boton1);
        boton1.addActionListener(this);
        boton2=new JButton("2");
        boton2.setBounds(110,100,90,30);
        add(boton2);
        boton2.addActionListener(this);
        boton3=new JButton("3");
        boton3.setBounds(210,100,90,30);
        add(boton3);
        boton3.addActionListener(this);
    }

    public void actionPerformed(ActionEvent e) {
        if (e.getSource()==boton1) {
            setTitle("boton 1");
        }
        if (e.getSource()==boton2) {
            setTitle("boton 2");
        }
        if (e.getSource()==boton3) {
            setTitle("boton 3");
        }
    }

    public static void main(String[] ar){
        Formulario formulario1=new Formulario();
        formulario1.setBounds(0,0,350,200);
        formulario1.setVisible(true);
    }
}
```

Debemos declarar 3 objetos de la clase JButton:

```
    private JButton boton1,boton2,boton3;
```

En el constructor creamos los tres objetos de la clase JButton y los ubicamos dentro del control JFrame (también llamamos al método addActionListener para enviarle la dirección del objeto de la clase Formulario):

```
    public Formulario() {
        setLayout(null);
        boton1=new JButton("1");
        boton1.setBounds(10,100,90,30);
        add(boton1);
        boton1.addActionListener(this);
        boton2=new JButton("2");
        boton2.setBounds(110,100,90,30);
        add(boton2);
        boton2.addActionListener(this);
```

```
        boton3=new JButton("3");
        boton3.setBounds(210,100,90,30);
        add(boton3);
        boton3.addActionListener(this);
    }
```

Cuando se presiona alguno de los tres botones se ejecuta el método actionPerformed y mediante tres if verificamos cual de los botones se presionó:

```
    public void actionPerformed(ActionEvent e) {
        if (e.getSource()==boton1) {
            setTitle("boton 1");
        }
        if (e.getSource()==boton2) {
            setTitle("boton 2");
        }
        if (e.getSource()==boton3) {
            setTitle("boton 3");
        }
    }
```

Según el botón presionado llamamos al método setTitle que se trata de un método heredado de la clase JFrame y que tiene por objetivo mostrar un String en el título de la ventana.

Problemas propuestos

1. Disponer dos objetos de la clase JButton con las etiquetas: "varón" y "mujer", al presionarse mostrar en la barra de títulos del JFrame la etiqueta del botón presionado.

- Swing - JTextField

Así como podríamos decir que el control JLabel remplaza a la salida estándar System.out.print, el control JTextField cumple la función de la clase Scanner para la entrada de datos.

El control JTextField permite al operador del programa ingresar una cadena de caracteres por teclado.

Problema 1:

Confeccionar un programa que permita ingresar el nombre de usuario y cuando se presione un botón mostrar el valor ingresado en la barra de títulos del JFrame.

Programa:

```
import javax.swing.*;
import java.awt.event.*;
public class Formulario extends JFrame implements ActionListener{
    private JTextField textfield1;
    private JLabel label1;
    private JButton boton1;
    public Formulario() {
        setLayout(null);
        label1=new JLabel("Usuario:");
        label1.setBounds(10,10,100,30);
        add(label1);
        textfield1=new JTextField();
        textfield1.setBounds(120,10,150,20);
        add(textfield1);
        boton1=new JButton("Aceptar");
        boton1.setBounds(10,80,100,30);
        add(boton1);
        boton1.addActionListener(this);
    }

    public void actionPerformed(ActionEvent e) {
        if (e.getSource()==boton1) {
            String cad=textfield1.getText();
            setTitle(cad);
        }
    }

    public static void main(String[] ar) {
```

```
        Formulario formulario1=new Formulario();
        formulario1.setBounds(0,0,300,150);
        formulario1.setVisible(true);
    }
}
```

Definimos los tres objetos que colaboran con nuestra aplicación:

```
public class Formulario extends JFrame implements ActionListener{
    private JTextField textfield1;
    private JLabel label1;
    private JButton boton1;
```

En el constructor creamos los tres objetos y los ubicamos:

```
    public Formulario() {
        setLayout(null);
        label1=new JLabel("Usuario:");
        label1.setBounds(10,10,100,30);
        add(label1);
        textfield1=new JTextField();
        textfield1.setBounds(120,10,150,20);
        add(textfield1);
        boton1=new JButton("Aceptar");
        boton1.setBounds(10,80,100,30);
        add(boton1);
        boton1.addActionListener(this);
    }
```

En el método actionPerformed se verifica si se presionó el objeto JButton, en caso afirmativo extraemos el contenido del control JTextField mediante el método getText:

```
    public void actionPerformed(ActionEvent e) {
        if (e.getSource()==boton1) {
            String cad=textfield1.getText();
            setTitle(cad);
        }
    }
```

En la variable auxiliar cad almacenamos temporalmente el contenido del JTextField y seguidamente actualizamos el título del control JFrame.

Problema 2:

Confeccionar un programa que permita ingresar dos números en controles de tipo JTextField, luego sumar los dos valores ingresados y mostrar la suma en la barra del título del control JFrame.

Programa:

```
import javax.swing.*;
import java.awt.event.*;
public class Formulario extends JFrame implements ActionListener{
    private JTextField textfield1,textfield2;
    private JButton boton1;
    public Formulario() {
        setLayout(null);
        textfield1=new JTextField();
        textfield1.setBounds(10,10,100,30);
        add(textfield1);
        textfield2=new JTextField();
        textfield2.setBounds(10,50,100,30);
        add(textfield2);
        boton1=new JButton("Sumar");
        boton1.setBounds(10,90,100,30);
        add(boton1);
        boton1.addActionListener(this);
    }

    public void actionPerformed(ActionEvent e) {
        if (e.getSource()==boton1) {
            String cad1=textfield1.getText();
            String cad2=textfield2.getText();
            int x1=Integer.parseInt(cad1);
            int x2=Integer.parseInt(cad2);
            int suma=x1+x2;
            String total=String.valueOf(suma);
            setTitle(total);
        }
    }

    public static void main(String[] ar) {
        Formulario formulario1=new Formulario();
        formulario1.setBounds(0,0,140,150);
        formulario1.setVisible(true);
    }
}
```

Definimos los tres objetos:

```
public class Formulario extends JFrame implements ActionListener{
    private JTextField textfield1,textfield2;
    private JButton boton1;
```

En el método actionPerformed es donde debemos sumar los valores ingresados en los controles de tipo JTextField. Para extraer el contenido de los controles debemos extraerlos con el método getText:

```
String cad1=textfield1.getText();
String cad2=textfield2.getText();
```

Como debemos sumar numéricamente los valores ingresados debemos convertir el contenido de las variables cad2 y cad2 a tipo de dato int:

```
int x1=Integer.parseInt(cad1);
int x2=Integer.parseInt(cad2);
```

El método parseInt de la clase Integer retorna el contenido de cad1 convertido a int (provoca un error si ingresamos caracteres en el control JTextField en lugar de números)

Una vez que tenemos los dos valores en formato numérico procedemos a sumarlos y almacenar el resultado en otra variable auxiliar:

```
int suma=x1+x2;
```

Ahora tenemos que mostrar el valor almacenado en suma en la barra de títulos del control JFrame, pero como el método setTitle requiere un String como parámetro debemos convertirlo a tipo String:

```
String total=String.valueOf(suma);
setTitle(total);
```

Como veremos de acá en adelante es muy común la necesidad de convertir String a enteros y enteros a String:

De String a int:

```
int x1=Integer.parseInt(cad1);
```

De int a String:

```
String total=String.valueOf(suma);
```

Problemas propuestos

1. Ingresar el nombre de usuario y clave en controles de tipo JTextField. Si se ingresa las cadena (usuario: juan, clave=abc123) luego mostrar en el título del JFrame el mensaje "Correcto" en caso contrario mostrar el mensaje "Incorrecto".

- Swing - JTextArea

El control de tipo JTextArea permite ingresar múltiples líneas, a diferencia del control de tipo JTextField.

Problema 1:

Confeccionar un programa que permita ingresar un mail en un control de tipo JTextField y el cuerpo del mail en un control de tipo JTextArea.

Programa:

```
import javax.swing.*;
public class Formulario extends JFrame{
    private JTextField textfield1;
    private JTextArea textarea1;
    public Formulario() {
        setLayout(null);
        textfield1=new JTextField();
        textfield1.setBounds(10,10,200,30);
        add(textfield1);
        textarea1=new JTextArea();
        textarea1.setBounds(10,50,400,300);
        add(textarea1);
    }

    public static void main(String[] ar) {
        Formulario formulario1=new Formulario();
        formulario1.setBounds(0,0,540,400);
        formulario1.setVisible(true);
```

```
    }
}
```

Como vemos crear un control de tipo JTextArea es similar a la creación de controles de tipo JTextField:

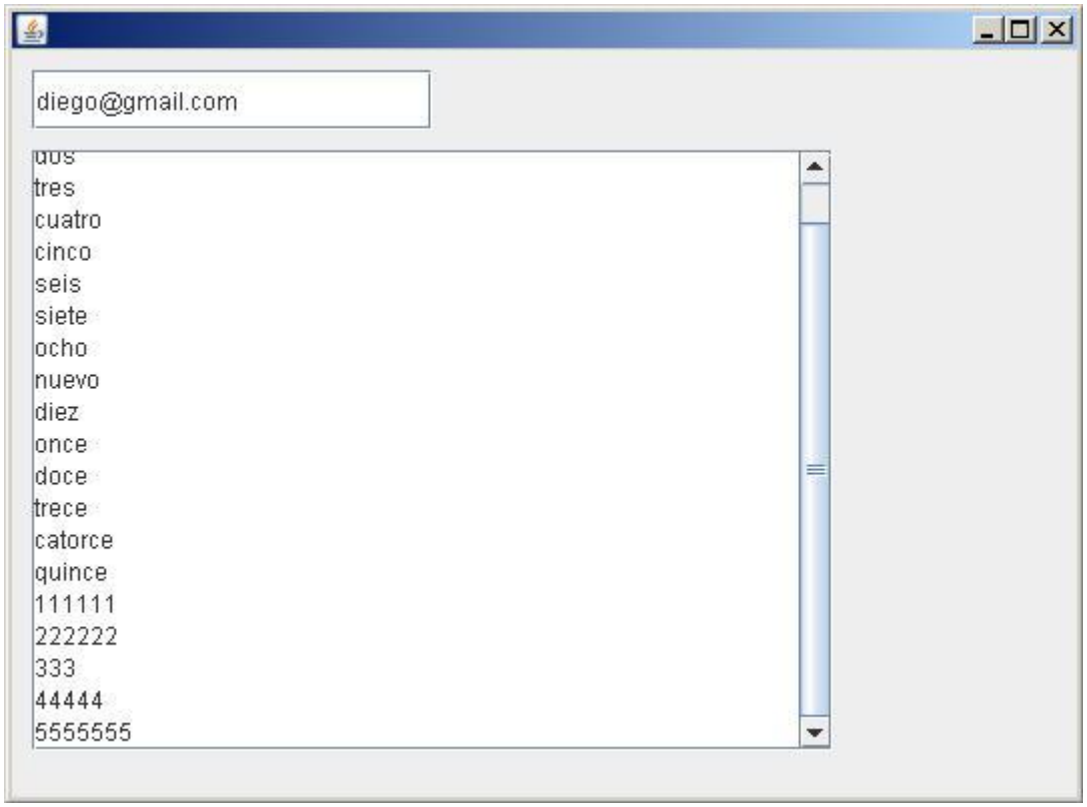
```
        textarea1=new JTextArea();
        textarea1.setBounds(10,50,400,300);
        add(textarea1);
```

El inconveniente que tiene este control es que si ingresamos más texto que el que puede visualizar no aparecen las barras de scroll y no podemos ver los caracteres tipeados.

Para salvar el problema anterior debemos crear un objeto de la clase JScrollPane y añadir en su interior el objeto de la clase JTextArea, luego el programa definitivo es el siguiente:

```
import javax.swing.*;
public class Formulario extends JFrame{
    private JTextField textfield1;
    private JScrollPane scrollpane1;
    private JTextArea textarea1;
    public Formulario() {
        setLayout(null);
        textfield1=new JTextField();
        textfield1.setBounds(10,10,200,30);
        add(textfield1);
        textarea1=new JTextArea();
        scrollpane1=new JScrollPane(textarea1);
        scrollpane1.setBounds(10,50,400,300);
        add(scrollpane1);
    }

    public static void main(String[] ar) {
        Formulario formulario1=new Formulario();
        formulario1.setBounds(0,0,540,400);
        formulario1.setVisible(true);
    }
}
```



Declaramos los dos objetos:

```
    private JScrollPane scrollpane1;
    private JTextArea textarea1;
```

Primero creamos el objeto de la clase JTextArea:

```
        textarea1=new JTextArea();
```

Seguidamente creamos el objeto de la clase JScrollPane y le pasamos como parámetro el objeto de la clase JTextArea:

```
        scrollpane1=new JScrollPane(textarea1);
```

Definimos la posición y tamaño del control de tipo JScrollPane (y no del control JTextArea):

```
        scrollpane1.setBounds(10,50,400,300);
```

Finalmente añadimos el control de tipo JScrollPane al JFrame:

```
        add(scrollpane1);
```

Problema 2:

Confeccionar un programa que permita ingresar en un control de tipo JTextArea una carta. Luego al presionar un botón mostrar un mensaje si la carta contiene el String "argentina".

Programa:

```
import javax.swing.*;
import java.awt.event.*;
public class Formulario extends JFrame implements ActionListener{
    private JScrollPane scrollpane1;
    private JTextArea textarea1;
    private JButton boton1;
    public Formulario() {
        setLayout(null);
        textarea1=new JTextArea();
        scrollpane1=new JScrollPane(textarea1);
        scrollpane1.setBounds(10,10,300,200);
        add(scrollpane1);
        boton1=new JButton("Verificar");
        boton1.setBounds(10,260,100,30);
        add(boton1);
        boton1.addActionListener(this);
    }

    public void actionPerformed(ActionEvent e) {
        if (e.getSource()==boton1) {
            String texto=textarea1.getText();
            if (texto.indexOf("argentina")!=-1) {
                setTitle("Si contiene el texto \"argentina\"");
            } else {
                setTitle("No contiene el texto \"argentina\"");
            }
        }
    }

    public static void main(String[] ar) {
        Formulario formulario1=new Formulario();
        formulario1.setBounds(0,0,400,380);
        formulario1.setVisible(true);
    }
}
```

Cuando se presiona el botón se ejecuta el método actionPerformed y procedemos a extraer el contenido del control TextArea a través del método getText:

```
            String texto=textarea1.getText();
```

Luego mediante el método indexOf de la clase String verificamos si el String "argentina" está contenido en la variable texto:

```
            if (texto.indexOf("argentina")!=-1) {
                setTitle("Si contiene el texto \"argentina\"");
            } else {
                setTitle("No contiene el texto \"argentina\"");
            }
```

Si queremos introducir una comilla doble dentro de un String de Java debemos antecederle la barra invertida (luego dicho caracter no se lo considera parte del String):

```
setTitle("Si contiene el texto \"argentina\"");
```

Problemas propuestos

1. Disponer dos controles de tipo JTextArea, luego al presionar un botón verificar si tienen exactamente el mismo contenido.