Interfaces Gráficas de Usuario

Ejemplo de construcción de una aplicación con GUI

Clases e Interfaces

- Crearemos las siguientes clases/interfaces (los nombres cambiarán según la aplicación):
  - Modelo: El modelo será una o más clases .java
  - Controlador: Clase `ControladorXXXXX.java`
  - Vista: Genera la interfaz gráfica de usuario
    - Interfaz `VistaXXXXX.java`
      - Un método para registrar el controlador
      - Un método para cada una las acciones que el controlador tenga que realizar sobre la GUI
    - Clase `PanelXXXXXX.java`, que implementa `VistaXXXXX.java`
      - `JPanel` con todos los elementos gráficos de la GUI
  - Aplicación: Clase `AplicaciónXXXXXX.java`

Modelo

- Clase Modelo.java

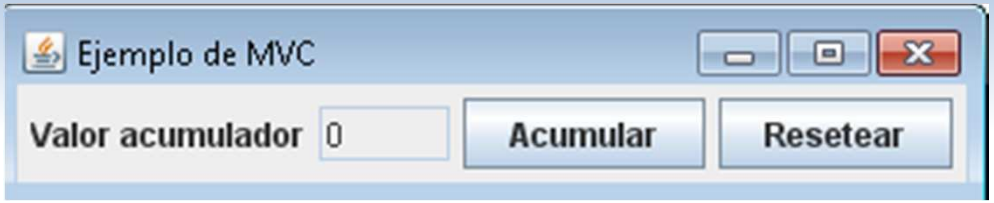
Modelo
- acumulador
+ Modelo() + acumular(int) + resetear(int) + int obtenerAcumulador()

```
public class Modelo {
    private int acumulador;

    public Modelo(){
        acumulador = 0;
    }
    public void acumular(int valor){
        acumulador = acumulador + valor;
    }
    public void resetear(){
        acumulador = 0;
    }
    public int obtenerAcumulador(){
        return acumulador;
    }
}
```

Vista – Vista.java

```
import java.awt.event.ActionListener;

public interface Vista {

//Definimos una constante por cada acción de la vista de
//la que el controlador tiene que estar pendiente
public static final String ACUMULAR = "ACUMULAR";
public static final String RESETEAR = "RESETEAR";

//Siempre tendremos este método para que el controlador
//pueda registrarse en la vista
public void controlador (ActionListener ctr);

//El resto de los métodos dependerá de cada aplicación.
//Son métodos para actualizar las distintas partes de la
//vista. Los llama el controlador
public void valorAcumulador(int valor);
}
```

Vista – Panel.java




```
import java.awt.FlowLayout;
import javax.swing.JPanel;
import javax.swing.JButton;
import javax.swing.JTextField;
import javax.swing.JLabel;
import java.awt.event.ActionListener;

@SuppressWarnings("serial")
public class Panel extends JPanel implements Vista{
        //Componentes que tendrá el panel
        JButton botonAcumular, botonResetear;
        JLabel etiquetaAcumulador;
        JTextField campoAcumulador;
```

Vista – Panel.java

```
... (Continúa de la página anterior)
//Constructor. Construimos la GUI en el constructor
public Panel(){
        //Paso 1. Establezco el esquema para la organización
        //de los elementos en el panel
        setLayout(new FlowLayout());

        //Paso 2. Construyo los componentes del panel
        botonAcumular = new JButton("Acumular");
        botonResetear = new JButton("Resetear");
        etiquetaAcumulador = new JLabel("Valor acumulador");
        campoAcumulador = new JTextField(4);
        campoAcumulador.setEditable(false);

        //Paso 3. Añado los componentes al panel
        add(etiquetaAcumulador);
        add(campoAcumulador);
        add(botonAcumular);
        add(botonResetear);
}
```

Vista – Panel.java

```
... (Continúa de la página anterior)
        //Implementamos interfaz Vista
        public void controlador (ActionListener ctr){
          /** Añadimos el controlador a los componentes de la vista
              que queremos controlar. Para cada componente que
              queramos controlar hay que hacer dos cosas:
                 1. Añadir el controlador a ese componente
                 2. Añadir un comando concreto para que luego
                    el controlador sepa que componente ha producido el
                    evento.
           */
           botonAcumular.addActionListener(ctr);
           botonAcumular.setActionCommand(Vista.ACUMULAR);

           botonResetear.addActionListener(ctr);
           botonResetear.setActionCommand(Vista.RESETEAR);
        }

        public void valorAcumulador(int valor){
           campoAcumulador.setText(Integer.toString(valor));
        }
}
```

Controlador

```
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class Controlador implements ActionListener {
        // Referencia a la Vista
        private Vista vista;

        // Referencia al Modelo
        private Modelo modelo;

        //En el constructor recibimos la vista y el modelo
        public Controlador(Vista vista, Modelo modelo) {
                this.modelo = modelo;
                this.vista = vista;
                //Actualizamos la vista para reflejar los valores
                //iniciales en el modelo
                vista.valorAcumulador(modelo.obtenerAcumulador());
}
```

Controlador

```
... (Continúa de la página anterior)
//Controlamos el modelo y la vista a tenor de los distintos
//eventos.
public void actionPerformed(ActionEvent ea) {
    //Tenemos una rama en el if por cada acción definida
    //antes en la vista
    if (ea.getActionCommand().equals(Vista.ACUMULAR)) {
        //Actualizamos el modelo y/o la vista
        modelo.acumular(1); //Acumular aquí es incrementar en 1
        vista.valorAcumulador(modelo.obtenerAcumulador());

    } else if (ea.getActionCommand().equals(Vista.RESETEAR)) {
        //Actualizamos el modelo y/o la vista
        modelo.resetear(); //Reseteamos el contador
        vista.valorAcumulador(modelo.obtenerAcumulador());
    }
}
}
```

Aplicación

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

public class Aplicacion {
    public static void main(String[] args) {
        //Creamos la vista
        Vista panel = new Panel();
        //Creamos el modelo
        Modelo modelo = new Modelo();
        //Creamos el controlador pasándole la vista y el modelo
        Controlador ctrl = new Controlador(panel,modelo);
        //Le decimos a la vista qué controlador va a controlar
        //los eventos que se produzcan
        panel.controlador(ctrl);

        //Creamos el "frame" principal de la aplicación
        JFrame ventana = new JFrame("Ejemplo de MVC");
        ventana.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        ventana.setContentPane((JPanel)panel);
        ventana.pack();
        ventana.setVisible(true);
}}
```