Estructura de las Aplicaciones Orientadas a Objetos

El patrón Modelo-Vista-Controlador (MVC)

Programación Orientada a Objetos
Facultad de Informática

Juan Pavón Mestras
Dep. Ingeniería del Software e Inteligencia Artificial
Universidad Complutense Madrid

El patrón MVC

- MVC: Modelo-Vista-Controlador
- Es un patrón de *arquitectura* de las aplicaciones software
  - Separa la lógica de negocio de la interfaz de usuario
    - Facilita la evolución por separado de ambos aspectos
    - Incrementa reutilización y flexibilidad

El patrón MVC

- Historia
  - Descrito por primera vez en 1979 para Smalltalk
    - http://heim.ifi.uio.no/~trygver/themes/mvc/mvc-index.html
  - Utilizado en múltiples frameworks
    - Java Swing
    - Java Enterprise Edition (J2EE)
    - XForms (Formato XML estándar del W3C para la especificación de un modelo de proceso de datos XML e interfaces de usuario como formularios web)
    - GTK+ (escrito en C, toolkit creado por Gnome para construir aplicaciones gráficas, inicialmente para el sistema X Window)
    - ASP.NET MVC Framework (Microsoft)
    - Google Web Toolkit (GWT, para crear aplicaciones Ajax con Java)
    - Apache Struts (framework para aplicaciones web J2EE)
    - Ruby on Rails (framework para aplicaciones web con Ruby)
    - Etc., etc., etc.

El patrón MVC

- Modelo-Vista-Controlador
  - Un modelo
  - Varias vistas
  - Varios controladores
  - Las vistas y los controladores suelen estar muy relacionados
    - Los controladores tratan los eventos que se producen en la interfaz gráfica (vista)
- Esta separación de aspectos de una aplicación da mucha flexibilidad al desarrollador

El patrón MVC

- Flujo de control
  1. El usuario realiza una acción en la interfaz
  2. El controlador trata el evento de entrada
     - Previamente se ha registrado
  3. El controlador notifica al modelo la acción del usuario, lo que puede implicar un cambio del estado del modelo (si no es una mera consulta)
  4. Se genera una nueva vista. La vista toma los datos del modelo
     - El modelo no tiene conocimiento directo de la vista
  5. La interfaz de usuario espera otra interacción del usuario, que comenzará otro nuevo ciclo

MVC en aplicaciones web

- Vista:
  - la página HTML
- Controlador:
  - código que obtiene datos dinámicamente y genera el contenido HTML
- Modelo:
  - la información almacenada en una base de datos o en XML
  - junto con las reglas de negocio que transforman esa información (teniendo en cuenta las acciones de los usuarios)

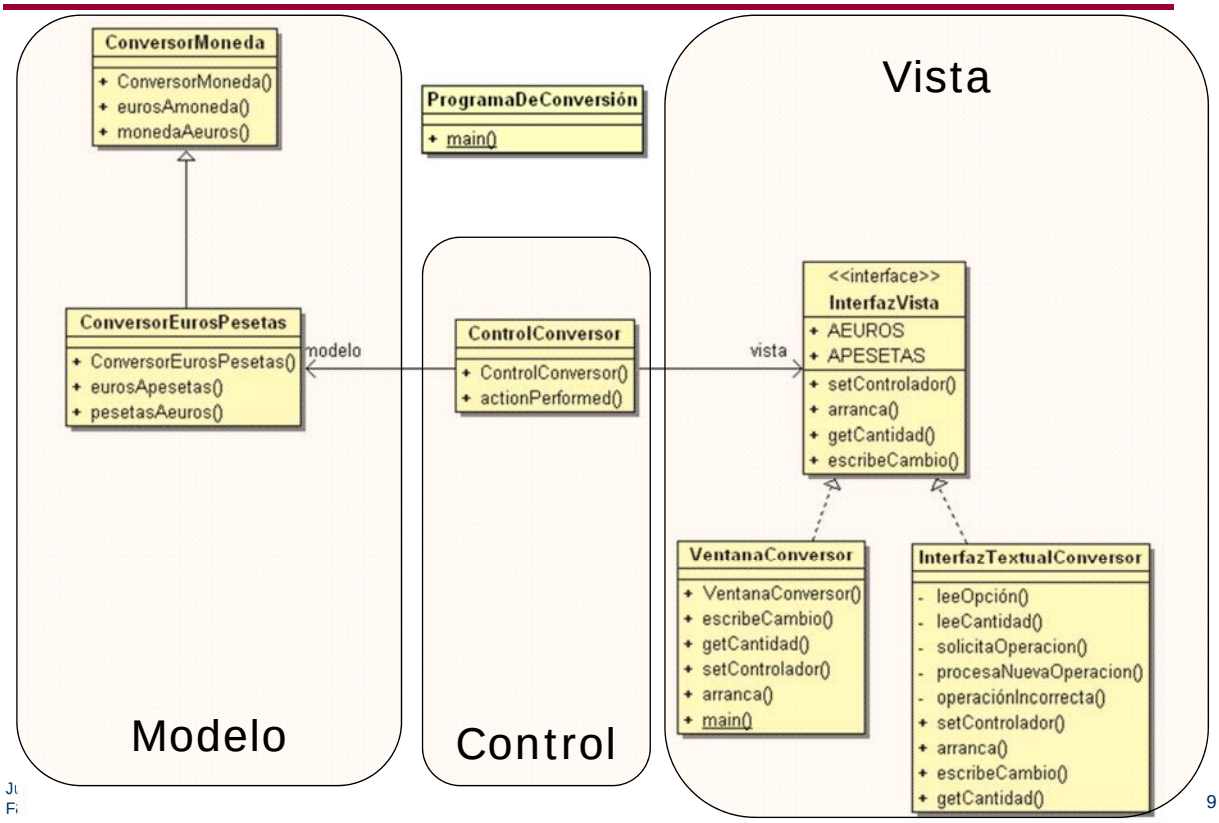
MVC en Java Swing

- Modelo:
  - El modelo lo realiza el desarrollador
- Vista:
  - Conjunto de objetos de clases que heredan de java.awt.Component
- Controlador:
  - El controlador es el thread de tratamiento de eventos, que captura y propaga los eventos a la vista y al modelo
  - Clases de tratamiento de los eventos (a veces como clases anónimas) que implementan interfaces de tipo EventListener (ActionListener, MouseListener, WindowListener, etc.)

Ejemplo: Calculadora de euros a pesetas

- Una clase sencilla que da operaciones para calcular la conversión entre euros y pesetas
  - En el constructor se indica el cambio
  - Métodos
    - Calcular pesetas de una cantidad en euros
    - Calcular euros de una cantidad en pesetas

Calculadora Euros-Pesetas



El modelo (1/2)

```
public class ConversorEuros {
   private double cambio;

   public ConversorEuros ( double valorCambio ) {
        // valor en la moneda de 1 euro
        cambio = valorCambio;
   }

   public double eurosAmoneda (double cantidad) {
        return cantidad * cambio;
   }

   public double monedaAeuros (double cantidad) {
        return cantidad / cambio;
   }
}
```

El modelo (2/2)

```
public class ConversorEurosPesetas extends ConversorEuros
{ // Adaptador de clase

   public ConversorEurosPesetas () {
        super(166.386D);
   }

   public double eurosApesetas(double cantidad) {
        return eurosAmoneda(cantidad);
   }

   public double pesetasAeuros(double cantidad) {
        return monedaAeuros(cantidad);
   }

}
```


El modelo (3/2)

```
public class ConversorEurosPesetas
{ // Adaptador de objetos

   private ConversorEuros conversor;

   public ConversorEurosPesetas () {
        Conversor = new ConversorEuros(166.386D);

   }

   public double eurosApesetas(double cantidad) {
        return conversor.eurosAmoneda(cantidad);
   }

   public double pesetasAeuros(double cantidad) {
        return conversor.monedaAeuros(cantidad);
   }
}
```

La Vista (1/4)

- Definimos una interfaz con las operaciones que el control puede necesitar para manipularla

```
public interface InterfazVista {
   void setControlador(ControlConversor c);
   void arranca();   // comienza la visualización

   double getCantidad();           // cantidad a convertir
   void escribeCambio(String s); //texto con la conversión

// Constantes que definen las posibles operaciones:
   static final String AEUROS="Pesetas a Euros";
   static final String APESETAS="Euros a Pesetas";
}
```


La Vista (2/4)

- Una ventana (JFrame) con
  - Un campo de texto (JTextField) para indicarla entrada
  - Una etiqueta (JLabel) para indicar el resultado de la conversión
  - Dos botones (JButton) para las dos operaciones

La Vista (3/4): Construcción de la ventana

```
public class VentanaConversor extends JFrame implements InterfazVista {
    private JButton convertirApesetas;
    private JButton convertirAeuros;
    private JTextField cantidad;
    private JLabel resultado;

    public VentanaConversor () {
          super("Conversor de Euros y Pesetas");
          setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

          JPanel panelPrincipal = new JPanel();
          panelPrincipal.setLayout(new BorderLayout(10,10));

          cantidad = new JTextField(8);
          JPanel panelaux = new JPanel(); panelaux.add(cantidad);
          panelPrincipal.add(panelaux, BorderLayout.NORTH);

          resultado = new JLabel("Indique una cantidad y pulse uno de los botones");
          JPanel panelaux2 = new JPanel(); panelaux2.add(resultado);
          panelPrincipal.add(panelaux2, BorderLayout.CENTER);

          convertirApesetas = new JButton("A pesetas");
          convertirApesetas.setActionCommand(APESETAS);
          convertirAeuros = new JButton("A euros");
          convertirAeuros.setActionCommand(AEUROS);
          JPanel botonera = new JPanel();
          botonera.add(convertirApesetas); botonera.add(convertirAeuros);
          panelPrincipal.add(botonera, BorderLayout.SOUTH);
          getContentPane().add(panelPrincipal);
}
```


La Vista (4/4)

```
// Métodos de la interfaz InterfazVista:
public void escribeCambio(String s) {
    resultado.setText(s);
}

public double getCantidad() {
    try {
          return Double.parseDouble(cantidad.getText());
    }
    catch (NumberFormatException e) {
          return 0.0D;
    }
}

public void setControlador(ControlConversor c) {
    convertirApesetas.addActionListener(c);
    convertirAeuros.addActionListener(c);
}

public void arranca() {
    pack();// coloca los componentes
    setLocationRelativeTo(null);// centra la ventana en la pantalla
    setVisible(true);// visualiza la ventana
}
```

El control

```
public class ControlConversor implements ActionListener {

    private InterfazVista vista;
    private ConversorEurosPesetas modelo;

    public ControlConversor(InterfazVista vista, ConversorEurosPesetas modelo)
    {
          this.vista = vista;
          this.modelo = modelo;
    }

    public void actionPerformed(ActionEvent evento) {
          double cantidad = vista.getCantidad();

          if ( evento.getActionCommand().equals(InterfazVista.AEUROS) ) {
                  vista.escribeCambio( cantidad + " pesetas son: “
                  + modelo.pesetasAeuros(cantidad) + " euros" );
          }
          else if ( evento.getActionCommand().equals(InterfazVista.APESETAS)) {
                  vista.escribeCambio( cantidad + " euros son: “
                  + modelo.eurosApesetas(cantidad) + " pesetas" );
          }
          else
                  vista.escribeCambio( "ERROR" );
    }
}
```


El programa

```
public class ProgramaDeConversión {
   public static void main(String[] args) {
// el modelo:
     ConversorEurosPesetas modelo = new ConversorEurosPesetas();

// la vista:
     InterfazVista vista = new VentanaConversor();

// y el control:
     ControlConversor control = new ControlConversor (vista,
   modelo);

// configura la vista
     vista.setControlador(control);

// y arranca la interfaz (vista):
     vista.arranca();
   }
}
```

Discusión

- El modelo
  - ¿Tiene algo de código que dependa de la vista o del controlador?
- El control
  - Manipula el modelo y gestiona la vista
- La vista
  - Tiene que implementar una interfaz predefinida para la aplicación
  - Tiene que configurar a quién le llegan los eventos que se produzcan sobre sus elementos

Otra vista

- Vista textual

```
Indica la operación que quiere realizar:
1: convertir euros a euros
2: convertir pesetas a pesetas
0: salir
1
Cantidad a convertir (formato 99.99): 2
2.0 pesetas son: 0.0120 euros
Indica la operación que quiere realizar:
1: convertir euros a euros
2: convertir pesetas a pesetas
0: salir
2
Cantidad a convertir (formato 99.99): 3
3.0 euros son: 499.158 pesetas
Indica la operación que quiere realizar:
1: convertir euros a euros
2: convertir pesetas a pesetas
0: salir
0
Adiós.
```

Vista textual (1/3)

```
import java.awt.event.ActionEvent;
import java.io.BufferedReader;
import java.io.InputStreamReader;

public class InterfazTextualConversor implements InterfazVista {
    private ControlConversor controlador;

    // Gestión de la entrada por teclado
    private BufferedReader in = new BufferedReader(new InputStreamReader(System.in));

    private int leeOpción() {
          String s = null;
          try {     s = in.readLine();
                    return Integer.parseInt(s);
          } catch(Exception e){
                    operaciónIncorrecta(); return 0;
          }
    }

    private double leeCantidad() {
          String s = null;
          try {     s = in.readLine();
                    return Double.parseDouble(s);
          } catch(Exception e){
                    System.out.println("Error en formato del número, tiene que ser 99.99: ");
                    return leeCantidad();
          }
    }
```


Vista textual (2/3)

```
private void solicitaOperación() {
    System.out.println("Indica la operación que quiere realizar:");
    System.out.println("1: convertir euros a euros");
    System.out.println("2: convertir pesetas a pesetas");
    System.out.println("0: salir");
}

private void procesaNuevaOperacion() {
    int operacion;
    solicitaOperación();
    operacion = leeOpción();
    if ( operacion == 0 ) {
          System.out.println("Adiós.");
          System.exit(0);
    }
    if (operacion == 1 ) {
          controlador.actionPerformed( new ActionEvent(this, nroOperacion, AEUROS) );
    }
    if (operacion == 2 ) {
          controlador.actionPerformed( new ActionEvent(this, nroOperacion, APESETAS) );
    }
    operaciónIncorrecta();
}

private void operaciónIncorrecta() {
    System.out.print("Operación incorrecta. ");
    procesaNuevaOperacion();
}
```

Vista textual (3/3)

```
// Métodos de la interfaz de la Vista:

public void setControlador(ControlConversor c) {
   controlador = c;
}

public void arranca() {
   procesaNuevaOperacion();
}

public void escribeCambio(String s) {
   // escribe el resultado:
   System.out.println(s);
   // y vuelve a solicitar al usuario una operación:
   procesaNuevaOperacion();
}

public double getCantidad() {
   System.out.print("Cantidad a convertir (formato 99.99): ");
   return leeCantidad();
}
```


Discusión final

- ¿Qué hay que cambiar en el modelo y el control para utilizar la vista textual en vez de la gráfica?
- ¿Qué hay que cambiar en el programa principal?