



DOCUMENTACION EJERCICIOS I & II

Edgar Núñez Arana

Tabla de contenido

EJERCICIO I	2
Modelo (M)	2
Controlador (C)	3
Vista (V)	6
Conclusión	7
 EJERCICIO II	8
VISTA	8
Ventana Principal	8
Ventana Crear Cuenta	9
Ventana Ver Cuenta	10
Ventana Actualizar Saldo	11
Ventana Borrar Cuenta	12
Guardar Cuenta	13
MODELO	14
CONTROLADOR	15
CONCLUSION	15

EJERCICIO I

En este ejercicio haremos una gestión de un banco desde Eclipse donde podremos crear cuentas y actuar sobre ellas, ver sus datos, borrarlas, o modificarlas para ello las reconoceremos con un ID que será el código identificativo de cada cuenta, todo ello estará en el formato *MVC* para el correcto funcionamiento y administración de cada clase del banco.

Modelo (M)

El apartado modelo (M) es el package, como se puede ver en la [imagen 1](#), interno de nuestro banco el cual tiene la clase cuenta.java y es donde se guardara nuestras cuentas en formato .dat.

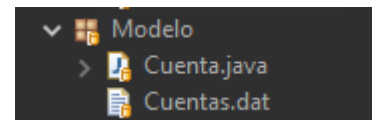


IMAGEN 1

Además la clase Cuenta tendrá unos atributos que serán los datos a ver de cada cuenta como puede ser Nombre del titular de la cuenta, numero de cuenta, saldo (tipo double) y por último el código identificativo que será el mas importante ya que con el podremos reconocer a que cuenta nos dirigimos. En esta clase Cuenta también tendremos su constructor y los "getter" y "setters", véase en la [imagen 2](#) (hay algunos getter y setter no todos).

```
package Modelo;

public class Cuenta {

    private String cod;
    private String nombreCuenta;
    private String numCuenta;
    private double saldo;

    public Cuenta(String cod, String nombreCuenta, String numCuenta, double saldo)
    {
        super();
        this.cod = cod;
        this.nombreCuenta = nombreCuenta;
        this.numCuenta = numCuenta;
        this.saldo = saldo;
    }

    public String getCod()
    {
        return cod;
    }

    public void setCod(String cod)
    {
        this.cod = cod;
    }

    public String getNombreCuenta()
    {
        return nombreCuenta;
    }
}
```

IMAGEN 2

Controlador (C)

En el apartado controlador (C) es el package, **imagen 3**, el cual es el encargado de dar las distintas funciones de nuestro banco.

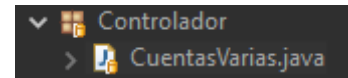


IMAGEN 3

Para ello crearemos una colección para poder actuar sobre ella.

```
package Controlador;

import java.io.DataOutputStream;

public class CuentasVarias
{
    //Clase CuentasVarias donde podremos encontrar los metodos.

    private List<Cuenta> cuentasBancarias = new LinkedList<>();/*
```

```
public void crearCuenta(Cuenta c) //Metodo crearCuenta objeto nuevo de Cuenta = c
{
    // Verificar si el código identificativo ya está en uso
    if (buscarCuentaPorCodigo(c.getCod()) == null)
    {
        cuentasBancarias.add(c); //Añadimos a nuestra coleccion cuentasBancarias nuestra cuenta c
        System.out.println("Cuenta registrada con éxito.");
    }
    else //Si no se cumple la condicion que al buscarCuentaPorCodigo = null mandará un msg de que ya existe una cuenta con ese cod por lo tanto no la creará.
    {
        System.out.println("Ya existe una cuenta con este código identificativo. No se pudo crear la cuenta.");
    }
}
```

```
private Cuenta buscarCuentaPorCodigo(String codigo) //Creacion del metodo buscarCuentaPorCodigo.
{
    for (Cuenta cuenta : cuentasBancarias) //Iteramos entre cuentas para poder buscar Cuentas
    {
        if (cuenta.getCod().equals(codigo)) //Si el código creado introducido coincide devolvemos cuenta
        {
            return cuenta;
        }
    }
    return null;
}
```

```

public void leerCuenta(String codigo)
{
    for(Cuenta cuenta : cuentasBancarias)//Iteramos entre nuestra coleccion de cuentas buscando lo
    {
        if(cuenta.getCod().equals(codigo))//Si el codigo de cuenta introducido en el main coincide
        {
            System.out.println(cuenta.toString());//En este caso escribe el toString
        }
        else
        {
            System.out.println("No se encontró el codigo de esa cuenta");
        }
    }
    if(cuentasBancarias.size() == 0)//Si no existe ninguna cuenta saldra el msg posterior, para qu
    {
        System.out.println("No hay cuentas agregadas en el banco en este momento");
    }
}

```

```

public void eliminarCuenta(String codigo)
{
    for(Cuenta cuenta : cuentasBancarias)//Iteramos entre nuestra coleccion de cuentas buscando los obj
    {
        if(cuenta.getCod().equals(codigo))//Si el codigo de cuenta introducido en el main coincide con
        {
            cuentasBancarias.remove(cuenta);//En esta caso se elimina la cuenta de la coleccion
            System.out.println("La cuenta con codigo " + codigo + " se ha eliminado correctamente");
        }
        else
        {
            System.out.println("No se encontró una cuenta con ese codigo");
        }
    }

    if(cuentasBancarias.size() == 0)//Si no existe ninguna cuenta saldra el msg posterior, para que ter
    {
        System.out.println("No hay cuentas agregadas en el banco en este momento");
    }
}

```

```

public void actualizarCuenta(String codigo, String nuevoNombre, String nuevoNumero, double nuevoSaldo) //Creación del m
{
    for (Cuenta cuenta : cuentasBancarias)//Iteramos entre nuestra coleccion de cuentas buscando los obj cuenta.
    {
        if (cuenta.getCod().equals(codigo))//Si el codigo de cuenta introducido en el main coincide con el codigo de u
        {
            cuenta.setNombreCuenta(nuevoNombre);
            cuenta.setNumCuenta(nuevoNumero);
            cuenta.setSaldo(nuevoSaldo);
            //En esta caso modificamos con los nuevos datos introducidos los datos anteriormente creados mediante cuen

            System.out.println("Los datos de la cuenta con codigo " + codigo + " se han actualizado correctamente.");
            return;
        }
        else
        {
            System.out.println("No se encontro una cuenta con ese codigo");
        }
    }

    if(cuentasBancarias.size() == 0)
    {
        System.out.println("No hay cuentas agregadas en el banco en este momento");
    }
}

```

```
public void guardarCuenta()//Metodo guardar cuenta
{
    String filePath = "./modelo/Cuentas.dat";
    try (FileOutputStream os = new FileOutputStream(filePath);
        DataOutputStream escritor = new DataOutputStream(os))
    {
        for(Cuenta c: cuentasBancarias)
        {
            escritor.writeUTF(c.getCod());
            escritor.writeUTF(c.getNombreCuenta());
            escritor.writeUTF(c.getNumCuenta());
            escritor.writeDouble(c.getSaldo());
        }

        System.out.println("Cuentas guardadas correctamente en el archivo " + filePath);
    }

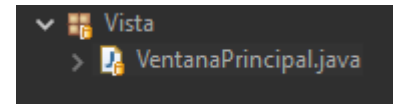
    catch(IOException e)
    {
        System.out.println("Se ha producido un error en el manejo del fichero " + filePath);
        System.out.println(e.getMessage());
    }

    finally
    {
        System.out.println("La escritura del fichero ha finalizado.");
    }
}
```

IMAGEN 4

Vista (V)

Por ultimo en el apartado vista, **imagen 5**, será el encargado de darle el feedback con el cliente donde este último podrá tener acciones con el programa escoger las opciones en el menú y crear borrar modificar cuentas...



Para ello crearemos un banco donde podremos realizar las acciones.

```
package Vista;

import java.util.Scanner;

public class VentanaPrincipal {

    public static void main(String[] args)
    {
        Scanner entrada = new Scanner(System.in);
        System.out.println("__ Banco Santander __");

        CuentasVarias banco = new CuentasVarias();//Creacion del banco para poder trabajar sobre él.

        boolean salir = false; //Inicializamos la variable salir = false para el case 6
        int seleccionar = 0;//creacion de la variable para "seleccionar" para poder escoger una opcion del menu.

        do
        {
            //menu de opciones de nuestro banco

            System.out.println("1. Crear Cuenta");
            System.out.println("2. Leer datos de Cuenta");
            System.out.println("3. Actualizar datos de Cuenta");
            System.out.println("4. Borrar Cuenta");
            System.out.println("5. Guardar datos de Cuenta");
            System.out.println("6. Salir");
            System.out.println("Seleccione una opción del menu escogiendola por su número");

            seleccionar = entrada.nextInt();
            switch(seleccionar)//switch para intercalar entre opciones escogidas por su numero
            {
                //
            }
        }
    }
}
```

```
switch(seleccionar)//switch para intercalar entre opciones escogidas por su numero
{
    case 1:
        //Creacion de la cuenta
        entrada.nextLine();

        System.out.println("Introduzca el código identificativo de la cuenta (Formato: BANK_XXX):");
        String cod = "";
        boolean codigoValido = false;

        while (!codigoValido)
        {
            cod = entrada.nextLine();
            if (Pattern.matches("BANK_\\d{3}", cod))
            {
                codigoValido = true;
            }
            else
            {
                System.out.println("Formato incorrecto. Introduzca un código válido (BANK_XXX):");
            }
        }
        //validador del codigo identificativo, tiene que cumplir el formato BANK_123 O BANK_000 el (vania el numero)
    }
}
```

```
System.out.println("Introduzca el nombre del titular la cuenta:");//Solicitud del nombre del titular de la cuenta sin formato.
String nombreCuenta = entrada.nextLine();

System.out.println("Introduzca el numero de cuenta (Ejemplo: 1234 1234 XX 1234567890):");
String numCuenta;

while (true)
{
    numCuenta = entrada.nextLine();
    if (Pattern.matches("\\d{4} \\d{4} [A-Z]{2} \\d{10}", numCuenta))
    {
        break;
    }
    else
    {
        System.out.println("Formato incorrecto. Introduzca un número de cuenta válido (Ejemplo: 1234 1234 XX 1234567890):");
    }
}
//validador del numero de cuenta, tiene que cumplir el formato 1234 1234 EE 1234567890 (Pueden variar tanto nums como letras)

Cuenta nuevaCuenta = new Cuenta(cod, nombreCuenta, numCuenta, 0);//Creacion de la cuenta (creacion del objeto) pasandole los d
banco.crearCuenta(nuevaCuenta);//Pasamos a nuestro banco el metodo crear cuenta para pasarle nuestro nueva cuenta y que quede
```

```
case 2:
    entrada.nextLine();

    System.out.println("Introduzca el código identificativo de la cuenta que desea leer sus datos (Formato: BANK_XXX):");
    String codigoL = entrada.nextLine();//Pedimos el codigo de la cuenta que deseemos leer sus datos, el codigo cumpleca co

    banco.leerCuenta(codigoL);//llamamos al metodo leerCuenta de la cuenta con el codigo pedido anteriormente (codigoL)

    break;
```

```

case 3:
    entrada.nextLine();

    System.out.println("Introduzca el código identificativo de la cuenta que desea actualizar (Formato: BANK_XXX):");
    String codigoA = entrada.nextLine();

    System.out.println("Introduzca el nuevo nombre del titular de la cuenta:"); // Pedimos otra vez los datos de la cuenta
    String nuevoNombre = entrada.nextLine();

    System.out.println("Introduzca el nuevo número de cuenta (Ejemplo: 1234 1234 XX 1234567890)"); /* Pedimos el número de
                                                                                             tendremos que pedir
                                                                                             no es como el código
                                                                                             el mismo para todos

    String nuevoNumero = entrada.nextLine();

    while (true)
    {
        nuevoNumero = entrada.nextLine();
        if (Pattern.matches("\\d{4} \\d{4} [A-Z]{2} \\d{10}", nuevoNumero))
        {
            break;
        }
        else
        {
            System.out.println("Formato incorrecto. Introduzca un número de cuenta válido (Ejemplo: 1234 1234 XX 1234567890)");
        }
    } // Patrón del formato

    System.out.println("Introduzca el nuevo saldo de la cuenta:");
    double nuevoSaldo = entrada.nextDouble(); // Al actualizar los datos de cuenta pedirá también el saldo .

    banco.actualizarCuenta(codigoA, nuevoNombre, nuevoNumero, nuevoSaldo); // Llamamos al método actualizarCuenta y pasamos los datos

break;

case 4:
    entrada.nextLine();

    System.out.println("Introduzca el código identificativo de la cuenta que desea borrar (Formato: BANK_XXX):"); // Pedimos el código de la cuenta que se va a borrar
    String codigoB = entrada.nextLine();

    banco.eliminarCuenta(codigoB); // Llamamos al método eliminarCuenta de la cuenta con el código pedido anteriormente

break;

case 5:

    banco.guardarCuenta(); /* Guardamos todas las cuentas, aunque borremos una cuenta y abramos el archivo cuenta.dat
                           no le damos a guardar cuenta no se actualizan los datos en cuenta.dat */

break;

case 6:

    System.out.println("Usted salió del programa correctamente");
    salir = true; // Declaramos salir = true para poder salir del menú.

break;

} // llave del Switch.

}while(!salir);

entrada.close();

```

IMAGEN 5

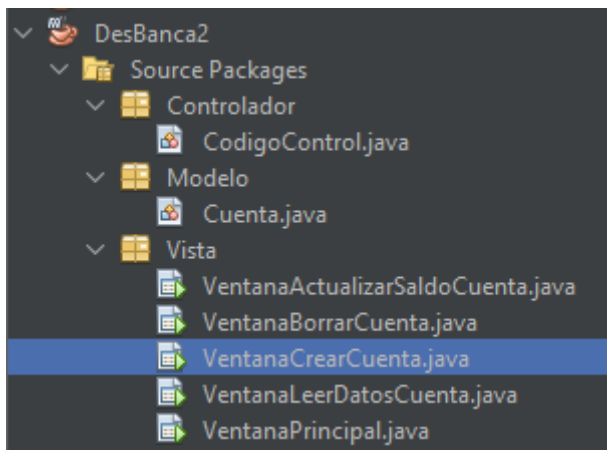
Se puede apreciar que cada case será una opción que se podrá escoger mediante el menú.

Conclusión

En conclusión, con esta organización MVC podremos gestionar de una manera óptima y organizada las clases para su correcto funcionamiento de banco.

EJERCICIO II

En este ejercicio haremos una gestión de un banco desde NetBeans el funcionamiento es parecido al del ejercicio I pero aquí podremos ver la interfaz gráfica de cómo se vería la aplicación de nuestro banco.

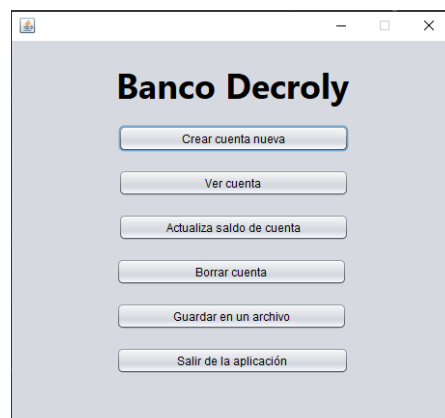


VISTA

Para empezar con el modelo vista tendremos las distintas ventanas que interactuara el usuario siendo la main la Ventana Principal:

Ventana Principal

Esta venta será un archivo JFrame y será la encargada de proporcionar un menú al usuario donde este podrá escoger la opción que desee.



Ventana Crear Cuenta

En la ventana crear cuenta (JDialog) nos dará la funcionalidad de crear cuenta donde escribiremos su ID nombre número de cuenta y saldo los cuales se guardarán en un nuevo objeto cuenta.

Si la cuenta se ha creado correctamente mandara un mensaje de se ha creado la cuenta con éxito.

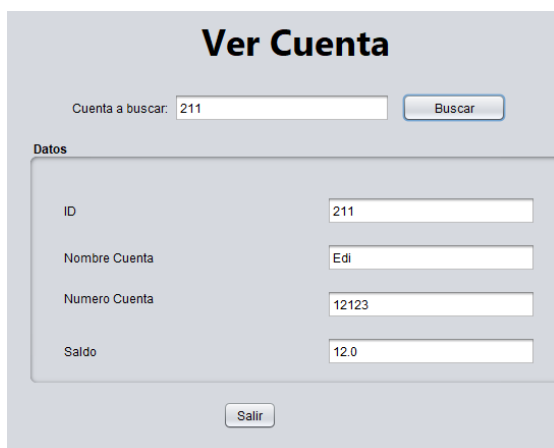
Ventana Ver Cuenta

En este caso tendremos la ventana ver cuenta (JDialog) la cual tendremos que introducir un ID de cuenta previamente creado para que nos muestre sus datos.



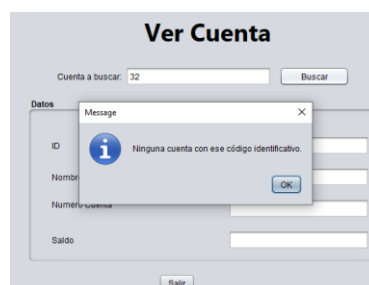
The screenshot shows a Java Swing window titled "Ver Cuenta". At the top, there is a label "Cuenta a buscar:" followed by a text input field and a "Buscar" button. Below this, there is a section titled "Datos" which contains four rows, each with a label and a text input field: "ID", "Nombre Cuenta", "Numero Cuenta", and "Saldo". At the bottom of the window is a "Salir" button. All input fields are currently empty.

Como se ve a continuación si introducimos el ID de la cuenta creada en el apartado anterior nos mostrara los datos de dicha cuenta.



This screenshot shows the "Ver Cuenta" window after a search. The "Cuenta a buscar:" field now contains the value "211". The "Datos" section has been populated with the following values: "ID" is "211", "Nombre Cuenta" is "Edi", "Numero Cuenta" is "12123", and "Saldo" is "12.0". The "Salir" button remains at the bottom.

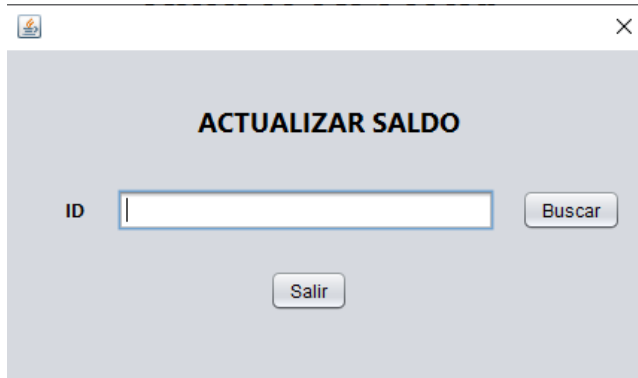
Si se introduce un ID de cuenta que aun no hayamos añadido antes saltará el mensaje de que no se ha podido encontrar dicha cuenta.



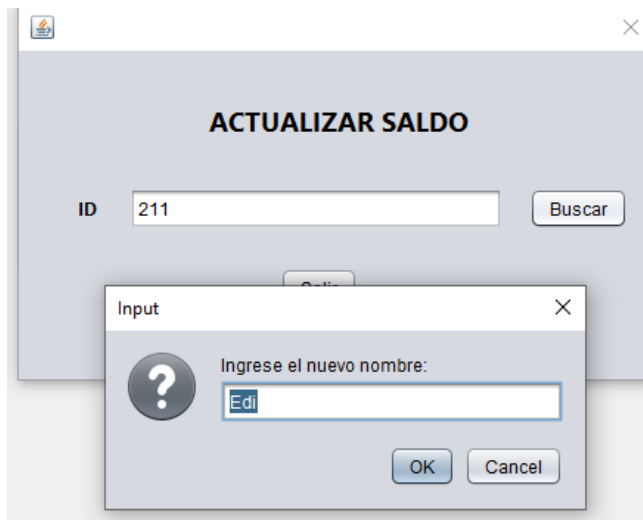
This screenshot shows the "Ver Cuenta" window with the "Cuenta a buscar:" field containing the value "32". A modal dialog box titled "Message" is displayed in the center. The dialog contains an information icon (i) and the text "Ninguna cuenta con ese código identificativo." (No account with that identification code). There is an "OK" button in the dialog. The background window is dimmed, showing the same fields as in the previous screenshots.

Ventana Actualizar Saldo

En la ventana actualizar saldo (jDialog) la función será cambiar los datos de cada cuenta, primero introduciremos el id de la cuenta para poder escoger que cuenta modificar.



Si la cuenta existe nos mostrará una nueva ventana donde podremos cambiar los datos en este caso modificar el nombre.

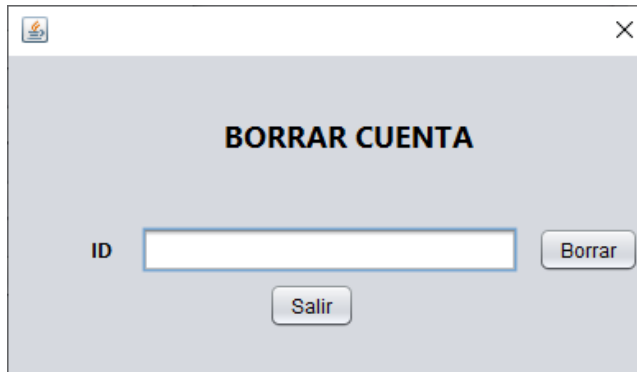


Si la cuenta con ese ID no existe saldra el mensaje de que no existe dicha cuenta.

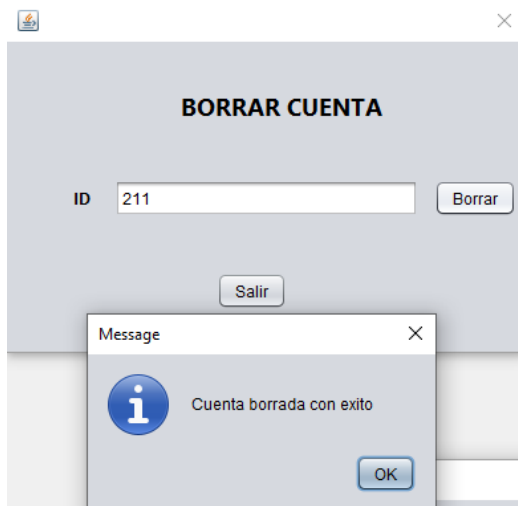


Ventana Borrar Cuenta

Ahora en la ventana borrar cuenta podremos borrar la cuenta de nuestra lista cuentas mediante su id.



Si la cuenta con el id existe se borrará de la lista cuentas y nos mostrará el siguiente mensaje.

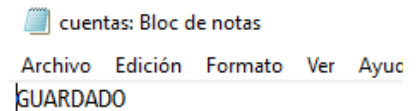
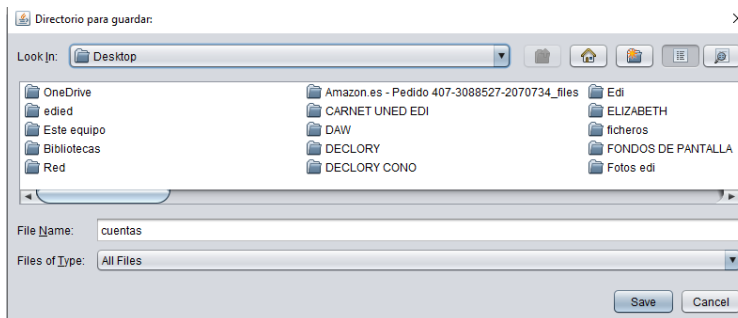


Si la cuenta con ese ID no existiera mostrara el mensaje de que no existe cuenta bancaria con ese código identificativo.



Guardar Cuenta

Aunque la opción guardar cuenta no se encuentre como otro JDialog en una ventana específica, está implementada en Ventana Principal la cual si está en Vista (MVC). Esta función será la encargada de guardar la cuenta en un directorio específico de nuestro equipo.



```
private void btnGuardarCuentaActionPerformed(java.awt.event.ActionEvent evt) {
    JFrame parentFrame = new JFrame();
    JFileChooser fileChooser = new JFileChooser();
    fileChooser.setDialogTitle("Directorio para guardar:");
    int userSelection = fileChooser.showSaveDialog(parentFrame);
    if (userSelection == JFileChooser.APPROVE_OPTION)
    {
        File fileToSave = fileChooser.getSelectedFile();
        System.out.println("Guardar como: " + fileToSave.getAbsolutePath());
        //editar con la información que se quiera guardar
        try {
            BufferedWriter out = new BufferedWriter(new FileWriter(fileToSave));
            out.write("GUARDADO");
            out.close();
        } catch (IOException ex) {
            Logger.getLogger(VentanaPrincipal.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
}
```

MODELO

En el package modelo nos encontraremos con la clase cuenta.java que es nuestro objeto principal el cual es usado en todo el programa.

```
package Modelo;

/**
 *
 * @author dawl
 */
public class Cuenta
{
    private String cod;
    private String nombreCuenta;
    private String numCuenta;
    private Double saldo;

    public Cuenta(String cod, String nombreCuenta, String numCuenta, Double saldo)
    {
        super();
        this.cod = cod;
        this.nombreCuenta = nombreCuenta;
        this.numCuenta = numCuenta;
        this.saldo = saldo;
    }

    public String getCod()
    {
        return cod;
    }

    public void setCod(String cod)
    {
        this.cod = cod;
    }

    public String getNombreCuenta()
    {
        return nombreCuenta;
    }

    public void setNombreCuenta(String nombreCuenta)
    {
        this.nombreCuenta = nombreCuenta;
    }

    public String getNumCuenta()
    {
        return numCuenta;
    }

    public void setNumCuenta(String numCuenta)
    {
        this.numCuenta = numCuenta;
    }

    public Double getSaldo()
    {
        return saldo;
    }

    public void setSaldo(Double saldo)
    {
        this.saldo = saldo;
    }
}
```

CONTROLADOR

En este caso nuestro package controlador estará vacío ya que decidí darle toda la funcionalidad dentro de las propias ventanas de VISTA.

CONCLUSION

Habiendo terminado el proyecto banco MVC desde "NetBeans" he concluido que podemos hacer la aplicación del banco con sus requisitos añadiéndole una interfaz gráfica con la que el usuario le será más fácil e intuitivo adaptarse y poder ejecutar nuestro programa con sus distintas funciones.