Informe de la PROG09

Ejercicio 1

Se trata de modificar la aplicación desarrollado en la Unidad de Trabajo 8, Ejercicio 1 para dar persistencia a los datos de cuentas bancarias. El nombre será PROG09_Ejerc1 Para ello:

- Cuando la aplicación finalice, es decir, el usuario seleccione la opción Salir, la aplicación volcará el contenido de la estructura de datos con las cuentas bancarias a un fichero binario denominado datoscuentasbancarias.dat.
- Cuando la aplicación inicie la ejecución, antes de mostrar el menú, deberá cargar en la estructura de datos el contenido del fichero datoscuentasbancarias.dat.

Como ya sabes, para poder realizar estas tareas es necesarios que nuestros objetos que representan cuentas bancarias sean serializables. Habrá que realizar las convenientes modificaciones a la clase CuentaBancaria.

Método guardarDatos de Banco
Guardar el cuentasBancarias en el fichero.
CuentasBancarias es un conjunto de tipo HashSet en el cual están todas las cuentas del Banco.

El método en si lo que hace es crear un objeto de tipo FileOutputStream para manipular el fichero y otro Objeto de tipo ObjectOutputStream para escribir en el fichero

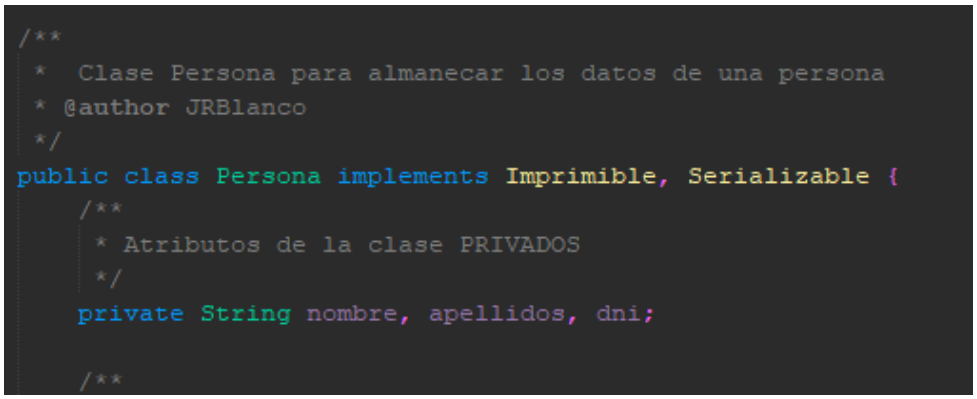
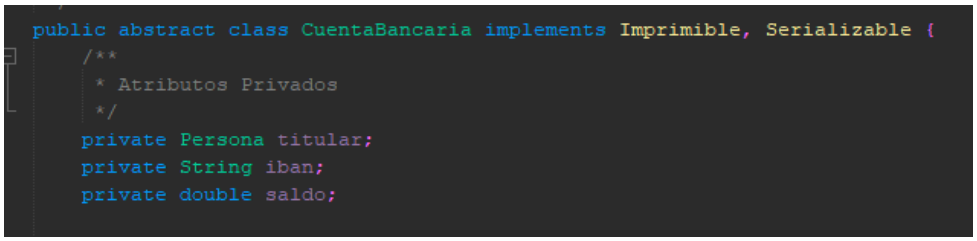
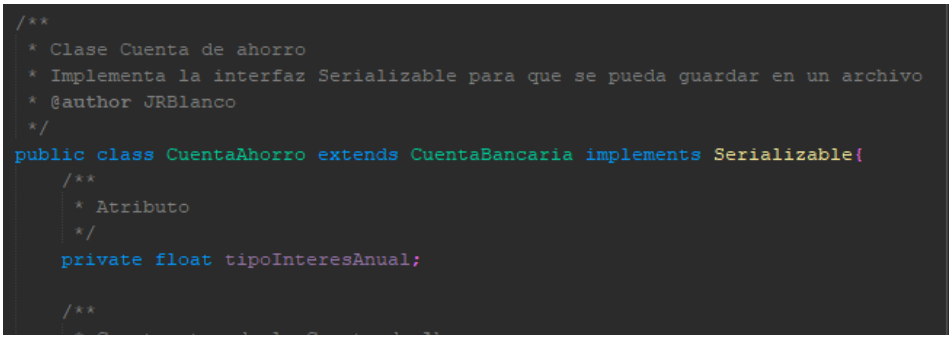
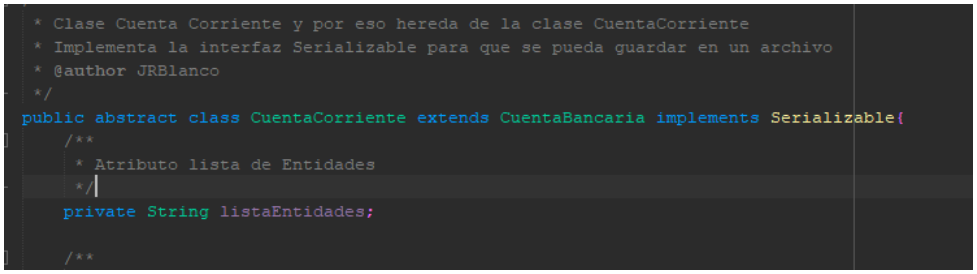
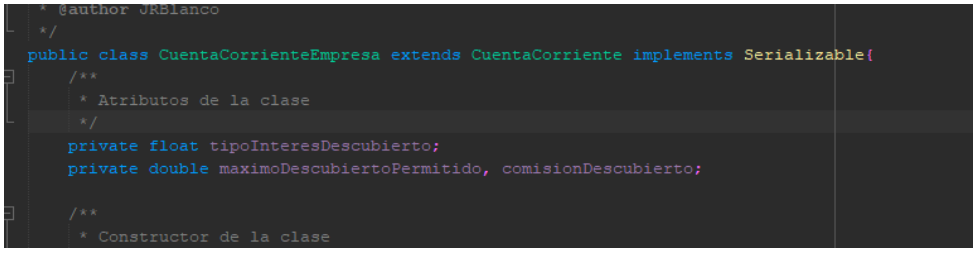
Si todo se ha creado correctamente guarda con writeObject el hashSet.

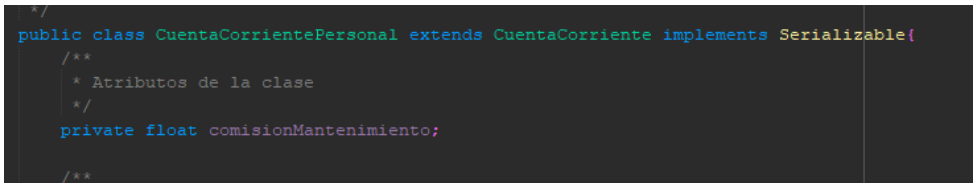
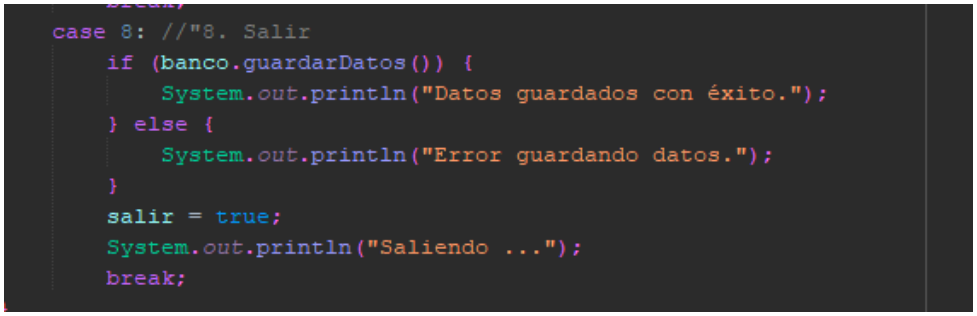
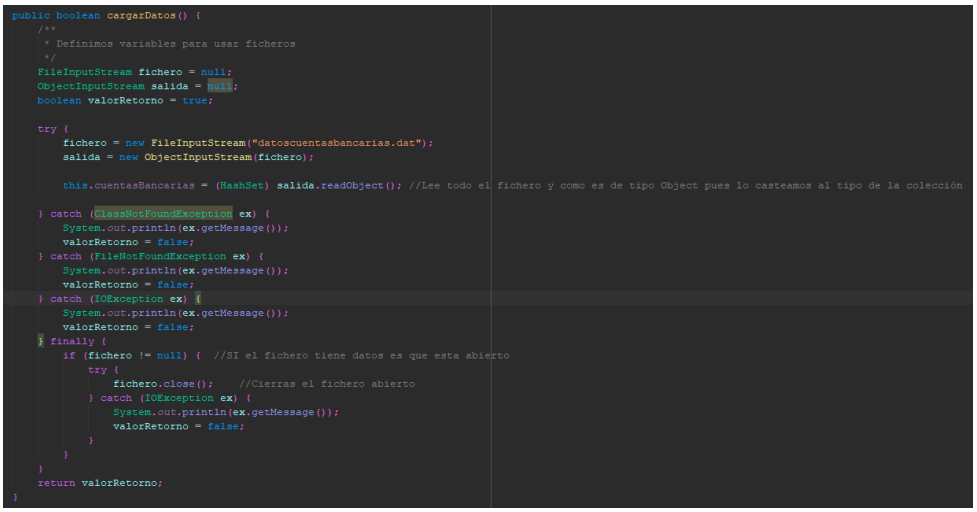
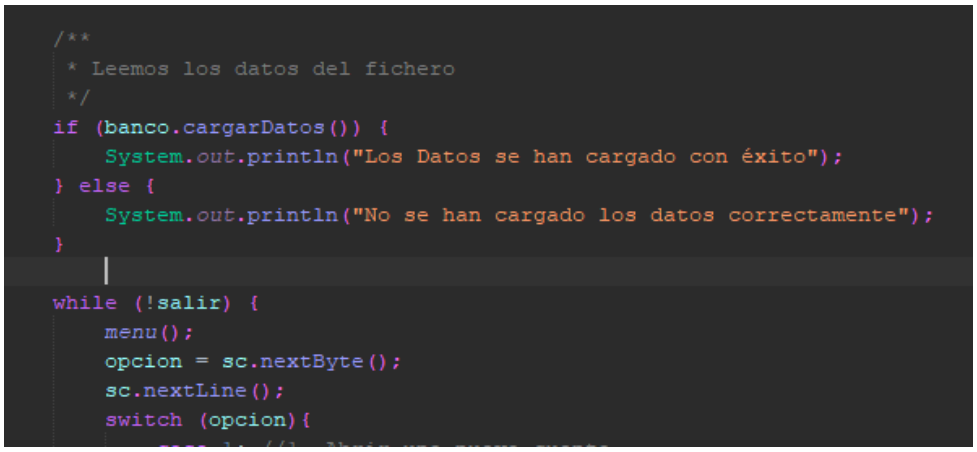
```java
public boolean guardarDatos() {
    /**
     * Definimos variables para usar ficheros
     */
    FileOutputStream fichero = null;
    ObjectOutputStream salida = null;
    boolean valorRetorno = true;

    try {
        fichero = new FileOutputStream("datoscuentasbancarias.dat");
        salida = new ObjectOutputStream(fichero);

        salida.writeObject(this.cuentasBancarias);

    } catch (FileNotFoundException ex) {
        System.out.println(ex.getMessage());
        valorRetorno = false;
    } catch (IOException ex) {
        System.out.println(ex.getMessage());
        valorRetorno = false;
    } finally {
        if (fichero != null) { //SI el fichero tiene datos es que esta abierto
            try {
                fichero.close();    //Cierra el fichero abierto
            } catch (IOException ex) {
                System.out.println(ex.getMessage());
                valorRetorno = false;
            }
        }
    }
    return valorRetorno;
}
```

Clase Persona Para que poder guardar el objeto cuentaBancaria he tenido que añadir a las diferentes clase la interfaz Serializable.	 ```java /** * Clase Persona para almanecar los datos de una persona * @author JRBlanco */ public class Persona implements Imprimible, Serializable { /** * Atributos de la clase PRIVADOS */ private String nombre, apellidos, dni; /** ```
Clase Abstracta CuentaBancaria	 ```java public abstract class CuentaBancaria implements Imprimible, Serializable { /** * Atributos Privados */ private Persona titular; private String iban; private double saldo; ```
Clase Cuenta Ahorro	 ```java /** * Clase Cuenta de ahorro * Implementa la interfaz Serializable para que se pueda guardar en un archivo * @author JRBlanco */ public class CuentaAhorro extends CuentaBancaria implements Serializable{ /** * Atributo */ private float tipoInteresAnual; /** ```
Clase Abstracta Cuenta Corrient	 ```java * Clase Cuenta Corriente y por eso hereda de la clase CuentaCorriente * Implementa la interfaz Serializable para que se pueda guardar en un archivo * @author JRBlanco */ public abstract class CuentaCorriente extends CuentaBancaria implements Serializable{ /** * Atributo lista de Entidades */ private String listaEntidades; /** ```
Clase CuentaCorrienteEmpresa	 ```java * @author JRBlanco */ public class CuentaCorrienteEmpresa extends CuentaCorriente implements Serializable{ /** * Atributos de la clase */ private float tipoInteresDescubierto; private double maximoDescubiertoPermitido, comisionDescubierto; /** * Constructor de la clase ```

Clase CuentaCorrientePersonal	
Y en la opción de salir llama al método y si todo ha ido correctamente crea el fichero con los datos.	
Método cargarDatos de Banco Este método carga los datos del disco y los guarda en el atributo cuentasBancarias. Para leer el fichero usamos readObject que devuelve un objeto de tipo Object por eso he tenido que castearlo con el HashSet.	
Antes de que se inicie el bucle principal se llama al método para que cargue los datos si les hubiera.	

Ejercicio 2

Añade una nueva opción al menú de la aplicación denominado "Listado clientes" de modo que al seleccionarla, se genere un fichero de texto denominado ListadoClientesCCC.txt que contenga una línea de texto por cada cuenta bancaria almacenada, donde se visualice nombre del propietario y CCC por cada una de ellas.

La última línea del fichero contendrá el número total de cuentas existente.

Método listadoClientes

Este método genera un fichero con el número de cuenta y el propietario de dicha cuenta.

El funcionamiento básicamente lo primero que hace es comprobar que hay datos en el banco y si es así entra y recorre la colección con un foreach y cada cuenta y propietario lo guarda.

Al final de recorrer todas las cuentas, almacena el numero de cuentas en el fichero de texto.

```java
public boolean listadoClientes() {
    /**
     * Definición de variables para manupular ficheros
     */
    FileWriter fichero = null;
    PrintWriter pw = null;
    boolean estado = true;

    //Inicializamos los objetos para manipular el fichero de
    if (!this.cuentasBancarias.isEmpty()) {
        try {
            fichero = new FileWriter("ListadoClientesCCC.txt"); //Si quiera concatenar contenido añado true
            pw = new PrintWriter(fichero);

            pw.println("-------------------------------------------------");
            pw.println("---------------Listado de Clientes---------------");
            pw.println("-------------------------------------------------");
            for (CuentaBancaria cb:cuentasBancarias){
                pw.println("{" + cb.getIban() + " , " +
                                    cb.getTitular().getNombre() + " " +
                                    cb.getTitular().getApellidos() + "}");
            }
            pw.println("-------------------------------------------------");
            pw.println("Total de cuentas existentes: " + cuentasBancarias.size());
            pw.println("-------------------------------------------------");
        } catch (IOException ex) {
            System.out.println(ex.getMessage());
            estado = false;
        } finally {
            if (fichero != null) {
                try {
                    fichero.close();
                } catch (IOException ex) {
                    System.out.println(ex.getMessage());
                }
            }
        }
    } else {
        estado = false;
    }
    return estado;
```

Llamo al método y si ha ido bien muestro mensaje de todo correcto y sino ha ido bien pongo un mensaje de error.

```java
case 7: //7. Listado clientes
    if (banco.listadoClientes()) {
        System.out.println("Listado de clientes guardado con éxito en: ListadoClientesCCC.txt");
    } else {
        System.out.println("Error guardando listado de clientes");
    }
    break;
```

Contenido del fichero de texto creado.