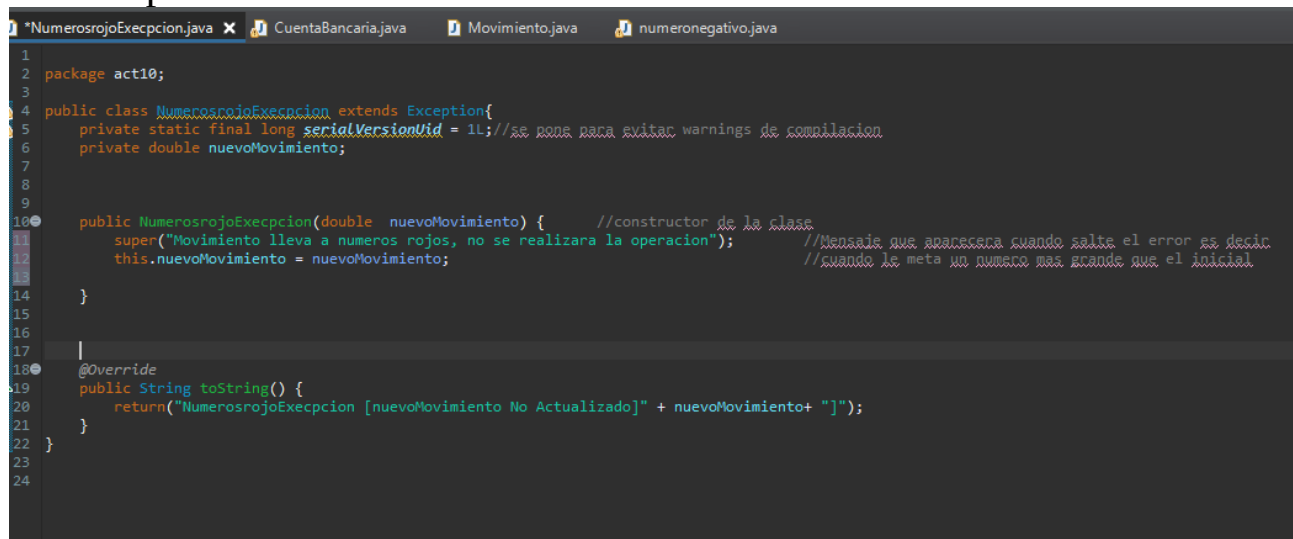


Actividad 10

NumerosRojosException

En esta actividad se nos pide simular el cajero de un banco y se nos pide hacer las excepciones de este de tal manera que si tienes una retirada de saldo mayor que el saldo inicial se ejecute una excepción que diga que no se puede.

Para esto creamos la clase NumerosRojosException que sera un extends de Exception.



```
1 package act10;
2
3 public class NumerosRojosException extends Exception{
4     private static final long serialVersionUID = 1L; //se pone para evitar warnings de compilacion
5     private double nuevoMovimiento;
6
7
8
9
10    public NumerosRojosException(double nuevoMovimiento) { //constructor de la clase
11        super("Movimiento lleva a numeros rojos, no se realizara la operacion"); //Mensaje que aparecera cuando salta el error es decir
12        this.nuevoMovimiento = nuevoMovimiento; //cuando le meta un numero mas grande que el inicial
13    }
14
15
16
17
18    |
19    @Override
20    public String toString() {
21        return("NumerosRojosException [nuevoMovimiento No Actualizado]" + nuevoMovimiento+ " ");
22    }
23
24 }
```

Es una clase simple en la que se tiene un toString para decir que se ha cancelado la operación.

Y en el constructor de la clase tenemos un mensaje que saldrá al llamarlo para informar.

La clase CuentaBancaria es una clase que tiene la entrada de datos de nombre del cliente numero de cuenta el saldo que tiene y un arrayList para los movimientos de retirada de dinero.

```

1 package act10;
2
3 import java.util.ArrayList;
4
5 public class CuentaBancaria {
6
7     private int miCuenta;
8     private String cliente;
9     private double saldo;
10    private ArrayList<Movimiento> movimientos;
11
12    public CuentaBancaria(int numeroCuenta, String cliente) {
13        this.miCuenta = numeroCuenta;
14        this.cliente = cliente;
15        this.saldo = 0;
16        this.movimientos = new ArrayList();
17    }
18
19    public void agregarMovimiento(String concepto, double cantidad) {
20        this.saldo = this.saldo + cantidad;
21        this.movimientos.add(new Movimiento(concepto, cantidad, saldo));
22    }
23
24    @Override
25    public String toString() {
26        return "Número de Cuenta= " + miCuenta + ", Cliente ->|" + cliente + ", Con un saldo de Saldo = " + saldo;
27    }
28
29    public String listaMovimientos() {
30        String listado = "";
31
32        for (Movimiento mov : this.movimientos) {
33            listado = listado + mov.toString()+"\n";
34        }
35        return listado;
36    }
37
38    public double ObtenerSaldo() throws NumerosrojoExcepcion{
39        int i = 0;
40        return(movimientos.get(i).getSaldo());
41    }
42 }

```

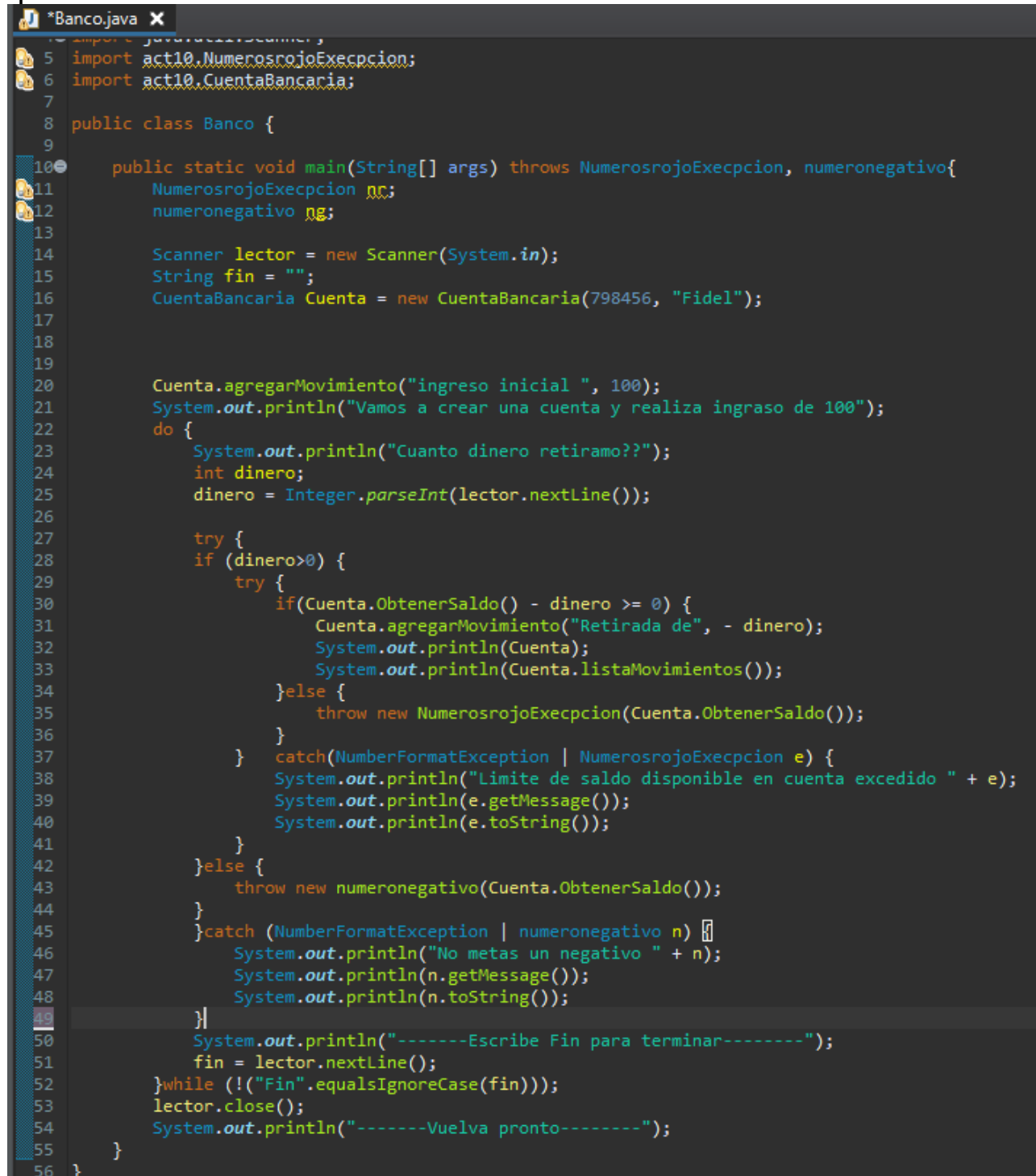
El método agregaMovimiento lo que hace es añadir al ArrayList un texto del movimiento que vas a hacer y el dinero que insertas y el total de lo que se va a quedar.

El método listamovimientos lo que hace es escribir todos los movimientos que se han realizado.

La clase Movimientos es una clase simple en la que se almacenan los datos de los movimientos y la fecha de cuando se realizan.

```
1 package act10;
2
3 import java.time.LocalDate;
4
5 public class Movimiento {
6     private LocalDate fecha;
7     private String concepto;
8     private double cantidad;
9     private double saldo;
10
11     public Movimiento(String concepto, double cantidad, double saldo) {
12         this.concepto = concepto;
13         this.cantidad = cantidad;
14         this.saldo = saldo;
15         this.fecha = LocalDate.now();
16     }
17
18     @Override
19     public String toString() {
20         return(fecha + " concepto = " + concepto + ", Cantidad = " + cantidad + ", Saldo = " + saldo);
21     }
22
23     public double getSaldo() {
24         return this.saldo;
25     }
26 }
27 }
```

El main se llama banco y aquí es donde se van a controlar las excepciones que se crearon.



```

1  import java.util.Scanner;
2
3  import act10.NumerosrojoExcepcion;
4  import act10.CuentaBancaria;
5
6  public class Banco {
7
8      public static void main(String[] args) throws NumerosrojoExcepcion, numeronegativo {
9          NumerosrojoExcepcion n;
10         numeronegativo ng;
11
12         Scanner lector = new Scanner(System.in);
13         String fin = "";
14         CuentaBancaria Cuenta = new CuentaBancaria(798456, "Fidel");
15
16         Cuenta.agregarMovimiento("ingreso inicial ", 100);
17         System.out.println("Vamos a crear una cuenta y realiza ingrasso de 100");
18         do {
19             System.out.println("Cuanto dinero retiramo??");
20             int dinero;
21             dinero = Integer.parseInt(lector.nextLine());
22
23             try {
24                 if (dinero > 0) {
25                     try {
26                         if (Cuenta.ObtenerSaldo() - dinero >= 0) {
27                             Cuenta.agregarMovimiento("Retirada de", - dinero);
28                             System.out.println(Cuenta);
29                             System.out.println(Cuenta.listaMovimientos());
30                         } else {
31                             throw new NumerosrojoExcepcion(Cuenta.ObtenerSaldo());
32                         }
33                     } catch (NumberFormatException | NumerosrojoExcepcion e) {
34                         System.out.println("Limite de saldo disponible en cuenta excedido " + e);
35                         System.out.println(e.getMessage());
36                         System.out.println(e.toString());
37                     }
38                 } else {
39                     throw new numeronegativo(Cuenta.ObtenerSaldo());
40                 }
41             } catch (NumberFormatException | numeronegativo n) {
42                 System.out.println("No metas un negativo " + n);
43                 System.out.println(n.getMessage());
44                 System.out.println(n.toString());
45             }
46         } while (!("Fin".equalsIgnoreCase(fin)));
47         lector.close();
48         System.out.println("-----Vuelva pronto-----");
49     }
50 }

```

Se utiliza el scanner para poder leer los datos por la terminal se crea un cuenta en la linea 20.

se inicia un do while que se cerrara cuando escribas fin (linea 52-54)

Dentro del do tenemos la creacion de un int llamado dinero que sera el que reciba los datos del scanner y iniciamos el primer try Este try lo explicare abajo del todo ya que no es lo que pedía la actividad.

El segundo try es el de la actividad este lo que hace es comprobar si los números que ingresas son mayores o iguales a 0 de tal manera que si son podrás retirar el dinero de la cuenta de no serlo se creara la excepciones.

Y el catch dira que no puedes sacar el dinero de la cuenta por el movito mencionado anteriormente.

Esto ocurrira llamando a los métodos de la linea 39 y 40.

El segundo try lo añadí ya que me di cuenta que al meter números negativos se sumaba al saldo.

Para esto cree una clase nueva igual que la de NumerosRojos solo que cambiaba el nombre.

```
package act10;

public class numeronegativo extends Exception{
    private static final long serialVersionUID = 1L;
    private double negativo;

    public numeronegativo(double nuevoMovimiento) {
        super("No puedes meter numeros negativos");
        this.negativo = nuevoMovimiento;
    }

    @Override
    public String toString() {
        return("numeronegativo| [nuevoMovimiento No Actualizado]" + negativo+ " ");
    }
}
```

Y añadí en la línea 27-28 el try con la condición para los números negativo además se añadió las líneas 42-48 que son iguales a las del NumeroRojo solo que modificadas para el negativo.

```
27     try {
28         if (dinero > 0) {
29             try {
30                 if (Cuenta.ObtenerSaldo() - dinero >= 0) {
31                     Cuenta.agregarMovimiento("Retirada de", - dinero);
32                     System.out.println(Cuenta);
33                     System.out.println(Cuenta.listaMovimientos());
34                 } else {
35                     throw new NumerosrojoExcepcion(Cuenta.ObtenerSaldo());
36                 }
37             } catch (NumberFormatException | NumerosrojoExcepcion e) {
38                 System.out.println("Limite de saldo disponible en cuenta excedido " + e);
39                 System.out.println(e.getMessage());
40                 System.out.println(e.toString());
41             }
42         } else {
43             throw new numeronegativo(Cuenta.ObtenerSaldo());
44         }
45     } catch (NumberFormatException | numeronegativo n) {
46         System.out.println("No metas un negativo " + n);
47         System.out.println(n.getMessage());
48         System.out.println(n.toString());
49     }
```