




Instituto Tecnológico
de Edix

UF8 · Actividad 10

Tarea Individual.

NumerosRojosException

MIGUEL ÁNGEL LOZANO BERMEJO



MP_0485
Programación

Actividad 10. Tarea individual. NumerosRojosException

Esta práctica consiste en la implementación de una clase de excepción llamada `NumerosRojosException`, que servirá para detectar en una clase llamada `CuentaBancaria` la situación en la que se saca de la cuenta más dinero del que disponemos, quedándonos en números rojos.

- Debes crear un proyecto en Eclipse con el nombre `EjercicioUF8-01` atendiendo a las especificaciones que se indican en este documento.
- Empaquetarás el proyecto en un archivo `.zip` que entregarás a tu tutor junto con un documento en formato Word o PDF donde realizarás una exposición sobre lo que has ido realizando y pegarás las partes principales del código.

Especificaciones

- Partirás de una copia de las clases `Movimiento` y `CuentaBancaria` que aparecían en la unidad anterior (7.2 Colecciones). En este mismo documento, más abajo, tienes el código para que puedas copiarlo y luego pegarlo en tu proyecto.
- Debes considerar como una excepción el hecho de realizar un movimiento que provoque números rojos. Si esto va a ocurrir, se producirá la excepción y el movimiento no debe llegar a realizarse. Para ello debes crear una clase de excepción que se llamará `NumerosRojosException` y será de tipo “comprobada”.
- La siguiente clase `Principal` crea un objeto `CuentaBancaria`, le agrega un saldo inicial de 100 euros mediante un ingreso. Por último permite al usuario realizar una retirada de dinero solicitando por teclado la cantidad.

Planteo la actividad como la realización de **Tarjeta Regalo / Fidelización** en la que se cargarán y descontarán puntos según el uso que se haga de ella.

Nos indican que la clase `NumerosRojosException` debe ser comprobada, por lo que nos obliga a crear un bloque `try...catch` para controlar la ejecución sin que sea abortada. Crearemos 4 clases para el funcionamiento del programa, la clase **Principal**, la clase **NumerosRojosException**, la clase **CuentaBancaria** y la clase **Movimiento**.

Clase Principal

A partir del código dado realizamos las modificaciones necesarias para poder ejecutar el programa. Lo primero que hacemos es insertar en el *bloque try* la parte del código susceptible de darnos excepciones, en esta parte haremos una carga inicial de puntos en la tarjeta que será enviada mediante el método `agregarMovimiento()`, a continuación realizamos tres operaciones para canjear puntos, si en alguno de los movimientos intentamos canjear más de los que dispone, saldrá un mensaje de error, en caso de disponer de puntos los restará al total acumulado. A continuación en el *bloque catch* capturamos el objeto de excepción si ocurre esa situación en el *bloque try*. Terminamos estos bloques con el *bloque finally* en el que colocamos una salida formal del programa

```
try {
    miCuenta.agregarMovimiento("Puntos a favor:", 50);
    System.out.println("Cuántos puntos va a utilizar: ");
    int dinero;
    dinero = Integer.parseInt(lector.nextLine());
    miCuenta.agregarMovimiento("Uso de puntos,", -dinero);
```

```

        System.out.println("Cuántos puntos va a utilizar: ");
        dinero = Integer.parseInt(lector.nextLine());
        miCuenta.agregarMovimiento("Uso de puntos,", -dinero);

        System.out.println("Cuántos puntos va a utilizar: ");
        dinero = Integer.parseInt(lector.nextLine());
        miCuenta.agregarMovimiento("Uso de puntos,", -dinero);
        lector.close();
    } catch (NumberFormatException | NumerosRojosException e) {
        System.out.println(e.getMessage());
        System.out.println("No ha introducido correctamente el valor " +
e.getClass().getName());
        System.out.println(e.toString());
    } finally {
        System.out.println("Gracias por confiar en nosotros\n");
    }
}

```

bloque try...catch

Clase NumerosRojosException

Esta clase al ser de tipo “comprobada” extiende de la clase base *Exception* por lo que incluimos *extends Exception* en la cabecera. En el código se añade la sentencia *private static final long serialVersionUID = 1L*; necesaria para evitar un warning en Eclipse. Creamos los constructores por defecto y el que recibe un parámetro, y por último sobre escribimos el método *toString()*.

```

public class NumerosRojosException extends Exception{
    // PROPIEDADES
    private static final long serialVersionUID = 1L;    private int
nuevoAcum;

    // CONSTRUCTOR
    public NumerosRojosException(int nuevoAcum) {
        super("\n***** No dispone de puntos en su Tarjeta Feliz
*****\n");
        this.nuevoAcum = nuevoAcum;
    }

    public NumerosRojosException() {
        super("\n***** No dispone de puntos en su Tarjeta Feliz
*****\n");
        this.nuevoAcum = 0;
    }

    @Override
    public String toString() {
        return "NumerosRojosException [La cantidad de puntos que tiene son: "
+ nuevoAcum + "]\n";
    }
}

```

Clase CuentaBancaria

Esta clase presenta un *ArrayList* que recoge los movimientos realizados. Primero importamos la clase *import java.util.ArrayList*; creamos dos constructores, uno por defecto, y dos métodos, uno donde agregamos los movimientos al *ArrayList* y otro donde quedan guardados en un lista para más tarde imprimirlos.

También hemos sobre escrito el método *toString()*.

```
import java.util.ArrayList;

public class CuentaBancaria {

    //PROPIEDADES
    private int numeroCuenta;
    private String cliente;
    private int saldo;
    private ArrayList<Movimiento> movimientos;

    // CONSTRUCTOR
    public CuentaBancaria() {
        this.numeroCuenta = 0;
        this.cliente = "";
        this.saldo = 0;
        this.movimientos = new ArrayList();
    }

    public CuentaBancaria(int numeroCuenta, String cliente) {
        this.numeroCuenta = numeroCuenta;
        this.cliente = cliente;
        this.saldo = 0;
        this.movimientos = new ArrayList();
    }

    // METODOS
    public void agregarMovimiento(String concepto, int cantidad) throws
    NumerosRojosException {
        this.saldo = this.saldo + cantidad;
        if (this.saldo<0) {
            throw new NumerosRojosException(this.saldo);
        } else
            this.movimientos.add(new Movimiento(concepto, cantidad, saldo));
    }

    public String listarMovimientos() {
        String listado = "";
        for (Movimiento mov : this.movimientos) {
            listado = listado + mov.toString()+"\n";
        }
        return listado;
    }

    @Override
    public String toString() {
        return "Número Tarjeta: " + this.numeroCuenta + ", Cliente: " + this.
        cliente + ", Puntos= " + this.saldo;
    }
}
```

Clase Movimiento

Con esta clase lo que ejecutamos son los movimientos que hace el usuario. creamos los dos constructores como en el resto de clases, por defecto y el que recibe los tres parámetros. Tenemos un método get para obtener el saldo y se construye un método *toString()* el cual será sobre escrito.

Hemos utilizado el método *LocalDateTime()* para recoger la hora exacta de la utilización de la tarjeta y ser más precisos.

```
import java.time.LocalDateTime;

public class Movimiento {

    // PROPIEDADES
    private LocalDateTime fecha;
    private String concepto;
    private double cantidad;
    private int saldo;

    // CONSTRUCTOR
    public Movimiento() {
        this.concepto = "";
        this.cantidad = 0;
        this.saldo = 0;
        this.fecha = LocalDateTime.now();
    }

    public Movimiento(String concepto, int cantidad, int saldo) {
        this.concepto = concepto;
        this.cantidad = cantidad;
        this.saldo = saldo;
        this.fecha = LocalDateTime.now();
    }

    // GETTERS SETTERS
    public int getSaldo() {
        return this.saldo;
    }

    //METODOS
    @Override
    public String toString() {
        return fecha + " -- Concepto: " + concepto + " \tCantidad = " +
cantidad + " \tPuntos en tarjeta = " + saldo;
    }

}
```