# E-COMMERCE

PÉREZ SELLERS, ALEJANDRO

PROGRAMACIÓN 1º DE DAW

## 1. Introducción

En esta práctica se ha desarrollado un sistema de pago para un e-commerce de cursos on-line, simulando una tienda digital que permite a los usuarios realizar pagos mediante distintos métodos. Los métodos de pago disponibles son: Tarjeta de crédito, Paypal y Bizum.

En esta práctica se utiliza programación orientada a objetos, utilizando la abstracción, herencia y polimorfismo.  Para ello se ha creado una clase abstracta común llamada MetododePago, de la cual heredan los distintos métodos de pago, implementando cada uno su propio comportamiento a través del método procesarpago().

Además, se han introducido diferentes validaciones para comprobar que los datos introducidos por el usuario sean correctos antes de realizar el pago, como el formato del numero de la tarjeta de crédito o el tipo de tarjeta.

Por último, se ha creado una clase Tienda que se encarga de gestionar el programa, permitiendo al usuario elegir entre las diferentes opciones del menú, además de procesar y validar.

## 2. Metodología

**Clase MetodoPago**

```
package org.example;

public abstract class MetodoPago {



    public abstract void procesarPago(double importe);

}
```

El método abstracto que debe ser ejecutado por cada método de pago, dependiendo del tipo de pago (tarjeta, paypal o Bizum), se ejecutará de manera diferente, todas las clases hijas están obligadas a implementar el método procesar pago. El importe es la cantidad de dinero que se tiene que pagar.

**Clase Tarjeta de crédito**

```
package org.example;

public class TarjetaCredito extends MetodoPago{
    private String nro_tarjeta;
    private String tipo;

    public TarjetaCredito(String nro_tarjeta, String tipo) {
        this.nro_tarjeta = nro_tarjeta;
        this.tipo = tipo;

    }
```

La clase representa un método de pago mediante Tarjeta de crédito, esta clase hereda de metodopago e implementa su propio comportamiento para procesar pagos y validar los datos de la tarjeta.

```
@Override
public void procesarPago(double importe) {
   System.out.println("Procesando pago de " +importe +" € con tarjeta de credito
"+tipo);
}

public boolean validarTarjeta() {
   if (nro_tarjeta.length() != 16 || !nro_tarjeta.matches ("\\d{16}") ||
!tipo.equalsIgnoreCase("VISA") && !tipo.equalsIgnoreCase("MASTERCARD") &&
!tipo.equalsIgnoreCase("MAESTRO")){
       return false;

   }
   return true;
}
```

Muestra un mensaje de procesando el pago con tarjeta, aquí se aplica el polimorfismo, ya que este método se ejecutará cuando el objeto sea de tipo tarjetaCredito.

A continuación, se valida la tarjeta mediante un método de tipo booleano con un condicional if que verifica que sea 16 caracteres, que sean dígitos y el tipo de tarjeta, le he introducido una exclamación para indicar que si es diferente al valor que indica devuelva un falso, si es verdadero un true.

**Clase paypal**

```
package org.example;

public class Paypal extends MetodoPago{
    private String cuenta;
    private double saldo;

    public Paypal(String cuenta) {
        this.cuenta = cuenta;
        this.saldo =23;
    }

    @Override
    public void procesarPago(double importe) {
        System.out.println("Procesando pago "+importe+" € con paypal");
    }

    public boolean validarPaypal(double importe) {
        if (!cuenta.matches("^[A-Za-z0-9+_.-]+@alu.edu.gva.es$") || importe>saldo ||
importe<=0){
            return false;
        }
        return true;
```

Esta clase hereda de MetodoPago e implementa la lógica para procesar pagos mediante una cuenta paypal, incluye validación de correo electrónico y comprobación del saldo. This.saldo = 23, es el saldo inicial por defecto.

El metodo boolean validarpaypal(double importe), valida mediante el matches el formato valido de correo según patrón, el importe debe ser mayor que 0 y el saldo suficiente para pagar el importe.

**Clase Bizum (String teléfono)**

```
public Bizum(String telefono) {
    this.telefono = telefono;
    String pin_aleat = "";
    Random aleatorio = new Random();
    for (int i =0; i<6; i++){
        int digito = aleatorio.nextInt(10);
        pin_aleat += digito; // va concatenando

    }
    this.pin = Integer.parseInt(pin_aleat);
    System.out.println("[Chivato Pin]: "+pin_aleat);

}

@Override
public void procesarPago(double importe) {
    System.out.println("Procesando pago de "+importe+ " € con bizum");
}

public boolean validarbizum(String pin_usuario) {


    if (telefono.length() != 9 || !telefono.matches("\\d{9}") ||
!pin_usuario.matches("\\d{6}")){
        return false;
    }

    return pin_usuario.equalsIgnoreCase(String.valueOf(pin));

}
```

El constructor de la clase bizum, al crear el objecto, se genera automáticamente un pin aleatorio de 6 dígitos, se imprime por pantalla como una trampa para que el usuario pueda meterlo y sea igual. Mediante int digito = aleatorio.nextInt(10); se genera un numero entre 0 y 9 y lo concatena con pin_aleat += digito;, después se convierte el String pin a int y se guarda en el atributo pin, this.pin = Integer.parseInt(pin_aleat);,

Por último, se valida el teléfono y el formato del pin con el matches para corroborar que son dígitos y length para la longitud, se compara pin usuario mediante el equalsIgnoreCase, convirtiendo el pin en en String con valueof, ya que deben ser del mismo tipo.

**Clase Tienda**

```java
public static void realizarPago (MetodoPago metodo){
    Scanner sc = new Scanner(System.in);
    System.out.println("¿que importe quieres pagar?");
    double importe = sc.nextDouble();

    metodo.procesarPago(importe);

}
```

Se solicita el importe a pagar y se ejecuta el polimorfismo ya que se llama al procesarPago() del método de pago en concreto.

```java
public static void iniciarPago() {
    Scanner sc = new Scanner(System.in);

    System.out.println("---- MENU DE METODOS DE PAGO ----");
    System.out.println("TARJETA DE CREDITO");
    System.out.println("PAYPAL");
    System.out.println("BIZUM");
    System.out.println("¿que metodo de pago quieres utilizar?");
    String metodo  = sc.nextLine();
    switch (metodo) {
```

Se realiza un método principal que inicia el proceso de pago, muestra el menú con los métodos disponibles y permite al usuario elegir uno, según la opción seleccionada, se crean objetos de TarjetaCredito, paypal o bizum, antes de procesar el pago se realizan validaciones para comprobar los datos.

```java
case "BIZUM":
    System.out.println("Introduce el numero de telefono vinculado con tu bizum: ");
    String telefono = sc.nextLine();
    Bizum bizum1 = new Bizum(telefono);
    System.out.println("introduce tu pin");
    int pin = sc.nextInt();
    String pin_string = String.valueOf(pin);
    System.out.println("validando Bizum...");
    if (bizum1.validarbizum(pin_string)) {
        realizarPago(bizum1);
        System.out.println("Pago aceptado. Muchas gracias");
    }else {
        System.out.println("Los datos de tu bizum no son correctos.");
    }
    break;
```

En el caso bizum se pide que se introduzca el numero de telegono, se crea un objeto bizum1 con el parámetro teléfono, después pide que se introduzca el pin generado anteriormente y se convierte el pin a String para poder validarlo con matches.

Posteriormente se valida Bizum y si es correcto se realiza el pago. Para las demás opciones aplica del mismo modo.

**Clase AppEcommerce ()**

```
package org.example;

public class AppEcommerce {
    public static void main(String[] args) {
        Tienda.iniciarPago();

    }
}
```

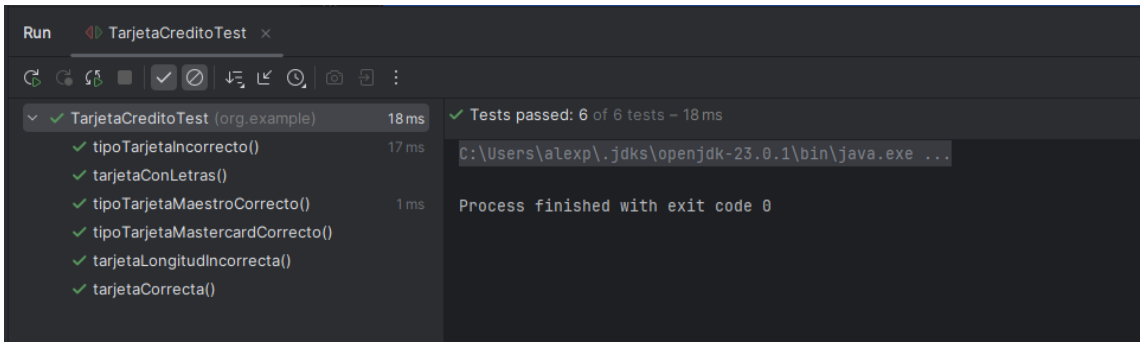
Aqui se inicia el programa ejecutando el menú de la tienda, es decir se inicia el sistema de pagos llamando al menú principal.

## 3. Pruebas con Junit

**Clase tarjeta**

```
package org.example;

import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.*;

public class TarjetaCreditoTest {

    @Test
    public void tarjetaCorrecta() {
        TarjetaCredito t = new TarjetaCredito("1234567891234567", "VISA");
        assertTrue(t.validarTarjeta());
    }

    @Test
    public void tarjetaLongitudIncorrecta() {
        TarjetaCredito t = new TarjetaCredito("12345", "VISA");
        assertFalse(t.validarTarjeta());
    }

    @Test
    public void tarjetaConLetras() {
        TarjetaCredito t = new TarjetaCredito("1234ABCD91234567", "VISA");
        assertFalse(t.validarTarjeta());
    }

    @Test
    public void tipoTarjetaIncorrecto() {
```

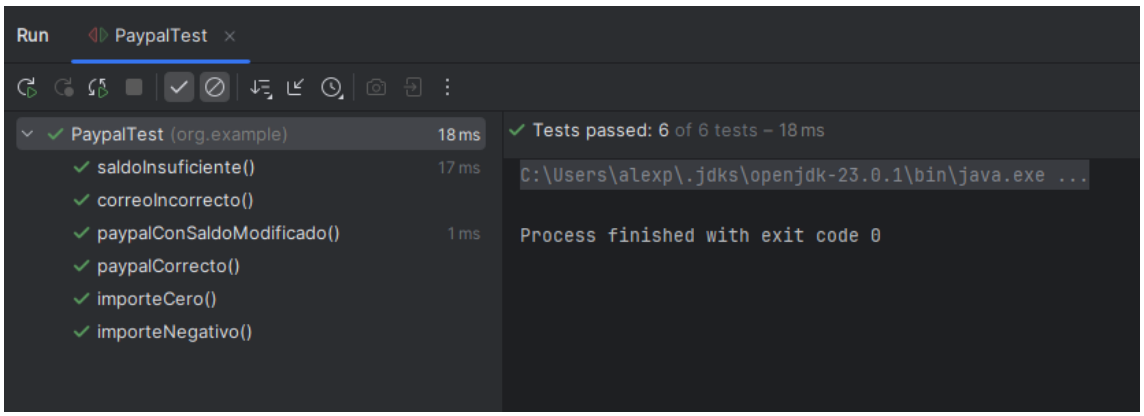
```
        TarjetaCredito t = new TarjetaCredito("1234567891234567", "AMEX");
        assertFalse(t.validarTarjeta());
    }

    @Test
    public void tipoTarjetaMaestroCorrecto() {
        TarjetaCredito t = new TarjetaCredito("1234567891234567", "MAESTRO");
        assertTrue(t.validarTarjeta());
    }

    @Test
    public void tipoTarjetaMastercardCorrecto() {
        TarjetaCredito t = new TarjetaCredito("1234567891234567", "MASTERCARD");
        assertTrue(t.validarTarjeta());
    }
}
```




**Pruebas con Paypaltest.**

```
package org.example;

import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.*;

public class PaypalTest {

    @Test
    public void paypalCorrecto() {
        Paypal paypal1 = new Paypal("usuario@alu.edu.gva.es");
        assertTrue(paypal1.validarPaypal(10));
    }

    @Test
    public void correoIncorrecto() {
        Paypal paypal2 = new Paypal("usuario@gmail.com");
```

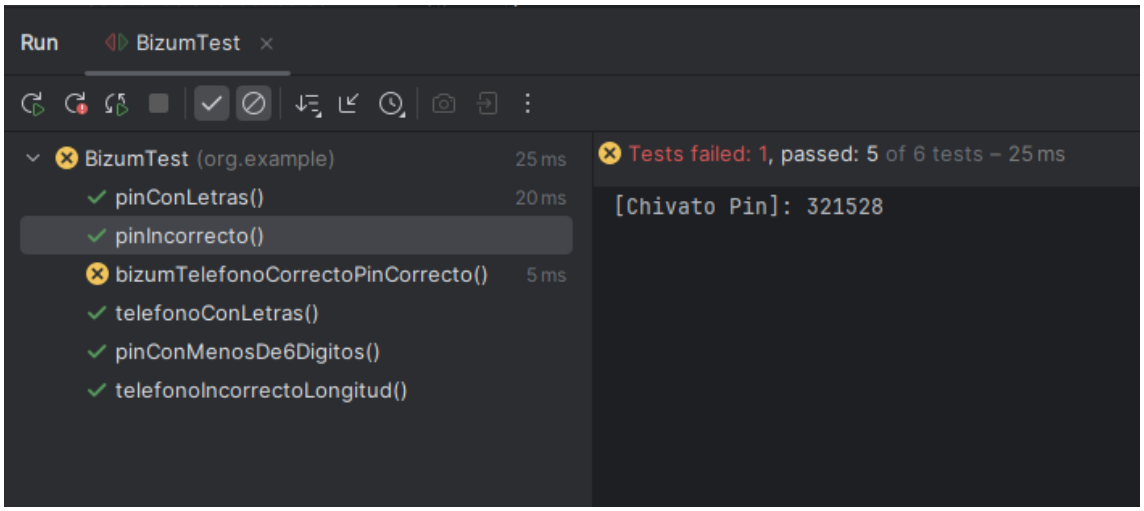
```
        assertFalse(paypal2.validarPaypal(10));
    }

    @Test
    public void saldoInsuficiente() {
        Paypal paypal3 = new Paypal("usuario@alu.edu.gva.es");
        assertFalse(paypal3.validarPaypal(50));
    }

    @Test
    public void importeCero() {
        Paypal paypal4 = new Paypal("usuario@alu.edu.gva.es");
        assertFalse(paypal4.validarPaypal(0));
    }

    @Test
    public void importeNegativo() {
        Paypal paypal5 = new Paypal("usuario@alu.edu.gva.es");
        assertFalse(paypal5.validarPaypal(-5));
    }

    @Test
    public void paypalConSaldoModificado() {
        Paypal paypal6 = new Paypal("usuario@alu.edu.gva.es");
        paypal6.setSaldo(100);
        assertTrue(paypal6.validarPaypal(50));
    }
}
```




**Pruebas con BizumTest**

```
package org.example;

import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.*;
```

```java
public class BizumTest {

    @Test
    public void bizumTelefonoCorrectoPinCorrecto() {
        Bizum bizum1 = new Bizum("612345678");
        String pinCorrecto = String.valueOf(bizum1.getPin());
        assertTrue(bizum1.validarbizum(pinCorrecto));
    }

    @Test
    public void telefonoIncorrectoLongitud() {
        Bizum bizum2 = new Bizum("12345");
        String pinCorrecto = String.valueOf(bizum2.getPin());
        assertFalse(bizum2.validarbizum(pinCorrecto));
    }

    @Test
    public void telefonoConLetras() {
        Bizum bizum3 = new Bizum("61A34567B");
        String pinCorrecto = String.valueOf(bizum3.getPin());
        assertFalse(bizum3.validarbizum(pinCorrecto));
    }

    @Test
    public void pinIncorrecto() {
        Bizum bizum4 = new Bizum("612345678");
        assertFalse(bizum4.validarbizum("111111"));
    }

    @Test
    public void pinConMenosDe6Digitos() {
        Bizum bizum5 = new Bizum("612345678");
        assertFalse(bizum5.validarbizum("123"));
    }

    @Test
    public void pinConLetras() {
        Bizum bizum6 = new Bizum("612345678");
        assertFalse(bizum6.validarbizum("12A45B"));
    }
}
```




## 4. Diagrama PlantUML