PROGRAMACIÓN II Trabajo Práctico 8: Interfaces y Excepciones en Java

Alumno: Zavatti Mauro – legajo 1215
Github: https://github.com/maurozavatti/UTN-TUPaD--P2

Parte 1: Interfaces en un sistema de E-commerce

1. Crear una interfaz **Pagable** con el método **calcularTotal().**

2. Clase **Producto**: tiene nombre y precio, implementa **Pagable.**

3. Clase **Pedido**: tiene una lista de productos, implementa **Pagable** y calcula el total del pedido.

4. Ampliar con interfaces **Pago** y **PagoConDescuento** para distintos medios de pago (**TarjetaCredito, PayPal**), con métodos **procesarPago(double)** y **aplicarDescuento(double)**.

5. Crear una interfaz **Notificable** para notificar cambios de estado. La clase **Cliente** implementa dicha interfaz y **Pedido** debe notificarlo al cambiar de estado.

```
public class MainEcommerce {
  public static void main(String[] args) {
    try {
      Cliente cliente = new Cliente("Mauro", "mauro@email.com");

      Pedido pedido = new Pedido(cliente);
      pedido.agregarProducto(new Producto("Mouse", 2500));
      pedido.agregarProducto(new Producto("Teclado", 4500));

      System.out.println(pedido);

      // Procesar pago con tarjeta
      PagoConDescuento pago = new TarjetaCredito("Mauro Zavatti");
      double totalConDescuento = pago.aplicarDescuento(pedido.calcularTotal());
```

```
            pago.procesarPago(totalConDescuento);

            // Cambiar estado del pedido
            pedido.cambiarEstado("Enviado");
            pedido.cambiarEstado("Entregado");

        } catch (PagoException e) {
            System.out.println("Error en el pago: " + e.getMessage());
        } catch (Exception e) {
            System.out.println("Error inesperado: " + e.getMessage());
        }
    }
}
interface Pagable {
    double calcularTotal();
}


interface Pago {
    void procesarPago(double monto) throws Exception;
}


interface PagoConDescuento extends Pago {
    double aplicarDescuento(double monto);
}
```

```
interface Notificable {
    void notificar(String mensaje);
}
class Producto implements Pagable {
    private String nombre;
    private double precio;

    public Producto(String nombre, double precio) {
        this.nombre = nombre;
        this.precio = precio;
    }

    public String getNombre() { return nombre; }
    public double getPrecio() { return precio; }

    @Override
    public double calcularTotal() {
        return precio;
    }

    @Override
    public String toString() {
        return nombre + " - $" + precio;
    }
}

import java.util.ArrayList;
import java.util.List;
```

```
class Pedido implements Pagable {
    private List<Producto> productos;
    private String estado;
    private Notificable cliente;

    public Pedido(Notificable cliente) {
        this.cliente = cliente;
        this.productos = new ArrayList<>();
        this.estado = "Pendiente";
    }

    public void agregarProducto(Producto p) {
        productos.add(p);
    }

    @Override
    public double calcularTotal() {
        double total = 0;
        for (Producto p : productos) {
            total += p.calcularTotal();
        }
        return total;
    }

    public void cambiarEstado(String nuevoEstado) {
        this.estado = nuevoEstado;
        cliente.notificar("El estado de tu pedido cambió a: " + nuevoEstado);
```

```
    }

    @Override
    public String toString() {
        return "Pedido (" + estado + ") - Total: $" + calcularTotal();
    }
}


class Cliente implements Notificable {
    private String nombre;
    private String email;

    public Cliente(String nombre, String email) {
        this.nombre = nombre;
        this.email = email;
    }

    @Override
    public void notificar(String mensaje) {
        System.out.println("Notificación a " + nombre + ": " + mensaje);
    }

    public String getNombre() { return nombre; }
}

// Excepción personalizada
class PagoException extends Exception {
```

```
    public PagoException(String mensaje) {
        super(mensaje);
    }
}

class TarjetaCredito implements PagoConDescuento {
    private String titular;

    public TarjetaCredito(String titular) {
        this.titular = titular;
    }

    @Override
    public double aplicarDescuento(double monto) {
        // 10% de descuento
        return monto * 0.9;
    }

    @Override
    public void procesarPago(double monto) throws PagoException {
        if (monto <= 0) {
            throw new PagoException("El monto debe ser mayor que 0.");
        }
        System.out.println("Pago de $" + monto + " procesado con Tarjeta de " +
titular);
    }
}
```

```
class PayPal implements PagoConDescuento {
    private String usuario;

    public PayPal(String usuario) {
        this.usuario = usuario;
    }

    @Override
    public double aplicarDescuento(double monto) {
        // 5% de descuento
        return monto * 0.95;
    }

    @Override
    public void procesarPago(double monto) throws PagoException {
        if (monto <= 0) {
            throw new PagoException("El monto debe ser mayor que 0.");
        }
        System.out.println("Pago de $" + monto + " procesado vía PayPal (" + usuario +
")");
    }
}

public class MainEcommerce {
    public static void main(String[] args) {
        try {
            Cliente cliente = new Cliente("Mauro", "mauro@email.com");
```

```
        Pedido pedido = new Pedido(cliente);
        pedido.agregarProducto(new Producto("Mouse", 2500));
        pedido.agregarProducto(new Producto("Teclado", 4500));

        System.out.println(pedido);

        // Procesar pago con tarjeta
        PagoConDescuento pago = new TarjetaCredito("Mauro Zavatti");
        double totalConDescuento = pago.aplicarDescuento(pedido.calcularTotal());
        pago.procesarPago(totalConDescuento);

        // Cambiar estado del pedido
        pedido.cambiarEstado("Enviado");
        pedido.cambiarEstado("Entregado");

      } catch (PagoException e) {
        System.out.println("Error en el pago: " + e.getMessage());
      } catch (Exception e) {
        System.out.println("Error inesperado: " + e.getMessage());
      }
    }
}
```

Parte 2: Ejercicios sobre Excepciones

1. **División segura**

○ Solicitar dos números y dividirlos. Manejar **ArithmeticException** si el divisor es cero.

2. **Conversión de cadena a número**

○ Leer texto del usuario e intentar convertirlo a int. Manejar **NumberFormatException** si no es válido.

3. **Lectura de archivo**

○ Leer un archivo de texto y mostrarlo. Manejar **FileNotFoundException** si el archivo no existe.

4. **Excepción personalizada**

○ Crear **EdadInvalidaException**. Lanzarla si la edad es menor a 0 o mayor a 120. Capturarla y mostrar mensaje.

5. **Uso de try-with-resources**

○ Leer un archivo con **BufferedReader** usando **try-with-resources**. Manejar **IOException** correctamente.

```
package Excepciones;


import java.util.Scanner;


public class DivisionSegura {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);


      try {
         System.out.print("Ingrese el dividendo: ");
         int a = sc.nextInt();
         System.out.print("Ingrese el divisor: ");
         int b = sc.nextInt();
```

```
            int resultado = a / b; // puede lanzar ArithmeticException
            System.out.println("Resultado: " + resultado);

        } catch (ArithmeticException e) {
            System.out.println("Error: no se puede dividir por cero.");
        } finally {
            System.out.println("Operación finalizada.");
        }
    }
}
```

```
package Excepciones;

import java.util.Scanner;

public class ConversionCadenaNumero {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        System.out.print("Ingrese un número en texto: ");
        String entrada = sc.nextLine();

        try {
            int numero = Integer.parseInt(entrada); // puede lanzar
NumberFormatException
            System.out.println("Número convertido correctamente: " + numero);
        } catch (NumberFormatException e) {
```

```
            System.out.println("Error: el texto ingresado no es un número válido.");
        }
    }
}

package Excepciones;



import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;

public class LecturaArchivo {
    public static void main(String[] args) {
        try {
            File archivo = new File("texto.txt");
            Scanner lector = new Scanner(archivo);

            System.out.println("Contenido del archivo:");
            while (lector.hasNextLine()) {
                System.out.println(lector.nextLine());
            }

            lector.close();

        } catch (FileNotFoundException e) {
            System.out.println("Error: el archivo no existe o no se puede abrir.");
        }
```

```
    }
}

package Excepciones;

// Definición de la excepción personalizada
class EdadInvalidaException extends Exception {
    public EdadInvalidaException(String mensaje) {
        super(mensaje);
    }
}

package Excepciones;

// EdadInvalida usa esta clase
public class TestEdadInvalida {
    public static void main(String[] args) {
        try {
            validarEdad(-5);
        } catch (EdadInvalidaException e) {
            System.out.println(" Excepción capturada: " + e.getMessage());
        }
    }

    public static void validarEdad(int edad) throws EdadInvalidaException {
        if (edad < 0 || edad > 120) {
            throw new EdadInvalidaException("La edad debe estar entre 0 y 120 años.");
        } else {
```

```
            System.out.println("Edad válida: " + edad);
        }
    }
}

package Excepciones;

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;

public class LecturaTryWithResources {
    public static void main(String[] args) {
        String ruta = "archivo.txt";

        try (BufferedReader reader = new BufferedReader(new FileReader(ruta))) {
            String linea;
            while ((linea = reader.readLine()) != null) {
                System.out.println(linea);
            }
        } catch (IOException e) {
            System.out.println(" Error al leer el archivo: " + e.getMessage());
        }
    }
}
```