TP 8 – Interfaces y Excepciones en Java

Objetivo:

Desarrollar habilidades en el uso de interfaces y manejo de excepciones en Java para fomentar la modularidad, flexibilidad y robustez del código. Comprender la definición e implementación de interfaces como contratos de comportamiento y su aplicación en el diseño orientado a objetos. Aplicar jerarquías de excepciones para controlar y comunicar errores de forma segura. Diferenciar entre excepciones comprobadas y no comprobadas, y utilizar bloques **try**, **catch**, **finally** y **throw** para garantizar la integridad del programa. Integrar interfaces y manejo de excepciones en el desarrollo de aplicaciones escalables y mantenibles.

Caso práctico:

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.

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.

CONCLUSIONES ESPERADAS

- Comprender la utilidad de las interfaces para lograr diseños desacoplados y reutilizables.
- Aplicar herencia múltiple a través de interfaces para combinar comportamientos.
- Utilizar correctamente estructuras de control de excepciones para evitar caídas del programa.
- Crear excepciones personalizadas para validar reglas de negocio.
- Aplicar buenas prácticas como **try-with-resources** y uso del bloque **finally** para manejar recursos y errores.
- Reforzar el diseño robusto y mantenible mediante la integración de interfaces y manejo de errores en Java.

Parte 1:

1)

```java
package Partel;

public interface Pagable {
    double calcularTotal();
}
```

2)

```java
package Partel;

public 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 "Producto{" + "nombre=" + nombre + ", precio=" + precio + '}';
    }

}
```

3)

```java
package Partel;

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

public class Pedido implements Pagable {
    private List<Producto> productos = new ArrayList<>();
    private String estado;
    private Cliente cliente;

    public Pedido(Cliente cliente) {
        this.cliente = cliente;
        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 pedido cambió de estado a: " + nuevoEstado);
    }

    public String getEstado() {
        return estado;
    }

    public List<Producto> getProductos() {
        return productos;
    }
}
```

4)

```java
package Partel;

public interface Pago {
    void procesarPago(double monto);
}
```

```java
package Partel;

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

```java
package Partel;

public class TarjetaCredito implements PagoConDescuento {
    private String numeroTarjeta;

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

    @Override
    public void procesarPago(double monto) {
        System.out.println("Pago de $" + monto + " procesado con tarjeta de crédito " + numeroTarjeta);
    }

    @Override
    public double aplicarDescuento(double porcentaje) {
        System.out.println("Aplicando descuento de " + porcentaje + "% con tarjeta de crédito");
        return porcentaje;
    }
}
```

```java
package Partel;

public class PayPal implements Pago {
    private String correo;

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

    @Override
    public void procesarPago(double monto) {
        System.out.println("Pago de $" + monto + " procesado con PayPal (" + correo + ")");
    }
}
```

5)

```java
package Partel;

public interface Notificable {
    void notificar(String mensaje);
}
```

```java
package Partel;

public class Cliente implements Notificable {
    private String nombre;
    private String correo;

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

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

    public String getNombre() {
        return nombre;
    }

    public String getCorreo() {
        return correo;
    }
}
```

```java
package Parte1;

import java.io.PrintStream;
import java.nio.charset.StandardCharsets;

public class Main {
    public static void main(String[] args) {
        System.setOut(new PrintStream(System.out, true, StandardCharsets.UTF_8));

        Cliente cliente = new Cliente("Juan Pérez", "juanp@example.com");
        Pedido pedido = new Pedido(cliente);

        Producto p1 = new Producto("Mouse", 1500);
        Producto p2 = new Producto("Teclado", 3000);

        pedido.agregarProducto(p1);
        pedido.agregarProducto(p2);

        System.out.println("Total del pedido: $" + pedido.calcularTotal());

        PagoConDescuento pago = new TarjetaCredito("1234-5678-9012-3456");
        double descuento = pago.aplicarDescuento(10);
        double totalConDescuento = pedido.calcularTotal() * (1 - descuento / 100);
        pago.procesarPago(totalConDescuento);

        pedido.cambiarEstado("Enviado");
    }

}
```

```
ut ×
Debugger Console × TP8 (run) ×
run:
Total del pedido: $4500.0
Aplicando descuento de 10.0% con tarjeta de crédito
Pago de $4050.0 procesado con tarjeta de crédito 1234-5678-9012-3456
Notificación a Juan Pérez (juanp@example.com): El pedido cambió de estado a: Enviado
BUILD SUCCESSFUL (total time: 0 seconds)
```

Parte 2:

1)

```java
package Parte2;

import java.util.Scanner;

public class Ejer1 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        try {
            System.out.print("Ingrese dividendo: ");
            int a = sc.nextInt();
            System.out.print("Ingrese divisor: ");
            int b = sc.nextInt();

            int resultado = a / b;
            System.out.println("Resultado: " + resultado);
        } catch (ArithmeticException e) {
            System.out.println("Error: No se puede dividir por cero.");
        }
    }
}
```

```
Debugger Console × TP8 (run) ×
run:
Ingrese dividendo: 2
Ingrese divisor: 0
Error: No se puede dividir por cero.
BUILD SUCCESSFUL (total time: 2 seconds)
```

2)

```java
package Parte2;

import java.io.PrintStream;
import java.nio.charset.StandardCharsets;
import java.util.Scanner;

public class Ejer2 {
    public static void main(String[] args) {
        System.setOut(new PrintStream(System.out, true, StandardCharsets.UTF_8));

        Scanner sc = new Scanner(System.in);
        try {
            System.out.print("Ingrese un número: ");
            String texto = sc.nextLine();
            int numero = Integer.parseInt(texto);
            System.out.println("Número ingresado: " + numero);
        } catch (NumberFormatException e) {
            System.out.println("Error: El texto ingresado no es un número válido.");
        }
    }
}
```

```
Parte2.Ejer2 > main > sc >
Debugger Console × TP8 (run) × TP8 (run) #2 ×
run:
Ingrese un número: hola
Error: El texto ingresado no es un número válido.
BUILD SUCCESSFUL (total time: 9 seconds)
```

3)

```java
package Parte2;

import java.io.File;
import java.io.FileNotFoundException;
import java.io.PrintStream;
import java.nio.charset.StandardCharsets;
import java.util.Scanner;

public class Ejer3 {
    public static void main(String[] args) {
        System.setOut(new PrintStream(System.out, true, StandardCharsets.UTF_8));
        try {
            File archivo = new File("archivo.txt");
            Scanner lector = new Scanner(archivo);

            while (lector.hasNextLine()) {
                System.out.println(lector.nextLine());
            }
            lector.close();
        } catch (FileNotFoundException e) {
            System.out.println("Error: El archivo no existe.");
        }
    }
}
```

```
Parte2.Ejer3
Debugger Console × TP8 (run) × TP8 (run) #2 ×
run:
Error: El archivo no existe.
BUILD SUCCESSFUL (total time: 0 seconds)
```

4) K

```java
package Parte2;

import java.io.PrintStream;
import java.nio.charset.StandardCharsets;
import java.util.Scanner;

public class Ejer4 {
    public static void main(String[] args) {
        System.setOut(new PrintStream(System.out, true, StandardCharsets.UTF_8));

        Scanner sc = new Scanner(System.in);
        try {
            System.out.print("Ingrese edad: ");
            int edad = sc.nextInt();

            if (edad < 0 || edad > 120) {
                throw new EdadInvalidaException("Edad fuera de rango permitido.");
            }

            System.out.println("Edad válida: " + edad);
        } catch (EdadInvalidaException e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
}
```

```
Parte2.Ejer4
Debugger Console × TP8 (run) × TP8 (run) #2 ×
run:
Ingrese edad: 121
Error: Edad fuera de rango permitido.
BUILD SUCCESSFUL (total time: 2 seconds)
```

```java
package Parte2;

public class EdadInvalidaException extends Exception {
    public EdadInvalidaException(String mensaje) {
        super(mensaje);
    }
}
```

5)

```java
package Parte2;

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

public class Ejer5 {
    public static void main(String[] args) {
        try (BufferedReader br = new BufferedReader(new FileReader("archivo.txt"))) {
            String linea;
            while ((linea = br.readLine()) != null) {
                System.out.println(linea);
            }
        } catch (IOException e) {
            System.out.println("Error al leer el archivo: " + e.getMessage());
        }
    }
}
```

```
ut ×
Debugger Console × TP8 (run) × TP8 (run) #2 ×
run:
Error al leer el archivo: archivo.txt (The system cannot find the file specified)
BUILD SUCCESSFUL (total time: 0 seconds)
```

[Repositorio de github]