PROGRAMACIÓN II

Trabajo Práctico 8: Interfaces & Excepciones

07/12/2025

Alumno:

- Federico Garcia Bengolea - feddericogarciaa@gmail.com
- Comisión: M2025-14
- Link GitHub: https://github.com/FeddericoGarcia/utn-tupad-p2

Profesores:

- **Profesor**: Alberto Cortez
- **Tutor**: Ramiro Hualpa

Objetivo General:

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

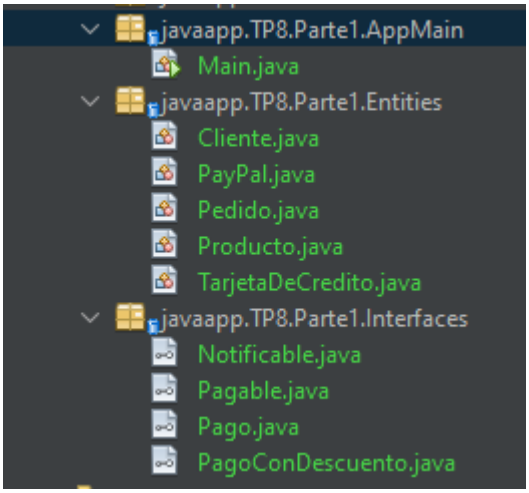
1. Crear una interfaz **Pagable** con el método **calcularTotal().**
2. Clase **Producto**: tiene nombre y precio, implementa **Pagable**.
3. Clase **Pedido**: tiene una lista de productos, implementa **Pagable** y calcula el total del pedido.
4. Ampliar con interfaces **Pago** y **PagoConDescuento** 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.

Parte 1: Interfaces en un sistema de E-commerce

- Estructura de folders:



- AppMain:

```java
package javaapp.TP8.Parte1.AppMain;

import javaapp.TP8.Parte1.Entities.PayPal;
import javaapp.TP8.Parte1.Entities.Producto;
import javaapp.TP8.Parte1.Entities.Pedido;
import javaapp.TP8.Parte1.Entities.TarjetaDeCredito;

public class Main {

    public static void main(String[] args) {
        Producto prod_1 = new Producto(1, "Alfajor", 50.0);
        Producto prod_2 = new Producto(2, "Chocolate", 100.0);
        Producto prod_3 = new Producto(3, "Coca-Cola 1.5ml", 75.0);
        Producto prod_4 = new Producto(4, "Sprite 500ml", 50.0);

        Pedido new_order = new Pedido();

        new_order.setProduct_list(prod_1);
        new_order.setProduct_list(prod_2);
        new_order.setProduct_list(prod_3);
        new_order.setProduct_list(prod_4);

        TarjetaDeCredito tc_1 = new TarjetaDeCredito();
        tc_1.setAmount(150000.0);

        PayPal pp_1 = new PayPal();
        pp_1.setAmount(200.0);

        System.out.println("----------------------------------");
        System.out.println("Productos agregados en el carrito: ");
        System.out.println(new_order.getProduct_list());
        System.out.println("----------------------------------");
        System.out.println("El total de la compra es de $"+ new_order.calcularTotal());
        System.out.println("----------------------------------");
        System.out.println("Realizar pago con TC: ");
        tc_1.procesarPago(new_order.calcularTotal());
        System.out.println("----------------------------------");
        System.out.println("Realizar pago con PayPal: ");
        pp_1.procesarPago(new_order.calcularTotal());
        System.out.println("----------------------------------");

    }

}
```

- Cliente:

```java
package javaapp.TP8.Parte1.Entities;

import javaapp.TP8.Parte1.Interfaces.Notificable;

public class Cliente implements Notificable{
    private int id;
    private String name, email;

    public Cliente(int id, String name, String email) {
        this.id = id;
        this.name = name;
        this.email = email;
    }

    public int getId() {...3 lines }

    public void setId(int id) {...3 lines }

    public String getName() {...3 lines }

    public void setName(String name) {...3 lines }

    public String getEmail() {...3 lines }

    public void setEmail(String email) {...3 lines }

    @Override
    public String toString() {
        return "Cliente{" + "id=" + id + ", name=" + name + ", email=" + email + '}';
    }

    @Override
    public void notifyToCustomer() {
        try{
            if(!email.isEmpty()){
                System.out.println("""
                                   ¡Notificación de compra confirmada!
                                   Queremos informarte que tu compra se realizo exitosamente.
                                   Que lo disfrutes!
                                   """);
            } else {
                throw new Exception("Email vacio");
            }
        }catch (Exception e){
            System.out.println("""
                               ¡Upps, Notificación fallida!
                               No hemos podido enviar el detalle de tu compra.
                               Intentá más tarde...
                               """);
        }
    }
```

- Producto:

```
package javaapp.TP8.Partel.Entities;
import javaapp.TP8.Partel.Interfaces.Pagable;

public class Producto implements Pagable{
    private int id;
    private String name;
    private double price;

    public Producto() {
    }

    public Producto(int id, String name, double price) {
        this.id = id;
        this.name = name;
        this.price = price;
    }

    public int getId() {...3 lines }

    public void setId(int id) {...3 lines }

    public String getName() {...3 lines }

    public void setName(String name) {...3 lines }

    public double getPrice() {...3 lines }

    public void setPrice(double price) {...3 lines }

    @Override
    public String toString() {
        return "\nProducto " + "id=" + id + ", name=" + name + ", price=" + price + "}";
    }

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

}
```

- Pedido:

```java
package javaapp.TP8.Partel.Entities;

import java.util.ArrayList;
import javaapp.TP8.Partel.Interfaces.Pagable;


public class Pedido implements Pagable{
    private ArrayList<Producto> product_list;
    private Cliente cte;

    public Pedido() {
        this.product_list = new ArrayList<>();
    }

    public Pedido(ArrayList<Producto> product_list) {
        this.product_list = new ArrayList<>();
    }

    public ArrayList<Producto> getProduct_list() {
        return product_list;
    }

    public void setProduct_list(Producto product) {
        product_list.add(product);
    }

    public void changeState(boolean new_state){
        if (new_state){
            cte.notifyToCustomer();
        }
    }

    @Override
    public String toString() {
        return "Pedido{" + "product_list=" + product_list + '}';
    }

    @Override
    public double calcularTotal() {
        double result = 0;
        for (int i = 0; i < product_list.size(); i++) {
            result += product_list.get(i).getPrice();
        }
        return result;
    }

}
```

- TarjetaDeCredito:

```java
package javaapp.TP8.Partel.Entities;

import javaapp.TP8.Partel.Interfaces.Pago;
import javaapp.TP8.Partel.Interfaces.PagoConDescuento;

public class TarjetaDeCredito implements Pago, PagoConDescuento{
    double amount;

    public void setAmount(double amount) {
        this.amount = amount;
    }

    @Override
    public void procesarPago(double payment) {
        double result = 0;
        try{
            if(payment <= amount){
                result += amount - payment;
                System.out.println("¡Pago aprobado!");
                System.out.println("El pago fue de $"+ payment);
            } else {
                throw new Exception("Fondos insuficientes en tu Tarjeta de Crédito");
            }
        } catch (Exception e){
            System.out.println("No fue posible continuar con la operación.");
        }
    }

    @Override
    public double aplicarDescuento(double total) {
        return total * 0.20;
    }

}
```

- PayPal:

```java
package javaapp.TP8.Partel.Entities;

import javaapp.TP8.Partel.Interfaces.Pago;
import javaapp.TP8.Partel.Interfaces.PagoConDescuento;

public class PayPal implements Pago, PagoConDescuento{
    double amount;

    public void setAmount(double amount) {
        this.amount = amount;
    }

    @Override
    public void procesarPago(double payment) {
        double result = 0;
        try{
            if(payment <= amount){
                result += amount - payment;
                System.out.println("¡Pago aprobado!");
                System.out.println("El pago fue de $"+ payment);
            } else {
                System.out.println("Fondos insuficientes en tu cuenta de PayPal");
                throw new Exception();
            }
        } catch (Exception e){
            System.out.println("No fue posible continuar con la operación.");
        }
    }

    @Override
    public double aplicarDescuento(double total) {
        return total * 0.15;
    }

}
```

- Notificable:

```java
package javaapp.TP8.Partel.Interfaces;

public interface Notificable {

    void notifyToCustomer();
}
```

- Pagable:

```
package javaapp.TP8.Partel.Interfaces;

public interface Pagable {

    double calcularTotal();

}
```

- Pago:

```
package javaapp.TP8.Partel.Interfaces;

public interface Pago {

    void procesarPago(double payment);
}
```

- PagoConDescuento:

```
package javaapp.TP8.Partel.Interfaces;

public interface PagoConDescuento {

    double aplicarDescuento(double total);
}
```

Parte 2: Ejercicios sobre Excepciones

- Main:

```
package javaapp.TP8.Parte2;

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

import javaapp.TP8.Parte2.CustomExceptions.EdadInvalidaException;


public class Main {

    public static void main(String[] args) throws FileNotFoundException, Exception {
        mostrarMenu();
    }
    public static void mostrarMenu() throws FileNotFoundException, Exception{
        Scanner sc = new Scanner(System.in);

        int opt = -1;

        while (opt != 0) {
            System.out.println("Ingresá la opción deseada: ");
            System.out.println("1. Division Segura");
            System.out.println("2. Conversion Cadena a Número");
            System.out.println("3. Lectura de archivo");
            System.out.println("4. Verificar Edad");
            System.out.println("5. Leer buffer");
            System.out.println("0. Salir");

            opt = sc.nextInt();

            switch(opt){
                case 1: divisionSegura(); break;
                case 2: conversionCadenaNumero(); break;
                case 3: lecturaArchivo(); break;
                case 4: verificarEdad(); break;
                case 5: leerBuffer(); break;
                case 0: {
                    System.out.println("Saliendo del sistema");
                    return;
                }
                default:
                    System.out.println("Opción incorrecta.");
            }
        }
    }
```

- División Segura:

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

    System.out.println("Divisón segura");
    int result, num1, num2;
    System.out.println("Ingresa el primer número");
    num1 = sc.nextInt();
    System.out.println("Ingresa el segundo número");
    num2 = sc.nextInt();

    try{
        result = num1 / num2;
        System.out.println("Tu resultado es: "+ result);
    } catch (ArithmeticException e){
        System.out.println("No es posible realizar una división por 0 (cero). \n"+ e.getMessage());
    }
    System.out.println("------------------------\n");

}
```

- Conversión de cadena a número:

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

    System.out.println("Conversion de String a Int");
    System.out.println("Ingresa una cadena de texto: ");
    String input = sc.nextLine();

    try {
        int result = Integer.parseInt(input);
        System.out.println("El número convertido es: "+ result);
    } catch(NumberFormatException e) {
        System.out.println("No es posible convertir tu texto a formato numérico. \n"+ e.getMessage());
    }
    System.out.println("------------------------\n");

}
```

- Lectura de archivo:

```java
public static void lecturaArchivo() throws FileNotFoundException, Exception{
    System.out.println("Lectura de archivo de texto");

    try{
       File file = new File("texto.txt");
       Scanner reader = new Scanner(file);

       System.out.println("-----------------------");
       while (reader.hasNextLine()){
           System.out.println(reader.nextLine());
       }

       reader.close();

    } catch (FileNotFoundException e){
        System.out.println("No se encontró el archivo seleccionado. \n"+ e.getMessage());
    }
    System.out.println("-----------------------\n");

}
```

- Excepción Personalizada:

```java
public static void verificarEdad() throws Exception{
    Scanner sc = new Scanner(System.in);
    System.out.println("Ingresa tu edad: ");
    int age;
    try {
        age = sc.nextInt();
        if (age <= 0 || age > 120){
            throw new EdadInvalidaException("Edad invalida");
        } else {
            System.out.println("Tenes "+ age +" años!");
        }
    } catch (EdadInvalidaException e){
        System.out.println("No es posible ingresar un valor menor a 0 y mayor a 120.\n");
        e.getMessage();
    }
    System.out.println("-----------------------\n");
}
```

```java
package javaapp.TP8.Parte2.CustomExceptions;

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

- Uso de try-with-resources:

```
    public static void leerBuffer() throws IOException{

        File file = new File("texto.txt");

        try (BufferedReader br = new BufferedReader(new java.io.FileReader(file))) {
            String linea;
            System.out.println("------------------------");
            while ((linea = br.readLine()) != null) {
                System.out.println(linea);
            }

        } catch (IOException e) {
            System.out.println("Ocurrió un error al leer el archivo:");
            System.out.println(e.getMessage());
        }

        System.out.println("------------------------\n");

    }

}
```