UNIVERSIDAD TECNOLOGICA NACIONAL

Trabajo Práctico 8

Interfaces y Excepciones

Programación II

Alumno

Alex Pedro Dauria

Fecha de Entrega

23 de Octubre de 2025

Índice

Link al Repositorio en GitHub

https://github.com/Alex-Dauria/UTN-TUPaD-P2/tree/main/08%20INTERFACES%20Y%20EXCEPCIONES

Parte 1: *Interfaces en sistema E-commerce*

1) Creé la interfaz Pagable con el método calcularTotal().

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

2) Clase Producto con nombre y precio, implementa Pagable.

```
public class Producto implements Pagable {
    private String nombre;
    private double precio;

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

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

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

3) Clase Pedido con lista de productos, implementa Pagable.

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

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

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

    public void cambiarEstado(String nuevoEstado) {
        this.estado = nuevoEstado;
        if (notificable != null) {
            notificable.notificar("El pedido cambió al estado: " + nuevoEstado);
        }
    }

    @Override
    public double calcularTotal() {
        return productos.stream()
                        .mapToDouble(Producto::calcularTotal)
                        .sum();
    }

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

4) Interfaces Pago y PagoConDescuento para medios de pago.

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

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

5) Clases TarjetaCredito y PayPal que implementan las interfaces.

```java
public class TarjetaCredito implements PagoConDescuento {
    private String numeroTarjeta;
    private String titular;

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

    @Override
    public void procesarPago(double monto) {
        System.out.println("Procesando pago con tarjeta de credito por: $" + monto);
        System.out.println("Tarjeta: " + numeroTarjeta.substring(numeroTarjeta.length() - 4));
    }

    @Override
    public double aplicarDescuento(double monto) {
        double descuento = monto * 0.10; // 10% de descuento
        double montoConDescuento = monto - descuento;
        System.out.println("Descuento aplicado: $" + descuento);
        System.out.println("Monto con descuento: $" + montoConDescuento);
        return montoConDescuento;
    }
}
```

```java
public class PayPal implements Pago {
    private String email;

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

    @Override
    public void procesarPago(double monto) {
        System.out.println("Procesando pago con PayPal por: $" + monto);
        System.out.println("Email: " + email);
    }
}
```

6) Interfaz Notificable y clase Cliente.

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

```java
public 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 para " + nombre + " (" + email + "): " + mensaje);
    }

    public String getNombre() { return nombre; }
    public String getEmail() { return email; }
}
```

7) Main para probar el sistema E-commerce.

```java
public class Main {
    public static void main(String[] args) {
        System.out.println("=== SISTEMA E-COMMERCE ===");

        // Parte 1: Sistema de E-commerce con Interfaces
        Cliente cliente = new Cliente("Juan Pérez", "juan@email.com");

        Producto producto1 = new Producto("Laptop", 1500.00);
        Producto producto2 = new Producto("Mouse", 25.50);
        Producto producto3 = new Producto("Teclado", 75.00);

        Pedido pedido = new Pedido(cliente);
        pedido.agregarProducto(producto1);
        pedido.agregarProducto(producto2);
        pedido.agregarProducto(producto3);

        System.out.println("\n--- Información del Pedido ---");
        System.out.println("Total del pedido: $" + pedido.calcularTotal());

        // Cambiar estado (debe notificar al cliente)
        pedido.cambiarEstado("PROCESANDO");
        pedido.cambiarEstado("ENVIADO");

        // Procesar pagos con diferentes métodos
        System.out.println("\n--- Procesamiento de Pagos ---");
        TarjetaCredito tarjeta = new TarjetaCredito("1234567812345678", "Juan Pérez");
        double totalConDescuento = tarjeta.aplicarDescuento(pedido.calcularTotal());
        tarjeta.procesarPago(totalConDescuento); // Paga el monto con descuento

        PayPal paypal = new PayPal("juan@email.com");
        paypal.procesarPago(pedido.calcularTotal());
```

Salida en consola

```
=== SISTEMA E-COMMERCE ===

--- Informaci▯n del Pedido ---
Total del pedido: $1600.5
Notificacion para Juan Perez (juan@email.com): El pedido cambio al estado: PROCESANDO
Notificacion para Juan Perez (juan@email.com): El pedido cambio al estado: ENVIADO

--- Procesamiento de Pagos ---
Descuento aplicado: $160.05
Monto con descuento: $1440.45
Procesando pago con tarjeta de credito por: $1440.45
Tarjeta: 5678
Procesando pago con PayPal por: $1600.5
Email: juan@email.com
```

Parte 2: *Ejercicios Excepciones*

1) División segura - Manejo de ArithmeticException

```java
public class EjerciciosExcepciones {

    // 1. División segura - Maneja ArithmeticException
    public static void divisionSegura() {
        Scanner scanner = new Scanner(System.in);
        try {
            System.out.print("Ingrese el dividendo: ");
            int dividendo = scanner.nextInt();
            System.out.print("Ingrese el divisor: ");
            int divisor = scanner.nextInt();

            int resultado = dividendo / divisor;
            System.out.println("Resultado: " + resultado);

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

Salida en consola

```
Ingrese el dividendo: 5
Ingrese el divisor: 0
Error: No se puede dividir por cero
```

```
Ingrese el dividendo: 50
Ingrese el divisor: 5
Resultado: 10
```

2) Conversión de cadena a número - Manejo de NumberFormatException

```java
// 2. Conversión de cadena a número - Maneja NumberFormatException
public static void conversionCadenaNumero() {
    Scanner scanner = new Scanner(System.in);
    System.out.print("Ingrese un número: ");
    String entrada = scanner.nextLine();

    try {
        int numero = Integer.parseInt(entrada);
        System.out.println("Número convertido: " + numero);
    } catch (NumberFormatException e) {
        System.out.println("Error: '" + entrada + "' no es un número válido");
    }
}
```

Salida en consola

```
Ingrese un numero: dd22
Error: 'dd22' no es un numero valido
```

```
Ingrese un numero: 150
Numero convertido: 150
```

3) Lectura de archivo - Manejo de FileNotFoundException

```java
// 3. Lectura de archivo - Maneja FileNotFoundException
public static void lecturaArchivo() {
    try {
        FileReader archivo = new FileReader("src/textos/archivo_ejemplo.txt");
        BufferedReader lector = new BufferedReader(archivo);
        String linea;

        System.out.println("Contenido del archivo:");
        while ((linea = lector.readLine()) != null) {
            System.out.println(linea);
        }

        lector.close();
    } catch (java.io.FileNotFoundException e) {
        System.out.println("Error: Archivo no encontrado");
    } catch (IOException e) {
        System.out.println("Error de lectura: " + e.getMessage());
    }
}
```

Salida en consola

```
Contenido del archivo:
Este es un archivo de ejemplo
para probar la lectura de archivos en el ejercicio de excepciones.
Linea 3 del archivo.
Linea 4 del archivo.
```

```
Error: Archivo no encontrado
```

4) Excepción personalizada - EdadInvalidaException

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

```java
// 4. Excepción personalizada - Valida edad
public static void validarEdad(int edad) throws EdadInvalidaException {
    if (edad < 0 || edad > 120) {
        throw new EdadInvalidaException("Edad " + edad + " no es válida. Debe estar entre 0 y 120");
    }
    System.out.println("Edad válida: " + edad);
}
```

Salida en consola

```
--- Validacion de Edad ---
Edad valida: 25
Excepcion capturada: Edad 150 no es valida. Debe estar entre 0 y 120
```

5) Uso de try-with-resources

```java
// 5. Try-with-resources - Manejo automático de recursos
public static void leerConTryWithResources() {
    try (BufferedReader lector = new BufferedReader(new FileReader("src/textos/archivo_ejemplo.txt"))) {
        String linea;
        System.out.println("Leyendo con try-with-resources:");
        while ((linea = lector.readLine()) != null) {
            System.out.println(linea);
        }
    } catch (IOException e) {
        System.out.println("Error: " + e.getMessage());
    }
}
```

Salida en consola

```
Leyendo con try-with-resources:
Este es un archivo de ejemplo
para probar la lectura de archivos en el ejercicio de excepciones.
Linea 3 del archivo.
Linea 4 del archivo.
```

```
Error: src\textos\archivo_ejemplo.txt (El sistema no puede encontrar el archivo especificado)
```

6) Main para probar todos los ejercicios

```java
// Parte 2: Ejercicios de Excepciones
EjerciciosExcepciones.divisionSegura();
EjerciciosExcepciones.conversionCadenaNumero();
EjerciciosExcepciones.lecturaArchivo();
EjerciciosExcepciones.leerConTryWithResources();

// Validación de edad con excepción personalizada
System.out.println("\n--- Validacion de Edad ---");
try {
    EjerciciosExcepciones.validarEdad(25);
    EjerciciosExcepciones.validarEdad(150); // Esto lanzará excepción
} catch (EdadInvalidaException e) {
    System.out.println("Excepcion capturada: " + e.getMessage());
}
```