

## PROGRAMACIÓN II

### TP 8: Interfaces y Excepciones en Java

#### 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.

Concepto	Aplicación en el proyecto
Interfaces	Definición de contratos de comportamiento común entre distintas clases
Herencia múltiple con interfaces	Permite que una clase implementa múltiples comportamientos sin herencia de estado
Implementación de interfaces	Uso de <b>implements</b> para que una clase cumpla con los métodos definidos en una interfaz
Excepciones	Manejo de errores en tiempo de ejecución mediante estructuras <b>try-catch</b>
Excepciones checked y unchecked	Diferencias y usos según la naturaleza del error
Excepciones personalizadas	Creación de nuevas clases que extienden <b>Exception</b>
finally y try-with-resources	Buenas prácticas para liberar recursos correctamente
Uso de throw y throws	Declaración y lanzamiento de excepciones

Interfaces	Definición de contratos de comportamiento común entre distintas clases
Herencia múltiple con interfaces	Permite que una clase implementa múltiples comportamientos sin herencia de estado

## MARCO TEÓRICO

### Caso Practico

#### 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.