Trabajo Práctico 8 – Interfaces y Excepciones en Java

Resumen General

Este trabajo práctico aborda dos ejes fundamentales del lenguaje Java:

1. Interfaces y diseño orientado a objetos.

2. Manejo de excepciones, tanto chequeadas como no chequeadas.

El objetivo es aplicar estos conceptos en un mini-sistema y en ejercicios independientes.

Parte 1 – Interfaces en un Sistema de E-commerce

Se desarrolla un pequeño sistema utilizando interfaces para representar comportamientos compartidos entre clases:

• Pagable: interfaz que define el método calcularTotal().

• Notificable: interfaz usada para notificar al cliente del estado de un pedido.

• Pago y PagoConDescuento: interfaces para medios de pago, incluyendo descuentos.

• Producto: clase que implementa Pagable.

• Cliente: implementa Notificable.

• Pedido: implementa Pagable y administra estados del pedido, notificando al cliente.

• TarjetaCredito y PayPal: medios de pago que implementan Pago/PagoConDescuento.

• MainEcommerce: clase principal que demuestra el funcionamiento del sistema.

Esta parte permite trabajar herencia múltiple mediante interfaces, polimorfismo y separación de responsabilidades.

Parte 2 – Ejercicios de Excepciones

Se trabajan los distintos tipos de excepciones y formas de manejo en Java:

1. División segura con captura de ArithmeticException e InputMismatchException.

2. Conversión segura de cadena a número con NumberFormatException.

3. Lectura de archivos con manejo de FileNotFoundException y IOException.

4. Creación de una excepción personalizada: EdadInvalidaException.

5. Uso de try-with-resources para cierre automático de recursos.

Propósito del manejo de excepciones:

• Evitar que el programa termine de forma abrupta.

• Detectar situaciones no válidas y actuar adecuadamente.

• Aprender a lanzar (throw) y declarar (throws) excepciones.

• Garantizar el cierre de recursos mediante finally o try-with-resources.

Conclusión

Este trabajo integra conceptos centrales del paradigma orientado a objetos (interfaces, polimorfismo y responsabilidad) junto con el uso avanzado de excepciones, permitiendo construir programas más robustos, seguros y mantenibles.