

# Trabajo Práctico: Interfaces y Excepciones en Java

Nombre: Alfredo de Inocenti

Materia: Programación II

Carrera: Tecnicatura Universitaria en Programación

## Repositorio:

[https://github.com/fonsecaFuentes/tps-Programacion\\_II/tree/main/Trabajo\\_Practico\\_7](https://github.com/fonsecaFuentes/tps-Programacion_II/tree/main/Trabajo_Practico_7)

---

## Objetivo general

Aplicar interfaces en un sistema de e-commerce para comprender desacoplamiento, extensibilidad y notificaciones de estado, y completar una serie de ejercicios de excepciones, reforzando el manejo correcto de errores en Java.

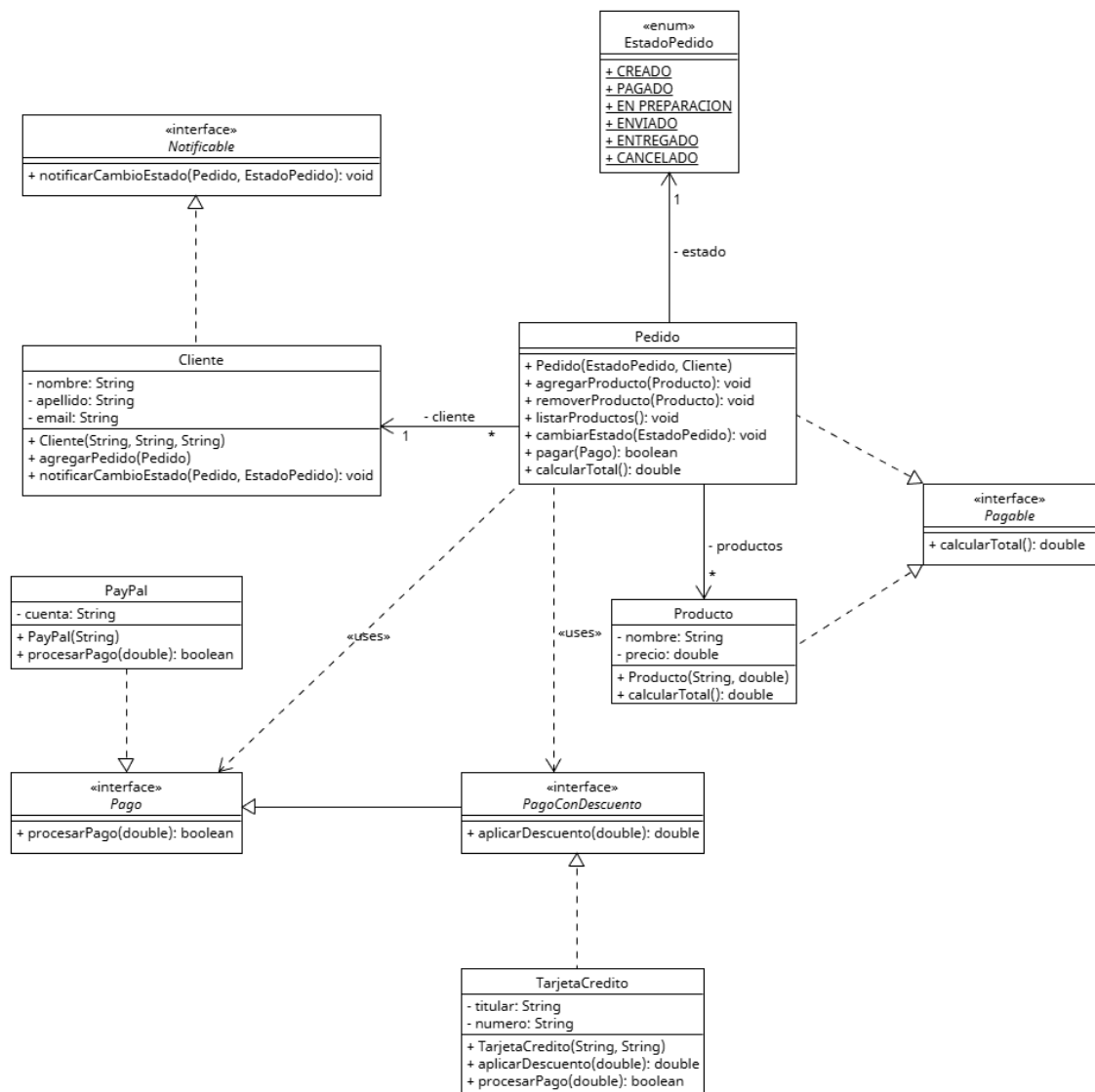
## Parte 1 — E-commerce con Interfaces

### Requerimientos (resumen de la consigna)

1. Interfaz Pagable con método calcularTotal().
2. Clase Producto: nombre, precio, implementa Pagable.
3. Clase Pedido: mantiene lista de Producto, implementa Pagable y calcula el total del pedido.
4. Medios de pago:
  - Interfaz Pago → procesarPago(double).
  - Interfaz PagoConDescuento (extiende Pago) → aplicarDescuento(double).
  - Clases: PayPal (implementa Pago) y TarjetaCredito (implementa PagoConDescuento).
5. Notificaciones:
  - Interfaz Notificable → notificarCambioEstado(Pedido, EstadoPedido).
  - Cliente implementa Notificable.
  - Pedido notifica a su Cliente al cambiar de estado.

### Diagrama (descripción)

- Clases/Interfaces:
  - Pagable ← Producto, Pedido
  - Pago ← PayPal
  - PagoConDescuento (extends Pago) ← TarjetaCredito
  - Notificable ← Cliente
  - Pedido → mantiene List<Producto> y tiene un Cliente
  - Pedido depende de Pago/PagoConDescuento (líneas de dependencia)
  - Pedido notifica a Cliente (invoca notificarCambioEstado)



## Parte 2 — Ejercicios sobre Excepciones

### 1) División segura

- Solicitar dos enteros y dividir.
- Manejar `ArithmeticException` (divisor = 0) y entradas inválidas.

### 2) Conversión de cadena a número

- Leer texto y convertir a `int` con `Integer.parseInt`.
- Manejar `NumberFormatException`.

### 3) Lectura de archivo (ruta por defecto)

- Leer archivo de texto y mostrar su contenido.
- Manejar `FileNotFoundException` (versión con `Scanner(File)`), y en variante `try-with-resources` manejar `IOException`.

#### 4) Excepción personalizada — EdadInvalidaException

- Versión enriquecida: guarda edad, min, max; genera mensaje por defecto; ofrece helpers (esMenorAlMinimo, esMayorAlMaximo).
- Lanzar si edad  $\notin$  [0,120]. Capturar y mostrar mensaje claro al usuario.

#### 5) try-with-resources con BufferedReader

- Leer archivo con BufferedReader en bloque *try-with-resources*.
- Manejar IOException correctamente.