TRABAJO PRÁCTICO 8

Interfaces y Excepciones en Java
Tecnicatura en Programación a Distancia
Programación II

Alumno: Fabricio Puccio
Fecha de Entrega: 21-11-2025
Repositorio GitHub:
https://github.com/FabricioPuccio/TP8-Interfaces-Excepciones

ÍNDICE

---

1. INTRODUCCIÓN TEÓRICA

1.1 Interfaces en Java

Las interfaces en Java son contratos que definen comportamientos que las clases deben implementar. Permiten lograr:

- Desacoplamiento entre componentes
- Polimorfismo sin herencia múltiple
- Flexibilidad en el diseño del software

1.2 Manejo de Excepciones

El manejo de excepciones garantiza la robustez del software mediante:

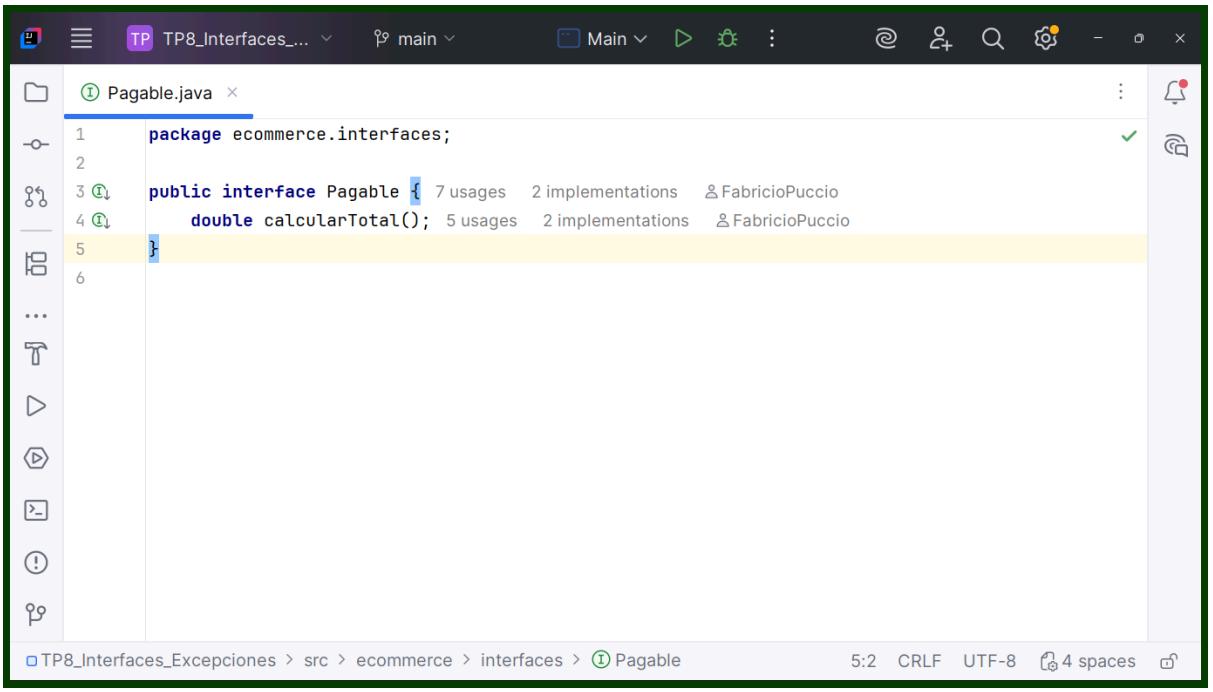
- Control de errores en tiempo de ejecución
- Recuperación graceful de fallos
- Mantenimiento de la integridad del programa

2. PARTE 1: SISTEMA E-COMMERCE

2.1 Diseño de Interfaces

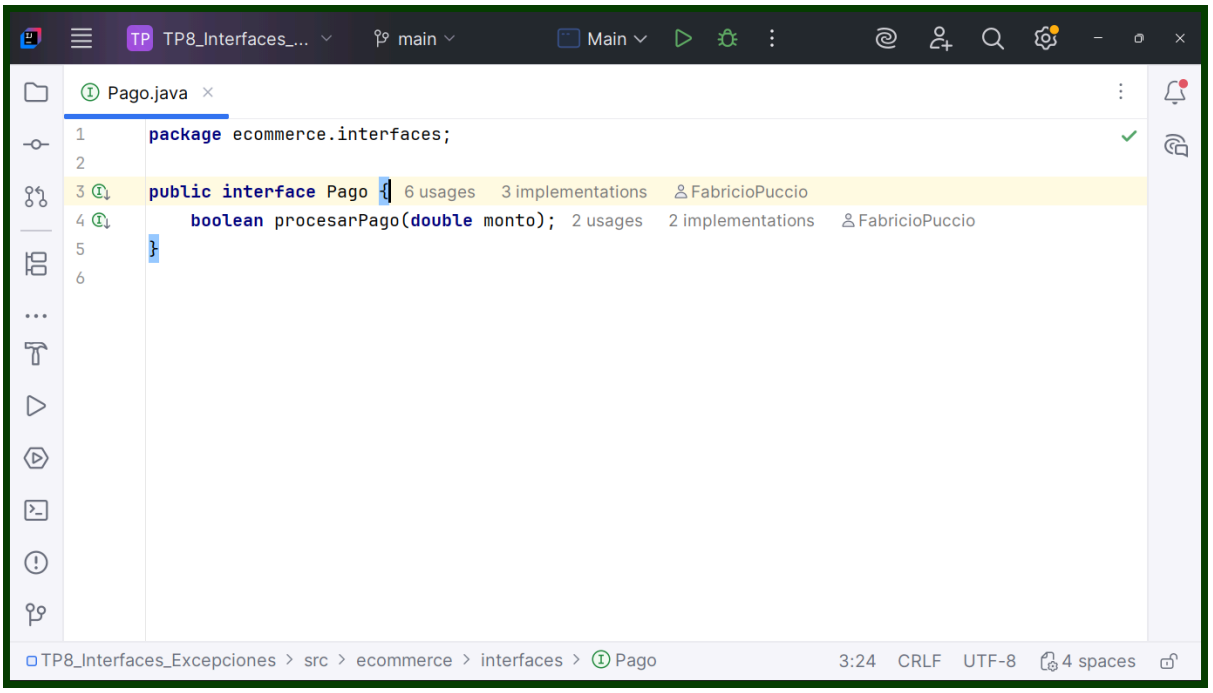
Interfaz Pagable

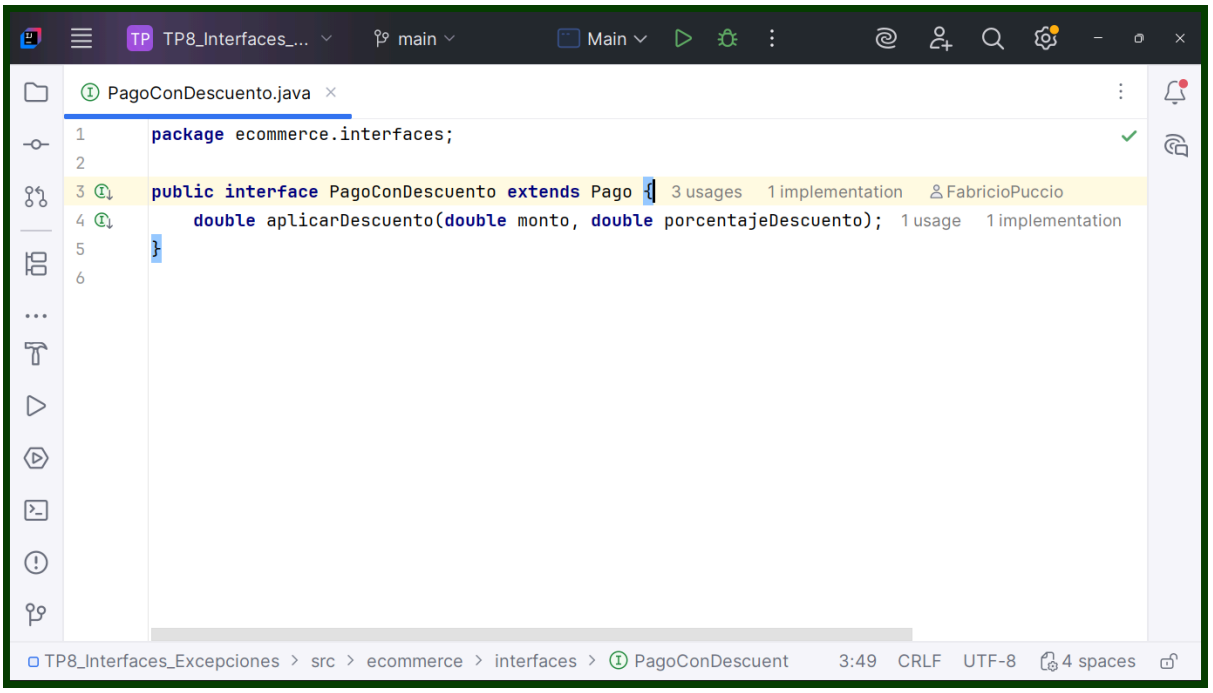
Define el contrato para elementos que pueden calcular un total.



Interfaz Pago y PagoConDescuento

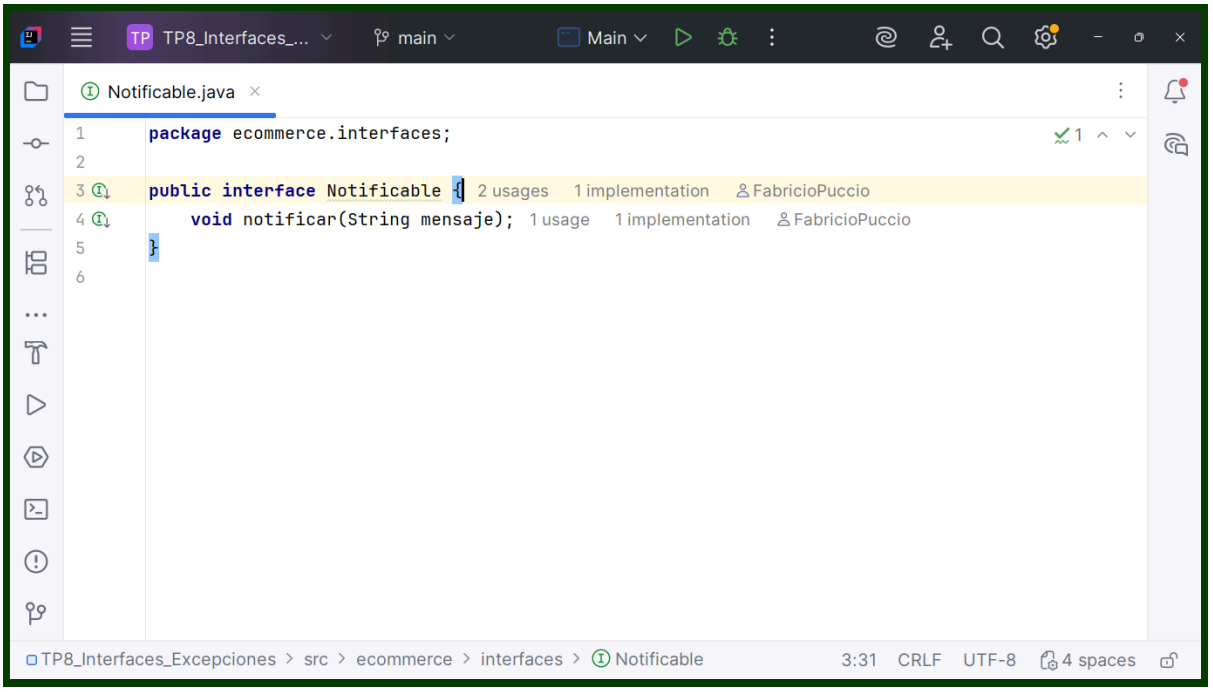
Especializan los métodos de procesamiento de pagos.





Interfaz Notificable

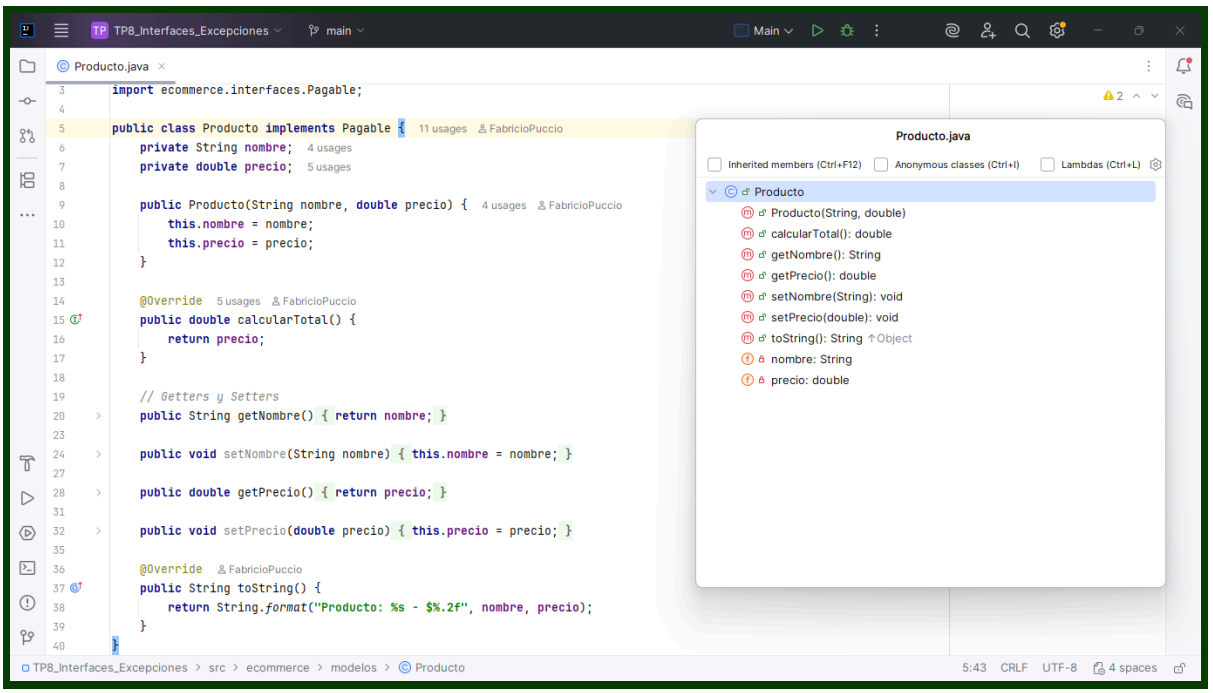
Permite notificar cambios de estado a los clientes.



2.2 Implementación de Clases

Clase Producto

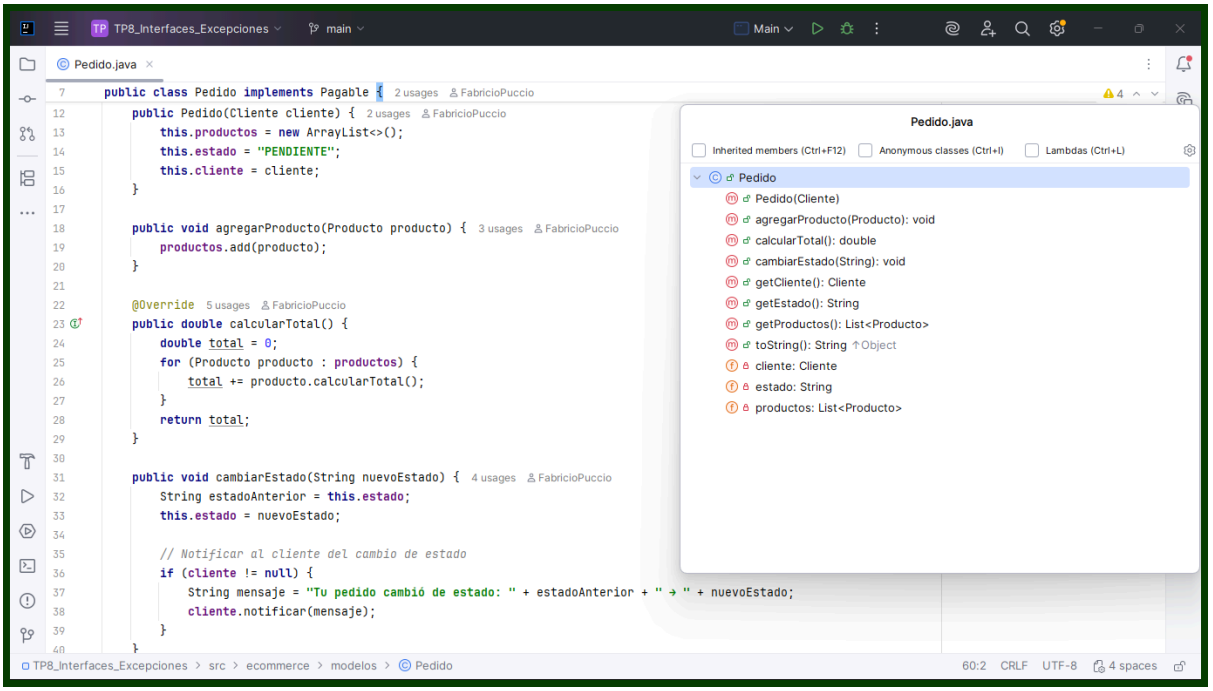
Implementa Pagable representando productos individuales.



```java
import ecommerce.interfaces.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;
    }

    // Getters y Setters
    public String getNombre() { return nombre; }

    public void setNombre(String nombre) { this.nombre = nombre; }

    public double getPrecio() { return precio; }

    public void setPrecio(double precio) { this.precio = precio; }

    @Override
    public String toString() {
        return String.format("Producto: %s - $%.2f", nombre, precio);
    }
}
```

Clase Pedido

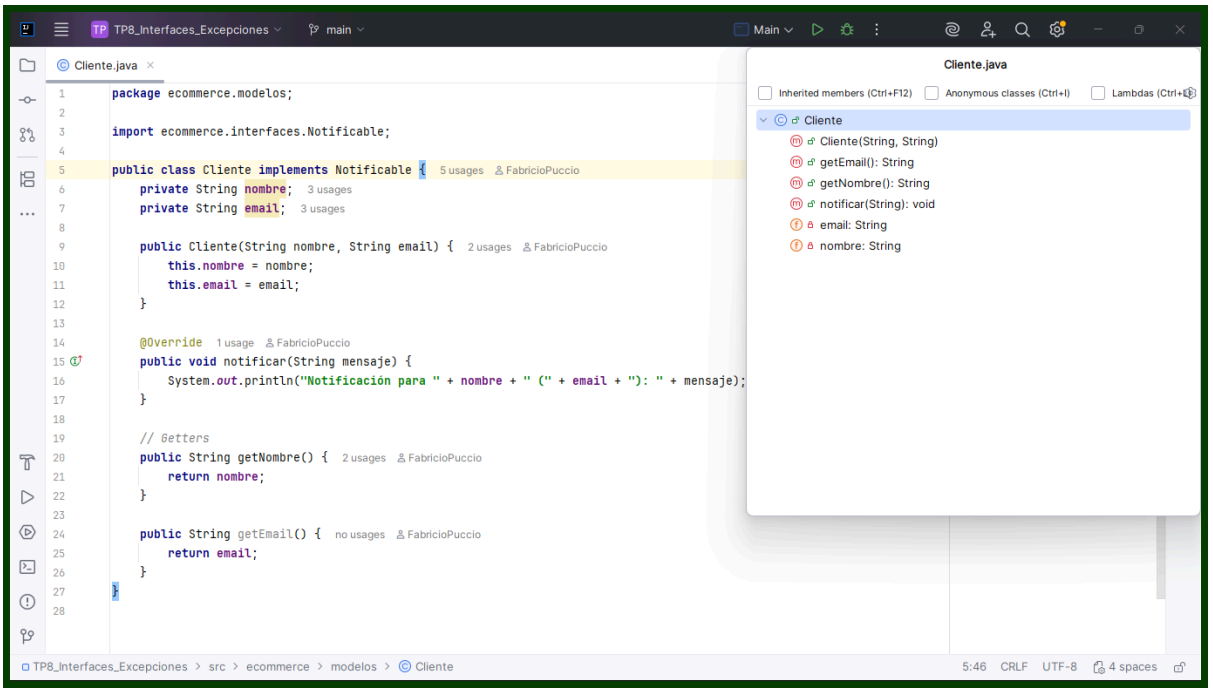
Gestiona múltiples productos y notifica cambios.



```java
public class Pedido implements Pagable {
    public Pedido(Cliente cliente) {
        this.productos = new ArrayList<>();
        this.estado = "PENDIENTE";
        this.cliente = cliente;
    }

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

    @Override
    public double calcularTotal() {
        double total = 0;
        for (Producto producto : productos) {
            total += producto.calcularTotal();
        }
        return total;
    }

    public void cambiarEstado(String nuevoEstado) {
        String estadoAnterior = this.estado;
        this.estado = nuevoEstado;

        // Notificar al cliente del cambio de estado
        if (cliente != null) {
            String mensaje = "Tu pedido cambió de estado: " + estadoAnterior + " → " + nuevoEstado;
            cliente.notificar(mensaje);
        }
    }
}
```

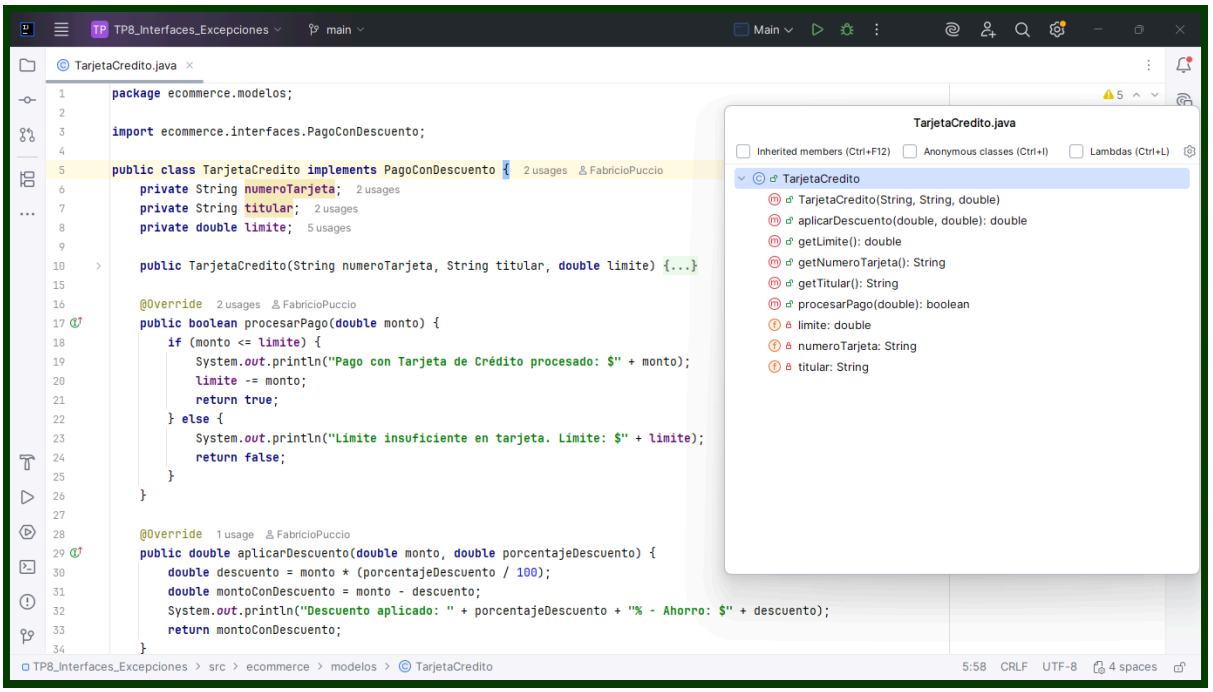
Clase Cliente

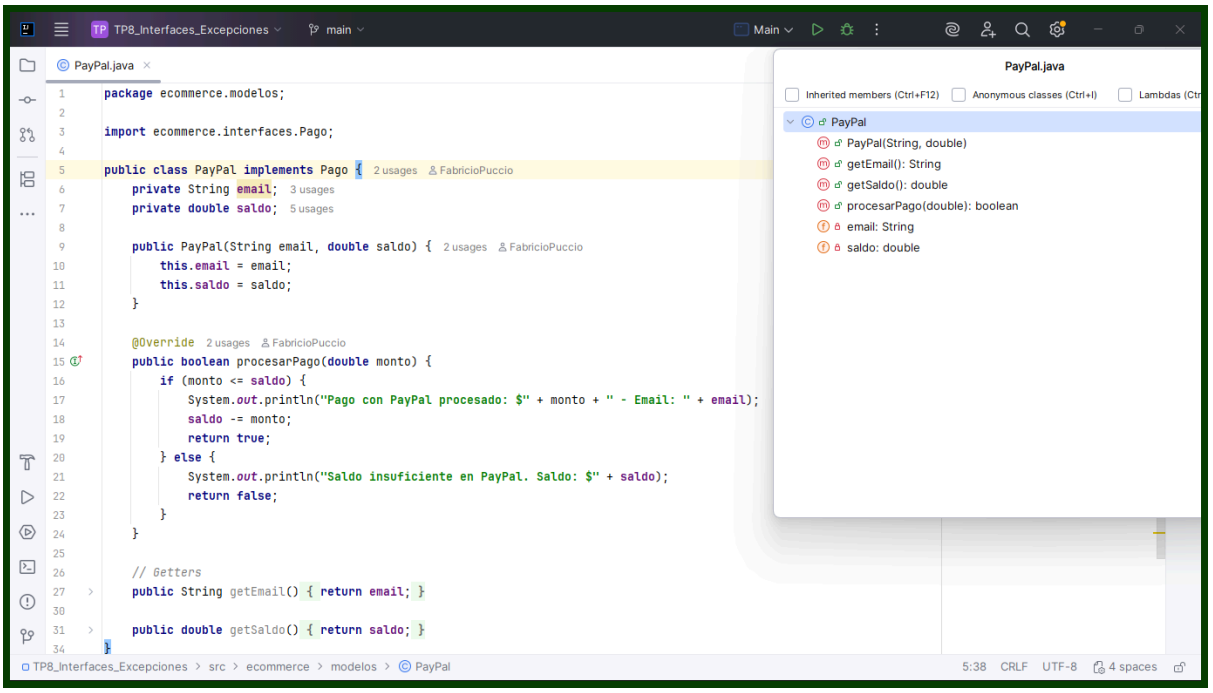
Recibe notificaciones sobre sus pedidos.



Medios de Pago

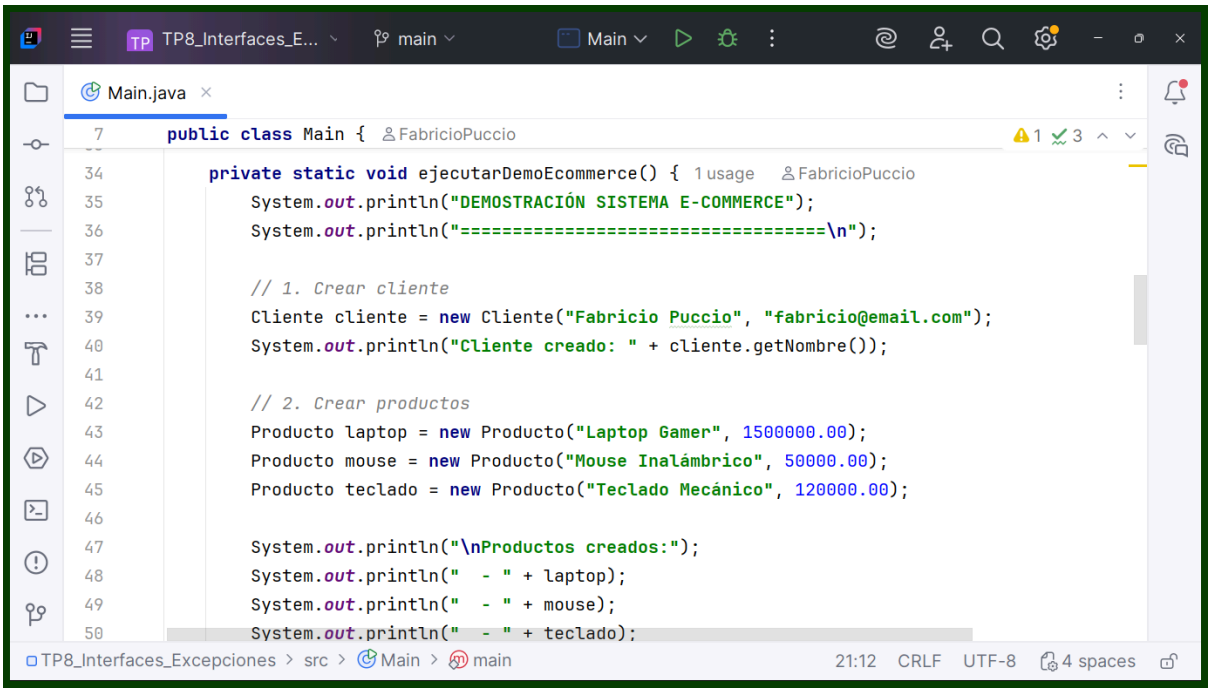
Implementaciones concretas de procesamiento de pagos.



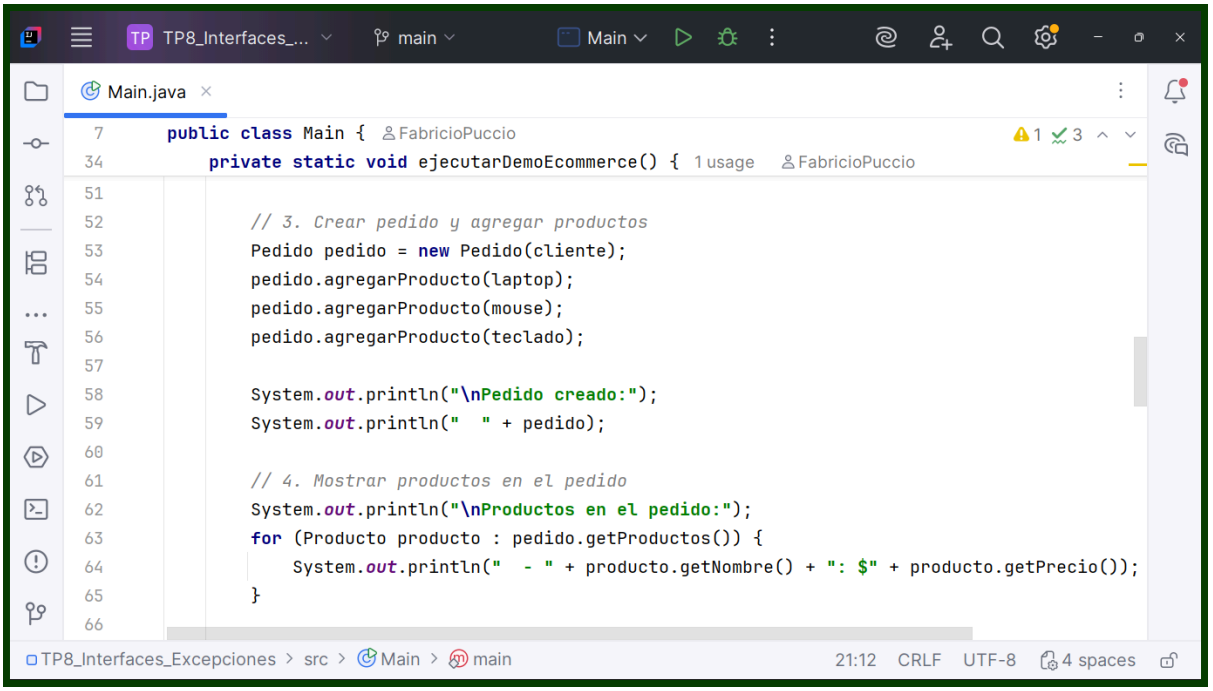


```java
package ecommerce.modelos;

import ecommerce.interfaces.Pago;

public class PayPal implements Pago {
    private String email;
    private double saldo;

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

    @Override
    public boolean procesarPago(double monto) {
        if (monto <= saldo) {
            System.out.println("Pago con PayPal procesado: $" + monto + " - Email: " + email);
            saldo -= monto;
            return true;
        } else {
            System.out.println("Saldo insuficiente en PayPal. Saldo: $" + saldo);
            return false;
        }
    }

    // Getters
    public String getEmail() { return email; }

    public double getSaldo() { return saldo; }
}
```

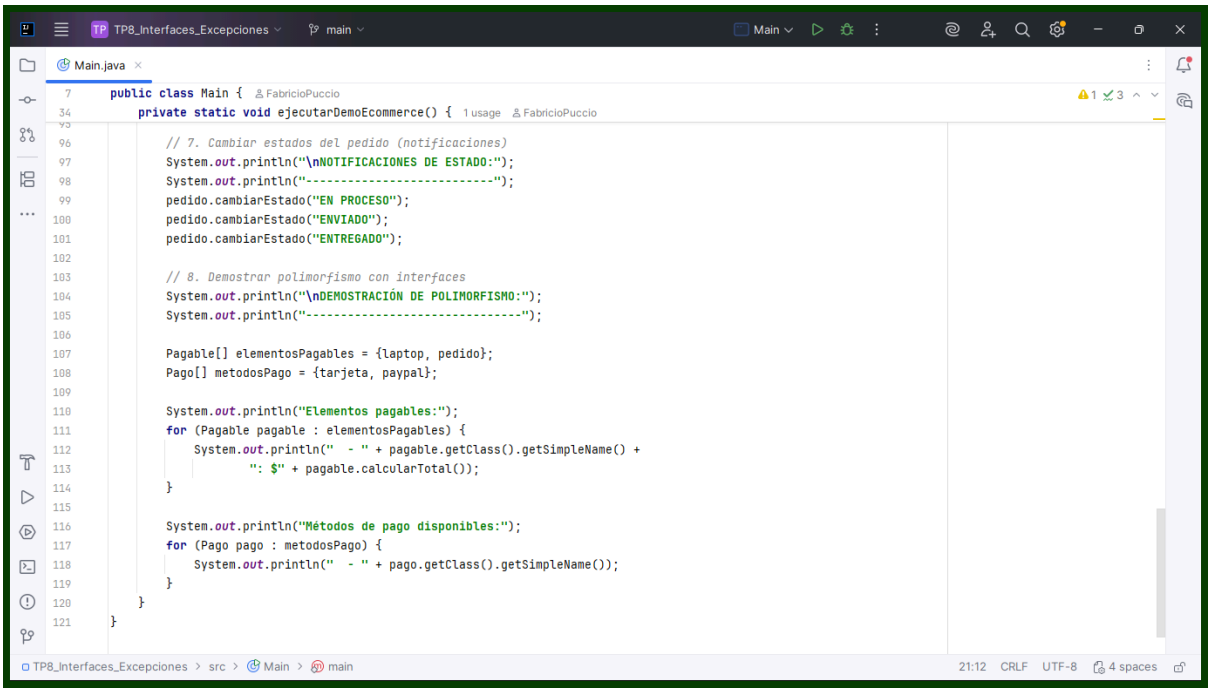
2.3 Clase Principal y Ejecución



```java
public class Main {
    private static void ejecutarDemoEcommerce() {
        System.out.println("DEMOSTRACIÓN SISTEMA E-COMMERCE");
        System.out.println("==================================\n");

        // 1. Crear cliente
        Cliente cliente = new Cliente("Fabricio Puccio", "fabricio@email.com");
        System.out.println("Cliente creado: " + cliente.getNombre());

        // 2. Crear productos
        Producto laptop = new Producto("Laptop Gamer", 1500000.00);
        Producto mouse = new Producto("Mouse Inalámbrico", 50000.00);
        Producto teclado = new Producto("Teclado Mecánico", 120000.00);

        System.out.println("\nProductos creados:");
        System.out.println("  - " + laptop);
        System.out.println("  - " + mouse);
        System.out.println("  - " + teclado);
```



```
public class Main {  FabricioPuccio
    private static void ejecutarDemoEcommerce() {  1 usage  FabricioPuccio

        // 3. Crear pedido y agregar productos
        Pedido pedido = new Pedido(cliente);
        pedido.agregarProducto(laptop);
        pedido.agregarProducto(mouse);
        pedido.agregarProducto(teclado);

        System.out.println("\nPedido creado:");
        System.out.println("  " + pedido);

        // 4. Mostrar productos en el pedido
        System.out.println("\nProductos en el pedido:");
        for (Producto producto : pedido.getProductos()) {
            System.out.println("  - " + producto.getNombre() + ": $" + producto.getPrecio());
        }
```



```
public class Main {  FabricioPuccio
    private static void ejecutarDemoEcommerce() {  1 usage  FabricioPuccio
        // 7. Cambiar estados del pedido (notificaciones)
        System.out.println("\nNOTIFICACIONES DE ESTADO:");
        System.out.println("--------------------------");
        pedido.cambiarEstado("EN PROCESO");
        pedido.cambiarEstado("ENVIADO");
        pedido.cambiarEstado("ENTREGADO");

        // 8. Demostrar polimorfismo con interfaces
        System.out.println("\nDEMOSTRACIÓN DE POLIMORFISMO:");
        System.out.println("-------------------------------");

        Pagable[] elementosPagables = {laptop, pedido};
        Pago[] metodosPago = {tarjeta, paypal};

        System.out.println("Elementos pagables:");
        for (Pagable pagable : elementosPagables) {
            System.out.println("  - " + pagable.getClass().getSimpleName() +
                    ": $" + pagable.calcularTotal());
        }

        System.out.println("Métodos de pago disponibles:");
        for (Pago pago : metodosPago) {
            System.out.println("  - " + pago.getClass().getSimpleName());
        }
    }
}
```

Resultados de Ejecución

=== TRABAJO PRÁCTICO 8 ===

Interfaces y Excepciones en Java

Alumno: Fabricio Puccio

==========

DEMOSTRACIÓN SISTEMA E-COMMERCE

==========

Cliente creado: Fabricio Puccio

Productos creados:

- Producto: Laptop Gamer - $1500000,00

- Producto: Mouse Inalámbrico - $50000,00

- Producto: Teclado Mecánico - $120000,00

Pedido creado:

Pedido [Cliente: Fabricio Puccio, Estado: PENDIENTE, Productos: 3, Total: $1670000,00]

Productos en el pedido:

- Laptop Gamer: $1500000.0

- Mouse Inalámbrico: $50000.0

- Teclado Mecánico: $120000.0

Total del pedido: $1670000,00

PROCESANDO PAGOS:

--- Pago con Tarjeta de Crédito ---

Descuento aplicado: 10.0% - Ahorro: $167000.0

Pago con Tarjeta de Crédito procesado: $1503000.0

Notificación para Fabricio Puccio (fabricio@email.com): Tu pedido cambió de estado: PENDIENTE → PAGADO

--- Pago con PayPal ---

Pago con PayPal procesado: $50000.0 - Email: fabricio.puccio@email.com

Pago PayPal exitoso

NOTIFICACIONES DE ESTADO:

Notificación para Fabricio Puccio (fabricio@email.com): Tu pedido cambió de estado: PAGADO → EN PROCESO

Notificación para Fabricio Puccio (fabricio@email.com): Tu pedido cambió de estado: EN PROCESO → ENVIADO

Notificación para Fabricio Puccio (fabricio@email.com): Tu pedido cambió de estado: ENVIADO → ENTREGADO

DEMOSTRACIÓN DE POLIMORFISMO:

Elementos pagables:

- **Producto: $1500000.0**
- **Pedido: $1670000.0**

Métodos de pago disponibles:

- **TarjetaCredito**
- **PayPal**

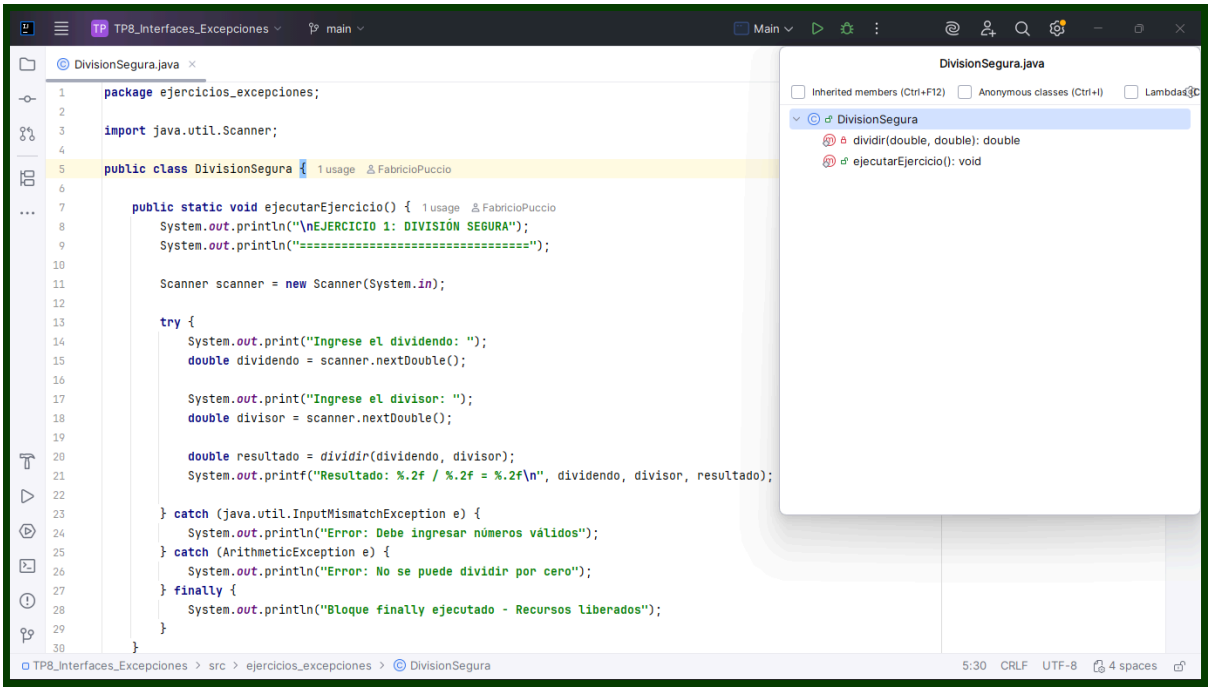
Análisis: El sistema demuestra polimorfismo mediante interfaces, notificaciones en tiempo real y procesamiento flexible de pagos.

3. PARTE 2: EJERCICIOS DE EXCEPCIONES

3.1 División Segura

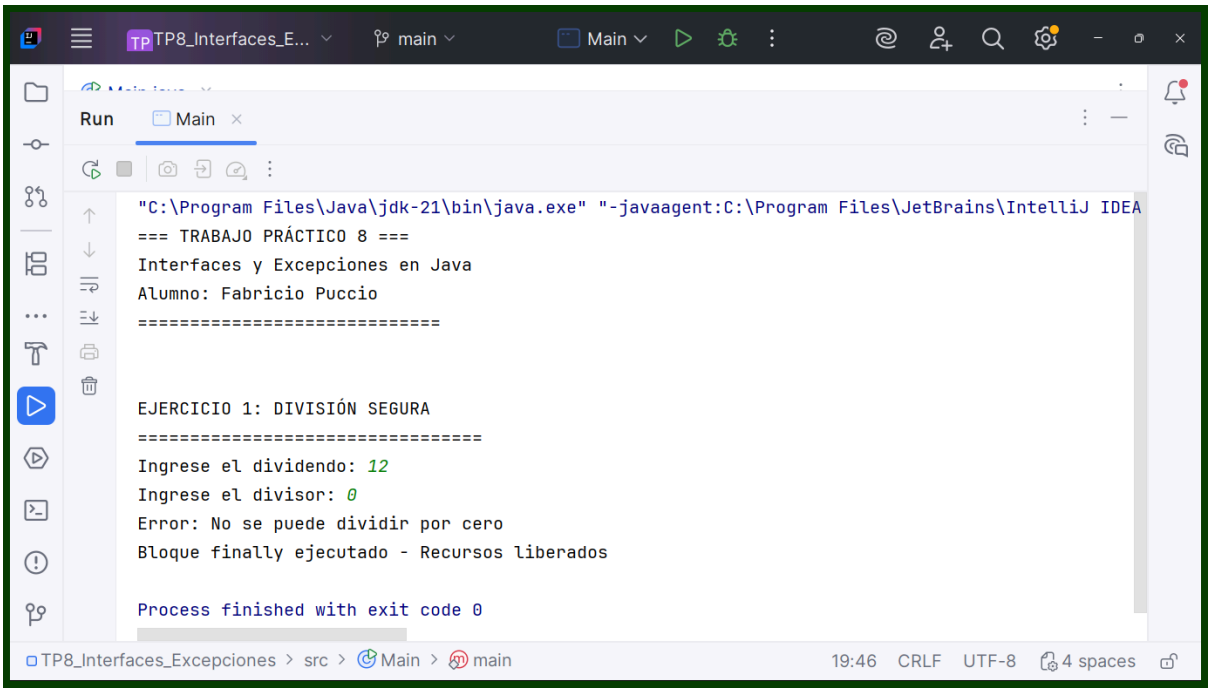
Manejo de ArithmeticException para división por cero.

Código:



```java
package ejercicios_excepciones;

import java.util.Scanner;

public class DivisionSegura {

    public static void ejecutarEjercicio() {
        System.out.println("\nEJERCICIO 1: DIVISIÓN SEGURA");
        System.out.println("==================================");

        Scanner scanner = new Scanner(System.in);

        try {
            System.out.print("Ingrese el dividendo: ");
            double dividendo = scanner.nextDouble();

            System.out.print("Ingrese el divisor: ");
            double divisor = scanner.nextDouble();

            double resultado = dividir(dividendo, divisor);
            System.out.printf("Resultado: %.2f / %.2f = %.2f\n", dividendo, divisor, resultado);

        } catch (java.util.InputMismatchException e) {
            System.out.println("Error: Debe ingresar números válidos");
        } catch (ArithmeticException e) {
            System.out.println("Error: No se puede dividir por cero");
        } finally {
            System.out.println("Bloque finally ejecutado - Recursos liberados");
        }
    }
```

Ejecución:

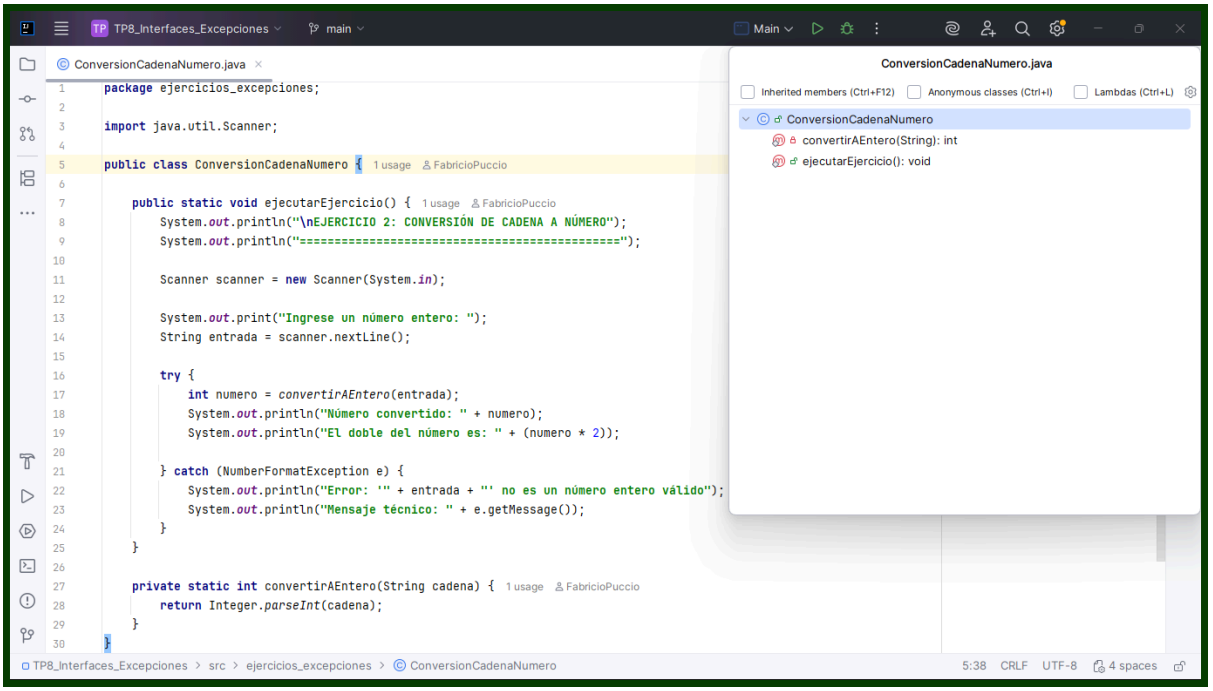


```
"C:\Program Files\Java\jdk-21\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA
=== TRABAJO PRÁCTICO 8 ===
Interfaces y Excepciones en Java
Alumno: Fabricio Puccio
==============================


EJERCICIO 1: DIVISIÓN SEGURA
==================================
Ingrese el dividendo: 12
Ingrese el divisor: 0
Error: No se puede dividir por cero
Bloque finally ejecutado - Recursos liberados

Process finished with exit code 0
```

3.2 Conversión de Cadena a Número

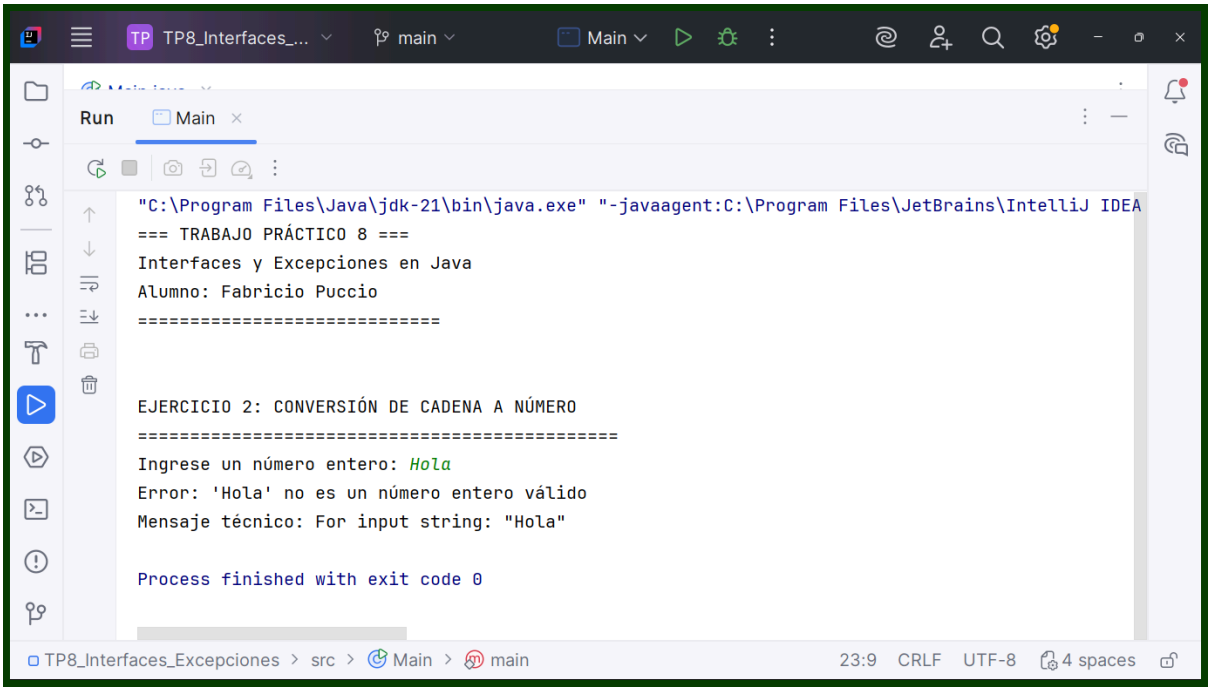
Manejo de NumberFormatException para conversiones inválidas.

Código:



```java
package ejercicios_excepciones;

import java.util.Scanner;

public class ConversionCadenaNumero {

    public static void ejecutarEjercicio() {
        System.out.println("\nEJERCICIO 2: CONVERSIÓN DE CADENA A NÚMERO");
        System.out.println("==========");

        Scanner scanner = new Scanner(System.in);

        System.out.print("Ingrese un número entero: ");
        String entrada = scanner.nextLine();

        try {
            int numero = convertirAEntero(entrada);
            System.out.println("Número convertido: " + numero);
            System.out.println("El doble del número es: " + (numero * 2));

        } catch (NumberFormatException e) {
            System.out.println("Error: '" + entrada + "' no es un número entero válido");
            System.out.println("Mensaje técnico: " + e.getMessage());
        }
    }

    private static int convertirAEntero(String cadena) {
        return Integer.parseInt(cadena);
    }
}
```

Ejecución:

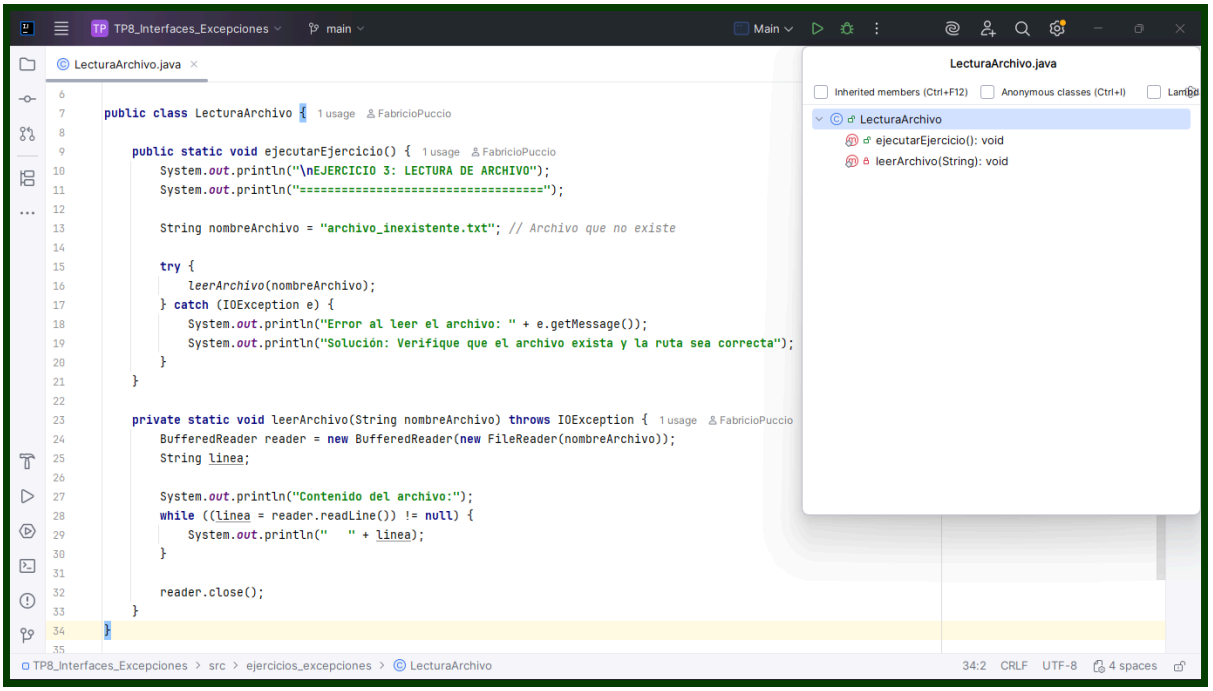


```
"C:\Program Files\Java\jdk-21\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA
=== TRABAJO PRÁCTICO 8 ===
Interfaces y Excepciones en Java
Alumno: Fabricio Puccio
==========


EJERCICIO 2: CONVERSIÓN DE CADENA A NÚMERO
==========
Ingrese un número entero: Hola
Error: 'Hola' no es un número entero válido
Mensaje técnico: For input string: "Hola"

Process finished with exit code 0
```

3.3 Lectura de Archivo

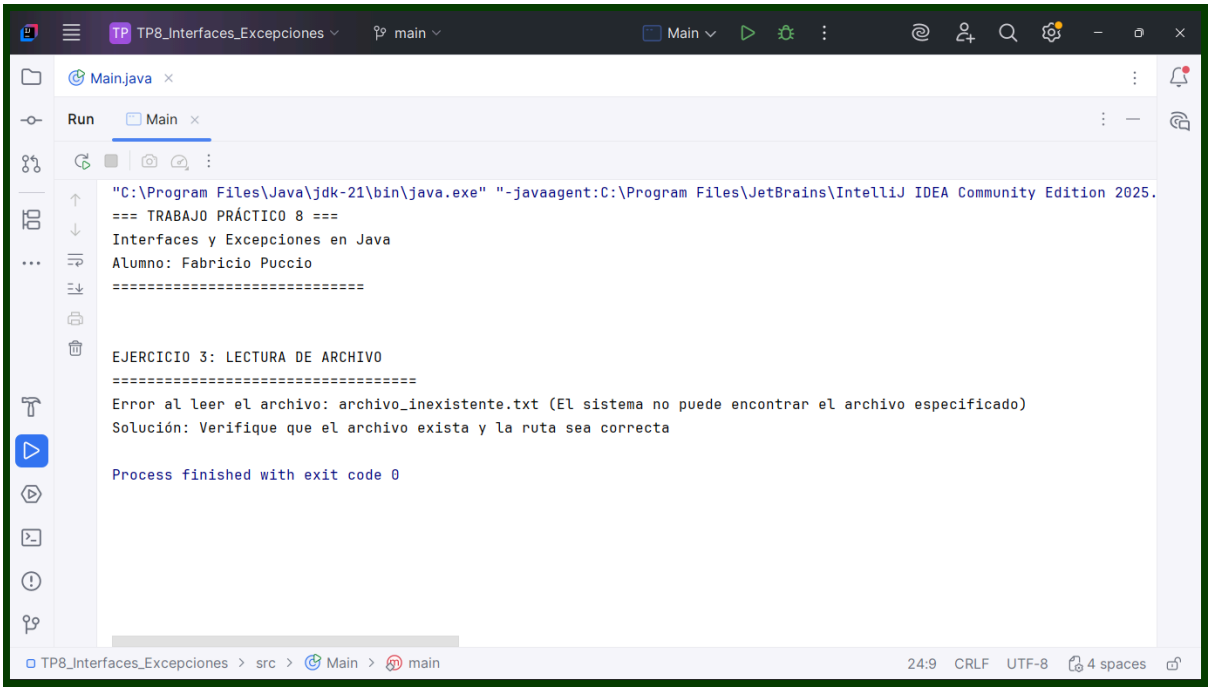
Manejo de FileNotFoundException para archivos inexistentes.

Código:



```java
public class LecturaArchivo {
    public static void ejecutarEjercicio() {
        System.out.println("\nEJERCICIO 3: LECTURA DE ARCHIVO");
        System.out.println("==================================");

        String nombreArchivo = "archivo_inexistente.txt"; // Archivo que no existe

        try {
            leerArchivo(nombreArchivo);
        } catch (IOException e) {
            System.out.println("Error al leer el archivo: " + e.getMessage());
            System.out.println("Solución: Verifique que el archivo exista y la ruta sea correcta");
        }
    }

    private static void leerArchivo(String nombreArchivo) throws IOException {
        BufferedReader reader = new BufferedReader(new FileReader(nombreArchivo));
        String linea;

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

        reader.close();
    }
}
```

Ejecución:

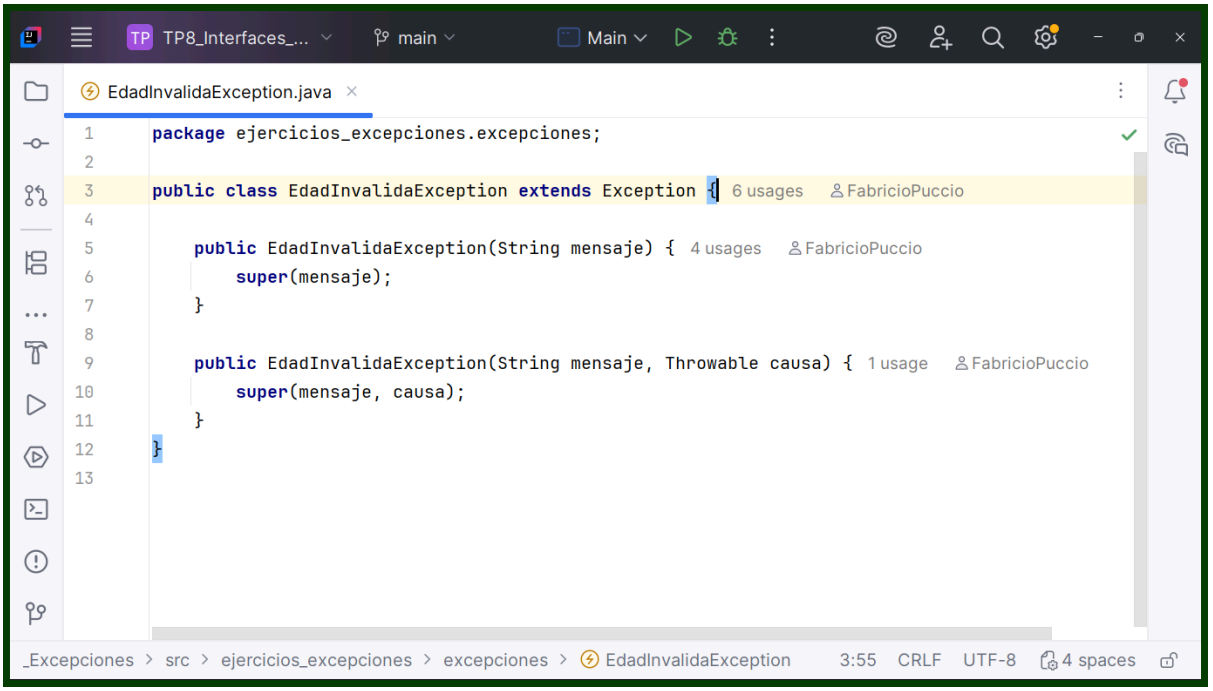


```
"C:\Program Files\Java\jdk-21\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2025.
=== TRABAJO PRÁCTICO 8 ===
Interfaces y Excepciones en Java
Alumno: Fabricio Puccio
==============================


EJERCICIO 3: LECTURA DE ARCHIVO
==================================
Error al leer el archivo: archivo_inexistente.txt (El sistema no puede encontrar el archivo especificado)
Solución: Verifique que el archivo exista y la ruta sea correcta

Process finished with exit code 0
```

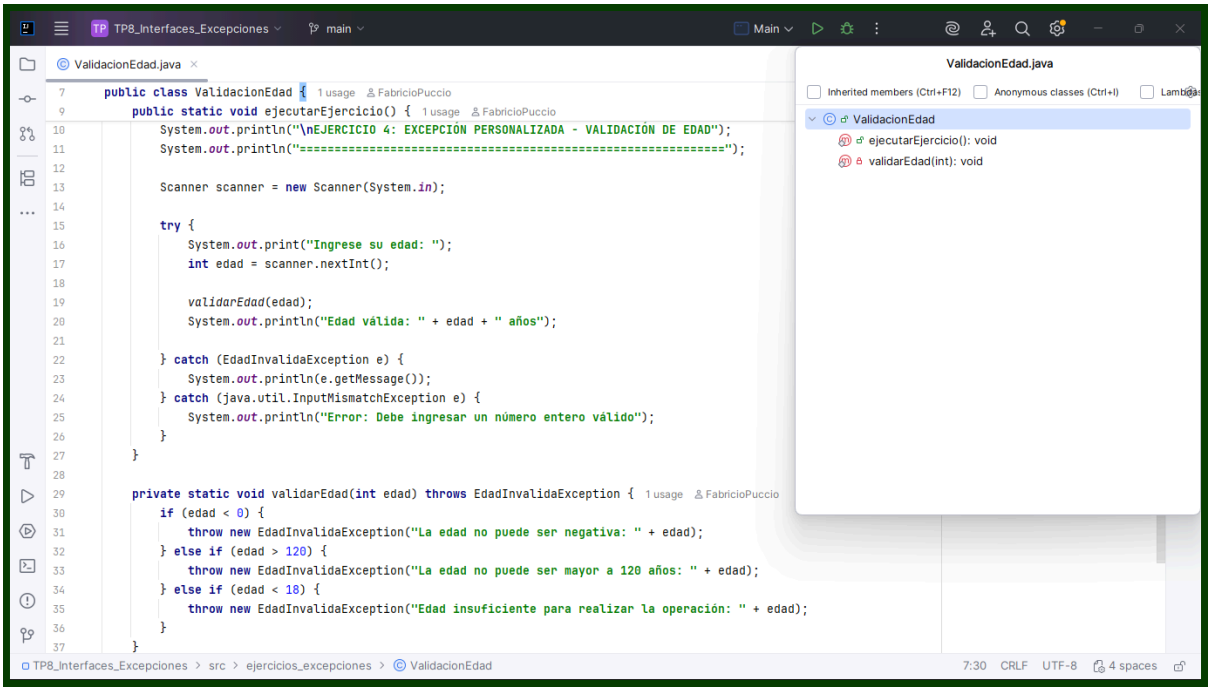
3.4 Excepción Personalizada

Creación y uso de EdadInvalidaException para validaciones.

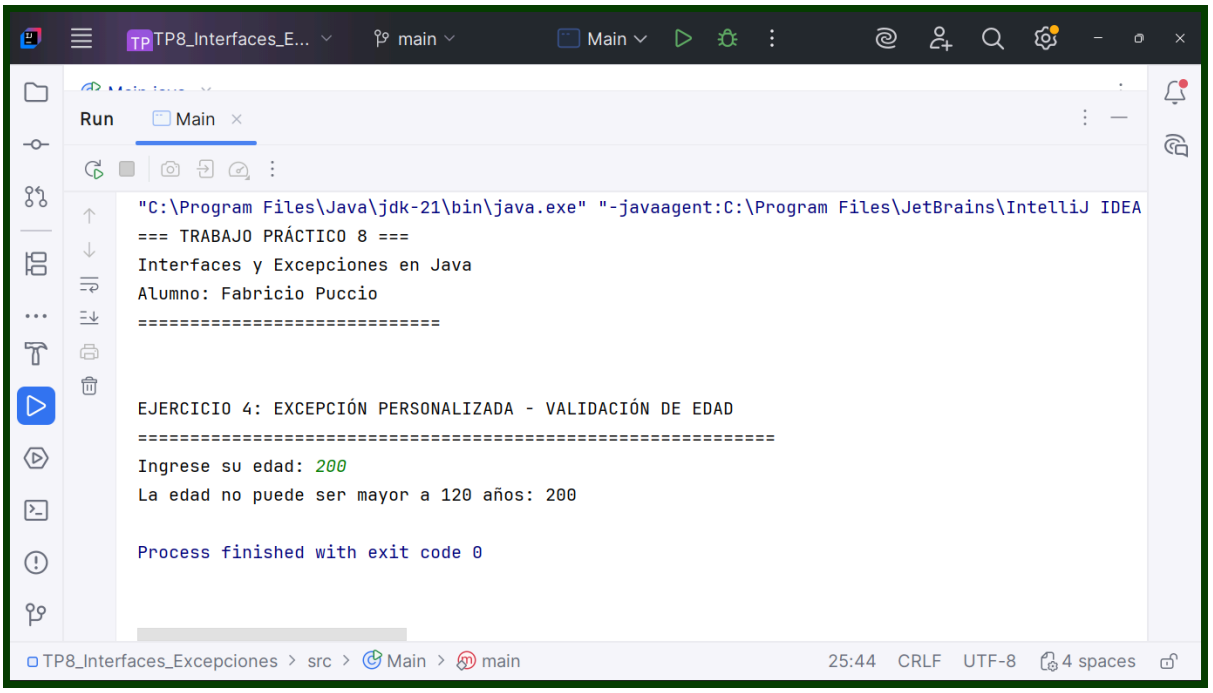
Excepción Personalizada:



Código de Validación:



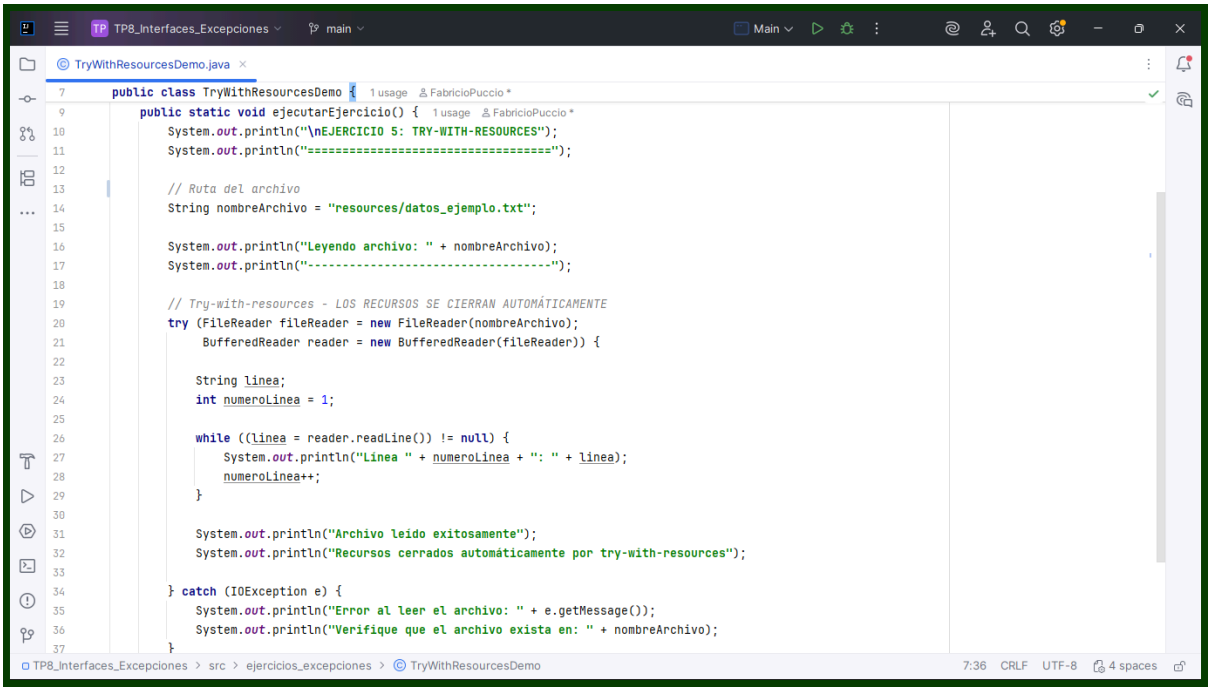
Ejecución:



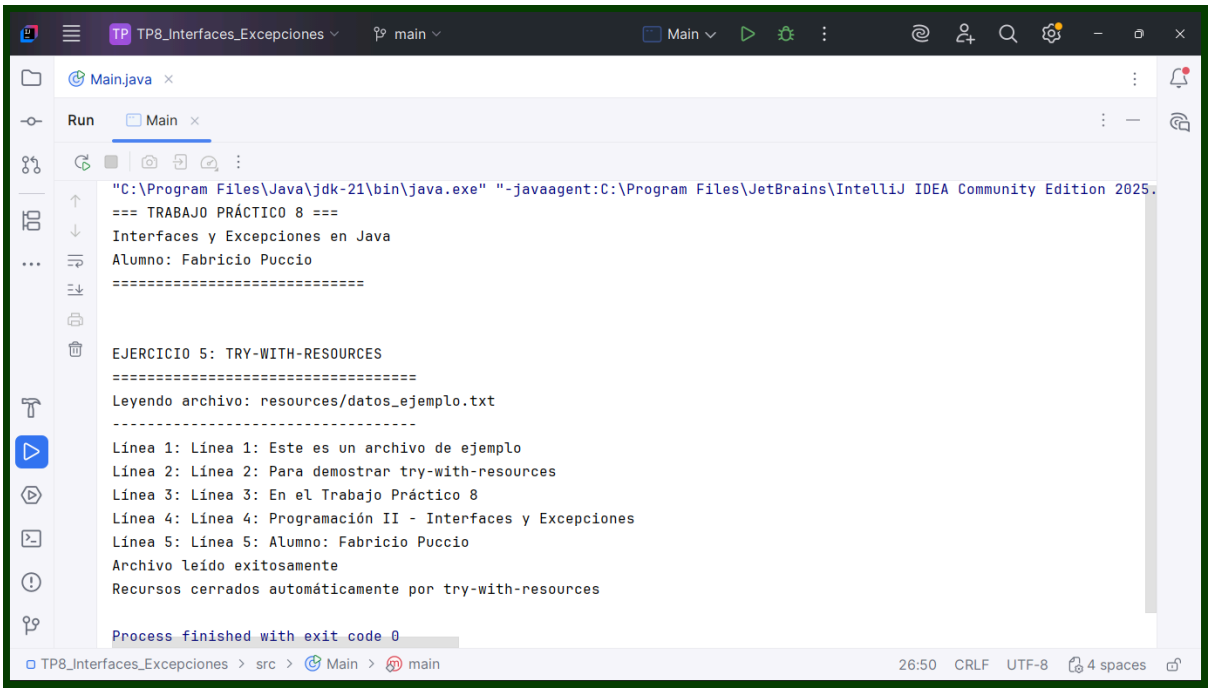
3.5 Try-with-Resources

Uso de try-with-resources para gestión automática de recursos.

Código:



Ejecución:



---

4. CONCLUSIÓN

4.1 Aprendizajes Adquiridos

- ❖ Interfaces como contratos para diseño modular
- ❖ Polimorfismo sin herencia múltiple mediante interfaces
- ❖ Manejo robusto de excepciones para software confiable
- ❖ Try-with-resources para gestión automática de recursos
- ❖ Excepciones personalizadas para reglas de negocio específicas

4.2 Dificultades Superadas

- Comprensión del problema del diamante y su solución con interfaces
- Diferenciación entre excepciones checked y unchecked
- Implementación de sistemas de notificación desacoplados
- Organización de paquetes y estructura modular

4.3 Aplicación Práctica

Los conocimientos adquiridos permiten desarrollar software más:

- Modular y mantenible
- Robusto ante errores
- Escalable para futuras extensiones
- Profesional en estándares de calidad

5. REPOSITORIO GITHUB

Enlace al código fuente completo:
https://github.com/FabricioPuccio/TP8-Interfaces-Excepciones

- El repositorio contiene:
- Código fuente completo y organizado
- README con documentación del proyecto
- Estructura de paquetes profesional
- Historial de commits del desarrollo

Fabricio Puccio - Programación II - 21-11-2025