

# PROGRAMACIÓN II

## Trabajo Práctico 8: Interfaces y Excepciones en Java

**Alumno :**

- González Federico

**Profesor :**

- Enferrel Ariel

**Tutor :**

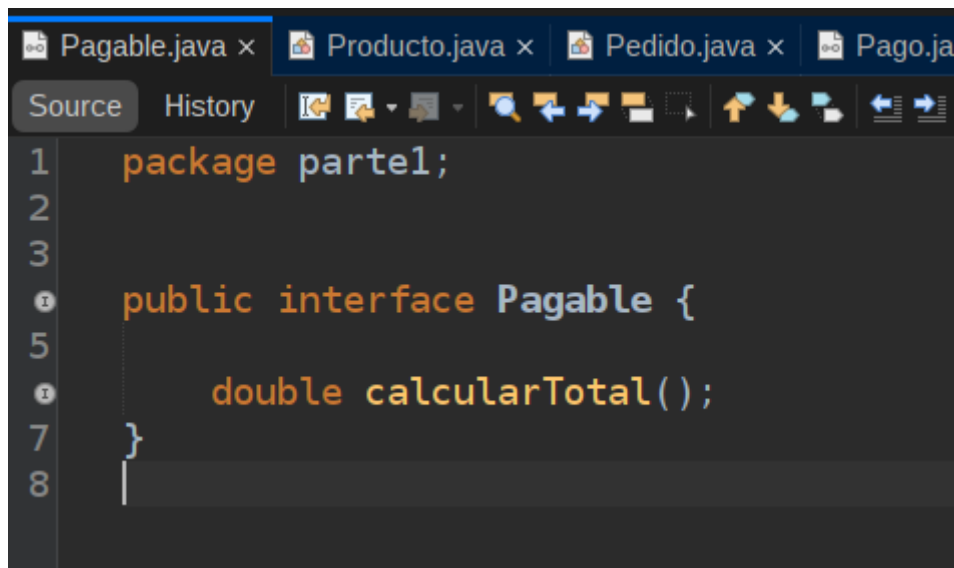
- Ferro Tomás

**GitHub:**

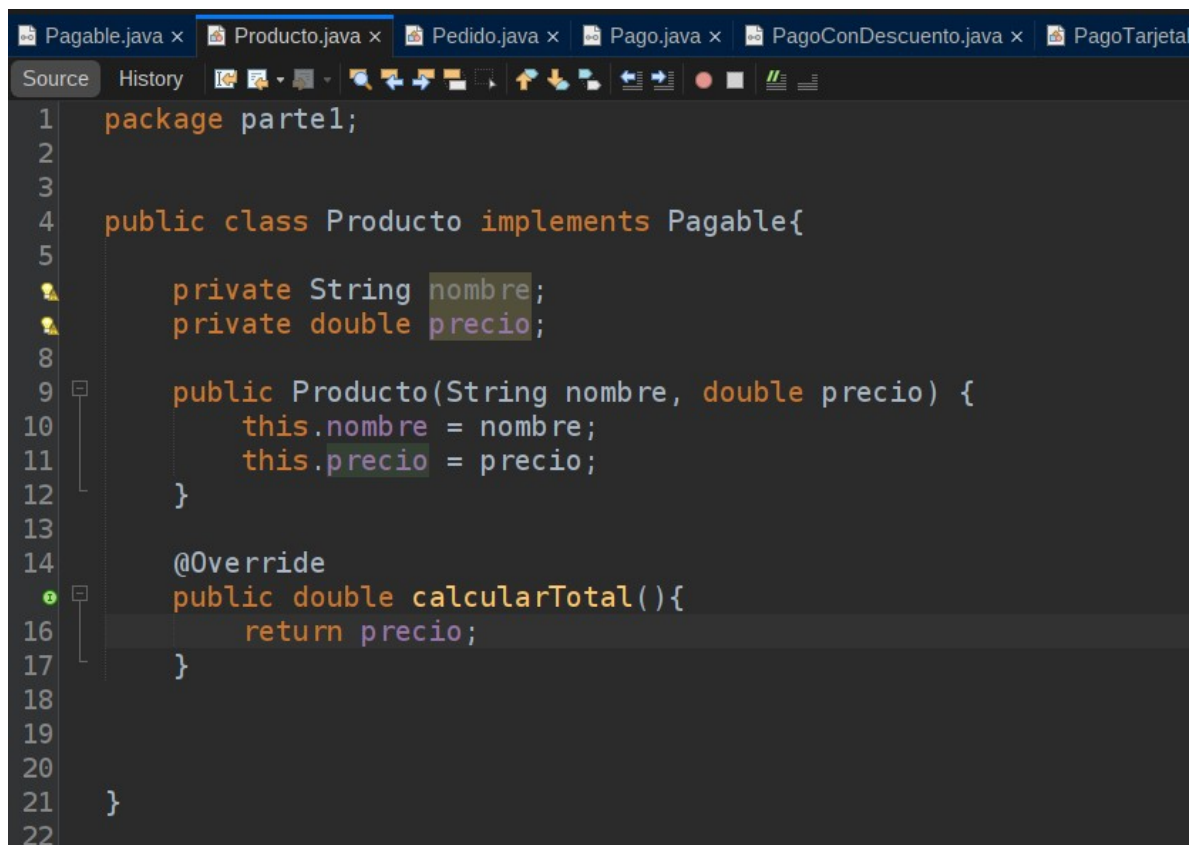
- <https://github.com/fedeglz/UTN-TUPaD-P2.git>

## 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.



```
1 package parte1;
2
3
4
5 public interface Pagable {
6
7     double calcularTotal();
8 }
```



```
1 package parte1;
2
3
4 public class Producto implements Pagable{
5
6     private String nombre;
7     private double precio;
8
9     public Producto(String nombre, double precio) {
10         this.nombre = nombre;
11         this.precio = precio;
12     }
13
14     @Override
15     public double calcularTotal(){
16         return precio;
17     }
18
19
20
21 }
22
```

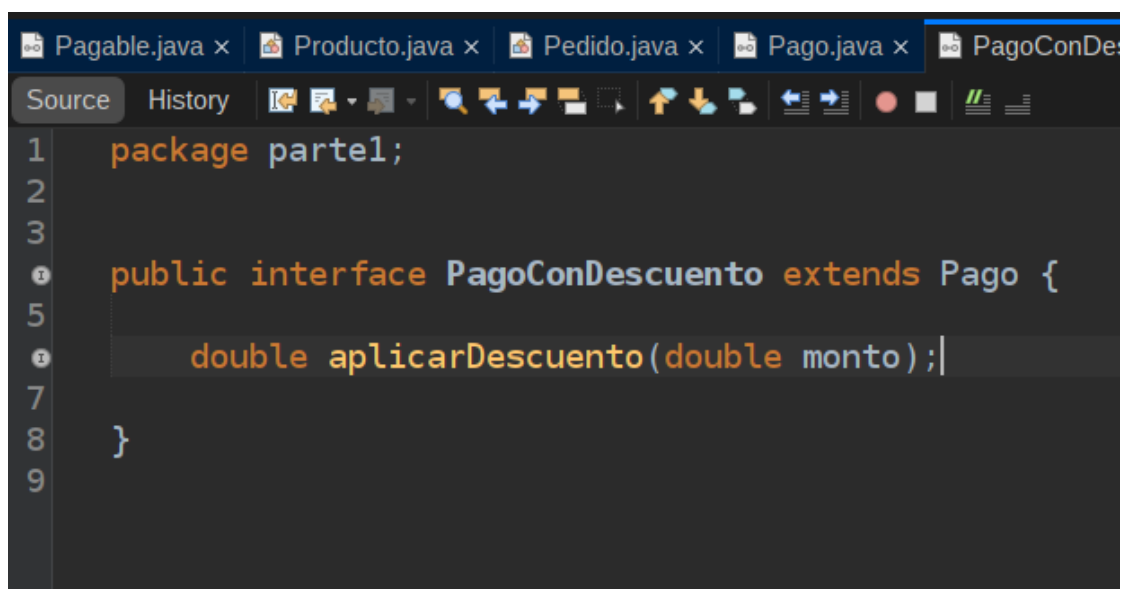
# TECNICATURA UNIVERSITARIA EN PROGRAMACIÓN A DISTANCIA

```

Pagable.java x Producto.java x Pedido.java x Pago.java x PagoConDescuento.java x PagoTarjetaDe
Source History
1 package partel;
2
3 import java.util.ArrayList;
4 import java.util.List;
5
6 public class Pedido implements Pagable {
7
8     private List<Producto> productos;
9     private String estado;
10    private Cliente cliente;
11
12    public Pedido(Cliente cliente) {
13        this.productos = new ArrayList<>();
14        this.estado = "Creado";
15        this.cliente = cliente;
16    }
17
18
19    public void agregarProducto(Producto p) {
20        productos.add(p);
21    }
22
23    @Override
24    public double calcularTotal() {
25        double total = 0;
26        for (Producto p : productos) {
27            total += p.calcularTotal();
28        }
29        return total;
30    }
31
32    public void setEstado(String nuevoEstado) {
33        this.estado = nuevoEstado;
34        notificarCambioEstado();
35    }
36
37    public String getEstado() {
38        return estado;
39    }
40
41    private void notificarCambioEstado() {
42        if (cliente != null) {
43            cliente.notificar("El estado de su pedido cambi  a: " + estado);
44        }
45    }
46
47    public List<Producto> getProductos() {
48        return productos;
49    }
50
51    @Override
52    public String toString() {
53        return "Pedido{"
54            + "cliente=" + cliente
55            + ", estado='" + estado + '\''
56            + ", total=" + calcularTotal()
57            + '}';
58    }
59
60 }
61
  
```



```
1 package parte1;
2
3
4
5 public interface Pago {
6
7     void procesarPago(double monto);
8
9 }
10
11
```



```
1 package parte1;
2
3
4
5 public interface PagoConDescuento extends Pago {
6
7     double aplicarDescuento(double monto);
8
9 }
```

TECNICATURA UNIVERSITARIA EN  
PROGRAMACIÓN A DISTANCIA

```

Pagable.java x Producto.java x Pedido.java x Pago.java x PagoConDescuento.java x PagoTarjetaDeCredito.java
Source History
1 package parte1;
2
3
4 public class PagoTarjetaDeCredito implements PagoConDescuento{
5
6     @Override
7     public double aplicarDescuento(double monto){
8         return monto * 0.9;
9     }
10
11     @Override
12     public void procesarPago(double monto){
13         System.out.println("Pago con tarjeta por $" + monto);
14     }
15
16 }

```

```

Pagable.java x Producto.java x Pedido.java x Pago.java x PagoConDescuento.java x PagoTarjetaDeCredito.java
Source History
1 package parte1;
2
3
4 public interface Notificable {
5
6     void notificar(String mensaje);
7
8
9 }
10

```

# TECNICATURA UNIVERSITARIA EN PROGRAMACIÓN A DISTANCIA

```

Pagable.java x Producto.java x Pedido.java x Pago.java x PagoConDescuento.java x PagoTarjetaDeCredito.java x
Source History
1 package parte1;
2
3
4 public class Cliente implements Notificable{
5
6     private String nombre;
7
8     public Cliente(String nombre) {
9         this.nombre = nombre;
10    }
11
12    @Override
13    public void notificar(String mensaje){
14        System.out.println("Notificacion a " + nombre + ": " + mensaje );
15    }
16
17 }
18

```

```

Proj... x Files Services ...
Source History
3
4 public class main {
5     public static void main(String[] args) {
6
7         Cliente cliente = new Cliente("Ana López");
8
9
10        Producto p1 = new Producto("Mouse Gamer", 15000);
11        Producto p2 = new Producto("Teclado Mecánico", 32000);
12        Producto p3 = new Producto("Auriculares Bluetooth", 28000);
13
14
15        Pedido pedido = new Pedido(cliente);
16        pedido.agregarProducto(p1);
17        pedido.agregarProducto(p2);
18        pedido.agregarProducto(p3);
19
20
21        double total = pedido.calcularTotal();
22        System.out.println("\n--- PEDIDO CREADO ---");
23        System.out.println("Total del pedido sin descuento: $" + total);
24
25
26
27        PagoConDescuento pago = new PagoTarjetaDeCredito();
28        double totalConDescuento = pago.aplicarDescuento(total);
29        System.out.println("Total con descuento (10%): $" + totalConDescuento);
30        pago.procesarPago(totalConDescuento);
31
32
33        pedido.setEstado("Pagado");
34        pedido.setEstado("Enviado");
35        pedido.setEstado("Entregado");
36
37
38        System.out.println("\n--- RESUMEN FINAL ---");
39        System.out.println(pedido);
40    }
41

```

Output - trabajoPractico08 (run)

```

run:
--- PEDIDO CREADO ---
Total del pedido sin descuento: $75000.0
Total con descuento (10%): $67500.0
Pago con tarjeta por $67500.0
Notificacion a Ana López: El estado de su pedido cambió a: Pagado
Notificacion a Ana López: El estado de su pedido cambió a: Enviado
Notificacion a Ana López: El estado de su pedido cambió a: Entregado
--- RESUMEN FINAL ---
Pedido{cliente=parte1.Cliente@7adf9f5f, estado='Entregado', total=75000.0}
BUILD SUCCESSFUL (total time: 0 seconds)

```

## Parte 2: Ejercicios sobre Excepciones

### 1. División segura

- Solicitar dos números y dividirlos. Manejar `ArithmeticException` si el divisor es cero.

```
DivisionSegura.java x
Source History
1 package parte2;
2
3 import java.util.Scanner;
4
5
6 public class DivisionSegura {
7     public static void main(String[] args) {
8
9         Scanner scanner = new Scanner(System.in);
10
11         try {
12             System.out.print("Ingrese el 1er número: ");
13             int a = scanner.nextInt();
14
15             System.out.print("Ingrese el 2do número: ");
16             int b = scanner.nextInt();
17
18             int resultado = a / b;
19             System.out.println("Resultado: " + resultado);
20
21         } catch (ArithmeticException e) {
22             System.out.println("Error: no se puede dividir por cero.");
23         } finally {
24             System.out.println("Operación finalizada.");
25         }
26
27         scanner.close();
28
29
30 }
```

Output - trabajoPractico08 (run)

```
run:
Ingrese el 1er número: 12
Ingrese el 2do número: 3
Resultado: 4
Operación finalizada.
BUILD SUCCESSFUL (total time: 9 seconds)
```

Output - trabajoPractico08 (run)

```
run:
Ingrese el 1er número: 12
Ingrese el 2do número: 0
Error: no se puede dividir por cero.
Operación finalizada.
BUILD SUCCESSFUL (total time: 5 seconds)
```



## TECNICATURA UNIVERSITARIA EN PROGRAMACIÓN A DISTANCIA

### 2. Conversión de cadena a número

- Leer texto del usuario e intentar convertirlo a int. Manejar `NumberFormatException` si no es válido.

```
ConversionDeCadenaANumero.java x
Source History
1 package parte2;
2
3 import java.util.Scanner;
4
5 public class ConversionDeCadenaANumero {
6
7     public static void main(String[] args) {
8         Scanner scanner = new Scanner(System.in);
9
10        try {
11            System.out.print("Ingrese un número entero: ");
12            String texto = scanner.nextLine();
13            int numero = Integer.parseInt(texto);
14            System.out.println("Número ingresado: " + numero);
15        } catch (NumberFormatException e) {
16            System.out.println("Error: el texto ingresado no es un número válido.");
17        }
18
19        scanner.close();
20    }
21 }
22
23 }
```

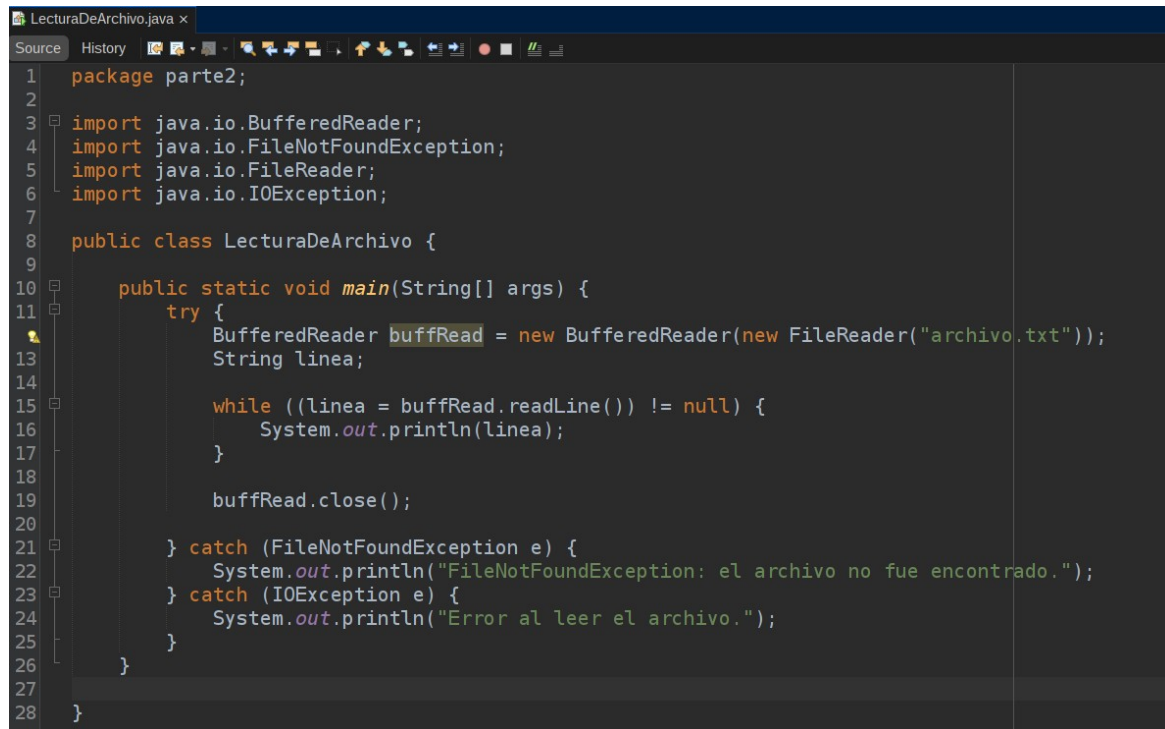
```
Output - trabajoPractico08 (run)
run:
Ingrese un número entero: 10
Número ingresado: 10
BUILD SUCCESSFUL (total time: 4 seconds)
```

```
Output - trabajoPractico08 (run)
run:
Ingrese un número entero: 12,4
Error: el texto ingresado no es un número válido.
BUILD SUCCESSFUL (total time: 4 seconds)
```

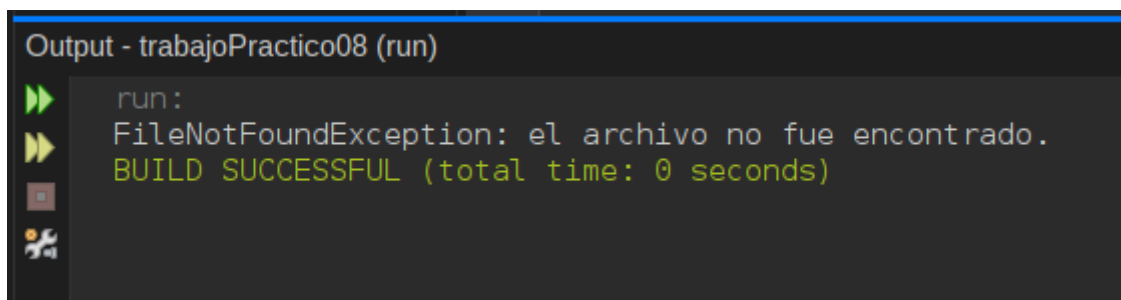
## TECNICATURA UNIVERSITARIA EN PROGRAMACIÓN A DISTANCIA

### 3. Lectura de archivo

- Leer un archivo de texto y mostrarlo. Manejar `FileNotFoundException` si el archivo no existe.



```
1 package parte2;
2
3 import java.io.BufferedReader;
4 import java.io.FileNotFoundException;
5 import java.io.FileReader;
6 import java.io.IOException;
7
8 public class LecturaDeArchivo {
9
10     public static void main(String[] args) {
11         try {
12             BufferedReader buffRead = new BufferedReader(new FileReader("archivo.txt"));
13             String linea;
14
15             while ((linea = buffRead.readLine()) != null) {
16                 System.out.println(linea);
17             }
18
19             buffRead.close();
20
21         } catch (FileNotFoundException e) {
22             System.out.println("FileNotFoundException: el archivo no fue encontrado.");
23         } catch (IOException e) {
24             System.out.println("Error al leer el archivo.");
25         }
26     }
27 }
28 }
```



```
Output - trabajoPractico08 (run)
run:
FileNotFoundException: el archivo no fue encontrado.
BUILD SUCCESSFUL (total time: 0 seconds)
```

## TECNICATURA UNIVERSITARIA EN PROGRAMACIÓN A DISTANCIA

### 4. Excepción personalizada

- Crear EdadInvalidaException. Lanzarla si la edad es menor a 0 o mayor a 120. Capturarla y mostrar mensaje.

```
ExcepcionPersonalizada.java x EdadInvalidaException.java x
Source History
1 package parte2;
2
3 import java.util.Scanner;
4
5 public class ExcepcionPersonalizada {
6
7     public static void main(String[] args) {
8         Scanner scanner = new Scanner(System.in);
9
10        try {
11            System.out.print("Ingrese su edad: ");
12            int edad = scanner.nextInt();
13            validarEdad(edad);
14            System.out.println("Edad válida: " + edad);
15
16        } catch (EdadInvalidaException e) {
17            System.out.println("Error: " + e.getMessage());
18        }
19
20        scanner.close();
21    }
22
23    public static void validarEdad(int edad) throws EdadInvalidaException {
24        if (edad < 0 || edad > 120) {
25            throw new EdadInvalidaException("La edad debe estar entre 0 y 120 años.");
26        }
27    }
28
29 }
```

```
ExcepcionPersonalizada.java x EdadInvalidaException.java x
Source History
1 package parte2;
2
3
4 public class EdadInvalidaException extends Exception{
5     public EdadInvalidaException(String mensaje) {
6         super(mensaje);
7     }
8 }
9
```

```
Output - trabajoPractico08 (run)
run:
Ingrese su edad: 25
Edad válida: 25
BUILD SUCCESSFUL (total time: 4 seconds)
```

```
Output - trabajoPractico08 (run)
run:
Ingrese su edad: 121
Error: La edad debe estar entre 0 y 120 años.
BUILD SUCCESSFUL (total time: 4 seconds)
```

## TECNICATURA UNIVERSITARIA EN PROGRAMACIÓN A DISTANCIA

### 5. Uso de try-with-resources

- Leer un archivo con `BufferedReader` usando `try-with-resources`.

Manejar `IOException` correctamente.

```
UsoDeTryWithResources.java x
Source History
1 package parte2;
2
3 import java.io.BufferedReader;
4 import java.io.FileReader;
5 import java.io.IOException;
6
7
8 public class UsoDeTryWithResources {
9
10     public static void main(String[] args) {
11         String ruta = "src/parte2/datos.txt";
12
13         try (BufferedReader br = new BufferedReader(new FileReader(ruta))) {
14
15             String linea;
16             System.out.println("=== Contenido del archivo ===");
17             while ((linea = br.readLine()) != null) {
18                 System.out.println(linea);
19             }
20
21         } catch (IOException e) {
22             System.out.println("Error al acceder o leer el archivo: " + e.getMessage());
23         }
24
25         System.out.println("Lectura finalizada correctamente.");
26     }
27
28 }
29
```

Output

Debugger Console x trabajoPractico08 (run) x

```
run:
=== Contenido del archivo ===
Hola, este es un archivo de prueba.

Fin del texto.

Lectura finalizada correctamente.
BUILD SUCCESSFUL (total time: 0 seconds)
|
```