TECNICATURA UNIVERSITARIA
EN PROGRAMACIÓN
A DISTANCIA

Trabajo Práctico 8: Interfaces y Excepciones en Java

Alumno : Francisco López (39.327.419)
GitHub: https://github.com/franciscolodev/UTN-TUPaD-P2

Primera parte

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.

Pagable.java

```java
package partel;

public interface Pagable {

    double calcularTotal();
}
```

Pagable.java | Producto.java

```java
package partel;

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;
    }
```

Pagable.java × Producto.java × Pedido.java ×

Source History

```java
package partel;

import java.util.ArrayList;
import java.util.List;

public class Pedido implements Pagable {

    private List<Producto> productos;
    private String estado;
    private Cliente cliente;

    public Pedido(Cliente cliente) {
        this.productos = new ArrayList<>();
        this.estado = "Creado";
        this.cliente = cliente;
    }


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

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

    public void setEstado(String nuevoEstado) {
        this.estado = nuevoEstado;
        notificarCambioEstado();
    }

    public String getEstado() {
        return estado;
    }

    private void notificarCambioEstado() {
        if (cliente != null) {
            cliente.notificar("El estado de su pedido cambió a: " + estado);
        }
    }

    public List<Producto> getProductos() {
        return productos;
    }

    @Override
    public String toString() {
        return "Pedido{"
                + "cliente=" + cliente
                + ", estado='" + estado + '\''
                + ", total=" + calcularTotal()
                + '}';
    }

}
```

Pagable.java × Producto.java × Pedido.java × Pago.java ×

Source History

```java
package partel;

public interface Pago {

    void procesarPago(double monto);

}
```

Pagable.java × Producto.java × Pedido.java × Pago.java × PagoTarjetaDeCredito.java ×

Source History

```java
package partel;

public class PagoTarjetaDeCredito implements PagoConDescuento{

    @Override
    public double aplicarDescuento(double monto){
        return monto * 0.9;
    }

    @Override
    public void procesarPago(double monto){
        System.out.println("Pago con tarjeta por $" + monto);
    }
}
```

Pagable.java × Producto.java × Pedido.java × Pago.java × PagoTarjetaDeCredito.java × PagoConDescuento.java ×

Source History

```java
package partel;


public interface PagoConDescuento extends Pago {

    double aplicarDescuento(double monto);

}
```

Pagable.java × Producto.java × Pedido.java × Pago.java × PagoTarjetaDeCredito.java × PagoConDescuento.java × Notificable.java ×

Source History

```java
package partel;

public interface Notificable {

    void notificar(String mensaje);

}
```

Pagable.java × Producto.java × Pedido.java × Pago.java × PagoTarjetaDeCredito.java × PagoConDescuento.java × Notificable.java × Cliente.java ×

Source History

```java
package parte1;

public class Cliente implements Notificable{

    private String nombre;

    public Cliente(String nombre) {
        this.nombre = nombre;
    }

    @Override
    public void notificar(String mensaje){
        System.out.println("Notificacion a " + nombre + ": " + mensaje );
    }

}
```

Pagable.java × Producto.java × Pedido.java × Pago.java × PagoTarjetaDeCredito.java × PagoConDescuento.java × Notificable.java × Cliente.java × main.java ×

Source History

```java
package parte1;


public class main {
    public static void main(String[] args) {


        Cliente cliente = new Cliente("Ana López");


        Producto p1 = new Producto("Mouse Gamer", 15000);
        Producto p2 = new Producto("Teclado Mecánico", 32000);
        Producto p3 = new Producto("Auriculares Bluetooth", 28000);


        Pedido pedido = new Pedido(cliente);
        pedido.agregarProducto(p1);
        pedido.agregarProducto(p2);
        pedido.agregarProducto(p3);


        double total = pedido.calcularTotal();
        System.out.println("\n--- PEDIDO CREADO ---");
        System.out.println("Total del pedido sin descuento: $" + total);


        PagoConDescuento pago = new PagoTarjetaDeCredito();
        double totalConDescuento = pago.aplicarDescuento(total);
        System.out.println("Total con descuento (10%): $" + totalConDescuento);
        pago.procesarPago(totalConDescuento);


        pedido.setEstado("Pagado");
        pedido.setEstado("Enviado");
        pedido.setEstado("Entregado");


        System.out.println("\n--- RESUMEN FINAL ---");
        System.out.println(pedido);
    }

}
```

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)
```

Segunda parte

Ejercicios sobre Excepciones:

1. División segura: Solicitar dos números y dividirlos. Manejar ArithmeticException si el divisor es cero.

DivisionSegura.java

```java
package parte2;

import java.util.Scanner;

public class DivisionSegura {
    public static void main(String[] args) {

        try (Scanner scanner = new Scanner(System.in)) {
            try {
                System.out.print("Ingrese el 1er número: ");
                int a = scanner.nextInt();

                System.out.print("Ingrese el 2do número: ");
                int b = scanner.nextInt();

                int resultado = a / b;
                System.out.println("Resultado: " + resultado);

            } catch (ArithmeticException e) {
                System.out.println("Error: no se puede dividir por cero.");
            } finally {
                System.out.println("Operación finalizada.");
            }

        }

    }
}
```

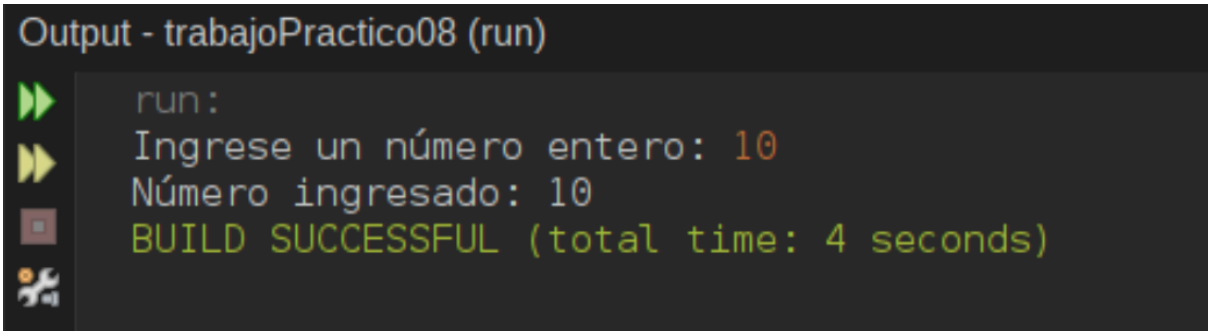
```
Output - trabajoPractico08 (run)
run:
Ingrese el ler n◆mero: 16
Ingrese el 2do n◆mero: 4
Resultado: 4
Operaci◆n finalizada.
BUILD SUCCESSFUL (total time: 9 seconds)
```

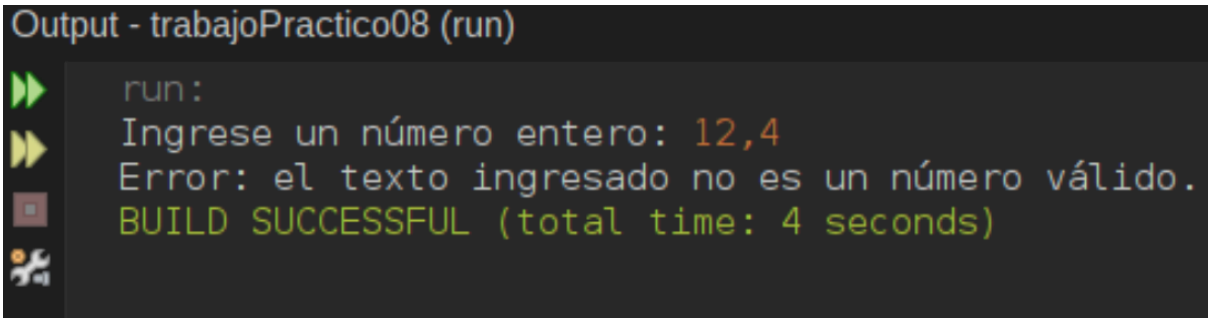
```
Output - trabajoPractico08 (run)
run:
Ingrese el ler n◆mero: 16
Ingrese el 2do n◆mero: 0
Error: no se puede dividir por cero.
Operaci◆n finalizada.
BUILD SUCCESSFUL (total time: 4 seconds)
```

2. Conversión de cadena a número: Leer texto del usuario e intentar convertirlo a int. Manejar NumberFormatException si no es válido.

ConversionDeCadenaANumero.java

```java
package parte2;

import java.util.Scanner;

public class ConversionDeCadenaANumero {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        try {
            System.out.print("Ingrese un número entero: ");
            String texto = scanner.nextLine();
            int numero = Integer.parseInt(texto);
            System.out.println("Número ingresado: " + numero);

        } catch (NumberFormatException e) {
            System.out.println("Error: el texto ingresado no es un número válido.");
        }
        scanner.close();
    }
}
```





3. Lectura de archivo: Leer un archivo de texto y mostrarlo. Manejar FileNotFoundException si el archivo no existe.

LecturaDeArchivo.java

```java
package parte2;

import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;

public class LecturaDeArchivo {

    public static void main(String[] args) {
        try {
            BufferedReader buffRead = new BufferedReader(new FileReader("archivo.txt"));
            String linea;

            while ((linea = buffRead.readLine()) != null) {
                System.out.println(linea);
            }

            buffRead.close();

        } catch (FileNotFoundException e) {
            System.out.println("FileNotFoundException: el archivo no fue encontrado.");
        } catch (IOException e) {
            System.out.println("Error al leer el archivo.");
        }
    }

}
```

Output - trabajoPractico08 (run)

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

4. Excepción personalizada: Crear EdadInvalidaException. Lanzarla si la edad es menor a 0 o mayor a 120. Capturarla y mostrar mensaje.

ExcepcionPersonalizada.java

```java
package parte2;

import java.util.Scanner;

public class ExcepcionPersonalizada {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        try {
            System.out.print("Ingrese su edad: ");
            int edad = scanner.nextInt();
            validarEdad(edad);
            System.out.println("Edad válida: " + edad);

        } catch (EdadInvalidaException e) {
            System.out.println("Error: " + e.getMessage());
        }

        scanner.close();
    }

    public static void validarEdad(int edad) throws EdadInvalidaException {
        if (edad < 0 || edad > 120) {
            throw new EdadInvalidaException("La edad debe estar entre 0 y 120 años.");
        }
    }
}
```

EdadInvalidaException.java

```java
package parte2;

public class EdadInvalidaException extends Exception{
    public EdadInvalidaException(String mensaje) {
            super(mensaje);
        }
}
```

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)
```

5. Uso de try-with-resources: Leer un archivo con BufferedReader usando try-with-resources. Manejar IOException correctamente.

UsoDeTryWithResources.java

```java
package parte2;

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;

public class UsoDeTryWithResources {

    public static void main(String[] args) {
        String ruta = "src/parte2/datos.txt";

        try (BufferedReader br = new BufferedReader(new FileReader(ruta))) {

            String linea;
            System.out.println("=== Contenido del archivo ===");
            while ((linea = br.readLine()) != null) {
                System.out.println(linea);
            }

        } catch (IOException e) {
            System.out.println("Error al acceder o leer el archivo: " + e.getMessage());
        }

        System.out.println("Lectura finalizada correctamente.");
    }
}
```

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

Fin del texto.

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