

PROGRAMACIÓN II

TP 8: Interfaces y Excepciones en Java

Alumna: Eugenia Demarchi

Comisión: 7

Parte 1:

Interfaces en un sistema de E-commerce

1. **Crear una interfaz Pagable con el método calcularTotal().**

```
public interface Pagable {  
    public double calcularTotal();  
}
```

2. **Clase Producto: tiene nombre y precio, implementa Pagable.**

```
public abstract class Producto implements Pagable{  
    private String nombre;  
    private double precio;  
  
    public double calcularTotal(){  
        return precio;  
    }  
  
    public String getNombre() {  
        return nombre;  
    }  
  
    public double getPrecio() {  
        return precio;  
    }  
  
    public Producto(String nombre, double precio) {  
        this.nombre = nombre;  
        this.precio = precio;  
    }  
}
```

3. **Clase Pedido: tiene una lista de productos, implementa Pagable y calcula el total del pedido.**

```
public class Pedido implements Pagable{
```

```
private List<Producto> productos = new ArrayList<>();
```

```
    @Override
    public double calcularTotal() {
        double acu = 0;
        for (Producto p : productos){

            acu += p.calcularTotal();
        }
        return acu;
    }
}
```

4. Ampliar con interfaces **Pago** y **PagoConDescuento** para distintos medios de pago (**TarjetaCredito**, **PayPal**), con métodos **procesarPago(double)** y **aplicarDescuento(double)**.

```
public interface Pago {
    public double procesarPago(double monto);
}

public interface PagoConDescuento extends Pago{
    public double aplicarDescuento(double monto, double porcentaje);
}
```

```
public class PayPal implements PagoConDescuento{
    @Override
    public double procesarPago(double monto) {
        System.out.println("Pago procesado: " + monto);
        return monto;
    }
    @Override
    public double aplicarDescuento(double monto, double porcentaje) {
        return monto * (1 - porcentaje / 100);
    }
}
```

```
public class TarjetaCredito implements PagoConDescuento{
    @Override
    public double aplicarDescuento(double monto, double porcentaje) {
        return monto * (1 - porcentaje / 100);
    }
    @Override
    public double procesarPago(double monto) {
        System.out.println("Pago procesado: " + monto);
        return monto;
    }
}
```

```
}  
}
```

5. Crear una interfaz Notificable para notificar cambios de estado. La clase Cliente implementa dicha interfaz y Pedido debe notificarlo al cambiar de estado.

```
public interface Notificable {  
    public void notificar(String mensaje);  
}  
  
public class Cliente implements Notificable{  
  
    @Override  
  
    public void notificar(String mensaje) {  
        System.out.println("Notificacion para cliente " + mensaje);  
    }  
}  
  
public class Pedido implements Pagable{  
    private List<Producto> productos = new ArrayList<>();  
    private List<Cliente> clientes = new ArrayList<>();  
    private String estado;  
  
    public void agregarCliente(Cliente cliente) {  
        clientes.add(cliente);  
    }  
    public void agregarProducto(Producto producto) {  
        productos.add(producto);  
    }  
    @Override  
    public double calcularTotal() {  
        double acu = 0;  
        for (Producto p : productos){  
            acu += p.calcularTotal();  
        }  
        return acu;  
    }  
  
    public void setEstado(String nuevoEstado) {  
        this.estado = nuevoEstado;  
        for (Cliente c : clientes) {  
            c.notificar("Tu pedido ha cambiado de estado a: " + nuevoEstado);  
        }  
    }  
}
```

```

public class Principal {

    public static void main(String[] args) {
        Pedido p = new Pedido();
        Cliente perez = new Cliente();
        ProductoNormal cartera =new ProductoNormal ("cartera", 70000);
        p.agregarProducto(cartera);
        p.agregarCliente(perez);
        p.setEstado("Entregado");
    }
}

```

Parte 2:

Ejercicios sobre Excepciones

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

```

public class DivisionSegura {
    public static void main(String[] args) {
        int x, y, z;
        System.out.println("Ingrese x: ");
        System.out.println("Ingrese y: ");
        try{
            Scanner sc = new Scanner(System.in);
            x = sc.nextInt();
            y = sc.nextInt();
            z = x/y;
            System.out.println("El resultado es " + z);
        }
        catch(ArithmeticException a)
        {
            System.out.println("No se puede dividir por 0");
        }
        catch(InputMismatchException a)
        {
            System.out.println("Deben ingresarse numeros enteros");
        } finally{

            System.out.println("Adios!");
        }
    }
}

```

```
run:
Ingrese x:
Ingrese y:
23
0
No se puede dividir por 0
Adios!
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.

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

        System.out.print("Ingresa un numero: ");

        try {
            Scanner sc = new Scanner(System.in);
            String texto = sc.nextLine();
            int numero = Integer.parseInt(texto);
        }
        catch (NumberFormatException n){
            System.out.println("Debes ingresar un numero");
        }
        finally{
            System.out.println("Gracias");
        }
    }
}
```

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

```
public class File {
    public static void main(String[] args) {
        VisorDeTxt v = new VisorDeTxt();
        v.mostrarTxt("C:\\Users\\Usuario\\Desktop\\PROGRAMACION\\UTN\\2NDO_CUATRIMESTRE\\
Programacion_2\\REPOSITORIO_TPS_PROGRAMACION_02\\Practico_08\\src\\Parte_2\\archi
vo.txt");
    }
}

public class VisorDeTxt {
    public void mostrarTxt(String ruta){
        java.io.File elArchivo = new java.io.File(ruta);
```

```
// try-with-resources para cerrar automáticamente el BufferedReader
try ( BufferedReader br = new BufferedReader(new FileReader(elArchivo))) {
String linea;
while ((linea = br.readLine()) != null) {
    System.out.println(linea);
}
} catch (FileNotFoundException fnfe){
    System.out.println("Archivo no encontrado: " + ruta);
}
catch(IOException ioe){
    System.out.println("Error E/S: " + ioe.getMessage());
}

}
}
```

run:

Archivo no encontrado:

C:\Users\Usuario\Desktop\PROGRAMACION\UTN\2NDO_CUATRIMESTRE\Programacion_2\RE
 POSITORIO_TPS_PROGRAMACION_02\Practico_08\src\Parte_2\archivo.txtn

BUILD SUCCESSFUL (total time: 0 seconds)

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

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

    System.out.print("Ingresa tu edad: ");

    try {
Scanner sc = new Scanner(System.in);
int edad = sc.nextInt();
if (edad < 0 || edad > 120){
    throw new EdadInvalidaException("La edad esta fuera de rango");
}

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

}
}
```

```

public class EdadInvalidaException extends Exception{

    public EdadInvalidaException() {
    }
    public EdadInvalidaException(String message) {
        super(message);
    }
    public EdadInvalidaException(String message, Throwable cause) {
        super(message, cause);
    }
    public EdadInvalidaException(Throwable cause) {
        super(cause);
    }
}

```

run:

Ingresa tu edad: -12

La edad esta fuera de rango

BUILD SUCCESSFUL (total time: 3 seconds)

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

```

public class VisorDeArchivo {
    public void mostrarTxt(String ruta){
        File elArchivo = new File(ruta);

        try ( BufferedReader br = new BufferedReader(new FileReader(elArchivo))) {
            System.out.println(br.readLine());
        } catch (IOException e){
            System.out.println("Error E/S " + e.getMessage());
        }

    }
}

```

```

public class Archivo {

```

```

    public static void main(String[] args) {
        VisorDeArchivo v = new VisorDeArchivo();
        // try {

```

```

        v.mostrarTxt("C:\\Users\\Usuario\\Desktop\\PROGRAMACION\\UTN\\2NDO_CUATRIMESTRE\\
        Programacion_2\\JAVA\\excepciones\\src\\txt\\MENSAJE_OCULTO.txt");
    }
}

```

```
}
```

Error E/S

C:\Users\Usuario\Desktop\PROGRAMACION\UTN\2NDO_CUATRIMESTRE\Programacion_2\JAVA\excepciones\src\txt\MENSAJE_OCULTO.txt (El sistema no puede encontrar el archivo especificado)

BUILD SUCCESSFUL (total time: 0 seconds)