PROGRAMACIÓN II

TRABAJO PRÁCTICO Nº8: INTERFACES Y EXCEPECIONES

CHRISTIAN EMMANUEL OLIVERO

https://github.com/CHRISTIAN2320/UTN-TUPAD-Programacion2

Parte 1: Interfaces en un sistema de E-commerce

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

```
public interface Pagable {

   double calcularTotal();
```

2. Clase Producto: tiene nombre y precio, implementa 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 this.precio;
   }
   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;
   }
}
}
```

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

```
package sistemaecommerce;
import java.util.ArrayList;
import java.util.List;

public class Pedido implements Pagable {
   private final List<Producto> productos;
   private Cliente cliente;
   public Pedido() {
      this.productos = new ArrayList<>();
   }
   public void agregarProducto(Producto producto) {
      this.productos.add(producto);
   }

   @Override
   public double calcularTotal() {
      double total = 0.0;
      for (Producto p
            : productos) {
         total
               += p.calcularTotal();
      }
      return total;
   }

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

   public void setCliente(Cliente cliente) {
      this.cliente = cliente;
   }

   public void cambiarEstadoYNotificar(String nuevoEstado) {
      System.out.println( "--- El estado del pedido ha cambiado a : " + nuevoEstado + " ---");
      if (this.cliente != null) {
         this.cliente.notificar("Estado actualizado: " + nuevoEstado);
      } else {
         System.out.println("No se pudo notificar: Cliente no asignado al pedido.");

      }
   }
}
```

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

```
public interface Pago {
   boolean procesarPago(double monto);
}
-----------------------------------------------------------------------------------------------------------------------
public interface PagoConDescuento {
   double aplicarDescuento(double monto);
}
-----------------------------------------------------------------------------------------------------------------------
public class Paypal implements Pago {
   @Override
   public boolean procesarPago(double monto) {
      System.out.println(
            "[PayPal] Iniciando sesion para el pago " + monto);
      if (monto < 0) {
         System.out.println("[PayPal] Monto invalido.");
         return false;
      }
      System.out.println("[PayPal] Pago procesado con exito");
      return true;
   }
}
-----------------------------------------------------------------------------------------------------------------------
public class TarjetaDeCredito implements PagoConDescuento {

   private final double descuentoPorcentaje;

   public TarjetaDeCredito(double descuentoPorcentaje) {
      this.descuentoPorcentaje = descuentoPorcentaje;
   }
   @Override
   public double
         aplicarDescuento(double monto) {
      double descuento = monto * (descuentoPorcentaje
            / 100);
      double montoFinal = monto - descuento;
      System.out.println("[Tarjeta] Aplicando descuento de " + descuentoPorcentaje + "%.
Total con desc.: " + montoFinal);
      return montoFinal;
   }
   public boolean procesarPago(double monto) {
      System.out.println("[Tarjeta] Procesando pago por "
            + monto);
      System.out.println("[Tarjeta] pago exitoso");
      return true;
   }
}
```

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 {
   void notificar(String mensaje);
}
```

Parte 2: Ejercicios sobre Excepciones

1. División segura

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

```
package javaapplication97
import java.util.InputMismatchException;
import java.util.Scanner;
public class Exception1 {

   public static void main(String[] args) {
      System.out.println("--- Divisionsegura ---");
      try (Scanner scanner = new Scanner(System.in)) {

         System.out.print("Ingrese el numerador entero: ");
         int numerador = scanner.nextInt();
         System.out.print("Ingrese el divisor entero:  ");
         int divisor = scanner.nextInt();

         int resultado = numerador / divisor;
         System.out.println("Resultado de la division: " + resultado);

      } catch (ArithmeticException e) {
         System.err.println("\n Error de calculo: ArithmeticException capturada.");
         System.err.println("Motivo: No se puede dividir por cero. El sistema continua.");

      } catch (InputMismatchException e) {
         System.err.println("\n Error de entrada: InputMismatchException capturada.");
         System.err.println("Por favor, ingrese solo numeros enteros.");

      }
      System.out.println("\n El programa ha finalizado el intento de division.");
   }
}
```

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

```
package javaapplication97;
import java.util.Scanner;
public class Exception2 {

    public static void main(String[] args) {
        System.out.println("--- Conversion ---");
        try (Scanner scanner = new Scanner(System.in)) {
            System.out.print("Ingrese una cadena de texto para convertirla a numero: ");
            String cadena = scanner.nextLine();

            int numero = Integer.parseInt(cadena);

            System.out.println("La conversion fue exitosa. Numero resultante: " + numero);

        } catch (NumberFormatException e) {
            System.err.println("\n ERROR DE CONVERSION: NumberFormatException
capturada.");
            System.err.println("La cadena ingresada no representa un numero entero valido.");
            }
        System.out.println("\n El sistema ha finalizado la conversion.");
    }
}
```

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

```
package javaapplication97;
import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;

public class Exception3 {
    public static void main(String[] args) {
        System.out.println("---Lectura de archivo ---");
        String nombreArchivo = "datos.txt";
        try (Scanner lectorArchivo = new Scanner(new File(nombreArchivo))) {
            System.out.println("Contenido del archivo "
                    + nombreArchivo + ":");
            // Leer y mostrar cada linea del archivo
            while (lectorArchivo.hasNextLine()) {
                String linea = lectorArchivo.nextLine();
                System.out.println(linea);
            }

        } catch (FileNotFoundException e) {
            System.err.println("\n Error de archivo: FileNotFoundException capturada.");
            System.err.println("El archivo "
                    + nombreArchivo + " no se encontro en la ruta del proyecto.");
            System.err.println("Cree el archivo en la raiz del proyecto.");

        }
        System.out.println("\n El sistema ha finalizado la lectura de archivo.");
    }
}
```

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

```
package javaapplication97;
import java.util.Scanner;
public class EdadInvalidaException extends Exception {
    public EdadInvalidaException(String mensaje) {
        super(mensaje);
    }
}
-----------------------------------------------------------------------------------------------------------------------

package javaapplication97;
import java.util.Scanner;
public class Exception4 {
    public static void validarEdad(int edad)
        throws EdadInvalidaException {
        if (edad < 0) {
            throw new EdadInvalidaException("La edad no puede ser negativa: " + edad);
        }
        if (edad > 120) {
            throw new EdadInvalidaException("La edad no puede ser mayor a 120: " + edad);
        }
        System.out.println("Edad validada con exito: "+ edad);
    }

    public static void main(String[] args) {
        System.out.println("--- Excepcion personalizada ---");
        try (Scanner scanner = new Scanner(System.in)) {
            System.out.print("Ingrese una edad para validar: ");
            int edadIngresada = scanner.nextInt();
            validarEdad(edadIngresada);
        } catch (EdadInvalidaException e) {
            System.err.println("\n Error de validacion: Excepcion personalizada capturada.");
            System.err.println("Mensaje: " + e.getMessage());
        } catch (Exception e) {
            System.err.println("\n Ocurrio un error inesperado.");
        }
        System.out.println("\n El sistema de validacion ha finalizado.");
    }
}
```

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

```
package javaapplication97;
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;

public class Exception5 {

    public static void main(String[] args) {
        System.out.println("--- "+ " de try-with-resources ---");

        String nombreArchivo = "documento.txt";
        String linea;

        // Bloque try-with-resources
        try (BufferedReader br = new BufferedReader(new FileReader(nombreArchivo))) {
            System.out.println("Leyendo archivo con BufferedReader:");

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

        } catch (IOException e) {
            System.err.println("\nError ENTRADA/SALIDA: IOException capturada.");
            System.err.println("Ocurrió un error al intentar leer el archivo: " + nombreArchivo);
            System.err.println("Mensaje: " + e.getMessage());
        }

        System.out.println("\nEl sistema ha finalizado el uso de try-with-resources.");
    }
}
```