TRABAJO PRACTICO N8

INTERFACES Y EXCEPCIONES

EMILIANO JARA

PROGRAMACION 2

ACTIVIDADES

Parte 1: Interfaces y sistema de E-commerce

1) Interfaz Pagable

Define el contrato para calcular totales

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

2) Clase Producto (implementa Pagable)

Tiene nombre y precio; su total es su propio precio.

```
public class Producto implements Pagable {
   private final String nombre;
   private final double precio;

   public Producto(String nombre, double precio) {
      if (precio < 0) throw new IllegalArgumentException("El precio no puede ser negativo");
      this.nombre = nombre;
      this.precio = precio;
   }

   public String getNombre() { return nombre; }
```

```
  public double getPrecio() { return precio; }

  @Override
  public double calcularTotal() { return precio; }

  @Override
  public String toString() {
    return nombre + " ($" + precio + ")";
  }
}
```

3) Clase Pedido (implementa Pagable)

Acumula productos y calcula el total; además **notifica** a un Cliente cuando cambia estado (usa Notificable).

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

public class Pedido implements Pagable {
  public enum Estado { CREADO, PAGADO, ENVIADO, ENTREGADO }

  private final List<Producto> productos = new ArrayList<>();
  private Estado estado = Estado.CREADO;
  private final Cliente cliente;

  public Pedido(Cliente cliente) { this.cliente = cliente; }

  public void agregarProducto(Producto p) { productos.add(p); }
  public List<Producto> getProductos() { return productos; }
```

```
    @Override
    public double calcularTotal() {
        return productos.stream().mapToDouble(Producto::getPrecio).sum();
    }

    public void cambiarEstado(Estado nuevo) {
        this.estado = nuevo;
        cliente.notificar("El pedido cambió de estado a: " + nuevo);
    }

    public boolean pagar(Pago pago) {
        double total = calcularTotal();
        if (pago instanceof PagoConDescuento) {
            total = ((PagoConDescuento) pago).aplicarDescuento(total);
        }
        boolean ok = pago.procesarPago(total);
        if (ok) cambiarEstado(Estado.PAGADO);
        return ok;
    }
}
```

4) Interfaces Pago y PagoConDescuento + medios de pago

PagoConDescuento extiende Pago (herencia de interfaces).

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

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

Implementaciones:

```
public class TarjetaCredito implements PagoConDescuento {
   private final String numero;
   private final String titular;

   public TarjetaCredito(String numero, String titular) {
      this.numero = numero;
      this.titular = titular;
   }

   @Override
   public double aplicarDescuento(double monto) { return monto * 0.95; } // 5%

   @Override
   public boolean procesarPago(double monto) {
      System.out.println("[TarjetaCredito] Procesando pago de $" + monto + " para "
+ titular);
      return true;
   }
}
public class PayPal implements Pago {
   private final String cuenta;
```

```
  public PayPal(String cuenta) { this.cuenta = cuenta; }

  @Override
  public boolean procesarPago(double monto) {
    System.out.println("[PayPal] Procesando pago de $" + monto + " para la cuenta " + cuenta);
    return true;
  }
}
```

5) Interfaz Notificable y clase Cliente

Cliente implementa Notificable y recibe mensajes al cambiar el estado del Pedido.

```
public interface Notificable {
  void notificar(String mensaje);
}
public class Cliente implements Notificable {
  private final String nombre;
  private final String email;

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

  public String getNombre() { return nombre; }
  public String getEmail() { return email; }

  @Override
```

```
  public void notificar(String mensaje) {
    System.out.println("[Notificación a " + nombre + "] " + mensaje);
  }
}
```

6) MainEcommerce (demo de todo junto)

```
public class MainEcommerce {
  public static void main(String[] args) {
    Cliente cliente = new Cliente("Emiliano", "emi@example.com");
    Pedido pedido = new Pedido(cliente);
    pedido.agregarProducto(new Producto("Mouse", 5000));
    pedido.agregarProducto(new Producto("Teclado", 12000));

    System.out.println("Total antes de pago: $" + pedido.calcularTotal());

    Pago tarjeta = new TarjetaCredito("4111-xxxx-xxxx-1111", "Emiliano");
    pedido.pagar(tarjeta);

    Pago paypal = new PayPal("emi@paypal");
    pedido.pagar(paypal);
  }
}
```

Parte 2: Ejercicios sobre Excepciones

Todos estos puntos están pedidos en el enunciado: división segura, conversión, lectura de archivo, excepción personalizada y try-with-resources.

1) División segura (ArithmeticException)

```
import java.util.Scanner;

public class DivisionSegura {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      try {
         System.out.print("Ingrese numerador: ");
         int a = Integer.parseInt(sc.nextLine());
         System.out.print("Ingrese denominador: ");
         int b = Integer.parseInt(sc.nextLine());
         int r = a / b; // puede lanzar ArithmeticException
         System.out.println("Resultado: " + r);
      } catch (ArithmeticException e) {
         System.out.println("Error: No se puede dividir por cero.");
      } catch (NumberFormatException e) {
         System.out.println("Error: Debe ingresar números enteros.");
      } finally {
         sc.close();
         System.out.println("Fin de DivisiónSegura.");
      }
   }
}
```

2) Conversión de cadena a número (NumberFormatException)

```
import java.util.Scanner;
```

```
public class ConversionCadenaANumero {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      try {
         System.out.print("Ingrese un número entero: ");
         String txt = sc.nextLine();
         int n = Integer.parseInt(txt); // puede lanzar NumberFormatException
         System.out.println("Ingresaste: " + n);
      } catch (NumberFormatException e) {
         System.out.println("Error: el texto no es un entero válido.");
      } finally {
         sc.close();
      }
   }
}
```

3) Lectura de archivo (FileNotFoundException, IOException)

```
import java.io.*;

public class LecturaArchivo {
   public static void main(String[] args) {
      if (args.length == 0) {
         System.out.println("Uso: java LecturaArchivo <ruta>");
         return;
```

```
        }
        File file = new File(args[0]);
        try (BufferedReader br = new BufferedReader(new FileReader(file))) {
            String line;
            while ((line = br.readLine()) != null) {
                System.out.println(line);
            }
        } catch (FileNotFoundException e) {
            System.out.println("Error: El archivo no existe: " + file.getAbsolutePath());
        } catch (IOException e) {
            System.out.println("Error de IO: " + e.getMessage());
        }
    }
}
```

4) Excepción personalizada (EdadInvalidaException)

```
public class EdadInvalidaException extends Exception {
    public EdadInvalidaException(String message) {
        super(message);
    }
}
import java.util.Scanner;

public class ValidadorEdad {
    public static void validar(int edad) throws EdadInvalidaException {
        if (edad < 0 || edad > 120) {
            throw new EdadInvalidaException("Edad inválida: " + edad);
```

```
    }
  }

  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    try {
      System.out.print("Ingrese edad: ");
      int edad = Integer.parseInt(sc.nextLine());
      validar(edad);
      System.out.println("Edad válida: " + edad);
    } catch (EdadInvalidaException e) {
      System.out.println("Error: " + e.getMessage());
    } catch (NumberFormatException e) {
      System.out.println("Error: Debe ingresar un número entero.");
    } finally {
      sc.close();
    }
  }
}
```

5) Try-with-resources (BufferedReader, IOException)

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

public class TryWithResourcesDemo {
```

```
    public static void main(String[] args) {
        if (args.length == 0) {
            System.out.println("Uso: java TryWithResourcesDemo <ruta>");
            return;
        }
        // try-with-resources asegura el cierre automático del recurso
        try (BufferedReader br = new BufferedReader(new FileReader(args[0]))) {
            String line;
            while ((line = br.readLine()) != null) {
                System.out.println(line);
            }
        } catch (IOException e) {
            System.out.println("Error de IO: " + e.getMessage());
        }
    }
}
```