

## Trabajo Práctico 8: Interfaces y Excepciones en Java

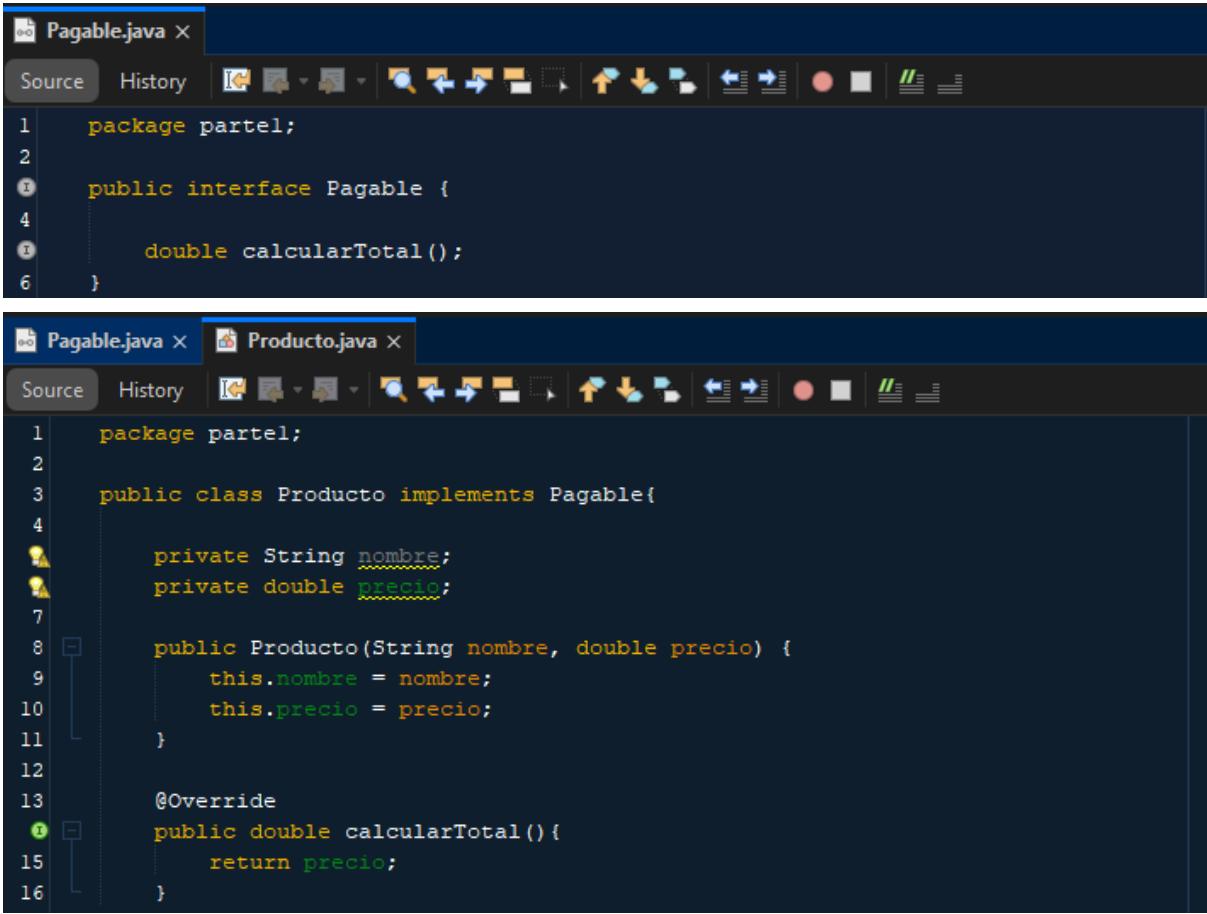
Alumno : Francisco López (39.327.419)

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.



The screenshot shows a Java development environment with two code editors open. The top editor contains the `Pagable.java` interface:

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

The bottom editor contains the `Producto.java` class, which implements the `Pagable` interface:

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

The screenshot shows a Java code editor with three tabs at the top: 'Pagable.java X', 'Producto.java X', and 'Pedido.java X'. The 'Pedido.java X' tab is active. The code editor has a dark theme with syntax highlighting. The code is as follows:

```
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    public void agregarProducto(Producto p) {
19        productos.add(p);
20    }
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 cambió a: " + estado);
44        }
45    }
46
47    public List<Producto> getProductos() {
48
49        public List<Producto> getProductos() {
50            return productos;
51        }
52
53        @Override
54        public String toString() {
55            return "Pedido{"
56                + "cliente=" + cliente
57                + ", estado='" + estado + '\''
58                + ", total=" + calcularTotal()
59                + '}';
60    }
61}
```

The screenshot shows a Java IDE interface with the Pago.java file open. The code defines a public interface named Pago with a single method: void procesarPago(double monto);.

```
1 package partel;
2
3 public interface Pago {
4
5     void procesarPago(double monto);
6
7 }
```

The screenshot shows a Java IDE interface with the PagoTarjetaDeCredito.java file open. This class implements the Pago interface and provides two implementations for the methods. It overrides the procesarPago method to print a message to the console.

```
1 package partel;
2
3 public class PagoTarjetaDeCredito implements PagoConDescuento{
4
5     @Override
6     public double aplicarDescuento(double monto){
7         return monto * 0.9;
8     }
9
10    @Override
11    public void procesarPago(double monto){
12        System.out.println("Pago con tarjeta por $" + monto);
13    }
14 }
```

The screenshot shows a Java IDE interface with the PagoConDescuento.java file open. This interface extends the Pago interface and adds a new method: double aplicarDescuento(double monto);.

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

The screenshot shows a Java IDE interface with the Notifiable.java file open. This interface defines a single method: void notificar(String mensaje);.

```
1 package partel;
2
3 public interface Notifiable {
4
5     void notificar(String mensaje);
6
7 }
```

```
1 package partel;
2
3 public class Cliente implements Notifiable{
4
5     private String nombre;
6
7     public Cliente(String nombre) {
8         this.nombre = nombre;
9     }
10
11    @Override
12    public void notificar(String mensaje){
13        System.out.println("Notificacion a " + nombre + ": " + mensaje );
14    }
15
16 }
17
18 }
```

```
1 package partel;
2
3
4 public class main {
5     public static void main(String[] args) {
6
7
8         Cliente cliente = new Cliente("Ana López");
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        PagoConDescuento pago = new PagoTarjetaDeCredito();
27        double totalConDescuento = pago.aplicarDescuento(total);
28        System.out.println("Total con descuento (10%): $" + totalConDescuento);
29        pago.procesarPago(totalConDescuento);
30
31
32        pedido.setEstado("Pagado");
33        pedido.setEstado("Enviado");
34        pedido.setEstado("Entregado");
35
36
37        System.out.println("\n--- RESUMEN FINAL ---");
38        System.out.println(pedido);
39
40    }
41
42 }
43 }
```

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=partel.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.

The screenshot shows a Java code editor with the file 'DivisionSegura.java' open. The code implements a simple division operation with exception handling. It uses a Scanner to read two integers from the user. A try block attempts the division, while a catch block handles the ArithmeticException if the divisor is zero. Finally, it prints a success message.

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

### Output - trabajoPractico08 (run)

```
▶ run:
▶ Ingrese el 1er número: 16
▶ Ingrese el 2do número: 4
▶ Resultado: 4
▶ Operación finalizada.
▶ BUILD SUCCESSFUL (total time: 9 seconds)
```

### Output - trabajoPractico08 (run)

```
▶ run:
▶ Ingrese el 1er número: 16
▶ Ingrese el 2do número: 0
▶ Error: no se puede dividir por cero.
▶ Operación 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.

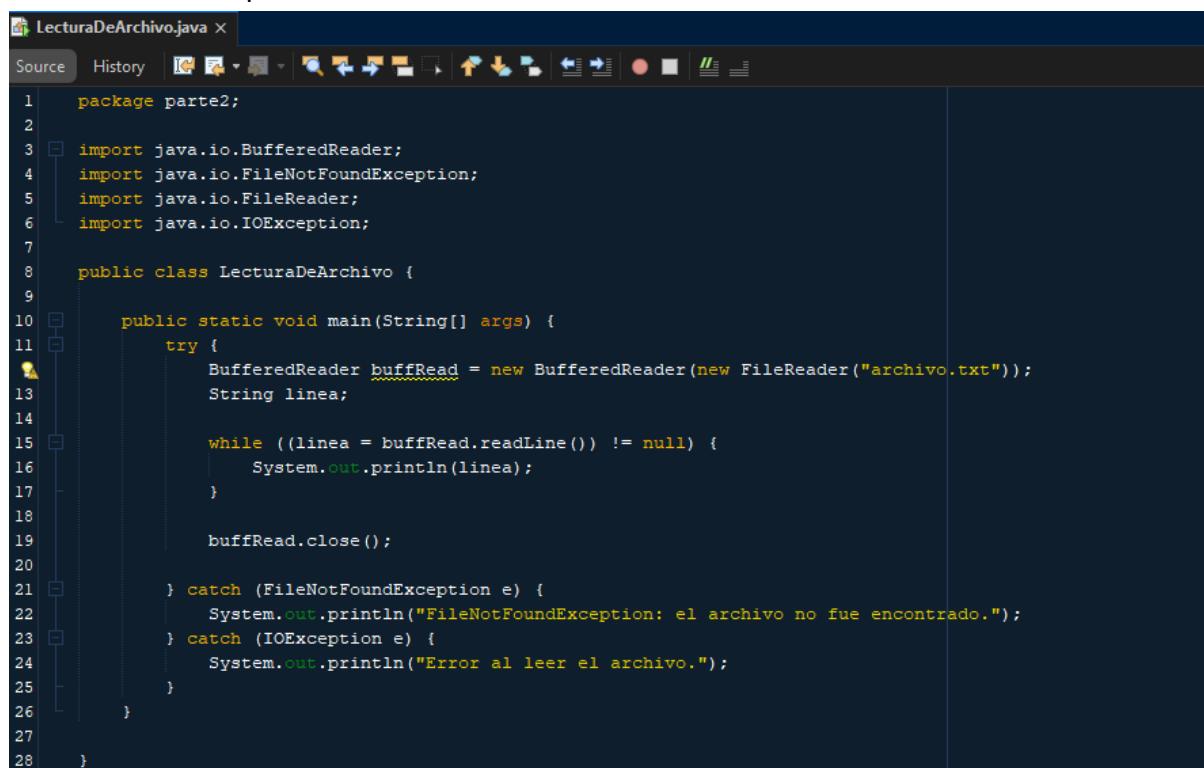
Output - trabajoPractico08 (run)

```
▶ 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)
```

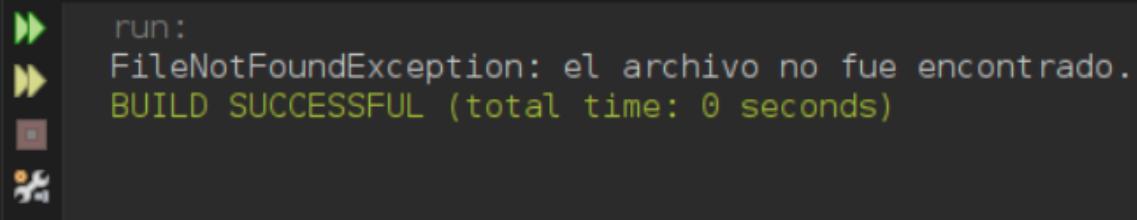
**3. Lectura de archivo:** Leer un archivo de texto y mostrarlo. Manejar FileNotFoundException si el archivo no existe.



The screenshot shows a Java code editor with the following code:

```
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 }
```

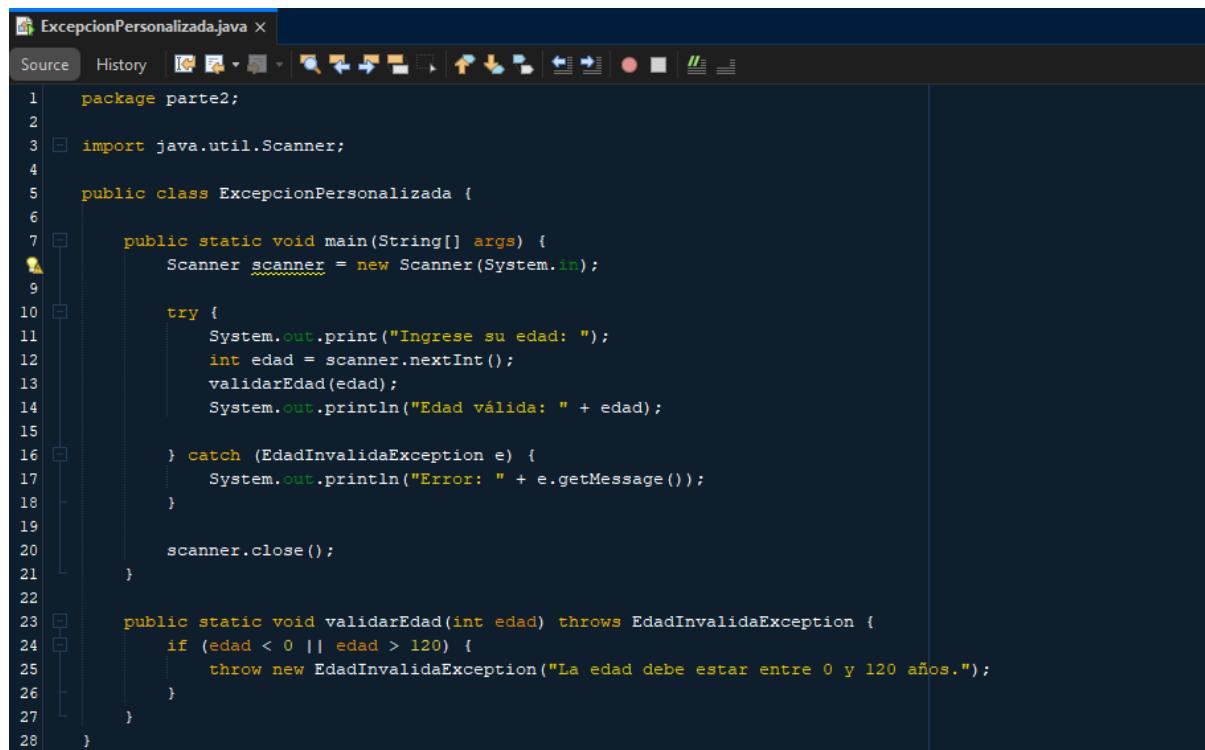
#### Output - trabajoPractico08 (run)



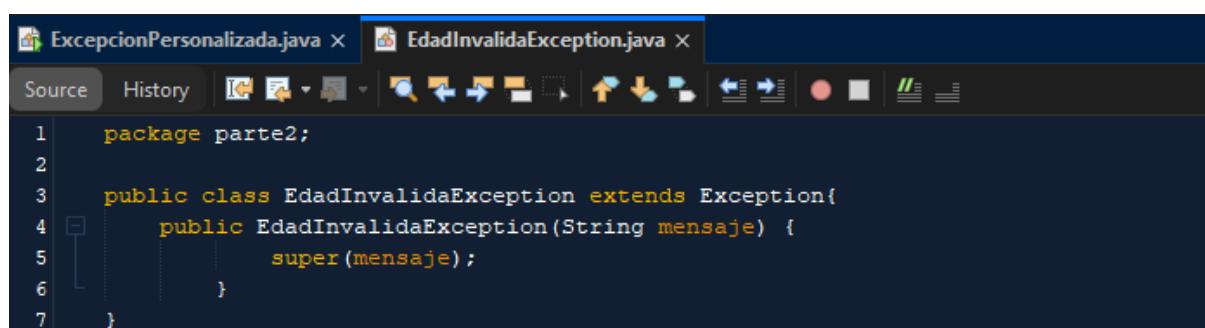
The terminal output shows the following results:

```
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.

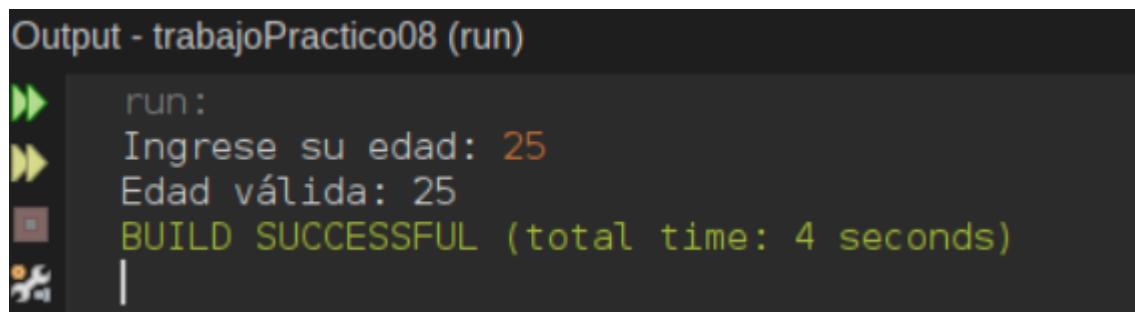


```
1 package parte2;
2
3 import java.util.Scanner;
4
5 public class ExpcionPersonalizada {
6
7     public static void main(String[] args) {
8         Scanner scanner = new Scanner(System.in);
9
10        try {
11            System.out.print("Ingrese su edad: ");
12            int edad = scanner.nextInt();
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}
```



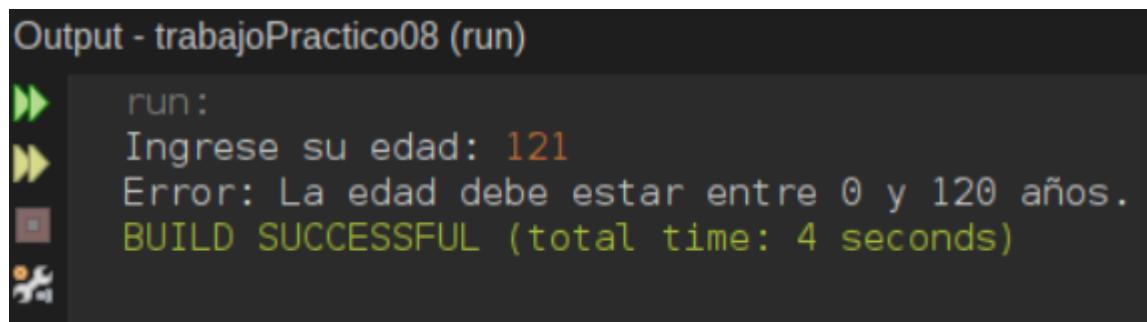
```
1 package parte2;
2
3 public class EdadInvalidaException extends Exception{
4     public EdadInvalidaException(String mensaje) {
5         super(mensaje);
6     }
7 }
```

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.

The screenshot shows a Java code editor with the following code:

```
1 package parte2;
2
3 import java.io.BufferedReader;
4 import java.io.FileReader;
5 import java.io.IOException;
6
7 public class UsoDeTryWithResources {
8
9     public static void main(String[] args) {
10         String ruta = "src/part2/datos.txt";
11
12         try (BufferedReader br = new BufferedReader(new FileReader(ruta))) {
13
14             String linea;
15             System.out.println("==== Contenido del archivo ====");
16             while ((linea = br.readLine()) != null) {
17                 System.out.println(linea);
18             }
19
20         } catch (IOException e) {
21             System.out.println("Error al acceder o leer el archivo: " + e.getMessage());
22         }
23
24         System.out.println("Lectura finalizada correctamente.");
25     }
26 }
```

```
run:  
==== Contenido del archivo ====  
Hola, este es un archivo de prueba.  
  
Fin del texto.  
  
Lectura finalizada correctamente.  
BUILD SUCCESSFUL (total time: 0 seconds)
```