

# PROGRAMACIÓN II

## Trabajo Práctico 2: Programación Estructurada

### OBJETIVO GENERAL

Desarrollar habilidades en programación estructurada en Java, abordando desde conceptos básicos como operadores y estructuras de control hasta temas avanzados como funciones, recursividad y estructuras de datos. Se busca fortalecer la capacidad de análisis y solución de problemas mediante un enfoque práctico,

### MARCO TEÓRICO

cepto	cación en el proyecto
cturas condicionales	ificación de edad, verificación de año sto
s (for, while, do-while)	etición de ingreso de datos y cálculos
iones	ulo modular de descuentos, envíos, stock
vs	ión de precios de productos
rsividad	esión recursiva de arrays

### Caso Práctico

Desarrollar los siguientes ejercicios en Java utilizando el paradigma de programación estructurada. Agrupados según el tipo de estructuras o conceptos aplicados:

#### Estructuras Condicionales:

1. Verificación de Año Bisiesto.

Escribe un programa en Java que solicite al usuario un año y determine si es bisiesto. Un año es bisiesto si es divisible por 4, pero no por 100, salvo que sea divisible por 400.

#### Ejemplo de entrada/salida:

Ingrese un año: 2024

El año 2024 es bisiesto.

Ingrese un año: 1900

El año 1900 no es bisiesto.

```
Ej1_anioBisiesto.java x
Source History
1 package Programacionestructurada_tp2;
2
3 import java.util.Scanner;
4
5 public class Ej1_anioBisiesto {
6
7     public static void main(String[] args) {
8         Scanner input = new Scanner(System.in);
9
10        System.out.print("Ingrese un año: ");
11        int año = input.nextInt();
12
13        // Verifica si el año es bisiesto
14        if ((año % 4 == 0 && año % 100 != 0) || (año % 400 == 0)) {
15            System.out.println("El año " + año + " es bisiesto.");
16        } else {
17            System.out.println("El año " + año + " no es bisiesto.");
18        }
19    }
20 }
21
```

## 2. Determinar el Mayor de Tres Números.

Escribe un programa en Java que pida al usuario tres números enteros y determine cuál es el mayor.

### Ejemplo de entrada/salida:

Ingrese el primer número: 8

Ingrese el segundo número: 12

Ingrese el tercer número: 5

El mayor es: 12

```
Ej2_mayorDeTres.java x
Source History
1 package programacionestructurada_tp2;
2
3 import java.util.Scanner;
4
5 public class Ej2_mayorDeTres {
6
7     public static void main(String[] args) {
8         Scanner input = new Scanner(System.in);
9
10        System.out.print("Ingrese el primer número: ");
11        int num1 = input.nextInt();
12
13        System.out.print("Ingrese el segundo número: ");
14        int num2 = input.nextInt();
15
16        System.out.print("Ingrese el tercer número: ");
17        int num3 = input.nextInt();
18
19        int mayor;
20
21        // Determinar cuál de los tres números es el mayor
22        if (num1 >= num2 && num1 >= num3) {
23            mayor = num1;
24        } else if (num2 >= num1 && num2 >= num3) {
25            mayor = num2;
26        } else {
27            mayor = num3;
28        }
29
30        //Mostrar resultado
31        System.out.println("El mayor es: " + mayor);
32    }
33 }
```

### 3. Clasificación de Edad.

Escribe un programa en Java que solicite al usuario su edad y clasifique su etapa de vida según la siguiente tabla:

Menor de 12 años: "Niño"

Entre 12 y 17 años: "Adolescente"

Entre 18 y 59 años: "Adulto"

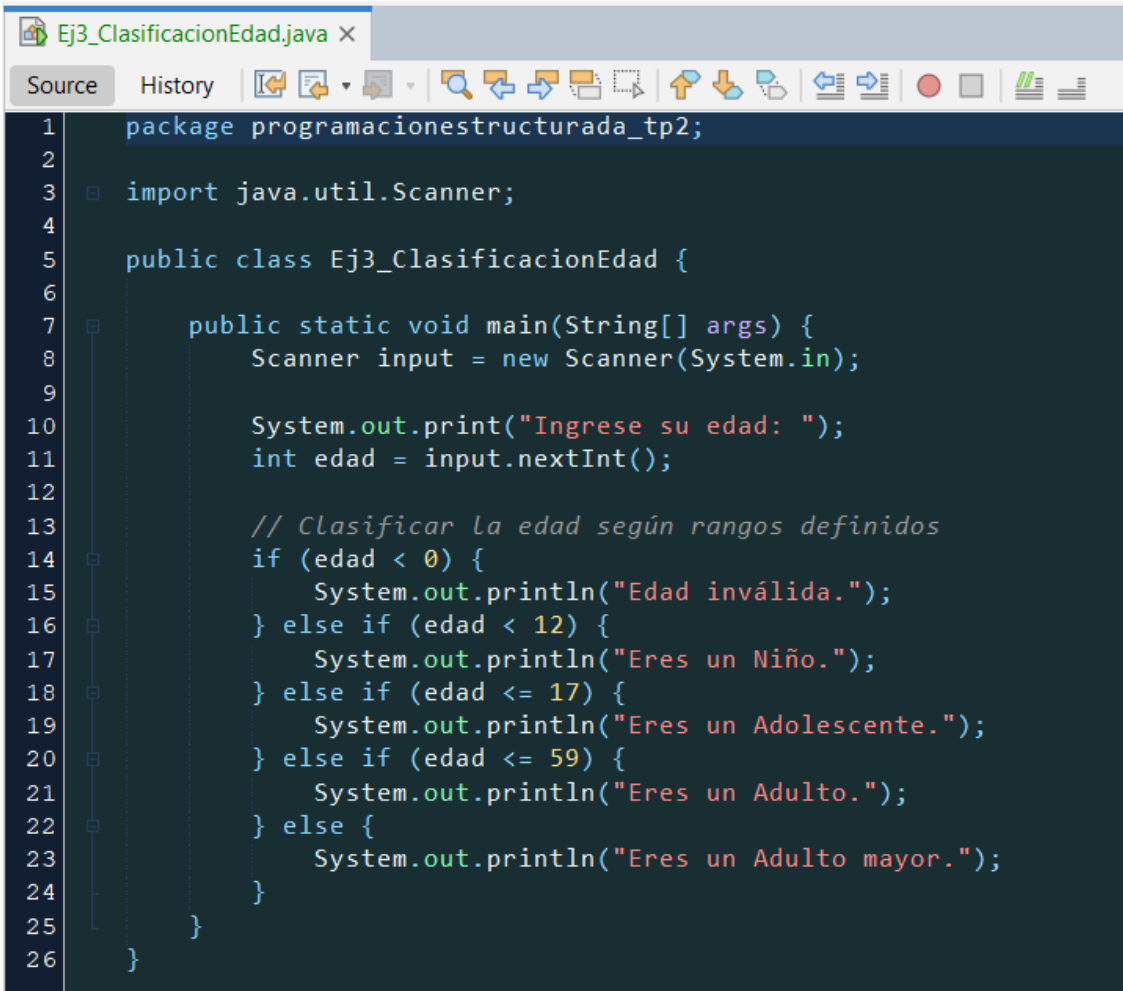
60 años o más: "Adulto mayor"

**Ejemplo de entrada/salida:**

Ingrese su edad: 25 Eres un Adulto.

Ingrese su edad: 10

Eres un Niño.



```
1 package programacionestructurada_tp2;
2
3 import java.util.Scanner;
4
5 public class Ej3_ClasificacionEdad {
6
7     public static void main(String[] args) {
8         Scanner input = new Scanner(System.in);
9
10        System.out.print("Ingrese su edad: ");
11        int edad = input.nextInt();
12
13        // Clasificar la edad según rangos definidos
14        if (edad < 0) {
15            System.out.println("Edad inválida.");
16        } else if (edad < 12) {
17            System.out.println("Eres un Niño.");
18        } else if (edad <= 17) {
19            System.out.println("Eres un Adolescente.");
20        } else if (edad <= 59) {
21            System.out.println("Eres un Adulto.");
22        } else {
23            System.out.println("Eres un Adulto mayor.");
24        }
25    }
26 }
```

#### 4. Calculadora de Descuento según categoría.

Escribe un programa que solicite al usuario el precio de un producto y su categoría (A, B o C).

Luego, aplique los siguientes descuentos:

Categoría A: 10% de descuento

Categoría B: 15% de descuento

Categoría C: 20% de descuento

El programa debe mostrar el precio original, el descuento aplicado y el precio final

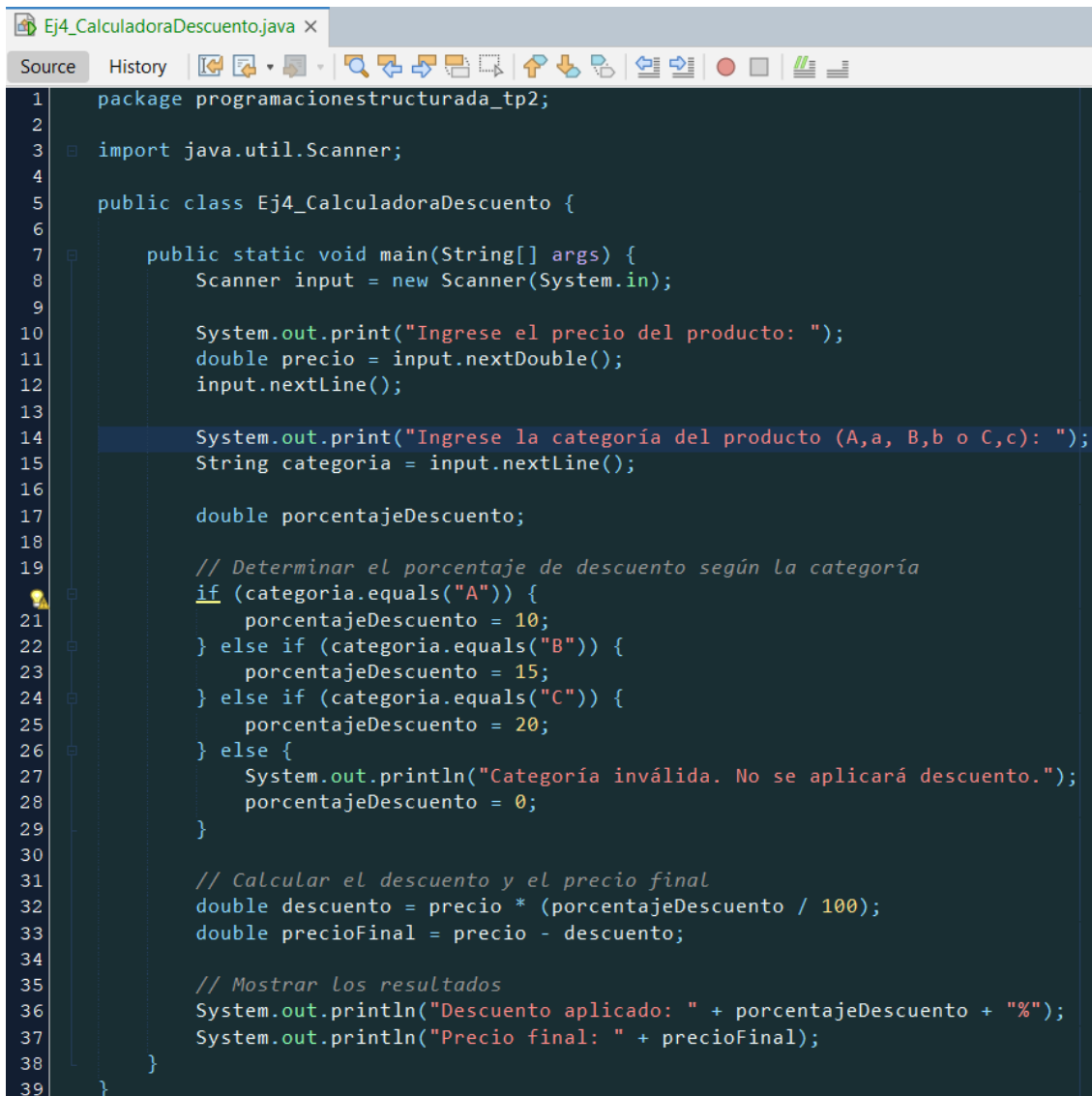
**Ejemplo de entrada/salida:**

Ingrese el precio del producto: 1000

Ingrese la categoría del producto (A, B o C): B

Descuento aplicado: 15%

Precio final: 850.0



```
1 package programacionestructurada_tp2;
2
3 import java.util.Scanner;
4
5 public class Ej4_CalculadoraDescuento {
6
7     public static void main(String[] args) {
8         Scanner input = new Scanner(System.in);
9
10        System.out.print("Ingrese el precio del producto: ");
11        double precio = input.nextDouble();
12        input.nextLine();
13
14        System.out.print("Ingrese la categoría del producto (A,a, B,b o C,c): ");
15        String categoria = input.nextLine();
16
17        double porcentajeDescuento;
18
19        // Determinar el porcentaje de descuento según la categoría
20        if (categoria.equals("A")) {
21            porcentajeDescuento = 10;
22        } else if (categoria.equals("B")) {
23            porcentajeDescuento = 15;
24        } else if (categoria.equals("C")) {
25            porcentajeDescuento = 20;
26        } else {
27            System.out.println("Categoría inválida. No se aplicará descuento.");
28            porcentajeDescuento = 0;
29        }
30
31        // Calcular el descuento y el precio final
32        double descuento = precio * (porcentajeDescuento / 100);
33        double precioFinal = precio - descuento;
34
35        // Mostrar los resultados
36        System.out.println("Descuento aplicado: " + porcentajeDescuento + "%");
37        System.out.println("Precio final: " + precioFinal);
38    }
39 }
```

### Estructuras de Repetición:

#### 5. Suma de Números Pares (while).

Escribe un programa que solicite números al usuario y sume solo los números pares. El ciclo debe continuar hasta que el usuario ingrese el número 0, momento en el que se debe mostrar la suma total de los pares ingresados.

#### Ejemplo de entrada/salida:

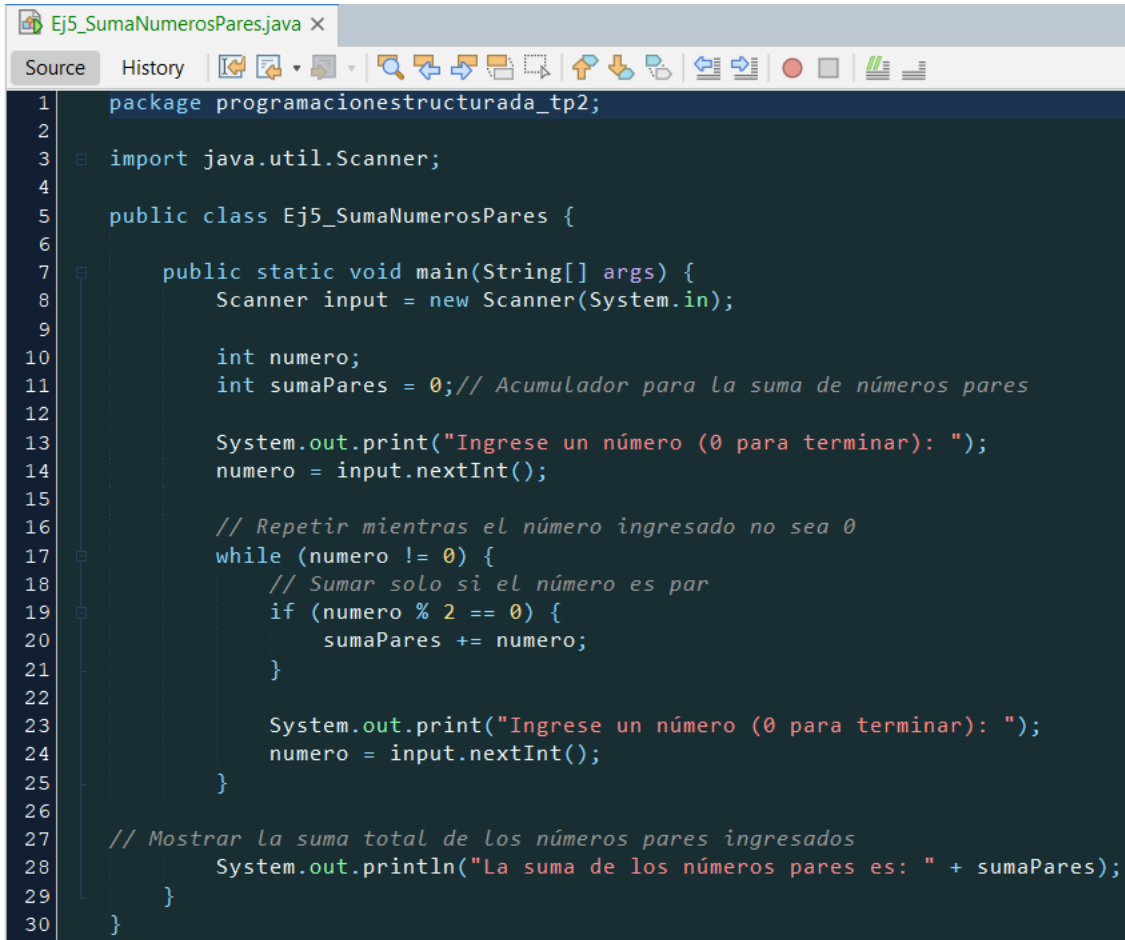
Ingrese un número (0 para terminar): 4

Ingrese un número (0 para terminar): 7

Ingrese un número (0 para terminar): 2

Ingrese un número (0 para terminar): 0

La suma de los números pares es: 6



```
1 package programacionestructurada_tp2;
2
3 import java.util.Scanner;
4
5 public class Ej5_SumaNumerosPares {
6
7     public static void main(String[] args) {
8         Scanner input = new Scanner(System.in);
9
10        int numero;
11        int sumaPares = 0; // Acumulador para la suma de números pares
12
13        System.out.print("Ingrese un número (0 para terminar): ");
14        numero = input.nextInt();
15
16        // Repetir mientras el número ingresado no sea 0
17        while (numero != 0) {
18            // Sumar solo si el número es par
19            if (numero % 2 == 0) {
20                sumaPares += numero;
21            }
22
23            System.out.print("Ingrese un número (0 para terminar): ");
24            numero = input.nextInt();
25        }
26
27        // Mostrar la suma total de los números pares ingresados
28        System.out.println("La suma de los números pares es: " + sumaPares);
29    }
30 }
```

6. Contador de Positivos, Negativos y Ceros (for).

Escribe un programa que pida al usuario ingresar 10 números enteros y cuente cuántos son positivos, negativos y cuántos son ceros.

### Ejemplo de entrada/salida:

Ingrese el número 1: -5

Ingrese el número 2: 3

Ingrese el número 3: 0

Ingrese el número 4: -1

Ingrese el número 5: 6

Ingrese el número 6: 0

Ingrese el número 7: 9

Ingrese el número 8: -3

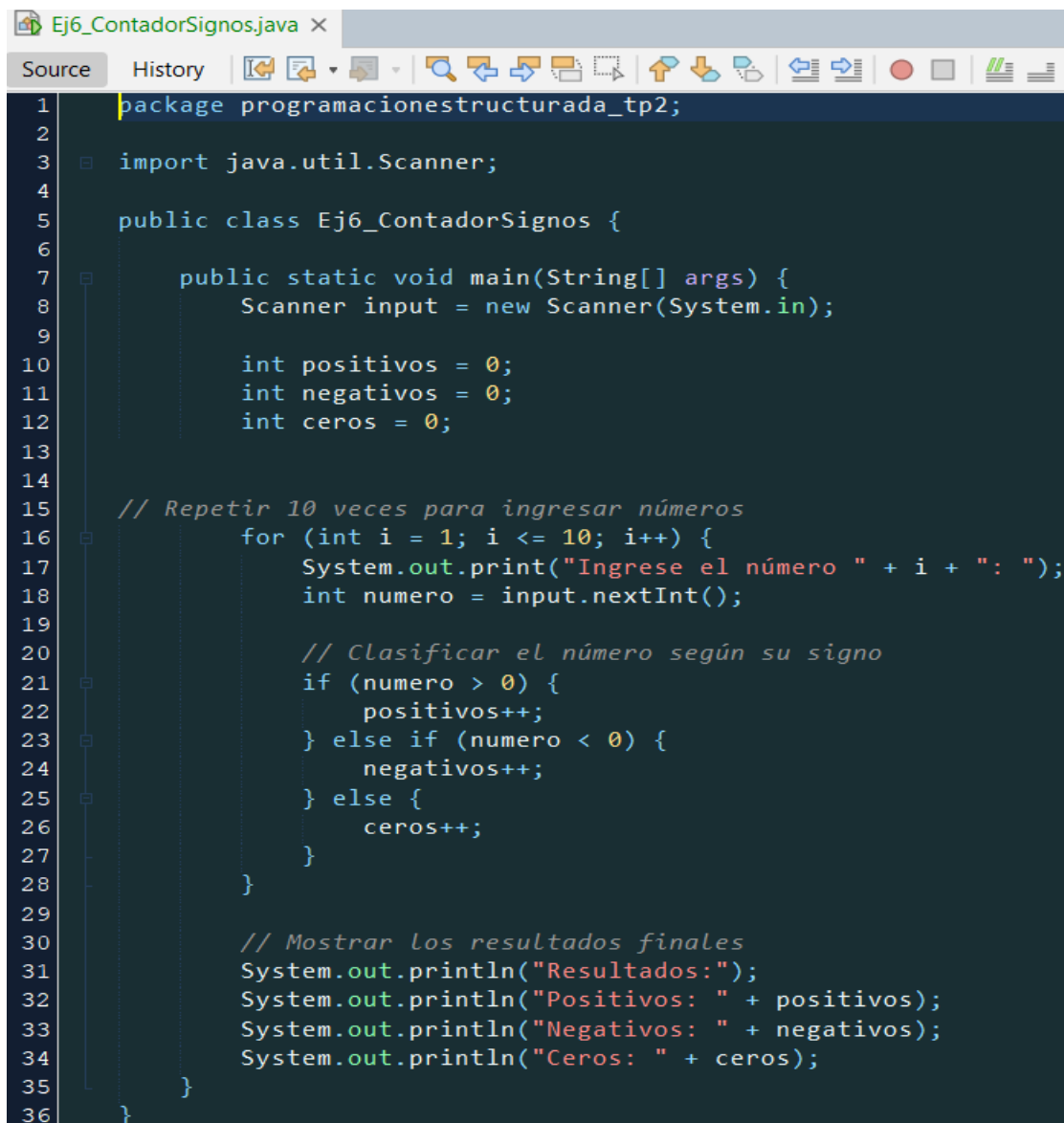
Ingrese el número 9: 4 Ingrese el número 10: -8

Resultados:

Positivos: 4

Negativos: 4

Ceros: 2



```
1 package programacionestructurada_tp2;
2
3 import java.util.Scanner;
4
5 public class Ej6_ContadorSignos {
6
7     public static void main(String[] args) {
8         Scanner input = new Scanner(System.in);
9
10        int positivos = 0;
11        int negativos = 0;
12        int ceros = 0;
13
14        // Repetir 10 veces para ingresar números
15        for (int i = 1; i <= 10; i++) {
16            System.out.print("Ingrese el número " + i + ": ");
17            int numero = input.nextInt();
18
19            // Clasificar el número según su signo
20            if (numero > 0) {
21                positivos++;
22            } else if (numero < 0) {
23                negativos++;
24            } else {
25                ceros++;
26            }
27        }
28
29        // Mostrar los resultados finales
30        System.out.println("Resultados:");
31        System.out.println("Positivos: " + positivos);
32        System.out.println("Negativos: " + negativos);
33        System.out.println("Ceros: " + ceros);
34    }
35 }
36 }
```

## 7. Validación de Nota entre 0 y 10 (do-while).

Escribe un programa que solicite al usuario una nota entre 0 y 10. Si el usuario ingresa un número fuera de este rango, debe seguir pidiéndole la nota hasta que ingrese un valor válido.

### Ejemplo de entrada/salida:

Ingrese una nota (0-10): 15

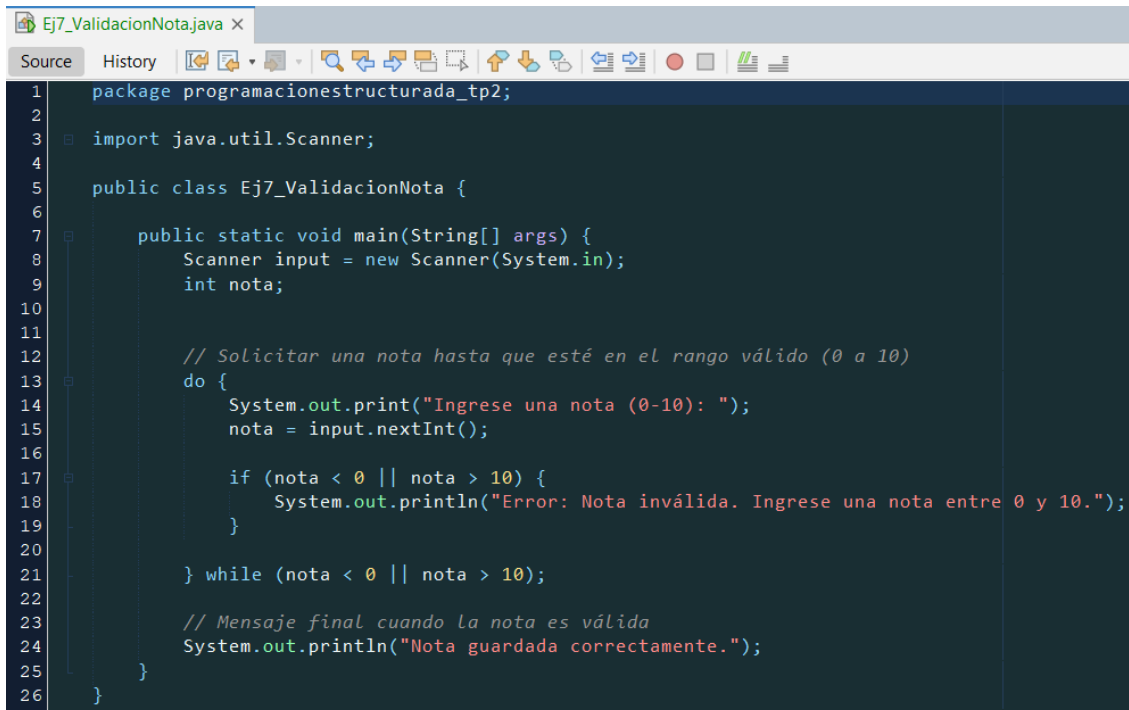
Error: Nota inválida. Ingrese una nota entre 0 y 10.

Ingrese una nota (0-10): -2

Error: Nota inválida. Ingrese una nota entre 0 y 10.

Ingrese una nota (0-10): 8

Nota guardada correctamente.



```
1 package programacionestructurada_tp2;
2
3 import java.util.Scanner;
4
5 public class Ej7_ValidacionNota {
6
7     public static void main(String[] args) {
8         Scanner input = new Scanner(System.in);
9         int nota;
10
11         // Solicitar una nota hasta que esté en el rango válido (0 a 10)
12         do {
13             System.out.print("Ingrese una nota (0-10): ");
14             nota = input.nextInt();
15
16             if (nota < 0 || nota > 10) {
17                 System.out.println("Error: Nota inválida. Ingrese una nota entre 0 y 10.");
18             }
19
20         } while (nota < 0 || nota > 10);
21
22         // Mensaje final cuando la nota es válida
23         System.out.println("Nota guardada correctamente.");
24     }
25 }
26
```

### Funciones:

## 8. Cálculo del Precio Final con impuesto y descuento.

Crea un método **calcularPrecioFinal(double impuesto, double descuento)** que calcule el precio final de un producto en un e-commerce. La fórmula es:

**PrecioFinal = PrecioBase + (PrecioBase×Impuesto) – (PrecioBase×Descuento)**

**PrecioFinal = PrecioBase + (PrecioBase \times Impuesto) - (PrecioBase \times Descuento)**



Desde main(), solicita el precio base del producto, el porcentaje de impuesto y el porcentaje de descuento, llama al método y muestra el precio final.

### Ejemplo de entrada/salida:

Ingrese el precio base del producto: 100

Ingrese el impuesto en porcentaje (Ejemplo: 10 para 10%): 10

Ingrese el descuento en porcentaje (Ejemplo: 5 para 5%): 5

El precio final del producto es: 105.0

```
Ej8_PrecioFinalImpuestoDescuento.java x
Source History
1 package programacionestructurada_tp2;
2
3 import java.util.Scanner;
4
5 public class Ej8_PrecioFinalImpuestoDescuento {
6
7     public static void main(String[] args) {
8         Scanner input = new Scanner(System.in);
9
10        // Solicitar los valores necesarios al usuario
11        System.out.print("Ingrese el precio base del producto: ");
12        double precioBase = input.nextDouble();
13
14        System.out.print("Ingrese el impuesto en porcentaje (Ejemplo: 10 para 10%): ");
15        double impuesto = input.nextDouble();
16
17        System.out.print("Ingrese el descuento en porcentaje (Ejemplo: 5 para 5%): ");
18        double descuento = input.nextDouble();
19
20        // Calcular el precio final usando el método
21        double precioFinal = calcularPrecioFinal(precioBase, impuesto, descuento);
22
23        // Mostrar el resultado
24        System.out.println("El precio final del producto es: " + precioFinal);
25    }
26
27    //Método que aplica la fórmula para calcular el precio final
28    public static double calcularPrecioFinal(double precioBase, double impuesto, double descuento) {
29        double impuestoDecimal = impuesto / 100;
30        double descuentoDecimal = descuento / 100;
31
32        return precioBase + (precioBase * impuestoDecimal) - (precioBase * descuentoDecimal);
33    }
34 }
```

9. Composición de funciones para calcular costo de envío y total de compra.

a. **calcularCostoEnvio(double peso, String zona)**: Calcula el costo de envío basado en la zona de envío (Nacional o Internacional) y el peso del paquete.

Nacional: \$5 por kg

Internacional: \$10 por kg

b. **calcularTotalCompra(double precioProducto, double costoEnvio)**: Usa **calcularCostoEnvio** para sumar el costo del producto con el costo de envío.

Desde **main()**, solicita el peso del paquete, la zona de envío y el precio del producto. Luego, muestra el total a pagar.

### Ejemplo de entrada/salida:

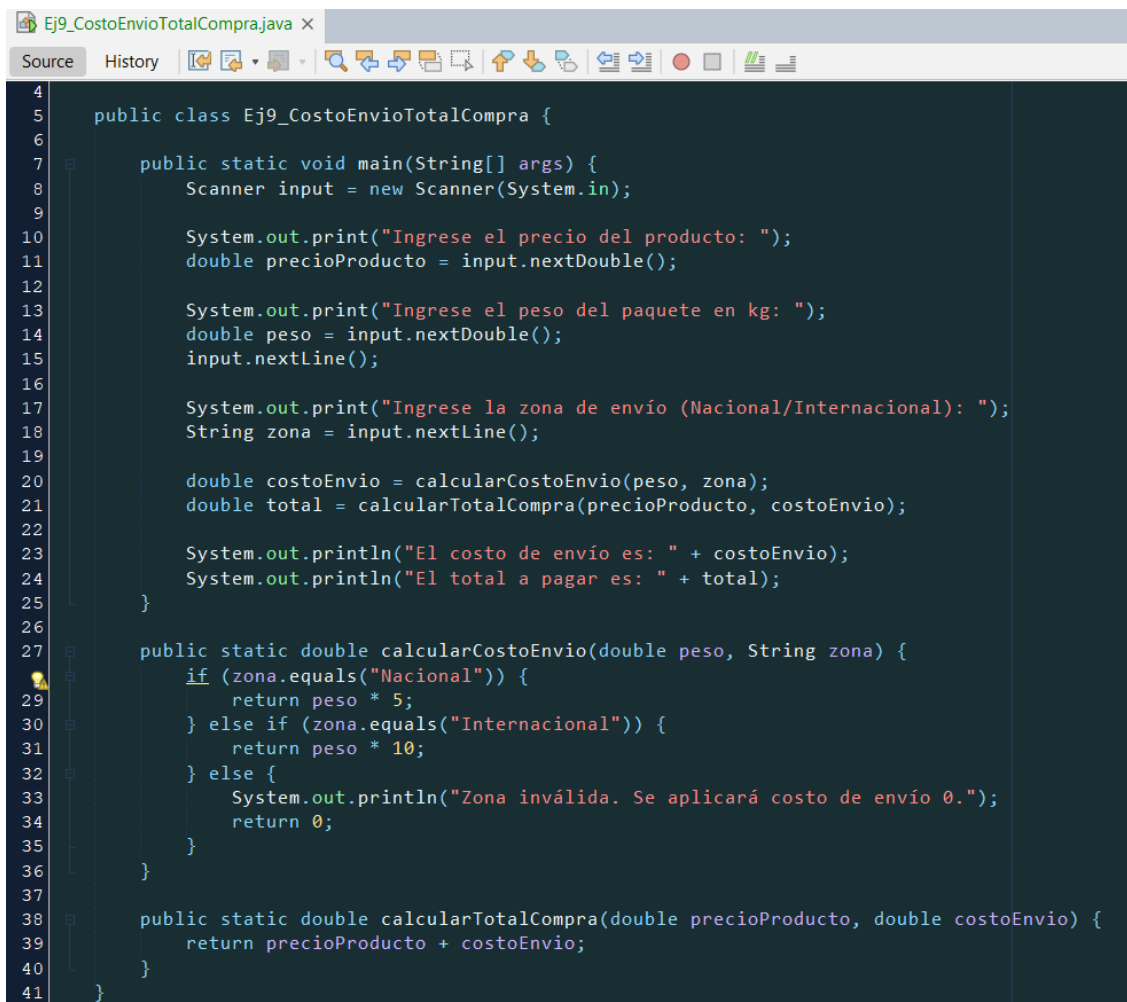
Ingrese el precio del producto: 50

Ingrese el peso del paquete en kg: 2

Ingrese la zona de envío (Nacional/Internacional): Nacional

El costo de envío es: 10.0

El total a pagar es: 60.0



```
4
5 public class Ej9_CostoEnvioTotalCompra {
6
7     public static void main(String[] args) {
8         Scanner input = new Scanner(System.in);
9
10        System.out.print("Ingrese el precio del producto: ");
11        double precioProducto = input.nextDouble();
12
13        System.out.print("Ingrese el peso del paquete en kg: ");
14        double peso = input.nextDouble();
15        input.nextLine();
16
17        System.out.print("Ingrese la zona de envío (Nacional/Internacional): ");
18        String zona = input.nextLine();
19
20        double costoEnvio = calcularCostoEnvio(peso, zona);
21        double total = calcularTotalCompra(precioProducto, costoEnvio);
22
23        System.out.println("El costo de envío es: " + costoEnvio);
24        System.out.println("El total a pagar es: " + total);
25    }
26
27    public static double calcularCostoEnvio(double peso, String zona) {
28        if (zona.equals("Nacional")) {
29            return peso * 5;
30        } else if (zona.equals("Internacional")) {
31            return peso * 10;
32        } else {
33            System.out.println("Zona inválida. Se aplicará costo de envío 0.");
34            return 0;
35        }
36    }
37
38    public static double calcularTotalCompra(double precioProducto, double costoEnvio) {
39        return precioProducto + costoEnvio;
40    }
41 }
```

10. Actualización de stock a partir de venta y recepción de productos. Crea un método

**actualizarStock(int stockActual, int cantidadVendida,**

**int cantidadRecibida)**, que calcule el nuevo stock después de una venta y recepción de productos:

**NuevoStock = StockActual – CantidadVendida + CantidadRecibida**

**NuevoStock = CantidadVendida + CantidadRecibida**

Desde **main()**, solicita al usuario el stock actual, la cantidad vendida y la cantidad recibida, y muestra el stock actualizado.

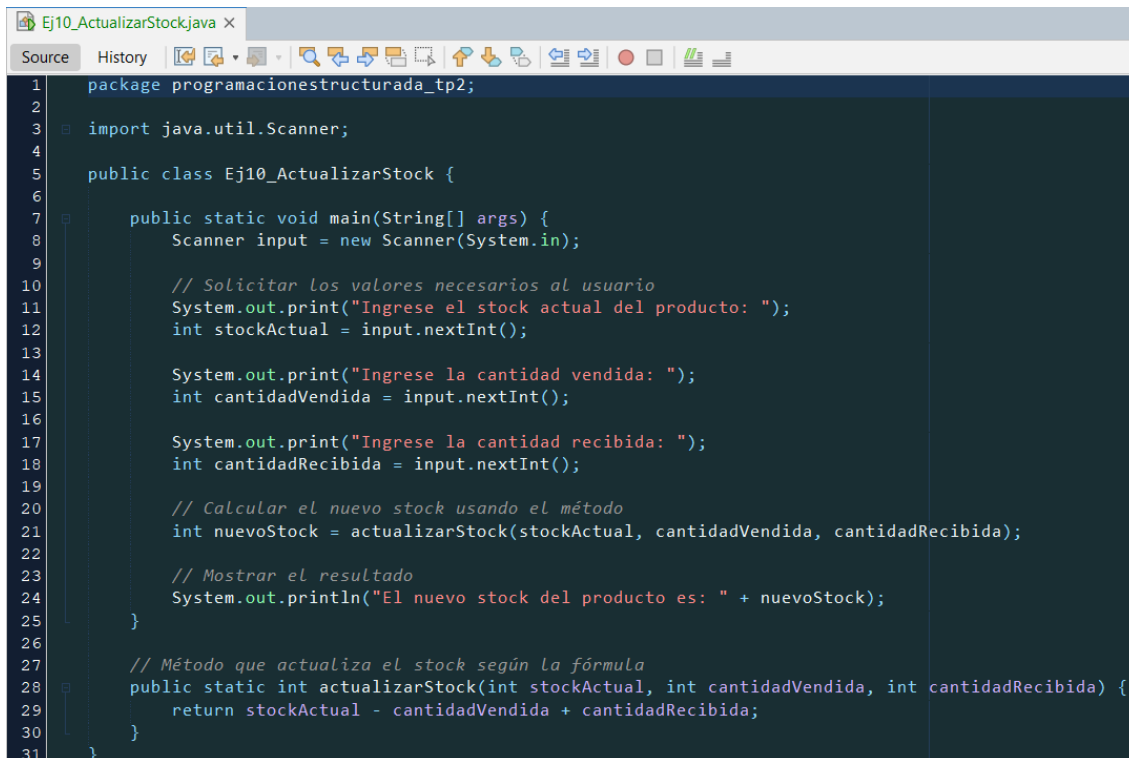
#### Ejemplo de entrada/salida:

Ingrese el stock actual del producto: 50

Ingrese la cantidad vendida: 20

Ingrese la cantidad recibida: 30

El nuevo stock del producto es: 60



```
1 package programacionestructurada_tp2;
2
3 import java.util.Scanner;
4
5 public class Ej10_ActualizarStock {
6
7     public static void main(String[] args) {
8         Scanner input = new Scanner(System.in);
9
10        // Solicitar los valores necesarios al usuario
11        System.out.print("Ingrese el stock actual del producto: ");
12        int stockActual = input.nextInt();
13
14        System.out.print("Ingrese la cantidad vendida: ");
15        int cantidadVendida = input.nextInt();
16
17        System.out.print("Ingrese la cantidad recibida: ");
18        int cantidadRecibida = input.nextInt();
19
20        // Calcular el nuevo stock usando el método
21        int nuevoStock = actualizarStock(stockActual, cantidadVendida, cantidadRecibida);
22
23        // Mostrar el resultado
24        System.out.println("El nuevo stock del producto es: " + nuevoStock);
25    }
26
27    // Método que actualiza el stock según la fórmula
28    public static int actualizarStock(int stockActual, int cantidadVendida, int cantidadRecibida) {
29        return stockActual - cantidadVendida + cantidadRecibida;
30    }
31 }
```

11. Cálculo de descuento especial usando variable global.

Declara una variable global **Ejemplo de entrada/salida:** = 0.10. Luego, crea un método **calcularDescuentoEspecial(double precio)** que use la variable global para calcular el descuento especial del 10%.

Dentro del método, declara una variable local **descuentoAplicado**, almacena el valor del descuento y muestra el precio final con descuento.

#### Ejemplo de entrada/salida:

Ingrese el precio del producto: 200

El descuento especial aplicado es: 20.0

El precio final con descuento es: 180.0

```
Ej11_DescuentoEspecialGlobal.java x
Source History
1 package programacionestructurada_tp2;
2
3 import java.util.Scanner;
4
5 public class Ej11_DescuentoEspecialGlobal {
6
7     // Variable global: porcentaje de descuento especial
8     static final double DESCUENTO_ESPECIAL = 0.10;
9
10    public static void main(String[] args) {
11        Scanner input = new Scanner(System.in);
12
13        System.out.print("Ingrese el precio del producto: ");
14        double precio = input.nextDouble();
15
16        calcularDescuentoEspecial(precio); // Llama al método para calcular el descuento
17    }
18
19    // Calcular y mostrar el descuento aplicado y el precio final
20    public static void calcularDescuentoEspecial(double precio) {
21        double descuentoAplicado = precio * DESCUENTO_ESPECIAL;
22        double precioFinal = precio - descuentoAplicado;
23
24        System.out.println("El descuento especial aplicado es: " + descuentoAplicado);
25        System.out.println("El precio final con descuento es: " + precioFinal);
26    }
27 }
```

## Arrays y Recursividad:

12. Modificación de un array de precios y visualización de resultados.

### Crea un programa que:

- Declare e inicialice un array con los precios de algunos productos.
- Muestre los valores originales de los precios.
- Modifique el precio de un producto específico.
- Muestre los valores modificados.

### Salida esperada:

Precios originales:

Precio: \$199.99

Precio: \$299.5

Precio: \$149.75

Precio: \$399.0

Precio: \$89.99

Precios modificados:

Precio: \$199.99

Precio: \$299.5

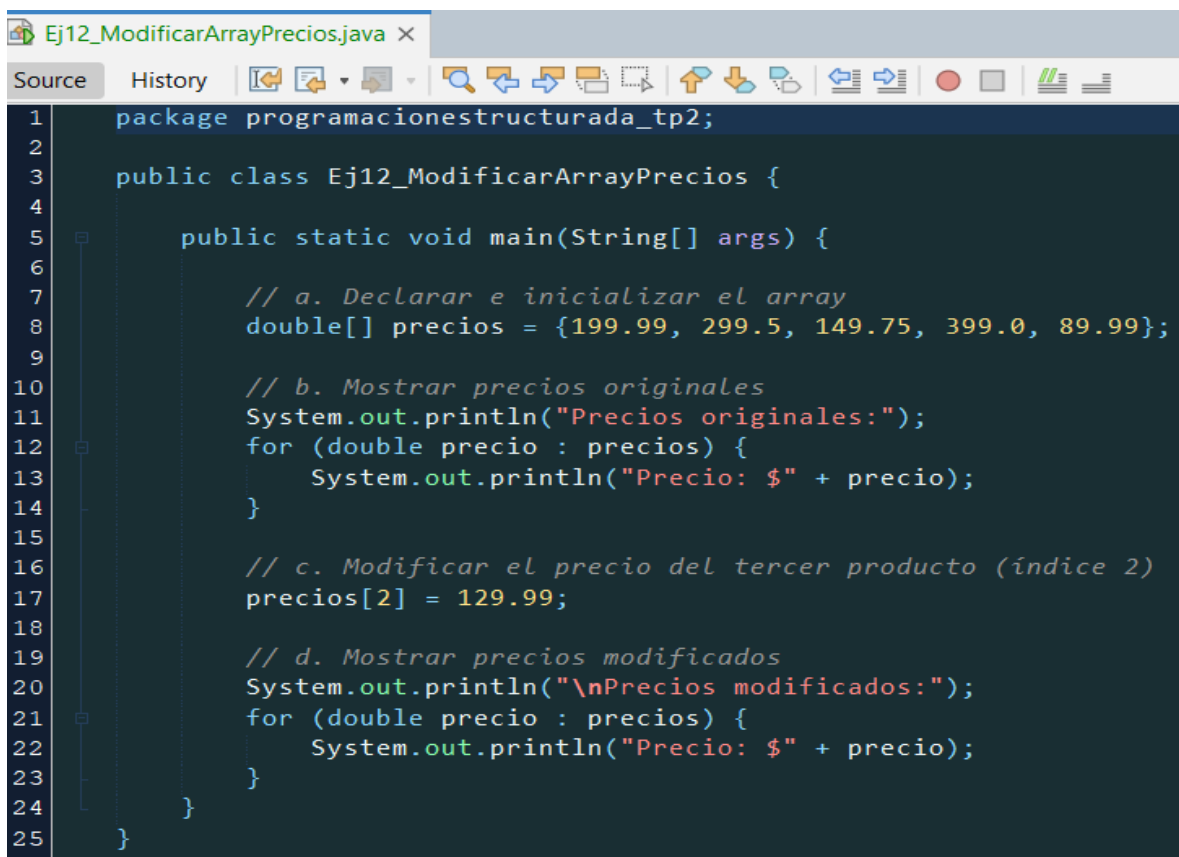
Precio: \$129.99

Precio: \$399.0

Precio: \$89.99

### Conceptos Clave Aplicados:

- ✓ Uso de arrays (double[]) para almacenar valores.
- ✓ Recorrido del array con for-each para mostrar valores.
- ✓ Modificación de un valor en un array mediante un índice.
- ✓ Reimpresión del array después de la modificación.



```
1 package programacionestructurada_tp2;
2
3 public class Ej12_ModificarArrayPrecios {
4
5     public static void main(String[] args) {
6
7         // a. Declarar e inicializar el array
8         double[] precios = {199.99, 299.5, 149.75, 399.0, 89.99};
9
10        // b. Mostrar precios originales
11        System.out.println("Precios originales:");
12        for (double precio : precios) {
13            System.out.println("Precio: $" + precio);
14        }
15
16        // c. Modificar el precio del tercer producto (índice 2)
17        precios[2] = 129.99;
18
19        // d. Mostrar precios modificados
20        System.out.println("\nPrecios modificados:");
21        for (double precio : precios) {
22            System.out.println("Precio: $" + precio);
23        }
24    }
25 }
```

13. Impresión recursiva de arrays antes y después de modificar un elemento.

**Crea un programa que:**

- Declare e inicialice un array con los precios de algunos productos.
- Use una función recursiva para mostrar los precios originales.
- Modifique el precio de un producto específico.
- Use otra función recursiva para mostrar los valores modificados.

**Salida esperada:**

Precios originales:

Precio: \$199.99

Precio: \$299.5

Precio: \$149.75

Precio: \$399.0

Precio: \$89.99

Precios modificados:

Precio: \$199.99

Precio: \$299.5

Precio: \$129.99

Precio: \$399.0

Precio: \$89.99

**Conceptos Clave Aplicados:**

- ✓ Uso de arrays (`double[]`) para almacenar valores.
- ✓ Recorrido del array con una función recursiva en lugar de un bucle.
- ✓ Modificación de un valor en un array mediante un índice.
- ✓ Uso de un índice como parámetro en la recursión para recorrer el array.

```
Ej13_ImpresionRecursivaArray.java x
Source History
1 package programacionestructurada_tp2;
2
3 public class Ej13_ImpresionRecursivaArray {
4
5     public static void main(String[] args) {
6
7         // a. Declarar e inicializar el array
8         double[] precios = {199.99, 299.5, 149.75, 399.0, 89.99};
9
10        // b. Mostrar precios originales con función recursiva
11        System.out.println("Precios originales:");
12        imprimirRecursivo(precios, 0);
13
14        // c. Modificar el precio del tercer producto (índice 2)
15        precios[2] = 129.99;
16
17        // d. Mostrar precios modificados con función recursiva
18        System.out.println("\nPrecios modificados:");
19        imprimirRecursivo(precios, 0);
20    }
21
22    // Función recursiva para imprimir los precios
23    public static void imprimirRecursivo(double[] array, int indice) {
24        if (indice < array.length) {
25            System.out.println("Precio: $" + array[indice]);
26            imprimirRecursivo(array, indice + 1);
27        }
28    }
29 }
```

## CONCLUSIONES ESPERADAS

- Aplicar estructuras de control y decisión para resolver problemas.
- Diseñar soluciones usando estructuras iterativas y condicionales.
- Modularizar el código utilizando funciones con y sin retorno.
- Utilizar arrays para almacenamiento y manipulación de datos.
- Comprender y aplicar la recursividad en casos simples.
- Trabajar con variables locales y globales de forma adecuada.
- Fortalecer la capacidad de análisis lógico y la resolución de errores.
- Consolidar el uso del lenguaje Java mediante la práctica estructurada.

ENLACE DE GITHUB: <https://github.com/GitGPais/TUPAD---P2---Java.git>