



# PROGRAMACIÓN II PROGRAMACIÓN ESTRUCTURADA

## Trabajo Práctico N°2

### Resumen

El objetivo de este trabajo es poner en práctica habilidades de programación estructurada en Java. Se desarrollan temas como operadores, estructuras de control, funciones, recursividad y estructuras de datos.

Nicolas Olima  
nicolima200@gmail.com

# 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

Concepto	Aplicación en el proyecto
Estructuras condicionales	Clasificación de edad, verificación de año bisiesto
Ciclos (for, while, do-while)	Repetición de ingreso de datos y cálculos
Funciones	Cálculo modular de descuentos, envíos, stock
Arrays	Gestión de precios de productos
Recursividad	Impresión 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.

```
10
11     package aniobisiesto;
12
13     import java.util.Scanner;
14     public class AnioBisiesto {
15     public static void main(String[] args) {
16         Scanner scan= new Scanner(System.in);
17         int anio;
18
19         System.out.println("Ingrese un año: ");
20         anio= Integer.parseInt(scan.nextLine());
21
22         if ((anio % 4 == 0 && anio % 100 != 0) || (anio % 400 == 0)){
23             System.out.println("Es bisiesto");
24         }
25         else{
26             System.out.println("No es Bisiesto");
27         }
28     }
29 }
30
```

Output - AnioBisiesto (run) x

```
run:
Ingrese un año:
2024
Es bisiesto
BUILD SUCCESSFUL (total time: 5 seconds)
```

### 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

```
8 package mayordetres;
9
10 import java.util.Scanner;
11 public class MayorDeTres {
12     public static void main(String[] args) {
13         Scanner scan = new Scanner(System.in);
14         int mayor,num,i;
15
16         System.out.print("Ingrese el 1º número: ");
17         mayor=Integer.parseInt(scan.nextLine());
18
19         for (i = 2; i <= 3; i++){
20             System.out.print("Ingrese el "+i+"º "+"número: ");
21             num = Integer.parseInt(scan.nextLine());
22
23             if (num > mayor){
24                 mayor=num;
25             }
26         }
27
28         System.out.println("El mayor es: "+ mayor);
29     }
30 }
```

Output x

Trabajos\_practicos - C:\Users\nicol\OneDrive\Documents\2025 TUPaD\2do cuatrimestre\2-

run:

Ingrese el 1º número: 45  
Ingrese el 2º número: 12  
Ingrese el 3º número: 34  
El mayor es: 45  
BUILD SUCCESSFUL (total time: 9 seconds)

### 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.

```
package clasificaedad;

import java.util.Scanner;
public class ClasificaEdad {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        int edad;

        do {
            System.out.println("Para salir ingrese un número negativo.");
            System.out.print("Ingrese su edad: ");
            edad = Integer.parseInt(scan.nextLine());

            if (0 <= edad && edad < 12){
                System.out.println("Niño");
            }else if (12 <= edad && edad <= 17){
                System.out.println("Adolescente");
            }else if (18 <= edad && edad <= 59){
                System.out.println("Adulto");
            }else if (edad > 60){
                System.out.println("Adulto mayor");
            }
        }
        while(edad >= 0);
        System.out.println("Hasta luego!");
    }
}
```

```
run:
Para salir ingrese un número negativo.
Ingrese su edad: 5
Niño
Para salir ingrese un número negativo.
Ingrese su edad: 13
Adolescente
Para salir ingrese un número negativo.
Ingrese su edad: 42
Adulto
Para salir ingrese un número negativo.
Ingrese su edad: 68
Adulto mayor
Para salir ingrese un número negativo.
Ingrese su edad: -1
Hasta luego!
BUILD SUCCESSFUL (total time: 22 seconds)
```

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

```
package descuentos;

import java.util.Scanner;
public class Descuentos {

    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        double precioOriginal, precioFinal, descuento;
        char cat;

        System.out.print("Ingrese el precio del producto: ");
        precioOriginal = Double.parseDouble(scan.nextLine().replaceAll(",", "."));

        System.out.print("Ingrese la categoría ('a', 'b' o 'c'): ");
        cat = Character.toLowerCase(scan.next().charAt(0));

        if (cat == 'a' || cat == 'b' || cat == 'c'){
            descuento = calcularDescuento(precioOriginal, cat);
            precioFinal = precioOriginal - descuento;
            System.out.println("Precio original: $ " + String.format("%.2f", precioOriginal));
            System.out.println("Descuento: $ -" + String.format("%.2f", descuento));
            System.out.println("PRECIO FINAL: $ " + String.format("%.2f", precioFinal));
        }else{
            System.out.println("La categoría es inválida");
        }
    }
}

final static double calcularDescuento(double precio, char cat) {
    double descuento=0;

    switch (cat) {
        case 'a' -> descuento = precio * 0.1;
        case 'b' -> descuento = precio * 0.15;
        case 'c' -> descuento = precio * 0.2;
    }
    return descuento;
}
}
```

```
Trabajos_practicos - C:\Users\nicol\OneDrive\Documents\20
run:
Ingrese el precio del producto: 1598.50
Ingrese la categoría ('a', 'b' o 'c'): b
Precio original: $ 1598,50
Descuento: $ -239,77
PRECIO FINAL: $ 1358,73
BUILD SUCCESSFUL (total time: 5 seconds)
```

## 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

```
package sumapares;

import java.util.Scanner;
public class SumaPares {

    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        double num,suma =0;

        do {
            System.out.print("Ingresa un número: ");
            num = Double.parseDouble(scan.nextLine());

            if (num % 2 == 0){
                suma+=num;
            }

        }while (num != 0);
        System.out.println("Suma TOTAL: "+suma);
    }
}
```

Trabajos\_practicos - C:\Users\nicol\OneDrive\Documents\

```
run:
Ingresa un número: 2
Ingresa un número: 5
Ingresa un número: 7
Ingresa un número: 9
Ingresa un número: -15.0
Ingresa un número: -21
Ingresa un número: 6
Ingresa un número: -4
Ingresa un número: 0
Suma TOTAL: 4.0
BUILD SUCCESSFUL (total time: 22 seconds)
```



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

```
package contadorposnegceros;

import java.util.Scanner;
public class ContadorPosNegCeros {

    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        int num,i;
        int neg=0,pos=0,cero=0;

        for (i=1; i<=10; i++){
            System.out.print("Ingresa un número: ");
            num= Integer.parseInt(scan.nextLine());

            if (num >0){
                pos+=1;
            }else if (num<0){
                neg+=1;
            }else{
                cero+=1;
            }
        }
        System.out.println("\n");
        System.out.println("Positivos: "+pos);
        System.out.println("Negavos: "+neg);
        System.out.println("Ceros: "+cero);
    }
}
```

```
run:
Ingresa un número: 1
Ingresa un número: -4
Ingresa un número: 0
Ingresa un número: 5
Ingresa un número: -17
Ingresa un número: 8
Ingresa un número: -0
Ingresa un número: 65
Ingresa un número: -45
Ingresa un número: 0
Positivos: 4
Negavos: 3
Ceros: 3
BUILD SUCCESSFUL (total time: 11 seconds)
```

#### 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.

```
package validanota;

import java.util.Scanner;
public class ValidaNota {

    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        int nota;

        do{
            System.out.print("Ingrese una nota: ");
            nota = Integer.parseInt(scan.nextLine());
            if (nota < 0 || nota > 10){
                System.out.println("Error: Nota inválida. Ingrese una nota entre 0 y 10.");
                System.out.println("");
            }
        }while (nota < 0 || nota > 10);
        System.out.println("Nota guardada correctamente");
    }
}

run:
Ingrese una nota: 15
Error: Nota inválida. Ingrese una nota entre 0 y 10.

Ingrese una nota: -5
Error: Nota inválida. Ingrese una nota entre 0 y 10.

Ingrese una nota: 5
Nota guardada correctamente
BUILD SUCCESSFUL (total time: 9 seconds)
```

### 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:

$$\text{PrecioFinal} = \text{PrecioBase} + (\text{PrecioBase} \times \text{Impuesto}) - (\text{PrecioBase} \times \text{Descuento})$$

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

```
package impuestos;

import java.util.Scanner;
public class Impuestos {

    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        double precio, impuesto, descuento;

        System.out.print("Ingresa el precio base: ");
        precio = Double.parseDouble(scan.nextLine());

        System.out.print("Ingresa el porcentaje de impuesto: ");
        impuesto = ((Double.parseDouble(scan.nextLine()))/100) + 1;

        System.out.print("Ingresa el porcentaje de descuento: ");
        descuento = ((Double.parseDouble(scan.nextLine()))/100) + 1;

        System.out.println("PRECIO FINAL: " + calcularPrecioFinal(precio, impuesto, descuento));
    }

    final static double calcularPrecioFinal(double precioBase, double impuesto, double descuento){
        double precioFinal = precioBase + precioBase*impuesto - precioBase*descuento;
        return precioFinal;
    }
}
```

Impuestos (run) x

```
run:
Ingresa el precio base: 100
Ingresa el porcentaje de impuesto: 10
Ingresa el porcentaje de descuento: 5
PRECIO FINAL: 105.0
BUILD SUCCESSFUL (total time: 3 seconds)
```

## 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

```
import java.util.Scanner;
public class CostoEnvio {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        double peso, precio, envio, precioFinal;
        String zona;

        System.out.print("Ingrese el PRECIO del producto: ");
        precio = Double.parseDouble(scan.nextLine());

        System.out.print("Ingrese el PESO del producto: ");
        peso = Double.parseDouble(scan.nextLine());

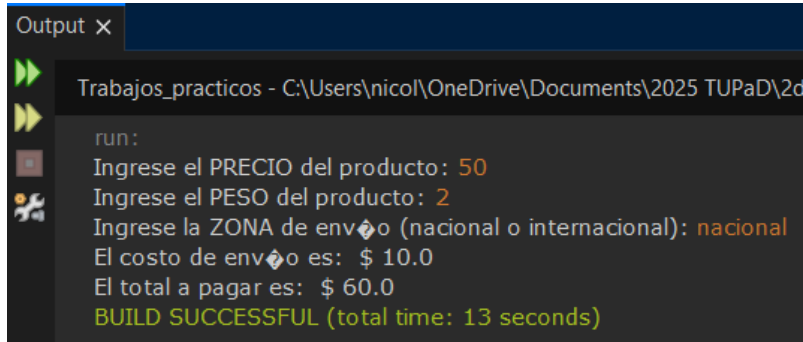
        System.out.print("Ingrese la ZONA de envío (nacional o internacional): ");
        zona = scan.nextLine();

        envio = calcularCostoEnvio(peso, zona);
        precioFinal = calcularTotalCompra(precio, envio);

        System.out.println("El costo de envío es: $" + envio);
        System.out.println("El total a pagar es: $" + precioFinal);
    } // Cierre main

    public static double calcularCostoEnvio(double peso, String zona){
        double costo;
        if (zona.equalsIgnoreCase("nacional")) {
            costo = peso * 5;
        } else if (zona.equalsIgnoreCase("internacional")) {
            costo = peso * 10;
        } else {
            System.out.println("!!! ZONA NO VÁLIDA !!!");
            return 00000.0000;
        }
        return costo;
    }

    public static double calcularTotalCompra(double precioProducto, double costoEnvio){
        double precioFinal = precioProducto + costoEnvio;
        return precioFinal;
    }
} // Cierre clase
```



```
Output x
Trabajos_practicos - C:\Users\nicol\OneDrive\Documents\2025 TUPaD\2d
run:
Ingrese el PRECIO del producto: 50
Ingrese el PESO del producto: 2
Ingrese la ZONA de envío (nacional o internacional): nacional
El costo de envío es: $ 10.0
El total a pagar es: $ 60.0
BUILD SUCCESSFUL (total time: 13 seconds)
```

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

```
package stock;

import java.util.Scanner;
public class Stock {

    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        int stockActual, stockRec, stockVen, nuevoStock;

        System.out.print("Ingrese el stock actual: ");
        stockActual = Integer.parseInt(scan.nextLine());

        System.out.print("Ingrese el stock vendido: ");
        stockVen = Integer.parseInt(scan.nextLine());

        System.out.print("Ingrese el stock recibido: ");
        stockRec = Integer.parseInt(scan.nextLine());

        nuevoStock=actualizarStock(stockActual, stockVen, stockRec);

        System.out.println("NUEVO STOCK DEL PRODUCTO: "+nuevoStock);
    }
    static int actualizarStock(int stockActual, int cantidadVendida, int cantidadRecibida){
        int nuevoStock = stockActual - cantidadVendida + cantidadRecibida;
        return nuevoStock;
    }
}
```

Trabajos\_practicos - C:\Users\nicol\OneDrive\Documents\2025 TUPaD\2do cuatrimestre\2-Programación 2\Trabajos\_practicos

run:  
Ingrese el stock actual: 50  
Ingrese el stock vendido: 30  
Ingrese el stock recibido: 02  
NUEVO STOCK DEL PRODUCTO: 22  
BUILD SUCCESSFUL (total time: 3 seconds)

#### 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

```
package descuentoespecial;

import java.util.Scanner;
public class DescuentoEspecial {

    final static double EJEMPLOENTRADASALIDA=0.10;
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        double precio, precioFinal, descuento;

        System.out.print("Ingresa el precio: ");
        precio = Double.parseDouble(scan.nextLine());

        descuento= calcularDescuentoEspecial(precio);
        precioFinal = precio - descuento;

        System.out.println("El descuento especial aplicado es: "+ descuento);
        System.out.println("El precio final con descuento es: "+ precioFinal);
    }
    static double calcularDescuentoEspecial(double precio) {
        double descuentoAplicado=precio * EJEMPLOENTRADASALIDA;
        return descuentoAplicado;
    }
}
```

Trabajos\_practicos - C:\Users\nicol\OneDrive\Documents\2025 TUPaD\2do cuatrimestre\2-Programación 2\T

run:  
Ingresa el precio: 200  
El descuento especial aplicado es: 20.0  
El precio final con descuento es: 180.0  
BUILD SUCCESSFUL (total time: 3 seconds)

## 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.

```
package arraysyrecursividad;
public class ArraysYRecursividad {
    public static void main(String[] args) {
        double precios[] = {10.99,54.10,125.00,5.45,28.99,87.90,66.00,31.50,2.00,46.75};

        System.out.println("Precios originales: ");
        mostrarPrecios(precios);

        System.out.println("");
        precios[4]=1; // Modificamos el precio del elemento de índice 4 . Nuevo valor: 1
        System.out.println("-----");

        System.out.println("Precios modificados: ");
        mostrarPrecios(precios);

        System.out.println("");
    }

    //Método que recorre el array recibido como argumento y muestra los elementos por pantalla.
    static void mostrarPrecios(double[] precios){
        for (double precio:precios){
            System.out.print("$"+precio+" ");
        }
    }
}
```

```
ArraysYRecursividad (run) x Trabajos_practicos - C:\Users\nicol\OneDrive\Documents\2
run:
Precios originales:
$10.99, $54.1, $125.0, $5.45, $28.99, $87.9, $66.0, $31.5, $2.0, $46.75,
-----
Precios modificados:
$10.99, $54.1, $125.0, $5.45, $1.0, $87.9, $66.0, $31.5, $2.0, $46.75,
BUILD SUCCESSFUL (total time: 0 seconds)
```

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.

```
package recursividad;

public class Recursividad {

    public static void main(String[] args) {
        double precios[] = {10.99,54.10,125.00,5.45,28.99,87.90,66.00,31.50,2.00,46.75};

        System.out.println("Precios originales: ");
        mostrarPrecios(precios, 0);
        System.out.println("");

        precios[9] = 0.0;

        System.out.println("Precios modificados: ");
        mostrarPrecios(precios, 0);

    }

    static void mostrarPrecios(double[] precios, int i){
        if (i == precios.length){
            return ;
        }
        System.out.println("Precio: $" + precios[i]);

        mostrarPrecios(precios, i+1);
    }
}
```

```
it - Recursividad (run) x
run:
Precios originales:
Precio: $10.99
Precio: $54.1
Precio: $125.0
Precio: $5.45
Precio: $28.99
Precio: $87.9
Precio: $66.0
Precio: $31.5
Precio: $2.0
Precio: $46.75

Precios modificados:
Precio: $10.99
Precio: $54.1
Precio: $125.0
Precio: $5.45
Precio: $28.99
Precio: $87.9
Precio: $66.0
Precio: $31.5
Precio: $2.0
Precio: $0.0
BUILD SUCCESSFUL (total time: 0 seconds)
```

## 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.