

# PROGRAMACIÓN II

## Trabajo Práctico 2: Programación Estructurada

**Alumna: Astrid Ayelen Añazco**

**Link repo GitHub: [LINK](#)**

### 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.

```
public class Punto01 {  
    public static void main(String[] args) {  
        Scanner scan = new Scanner(System.in);  
  
        System.out.println("Ingrese un año: ");  
        int anio = scan.nextInt();  
  
        if (esBisiesto(anio)) {  
            System.out.println(";" + anio + " es Bisiesto!");  
        } else {  
            System.out.println(anio + " no es Bisiesto.");  
        }  
    } //cierre de main  
  
    static boolean esBisiesto(int n) {  
        boolean esMultiploDe4 = n % 4 == 0;  
        boolean esMultiploDe100 = n % 100 == 0;  
        boolean esMultiploDe400 = n % 400 == 0;  
        boolean esExcepcion = esMultiploDe100 && !esMultiploDe400;  
  
        return esMultiploDe4 && !esExcepcion;  
  
        /* tambien puedo simplificar la funcion de esta manera:  
        static boolean esBisiesto(int n) {  
            return (n % 4 == 0) && (n % 100 != 0 || n % 400 == 0);  
        }  
        */  
    }  
}
```

**Ejemplo de entrada/salida:**

Ingrese un año: 2024

El año 2024 es bisiesto.

```
Ingrese un anio:  
2024  
? 2024 es Bisiesto!  
BUILD SUCCESSFUL (total time: 8 seconds)
```

Ingrese un año: 1900

El año 1900 no es bisiesto.

```
run:  
Ingrese un anio:  
1900  
1900 no 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.

```
public class Punto02 {  
    public static void main(String[] args) {  
        int num1, num2, num3;  
  
        Scanner input = new Scanner(System.in);  
  
        System.out.print("Ingrese el primer numero: ");  
        num1 = input.nextInt();  
        System.out.print("Ingrese el segundo numero: ");  
        num2 = input.nextInt();  
        System.out.print("Ingrese el tercer numero: ");  
        num3 = input.nextInt();  
  
        // verifico las opciones que se podrian dar  
        if (num1 == num2 && num2 == num3) {  
            System.out.println("Los tres números son iguales: " + num1);  
        } else if (num1 >= num2 && num1 >= num3) {  
            System.out.println("El mayor es: " + num1);  
        } else if (num2 >= num1 && num2 >= num3) {  
            System.out.println("El mayor es: " + num2);  
        } else {  
            System.out.println("El mayor es: " + num3);  
        }  
    }  
}
```

### Ejemplo de entrada/salida:

Ingrese el primer número: 8

Ingrese el segundo número: 12

Ingrese el tercer número: 5

El mayor es: 12

```
run:  
Ingrese el primer numero: 8  
Ingrese el segundo numero: 12  
Ingrese el tercer numero: 5  
El mayor es: 12  
BUILD SUCCESSFUL (total time: 11 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"

```
15  * @author Astrid
16  */
17  public class Ejercicio1 {
18      public static void main(String[] args) {
19          int edad;
20
21          Scanner input = new Scanner(System.in);
22
23          System.out.print("Ingrese su edad: ");
24          edad = input.nextInt();
25
26          if (edad >= 0) {
27              if (edad < 12) {
28                  System.out.println("Eres un niño");
29              } else if ( edad >= 12 && edad <= 17) {
30                  System.out.println("Eres un Adolescente");
31              } else if ( edad >= 18 && edad <= 59 ) {
32                  System.out.println("Eres un Adulto");
33              } else {
34                  System.out.println("Eres un Adulto mayor");
35              }
36          } else {
37              System.out.println("Incorrecto, ingrese un numero mayor o igual a 0");
38          }
39      }
40  }
41  }
```

#### Ejemplo de entrada/salida:

Ingrese su edad: 25

Eres un Adulto.

Ingrese su edad: 10

Eres un Niño.

```
run:
Ingrese su edad: 27
Eres un Adulto
BUILD SUCCESSFUL (total time: 6 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

```
19 public static void main(String[] args) {
20     double precio, precioDescuento;
21     char categoria;
22     String descAplicado;
23
24     Scanner input = new Scanner(System.in);
25
26     System.out.print("Ingrese el precio del producto: ");
27     precio = Double.parseDouble(input.nextLine());
28
29     System.out.print("Ingrese la categoria del producto (A, B o C): ");
30     categoria = Character.toUpperCase(input.nextLine().charAt(0));
31
32     switch (categoria) {
33     case 'A':
34         descAplicado = "10%";
35         precioDescuento = precio - ( precio * 0.10);
36         break;
37     case 'B':
38         descAplicado = "15%";
39         precioDescuento = precio - ( precio * 0.15);
40         break;
41     case 'C':
42         descAplicado = "20%";
43         precioDescuento = precio - ( precio * 0.20);
44         break;
45     default:
46         descAplicado = "Sin descuento.";
47         precioDescuento = precio;
48         break;
49     }
50
51     System.out.println("Precio Original: " + precio + "\nDescuento aplicado: " + descAplicado);
52     System.out.println("Precio final: " + precioDescuento);
53 }
54
55 }
```

### 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

```
ut - 02 - Programacion Estructurada (run)

run:
Ingrese el precio del producto: 1000
Ingrese la categoria del producto (A, B o C): b
Precio Original: 1000.0
Descuento aplicado: 15%
Precio final: 850.0
BUILD SUCCESSFUL (total time: 7 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.

```
16 public static void main(String[] args) {
17     int num, suma;
18     suma = 0;
19
20     Scanner input = new Scanner(System.in);
21     System.out.println("Este programa solo sumara numeros pares mayores a cero");
22
23     System.out.print("Ingrese un numero mayor a cero: ");
24     num = Integer.parseInt(input.nextLine());
25
26     while (num != 0) {
27         if (num % 2 == 0) {
28             suma += num;
29         }
30         System.out.print("Ingrese un numero (0 para terminar): ");
31         num = Integer.parseInt(input.nextLine());
32     }
33
34     System.out.println("La suma de los numeros pares es: " + suma);
35 }
36
37
38 }
```

#### 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

```
run:
Este programa solo sumara numeros pares mayores a cero
Ingrese un numero mayor a cero: 4
Ingrese un numero (0 para terminar): 7
Ingrese un numero (0 para terminar): 2
Ingrese un numero (0 para terminar): 0
La suma de los numeros pares es: 6
BUILD SUCCESSFUL (total time: 14 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.

```
5 public class Punto06 {
6     public static void main(String[] args) {
7         int num, positivos, negativos, ceros;
8         Scanner input = new Scanner(System.in);
9         positivos = 0;
10        negativos = 0;
11        ceros = 0;
12        for (int i = 1; i < 11; i++) {
13            System.out.print("Ingrese el numero " + i + ": ");
14            num = Integer.parseInt(input.nextLine());
15            if (num > 0) {
16                positivos += 1;
17            } else if (num < 0){
18                negativos += 1;
19            } else {
20                ceros +=1;
21            }
22        }
23        System.out.println("Resultados: \nPositivos: " + positivos);
24        System.out.println("Negativos: " + negativos + "\nCeros: " + ceros);
25    } //cierre main
26 } //cierre class
```

**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

```
Output - 02 - Programacion Estructurada (run)

run:
Ingrese el numero 1: -5
Ingrese el numero 2: 3
Ingrese el numero 3: 0
Ingrese el numero 4: -1
Ingrese el numero 5: 6
Ingrese el numero 6: 0
Ingrese el numero 7: 9
Ingrese el numero 8: -3
Ingrese el numero 9: 4
Ingrese el numero 10: -8
Resultados:
Positivos: 4
Negativos: 4
Ceros: 2
BUILD SUCCESSFUL (total time: 32 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.

```
17 public static void main(String[] args) {
18     int nota;
19     Scanner input = new Scanner(System.in);
20
21     do {
22         System.out.print("Ingrese una nota (0-10): ");
23         nota = Integer.parseInt(input.nextLine());
24         if (nota < 0 || nota > 10) {
25             System.out.println("Error: Nota inválida. Ingrese una nota entre 0 y 10.");
26         }
27     } while (nota < 0 || nota > 10);
28     System.out.print("Nota guardada correctamente.");
29 }
30 }
```

#### 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.

```
Output - 02 - Programacion Estructurada (run)
run:
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.BUILD SUCCESSFUL (total time: 17 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:

**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.

```
21 public static void main(String[] args) {
22     double precioBase, impuesto, descuento, precioFinal;
23     Scanner input = new Scanner(System.in);
24
25     System.out.print("Ingrese el precio base del producto: ");
26     precioBase = Double.parseDouble(input.nextLine());
27
28     System.out.print("Ingrese el impuesto en porcentaje (Ej 10 para 10%: ");
29     impuesto = Double.parseDouble(input.nextLine()) /100;
30
31     System.out.print("Ingrese el descuento en porcentaje (Ej 5 para 5%): ");
32     descuento = Double.parseDouble(input.nextLine()) /100;
33
34     precioFinal = calcularPrecioFinal(precioBase, impuesto, descuento);
35
36     System.out.println("El precio final del producto es: " + precioFinal);
37 } //cierre main
38
39 //funcion
40 static double calcularPrecioFinal(double precioBase, double impuesto, double descuento) {
41     double precioFinal = precioBase + (precioBase * impuesto) - (precioBase * descuento);
42
43     return precioFinal;
44 }
45
46 } //cierre class
47
```

### 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

```
Output - 02 - Programacion Estructurada (run)
run:
Ingrese el precio base del producto: 100
Ingrese el impuesto en porcentaje (Ej 10 para 10%: 10
Ingrese el descuento en porcentaje (Ej 5 para 5%): 5
El precio final del producto es: 105.0
BUILD SUCCESSFUL (total time: 13 seconds)
```

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

```
public class Punto05 {  
    public static void main(String[] args) {  
        double peso, precioProducto, costoEnvio, precioFinal;  
        String zona;  
        Scanner input = new Scanner(System.in);  
  
        System.out.print("Ingrese el precio del producto: ");  
        precioProducto = Double.parseDouble(input.nextLine());  
  
        System.out.print("Ingrese el peso del paquete: ");  
        peso = Double.parseDouble(input.nextLine());  
  
        System.out.print("Ingrese la zona de envio (Nacional/Internacional): ");  
        zona = input.nextLine();  
  
        costoEnvio = calcularCostoEnvio(peso, zona);  
  
        if (costoEnvio == -1) {  
            System.out.println("Zona invalida. No se puede calcular el envio");  
        } else {  
            precioFinal = calcularTotalCompra(precioProducto, costoEnvio);  
            System.out.println("El costo de envio es: " + costoEnvio);  
            System.out.println("El total a pagar es: " + precioFinal);  
        }  
    }  
} //cierra main
```

```
static double calcularCostoEnvio(double peso, String zona){  
    double costoEnvio;  
    if (zona.equalsIgnoreCase("nacional")) {  
        costoEnvio = 5 * peso;  
    } else if (zona.equalsIgnoreCase("internacional")) {  
        costoEnvio = 10 * peso;  
    } else {  
        costoEnvio = -1;  
    }  
    return costoEnvio;  
}  
  
static double calcularTotalCompra(double precioProducto, double costoEnvio) {  
    double totalCompra = precioProducto + costoEnvio;  
    return totalCompra;  
}  
  
} //cierra class
```

### 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

```
02 - Programacion Estructurada (run) x UTN-TUPaD-P2 - C:\Users\Astrid\OneDrive\Escritorio\UNI\UTN\Programacion II\Re
run:
Ingrese el precio del producto: 50
Ingrese el peso del paquete: 2
Ingrese la zona de envio (Nacional/Internacional): nacional
El costo de envio es: 10.0
El total a pagar es: 60.0
BUILD SUCCESSFUL (total time: 11 seconds)
```

```
run:
Ingrese el precio del producto: 55
Ingrese el peso del paquete: 3
Ingrese la zona de envio (Nacional/Internacional): papa
Zona invalida. No se puede calcular el envio
BUILD SUCCESSFUL (total time: 11 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.

```
public class Ejemplo {  
    public static void main(String[] args) {  
        int stockActual, cantVendida, cantRecibida, stockActualizado;  
        Scanner input = new Scanner(System.in);  
  
        System.out.print("Ingrese el stock actual del producto: ");  
        stockActual = Integer.parseInt(input.nextLine());  
  
        System.out.print("Ingrese la cantidad vendida: ");  
        cantVendida = Integer.parseInt(input.nextLine());  
  
        System.out.print("Ingrese la cantidad recibida: ");  
        cantRecibida = Integer.parseInt(input.nextLine());  
  
        stockActualizado = actualizarStock(stockActual, cantVendida, cantRecibida);  
        System.out.println("El nuevo stock del producto es: " + stockActualizado);  
    } // cierre main  
    static int actualizarStock(int stockActual, int cantVendida, int cantRecibida) {  
        int nuevoStock = stockActual - cantVendida + cantRecibida;  
        return nuevoStock;  
    }  
} //cierre class
```

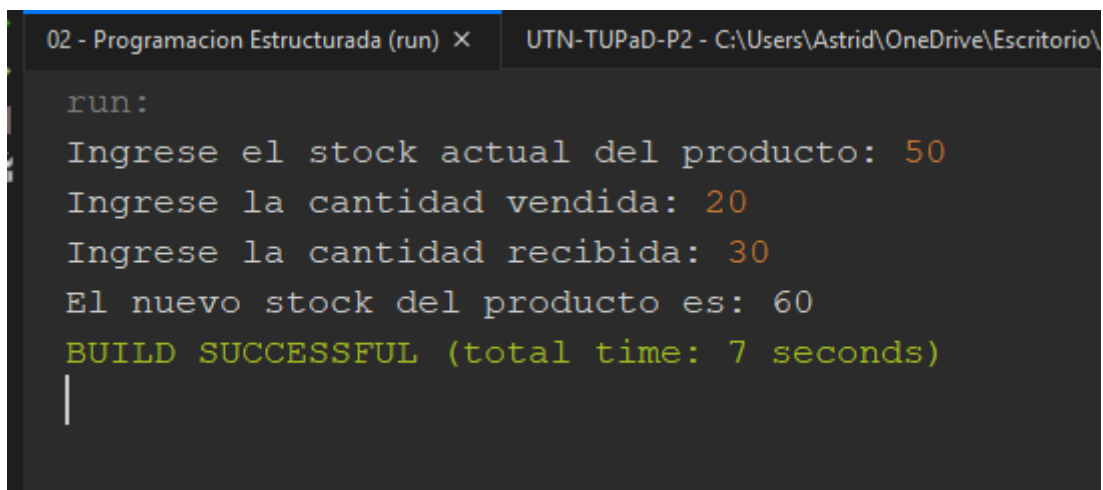
#### Ejemplo de entrada/salida:

Ingrese el stock actual del producto: 50

Ingrese la cantidad vendida: 20

Ingrese la cantidad recibida: 30

El nuevo stock del producto es: 60



```
02 - Programacion Estructurada (run) x UTN-TUPaD-P2 - C:\Users\Astrid\OneDrive\Escritorio\
run:
Ingrese el stock actual del producto: 50
Ingrese la cantidad vendida: 20
Ingrese la cantidad recibida: 30
El nuevo stock del producto es: 60
BUILD SUCCESSFUL (total time: 7 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.

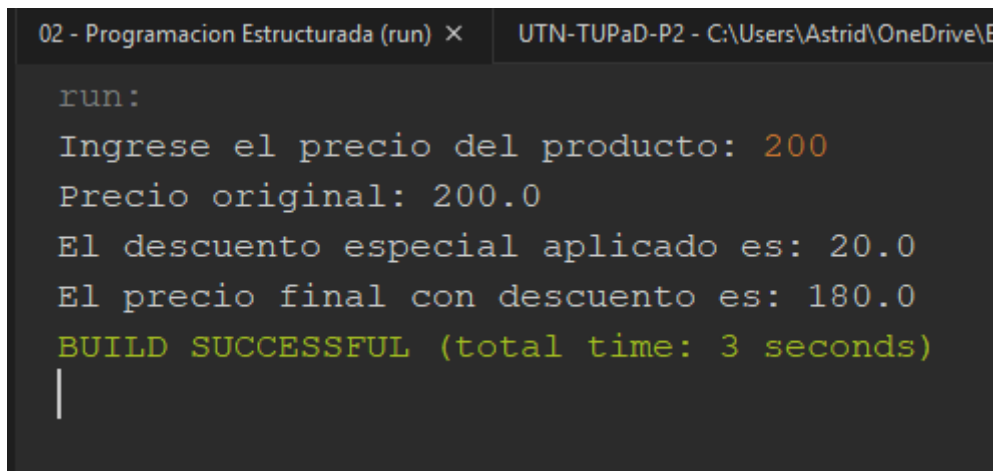
```
public class Punto1 {  
    static double descuento = 0.10; //variable global  
  
    public static void main(String[] args) {  
        Scanner input = new Scanner(System.in);  
  
        System.out.print("Ingrese el precio del producto: ");  
        double precio = input.nextDouble();  
  
        calcularDescuentoEspecial(precio);  
    } //cierre main  
  
    static void calcularDescuentoEspecial(double precio) {  
        double descuentoAplicado = precio * descuento; //variable local  
        double precioFinal = precio - descuentoAplicado;  
  
        System.out.println("Precio original: " + precio);  
        System.out.println("El descuento especial aplicado es: " + descuentoAplicado);  
        System.out.println("El precio final con descuento es: " + precioFinal);  
    }  
} // cierre class
```

#### 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



```
02 - Programacion Estructurada (run) X  UTN-TUPaD-P2 - C:\Users\Astrid\OneDrive\B  
run:  
Ingrese el precio del producto: 200  
Precio original: 200.0  
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:**

- a. Declare e inicialice un array con los precios de algunos productos.



- b. Muestre los valores originales de los precios.
- c. Modifique el precio de un producto específico.
- d. Muestre los valores modificados.

```
public class Puntol3 {  
    public static void main(String[] args) {  
        //declaro e inicio un array  
        double[] precios = {199.99, 299.5, 149.75, 399.0, 89.99};  
  
        //precios originales  
        System.out.println("Precios originales: ");  
  
        //for-each  
        for (double i : precios) {  
            System.out.println("Precio: $" + i);  
        }  
  
        //modifico el precio de un producto  
        precios[2] = 129.99;  
  
        //precios modificados  
        System.out.println("Precios modificados: ");  
        for (double i : precios) {  
            System.out.println("Precios modificados: $" + i);  
        }  
    }  
}
```

**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

```
02 - Programacion Estructurada (run) ×  UTN-TUPaD-P2 - C:\Users\Astrid\OneDrive\Escritorio
run:
Precios originales:
Precio: $199.99
Precio: $299.5
Precio: $149.75
Precio: $399.0
Precio: $89.99
Precios modificados:
Precios modificados: $199.99
Precios modificados: $299.5
Precios modificados: $129.99
Precios modificados: $399.0
Precios modificados: $89.99
BUILD SUCCESSFUL (total time: 0 seconds)
```

### 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.

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.

```
public class Punto13 {  
    public static void main(String[] args) {  
        double[] precios = {199.99, 299.5, 149.75, 399.0, 89.99};  
  
        System.out.println("Precios originales: ");  
        imprimirArray(precios, 0);  
  
        // modificar precio  
        precios[2] = 129.99;  
  
        System.out.println("\nPrecios modificados: ");  
        imprimirArray(precios, 0);  
    }  
  
    // funcion recursiva  
    static void imprimirArray(double[] precio, int indice) {  
        if (indice == precio.length) {  
            return; // caso base  
        }  
        System.out.println("Precio: $" + precio[indice]);  
        imprimirArray(precio, indice + 1); // paso recursivo  
    }  
}
```

#### 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

```
02 - Programacion Estructurada (run) × UTN-TUPaD-P2 - C:\Users\Astrid\OneDrive\Escritorio\UNI\UT
run:
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
BUILD SUCCESSFUL (total time: 0 seconds)
```

#### 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.

## 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.