



# PROGRAMACIÓN II

## Trabajo Práctico 2: Programación Estructurada

Alumno: Victor Barroeta

Comisión: M2025 – 2

Matricula: 100393

Link de Github:

[https://github.com/victormbar/UTN\\_Prog2\\_VictorBarroeta/tree/main/javaUTN\\_TPs/src/TPs\\_JAVA](https://github.com/victormbar/UTN_Prog2_VictorBarroeta/tree/main/javaUTN_TPs/src/TPs_JAVA)

### 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:

Ingresé un año: 2024

El año 2024 es bisiesto.

Ingresé un año: 1900

El año 1900 no es bisiesto.

```
public static void main(String[] args) {
    System.out.println("1-----");
    Scanner sc = new Scanner(System.in);
    System.out.println("Programa de años Bisiestos");
    System.out.println("Ingrese un año: ");
    int year = Integer.parseInt(sc.nextLine());
    if (year % 4 == 0 && year % 100 != 0 || year % 400 == 0) {
        System.out.println("El año " + year + " es bisiesto");
    } else {
        System.out.println("El año " + year + " no es bisiesto");
    }
    System.out.println("2-----");
}
```

Ps\_JAVA.TP2 > main >

- UTN\_Prog2\_VictorBarroeta (run) < Git - UTN\_Prog2\_VictorBarroeta []

run:

1-----  
Programa de años Bisiestos  
Ingrese un año:  
2024  
El año 2024 es bisiesto  
2-----

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

### 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.

The screenshot shows an IDE interface with a Java code editor and a terminal window. The code is a simple program that prompts the user for their age and prints a classification based on the input. The terminal window shows the program's output when run with the input '25'.

```
//  
//  
System.out.println("3-----");  
/* Escribe un programa en Java que solicite al usuario su edad y clasifique su etapa de vida según la siguiente tabla:  
*/  
  
Scanner input = new Scanner(System.in);  
  
System.out.println("Clasificacion de edades segun el usuario:");  
System.out.println("Ingrese su edad: ");  
int edad = Integer.parseInt(input.nextLine());  
  
if (edad >= 0 && edad < 12) {  
    System.out.println("Eres un/a niño/a");  
} else if (edad >= 12 && edad <= 17) {  
    System.out.println("Eres Adolescente");  
} else if (edad >= 18 && edad <= 59) {  
    System.out.println("Eres Adulto/a");  
} else { System.out.println("Eres Adulto/a mayor");  
}  
  
TPs_JAVA_TP2 > main >  
put - UTN_Prog2_VictorBarroeta (run) × Git - UTN_Prog2_VictorBarroeta []  
-----  
3-----  
Clasificacion de edades segun el usuario:  
Ingrese su edad:  
25  
Eres Adulto/a  
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

**Ejemplo de entrada/salida:**

Ingresé el precio del producto: 1000

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

Descuento aplicado: 15%

Precio final: 850.0

The screenshot shows an IDE interface with a Java file named TP2.java open. The code implements a switch statement to calculate a discount based on the category (A, B, or C) and prints the original price, discount amount, and final price. The output window shows a sample run where the user inputs a price of 150 and category 'B', resulting in a final price of \$127.5.

```
TP2.java
Source History
81 System.out.println("Calculadora de Descuento según categoría: ");
82
83 System.out.print("Ingresé el precio del producto: ");
84 double precio = Double.parseDouble(input.nextLine());
85
86 System.out.print("Ingresé su categoría (A,B o C) ");
87 String cat = input.nextLine();
88
89 String catMayus = cat.toUpperCase();
90
91 double precioDes;
92
93 switch (catMayus) {
94     case "A" -> precioDes = 0.10;
95     case "B" -> precioDes = 0.15;
96     case "C" -> precioDes = 0.20;
97     default -> {
98         System.out.println("Categoria no valida. No se aplica descuento.");
99         precioDes = 0.0;
100    }
101 }
102 System.out.println("- El precio original es: " + precio);
103 System.out.println("- Descuento aplicado es: " + (precioDes * 100) + "%");
104 System.out.println("- El precio total es $" + (precio - (precio * precioDes)));
105
106
TPs_JAVA.TP2 > main > switch(catMayus)
Output - UTN_Prog2_VictorBarroeta (run) > Git - UTN_Prog2_VictorBarroeta []
▶ Calculadora de Descuento seg n categor a:
▶ Ingrese el precio del producto:
▶ 150
▶ Ingrese su categoria (A,B o C)
▶ b
- El precio original es: 150.0
- Descuento aplicado es: 15.0%
- El precio total es $127.5
```

**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:**

Ingresar un número (0 para terminar): 4

Ingresar un número (0 para terminar): 7

Ingresar un número (0 para terminar): 2

Ingresar un número (0 para terminar): 0

La suma de los números pares es: 6

```
System.out.println("5-----");

/* Escribe un programa que solicite al usuario el precio de un producto y su categoría (A, B o C).
 */
System.out.println("Suma de Números Pares (while).");

int sumaPares = 0;

System.out.print("Ingresa un número (0 para terminar):\n");
int numero = Integer.parseInt(input.nextLine());

while (numero != 0) {
    if (numero % 2 == 0){
        sumaPares += numero;
    }
    System.out.println(" Ingresa otro número (0 para terminar): ");
    numero = Integer.parseInt(input.nextLine());
}

System.out.println("\n La suma total de los numeros pares es: " + sumaPares);
```



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:**

Ingresé el número 1: -5

Ingresé el número 2: 3

Ingresé el número 3: 0

Ingresé el número 4: -1

Ingresé el número 5: 6

Ingresé el número 6: 0

Ingresé el número 7: 9

Ingresé el número 8: -3

Ingresé el número 9: 4

Ingresé el número 10: -8

Resultados:

Positivos: 4

Negativos: 4

Ceros: 2

```
System.out.println("Contador de Positivos, Negativos y Ceros (for).");
// 1. Inicializamos los contadores ANTES del bucle
int positivos = 0;
int negativos = 0;
int ceros = 0;

System.out.println("Por favor, ingrese 10 números enteros.");

//Se usa ciclo FOR para la cant exacta de repeticiones
for (int i = 1; i <= 10; i++) {
    System.out.print("Ingresé el número " + i + ": ");
    int numero = Integer.parseInt(input.nextLine());

    // Se realiza un condicional dentro del ciclo para dejar un contador

    if (numero > 0) {
        positivos++; // '++' es un atajo para 'positivos = positivos + 1'
    } else if (numero < 0) {
        negativos++;
    } else {
        ceros++;
    }
}
System.out.println("\nResultados:");
System.out.println("Positivos: " + positivos);
System.out.println("Negativos: " + negativos);
System.out.println("Ceros: " + ceros);
```

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:**

Ingresar una nota (0-10): 15

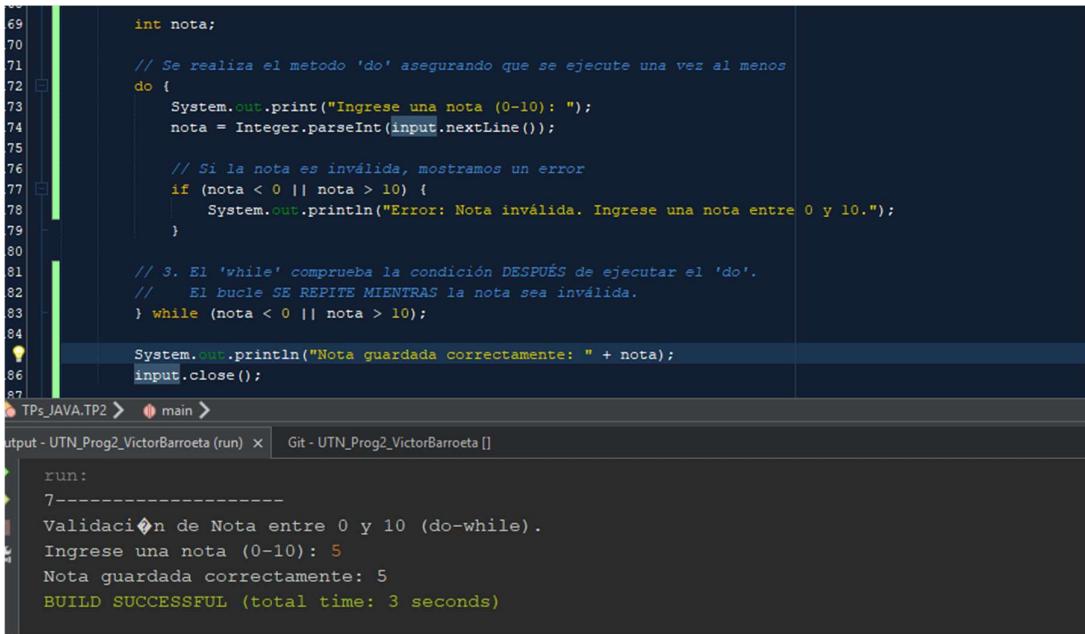
Error: Nota inválida. Ingrese una nota entre 0 y 10.

Ingresar una nota (0-10): -2

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

Ingresar una nota (0-10): 8

Nota guardada correctamente.



```
69 int nota;
70
71 // Se realiza el método 'do' asegurando que se ejecute una vez al menos
72 do {
73     System.out.print("Ingrese una nota (0-10): ");
74     nota = Integer.parseInt(input.nextLine());
75
76     // Si la nota es inválida, mostramos un error
77     if (nota < 0 || nota > 10) {
78         System.out.println("Error: Nota inválida. Ingrese una nota entre 0 y 10.");
79     }
80
81     // El 'while' comprueba la condición DESPUÉS de ejecutar el 'do'.
82     // El bucle SE REPITE MIENTRAS la nota sea inválida.
83 } while (nota < 0 || nota > 10);
84
85 System.out.println("Nota guardada correctamente: " + nota);
86 input.close();
87
```

TPs\_JAVA.TP2 > main >

run: 7-----  
Validaci n de Nota entre 0 y 10 (do-while).  
Ingrese una nota (0-10): 5  
Nota guardada correctamente: 5  
BUILD SUCCESSFUL (total time: 3 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.

## Ejemplo de entrada/salida:

Ingrese el precio base del producto: 100

Ingres 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



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:**

Ingresé el precio del producto: 50

Ingresé el peso del paquete en kg: 2

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

El costo de envío es: 10.0

El total a pagar es: 60.0

```
System.out.println("9-----");

System.out.println("Composición de funciones para calcular costo de envío y total de compra");
System.out.print("Ingresé el precio del producto: ");
double precioProducto = Double.parseDouble(input.nextLine());

System.out.print("Ingresé el peso del paquete en kg: ");
double peso = Double.parseDouble(input.nextLine());

System.out.print("Ingresé la zona de envío (Nacional/Internacional): ");
String zona = input.nextLine();

// Se llama al primer método
double costoEnvio = calcularCostoEnvio(peso, zona);

// 2. Se usa el resultado anterior para llamar al segundo método
double totalCompra = calcularTotalCompra(precioProducto, costoEnvio);

System.out.println("El costo de envío es: " + costoEnvio);
System.out.println("El total a pagar es: " + totalCompra);
input.close();
}
```



```
/*
 *9- a. Calcula el costo de envío según peso y zona.
 */
public static double calcularCostoEnvio(double peso, String zona) {
    // Usamos .equalsIgnoreCase() para aceptar "Nacional" o "nacional"
    if (zona.equalsIgnoreCase("Nacional")) {
        return peso * 5.0;
    } else if (zona.equalsIgnoreCase("Internacional")) {
        return peso * 10.0;
    } else {
        System.out.println("Zona no reconocida, costo de envío 0.");
        return 0.0; // Zona no válida
    }
}

/*
 * 9- b. Calcula el total sumando producto y envío.
 */
public static double calcularTotalCompra(double precioProducto, double costoEnvio) {
    return precioProducto + costoEnvio;
}
```

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:

$$\text{NuevoStock} = \text{StockActual} - \text{CantidadVendida} + \text{CantidadRecibida}$$

$$\text{NuevoStock} = \text{CantidadVendida} + \text{CantidadRecibida}$$

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



```
System.out.println("10-----");

System.out.println("Actualización de stock a partir de venta y recepción de productos");
System.out.print("Ingrese el stock actual del producto: ");
int stockActual = Integer.parseInt(input.nextLine());

System.out.print("Ingrese la cantidad vendida: ");
int cantidadVendida = Integer.parseInt(input.nextLine());

System.out.print("Ingrese la cantidad recibida: ");
int cantidadRecibida = Integer.parseInt(input.nextLine());

// Se llama al método con los 3 valores
int nuevoStock = actualizarStock(stockActual, cantidadVendida, cantidadRecibida);

System.out.println("El nuevo stock del producto es: " + nuevoStock);
input.close();

/**
 * 10 Calcula el nuevo stock basado en ventas y recepciones.
 */
public static int actualizarStock(int stockActual, int cantidadVendida, int cantidadRecibida) {
    // El método aplica la fórmula y devuelve el resultado
    return stockActual - cantidadVendida + cantidadRecibida;
}
```

### 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

```
// Variables globales
// Se declara 'static' para que main pueda verla.
public static final double DESCUENTO_ESPECIAL = 0.10; // 10%
```



```
System.out.println("11-----");

System.out.println("Cálculo de descuento especial usando variable global.");

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

// Se llama al método
calcularDescuentoEspecial(precio);

input.close();
}

/*
put Previous Next Select aA " * Git - UTN_Prog2_VictorBarroeta []
JAVA.TP2 > main >
- UTN_Prog2_VictorBarroeta (run) X Git - UTN_Prog2_VictorBarroeta []
run:
11-----
Cálculo de descuento especial usando variable global.
Ingrese el precio del producto: 200
El descuento especial aplicado es: 20.0
El precio final con descuento es: 180.0
BUILD SUCCESSFUL (total time: 6 seconds)
```

```
public static void calcularDescuentoEspecial(double precio) {

    // 3. El método usa la variable global DESCUENTO_ESPECIAL
    double descuentoAplicado = precio * DESCUENTO_ESPECIAL;

    // 4. 'descuentoAplicado' es una variable local, solo existe aquí
    double precioFinal = precio - descuentoAplicado;

    System.out.println("El descuento especial aplicado es: " + descuentoAplicado);
    System.out.println("El precio final con descuento es: " + precioFinal);
}
```

### 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.

**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

```
System.out.println("12-----");

System.out.println("Arrays y Recursividad:");
// a. Declarar e inicializar el array
double[] precios = {199.99, 299.5, 149.75, 399.0, 89.99};

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

// b. Recorrer y mostrar con un bucle for-each
for (double precio : precios) {
    System.out.println("Precio: $" + precio);
}

// c. Modificar un valor por su índice
// El índice 2 es el TERCER elemento
precios[2] = 129.99;

System.out.println("\nPrecios modificados:");
// d. Volver a recorrer el array
for (double precio : precios) {
    System.out.println("Precio: $" + precio);
}
```

### 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:**

- a. Declare e inicialice un array con los precios de algunos productos.
- b. Use una función recursiva para mostrar los precios originales.
- c. Modifique el precio de un producto específico.
- d. 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

```
System.out.println("13-----");

System.out.println("Impresión recursiva de arrays antes y después de modificar un elemento.");
// 1. Declarar e inicializar el array
double[] precios = {199.99, 299.5, 149.75, 399.0, 89.99};

System.out.println("Precios originales:");
// 2. Iniciar la recursión desde el indice 0
imprimirRecursivo(precios, 0);

// 3. Modificar el valor (igual que en el ej. 7)
precios[2] = 129.99;

System.out.println("\nPrecios modificados:");
// 4. Volver a iniciar la recursión desde el indice 0
imprimirRecursivo(precios, 0);
}
```



```
public static void imprimirRecursivo(double[] array, int indice) {  
  
    // CASO BASE:  
    // Si el índice es igual al largo, nos pasamos, así que terminamos.  
    if (indice >= array.length) {  
        return; // Termina la llamada actual  
    }  
  
    // CASO RECURSIVO:  
    // Imprime el elemento del índice actual  
    System.out.println("Precio: $" + array[indice]);  
  
    // Vuelve a llamar al método, pero para el SIGUIENTE índice  
    imprimirRecursivo(array, indice + 1);  
}
```

### 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.