

# PROGRAMACIÓN II

## Trabajo Práctico 2: Programación Estructurada

**Estudiante:** Roqué, Gabriel Osvaldo

**Matricula:** 101636

**Link GitHub:** <https://github.com/Ozzetas/Programacion2.git>

### 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:

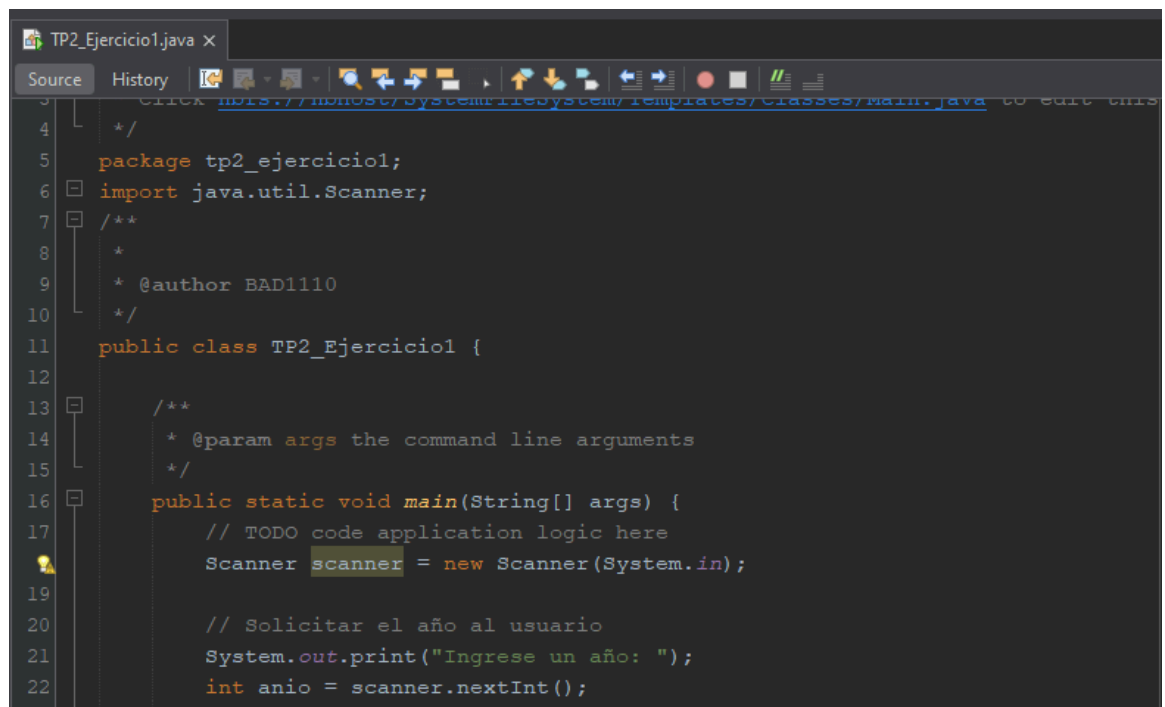
Ingrese un año: 2024

El año 2024 es

bisiesto. Ingrese un

año: 1900

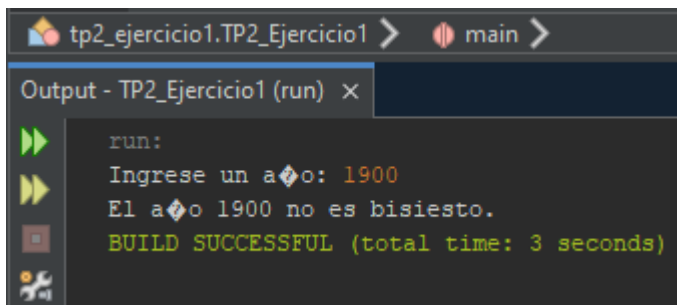
El año 1900 no es bisiesto.



```
TP2_Ejercicio1.java x
Source History
4  */
5  package tp2_ejercicio1;
6  import java.util.Scanner;
7  /**
8   *
9   * @author BAD1110
10  */
11  public class TP2_Ejercicio1 {
12
13      /**
14       * @param args the command line arguments
15       */
16      public static void main(String[] args) {
17          // TODO code application logic here
18          Scanner scanner = new Scanner(System.in);
19
20          // Solicitar el año al usuario
21          System.out.print("Ingrese un año: ");
22          int anio = scanner.nextInt();
```

```
24 // Verificar si es bisiesto
25 boolean esBisiesto = false;
26 if ((anio % 4 == 0 && anio % 100 != 0) || (anio % 400 == 0)) {
27     esBisiesto = true;
28 }
29
30 // Mostrar resultado
31 if (esBisiesto) {
32     System.out.println("El año " + anio + " es bisiesto.");
33 } else {
34     System.out.println("El año " + anio + " no es bisiesto.");
35 }
36
37 scanner.close();
38 }
39
40 }
```

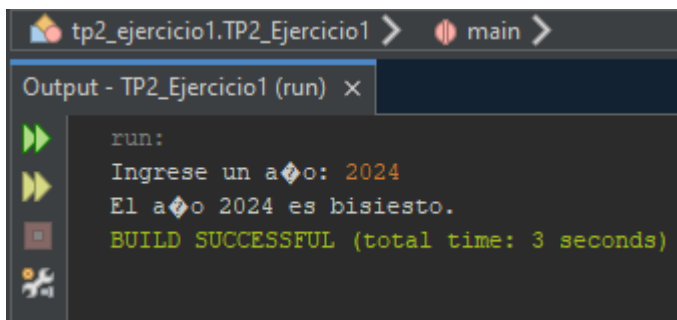
Salidas:



tp2\_ejercicio1.TP2\_Ejercicio1 > main >

Output - TP2\_Ejercicio1 (run) x

run:  
Ingrese un año: 1900  
El año 1900 no es bisiesto.  
BUILD SUCCESSFUL (total time: 3 seconds)



tp2\_ejercicio1.TP2\_Ejercicio1 > main >

Output - TP2\_Ejercicio1 (run) x

run:  
Ingrese un año: 2024  
El año 2024 es bisiesto.  
BUILD SUCCESSFUL (total time: 3 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

```
TP2_Ejercicio1.java x TP2_Ejercicio2.java x
Source History
1  /*
2   * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses
3   * Click nbfs://nbhost/SystemFileSystem/Templates/Class
4   */
5   package tp2_ejercicio2;
6   import java.util.Scanner;
7   /**
8    *
9    * @author BAD1110
10   */
11   public class TP2_Ejercicio2 {
12
13       /**
14        * @param args the command line arguments
15        */
16       public static void main(String[] args) {
17           // TODO code application logic here
18           Scanner scanner = new Scanner(System.in);
19
20           // Solicitar los tres números al usuario
21           System.out.print("Ingrese el primer número: ");
22           int num1 = scanner.nextInt();
23           System.out.print("Ingrese el segundo número: ");
24           int num2 = scanner.nextInt();
25           System.out.print("Ingrese el tercer número: ");
26           int num3 = scanner.nextInt();
27
28           // Determinar el mayor
29           int mayor = num1; // Suponemos que num1 es el mayor inicialmente
30           if (num2 > mayor) {
31               mayor = num2;
32           }
33           if (num3 > mayor) {
34               mayor = num3;
35           }
36
37           // Mostrar el resultado
38           System.out.println("El mayor es: " + mayor);
39
40           scanner.close();
41       }
```

Salidas:

```
Output - TP2_Ejercicio2 (run) x
run:
Ingrese el primer número: -3
Ingrese el segundo número: 0
Ingrese el tercer número: 2
El mayor es: 2
BUILD SUCCESSFUL (total time: 4 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](#)

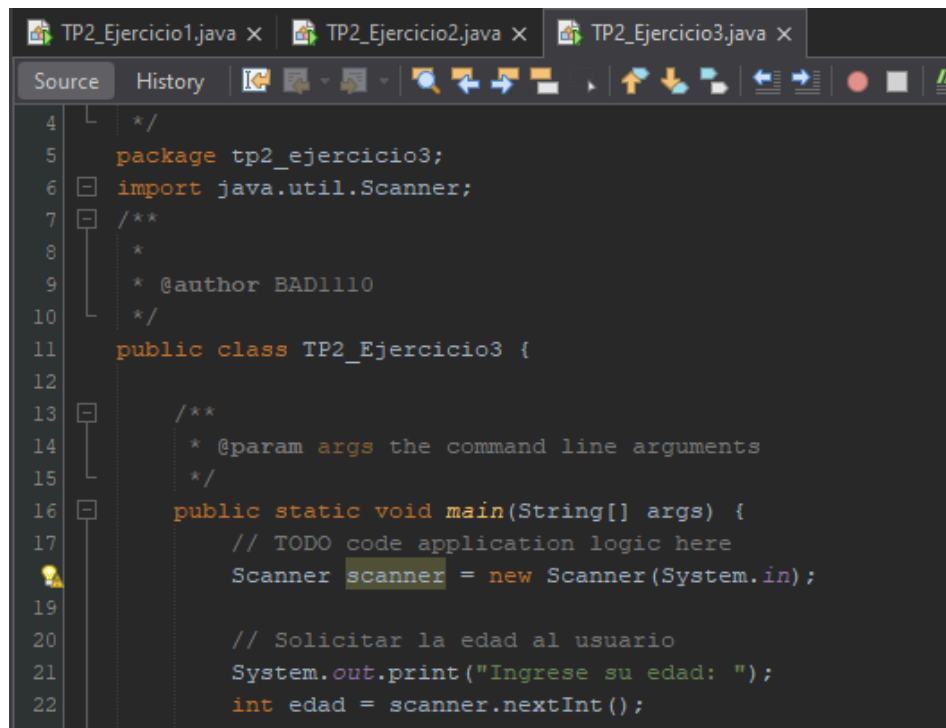
[entrada/salida](#): Ingrese su

edad: 25

Eres un Adulto.

Ingrese su edad: 10

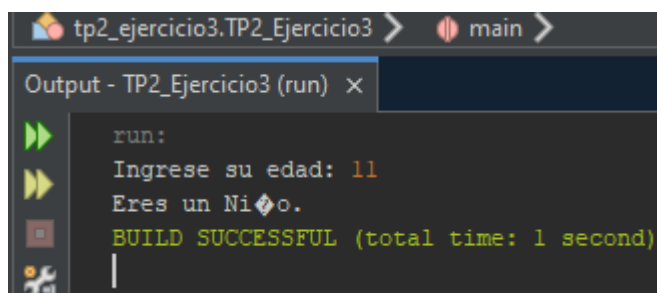
Eres un Niño.



```
4  */
5  package tp2_ejercicio3;
6  import java.util.Scanner;
7  /**
8   *
9   * @author BAD1110
10  */
11  public class TP2_Ejercicio3 {
12
13      /**
14       * @param args the command line arguments
15       */
16      public static void main(String[] args) {
17          // TODO code application logic here
18          Scanner scanner = new Scanner(System.in);
19
20          // Solicitar la edad al usuario
21          System.out.print("Ingrese su edad: ");
22          int edad = scanner.nextInt();
23      }
```

```
24 // Determinar la etapa de vida
25 String etapa;
26 if (edad < 12) {
27     etapa = "Niño";
28 } else if (edad >= 12 && edad <= 17) {
29     etapa = "Adolescente";
30 } else if (edad >= 18 && edad <= 59) {
31     etapa = "Adulto";
32 } else {
33     etapa = "Adulto mayor";
34 }
35
36 // Mostrar el resultado
37 System.out.println("Eres un " + etapa + ".");
38
39 scanner.close();
40 }
```

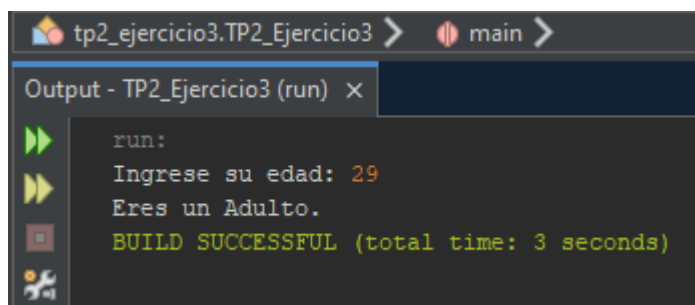
Salidas:



tp2\_ejercicio3.TP2\_Ejercicio3 > main >

Output - TP2\_Ejercicio3 (run) x

run:  
Ingrese su edad: 11  
Eres un Niño.  
BUILD SUCCESSFUL (total time: 1 second)



tp2\_ejercicio3.TP2\_Ejercicio3 > main >

Output - TP2\_Ejercicio3 (run) x

run:  
Ingrese su edad: 29  
Eres un Adulto.  
BUILD SUCCESSFUL (total time: 3 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

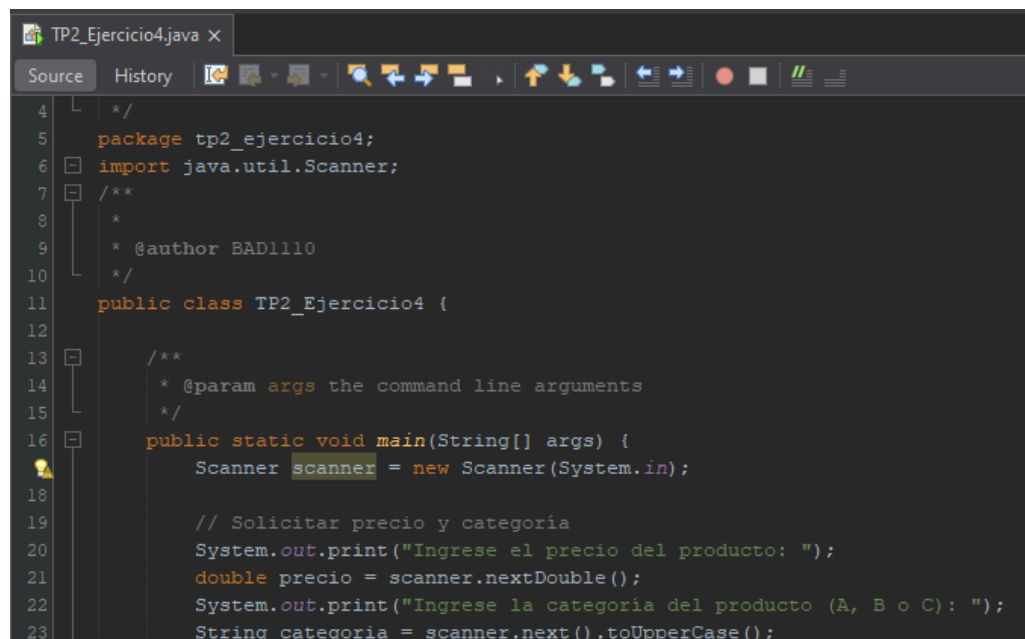
**Ejemplo de entrada/salida:**

Ingrese el precio del producto: 1000

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

Descuento aplicado: 15%

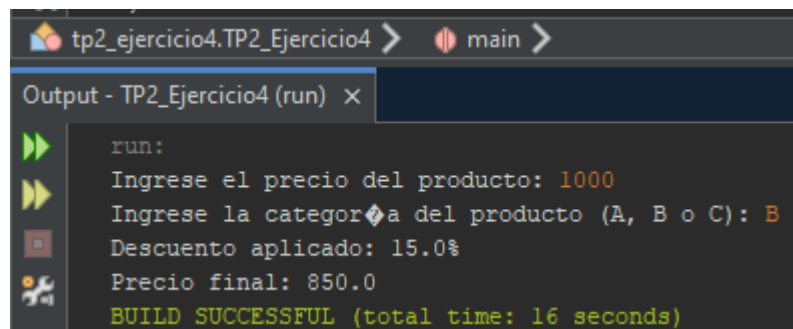
Precio final: 850.0



```
TP2_Ejercicio4.java x
Source History
4  /*
5  package tp2_ejercicio4;
6  import java.util.Scanner;
7  /**
8   *
9   * @author BAD1110
10  */
11  public class TP2_Ejercicio4 {
12
13      /**
14       * @param args the command line arguments
15       */
16      public static void main(String[] args) {
17          Scanner scanner = new Scanner(System.in);
18
19          // Solicitar precio y categoría
20          System.out.print("Ingrese el precio del producto: ");
21          double precio = scanner.nextDouble();
22          System.out.print("Ingrese la categoría del producto (A, B o C): ");
23          String categoria = scanner.next().toUpperCase();
```

```
23 String categoria = scanner.next().toUpperCase();
24
25 // Validar categoría y calcular descuento
26 double porcentajeDescuento = 0.0;
27 boolean categoriaValida = true;
28
29 switch (categoria) {
30     case "A":
31         porcentajeDescuento = 0.10; // 10%
32         break;
33     case "B":
34         porcentajeDescuento = 0.15; // 15%
35         break;
36     case "C":
37         porcentajeDescuento = 0.20; // 20%
38         break;
39     default:
40         categoriaValida = false;
41         System.out.println("Error: Categoría inválida. Debe ser A, B o C.");
42
43 }
44
45 // Calcular y mostrar resultados si la categoría es válida
46 if (categoriaValida) {
47     double descuento = precio * porcentajeDescuento;
48     double precioFinal = precio - descuento;
49
50     System.out.println("Descuento aplicado: " + (porcentajeDescuento * 100) + "%");
51     System.out.println("Precio final: " + precioFinal);
52 }
53 scanner.close();
54 }
55 }
56 }
```

Salidas:



```
tp2_ejercicio4.TP2_Ejercicio4 > main >
Output - TP2_Ejercicio4 (run) x
run:
Ingrese el precio del producto: 1000
Ingrese la categoría del producto (A, B o C): B
Descuento aplicado: 15.0%
Precio final: 850.0
BUILD SUCCESSFUL (total time: 16 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

```
TP2_Ejercicio4.java x TP2_Ejercicio5.java x
Source History
4  */
5  package tp2_ejercicio5;
6  import java.util.Scanner;
7  /**
8   *
9   * @author BAD1110
10  */
11  public class TP2_Ejercicio5 {
12
13      /**
14       * @param args the command line arguments
15       */
16      public static void main(String[] args) {
17          Scanner scanner = new Scanner(System.in);
18          int sumaPares = 0;
19          int numero;
20
21          // Ciclo para solicitar números hasta que se ingrese 0
22          do {
```

```
23              System.out.print("Ingrese un número (0 para terminar): ");
24              numero = scanner.nextInt();
25
26              // Sumar si el número es par
27              if (numero % 2 == 0 && numero != 0) {
28                  sumaPares += numero;
29              }
30          } while (numero != 0);
31
32          // Mostrar la suma total de los números pares
33          System.out.println("La suma de los números pares es: " + sumaPares);
34
35          scanner.close();
36      }
37
38  }
```

Salidas:

```
Output - TP2_Ejercicio5 (run) ×
run:
Ingrese un número (0 para terminar): 5
Ingrese un número (0 para terminar): 2
Ingrese un número (0 para terminar): 4
Ingrese un número (0 para terminar): 0
La suma de los números pares es: 6
BUILD SUCCESSFUL (total time: 9 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

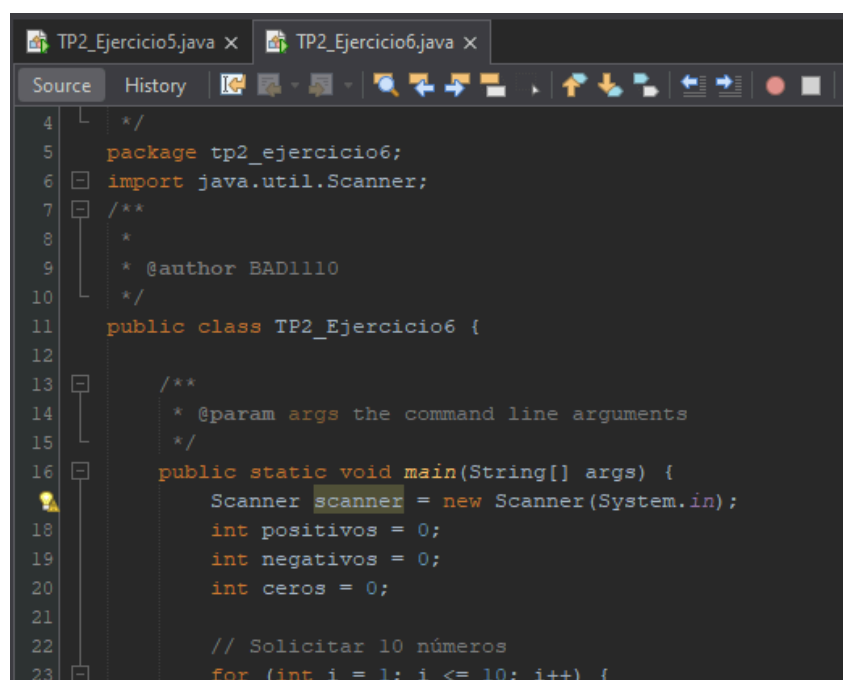
Ingrese el número 10: -8

Resultados:

Positivos: 4

Negativos: 4

Ceros: 2



```
4  /*
5  package tp2_ejercicio6;
6  import java.util.Scanner;
7  /**
8   *
9   * @author BAD1110
10  */
11  public class TP2_Ejercicio6 {
12
13      /**
14       * @param args the command line arguments
15       */
16      public static void main(String[] args) {
17          Scanner scanner = new Scanner(System.in);
18          int positivos = 0;
19          int negativos = 0;
20          int ceros = 0;
21
22          // Solicitar 10 números
23          for (int i = 1; i <= 10; i++) {
```

```
23 for (int i = 1; i <= 10; i++) {
24     System.out.print("Ingrese el número " + i + ": ");
25     int numero = scanner.nextInt();
26
27     // Contar según el valor del número
28     if (numero > 0) {
29         positivos++;
30     } else if (numero < 0) {
31         negativos++;
32     } else {
33         ceros++;
34     }
35 }
36
37 // Mostrar resultados
38
39 // Mostrar resultados
40 System.out.println("Resultados:");
41 System.out.println("Positivos: " + positivos);
42 System.out.println("Negativos: " + negativos);
43 System.out.println("Ceros: " + ceros);
44 scanner.close();
45 }
46 }
```

Salidas:

```
Output - TP2_Ejercicio6 (run) x
run:
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
BUILD SUCCESSFUL (total time: 30 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.

```
5 package tp2_ejercicio7;
6 import java.util.Scanner;
7 /**
8  *
9  * @author BAD1110
10  */
11 public class TP2_Ejercicio7 {
12
13     /**
14     * @param args the command line arguments
15     */
16     public static void main(String[] args) {
17         Scanner scanner = new Scanner(System.in);
18         double nota;
19
20         // Ciclo para validar la nota
21         do {
22             System.out.print("Ingrese una nota (0-10): ");
23             nota = scanner.nextDouble();
24
25             if (nota < 0 || nota > 10) {
26                 System.out.println("Error: Nota inválida. Ingrese una nota entre 0 y 10.");
27             }
28         } while (nota < 0 || nota > 10);
29
30         // Mostrar mensaje de confirmación
31         System.out.println("Nota guardada correctamente.");
32
33         scanner.close();
34     }
35 }
36 }
```

Salidas:

```
Output - TP2_Ejercicio7 (run) x
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: 18 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:

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

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

```
5 package tp2_ejercicio8;
6 import java.util.Scanner;
7 /**
8  *
9  * @author BAD1110
10  */
11 public class TP2_Ejercicio8 {
12
13     // Método para calcular el precio final
14     public static double calcularPrecioFinal(double precioBase, double impuesto, double descuento) {
15         return precioBase + (precioBase * impuesto) - (precioBase * descuento);
16     }
17     /**
18     * @param args the command line arguments
19     */
20     public static void main(String[] args) {
21         Scanner scanner = new Scanner(System.in);
22
23         // Solicitar datos al usuario
24         System.out.print("Ingrese el precio base del producto: ");
25         double precioBase = scanner.nextDouble();
26         System.out.print("Ingrese el impuesto en porcentaje (Ejemplo: 10 para 10%): ");
```

```
26      System.out.print("Ingrese el impuesto en porcentaje (Ejemplo: 10 para 10%): ");
27      double impuesto = scanner.nextDouble() / 100; // Convertir a decimal
28      System.out.print("Ingrese el descuento en porcentaje (Ejemplo: 5 para 5%): ");
29      double descuento = scanner.nextDouble() / 100; // Convertir a decimal
30
31      // Calcular el precio final llamando al método
32      double precioFinal = calcularPrecioFinal(precioBase, impuesto, descuento);
33
34      // Mostrar el resultado
35      System.out.println("El precio final del producto es: " + precioFinal);
36
37      scanner.close();
38  }
39
40  }
```

Salidas:

```
Output - TP2_Ejercicio8 (run) x
run:
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
BUILD SUCCESSFUL (total time: 14 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
  - 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

```
5 package tp2_ejercicio9;
6 import java.util.Scanner;
7 /**
8  *
9  * @author BAD1110
10  */
11 public class TP2_Ejercicio9 {
12
13     // Método para calcular el costo de envío
14     public static double calcularCostoEnvio(double peso, String zona) {
15         if (zona.equalsIgnoreCase("Nacional")) {
16             return peso * 5.0; // $5 por kg
17         } else if (zona.equalsIgnoreCase("Internacional")) {
18             return peso * 10.0; // $10 por kg
19         } else {
20             return -1; // Valor inválido para indicar error
21         }
22     }
23
24     // Método para calcular el total de la compra
25     public static double calcularTotalCompra(double precioProducto, double costoEnvio) {
26
27         return precioProducto + costoEnvio;
28     }
29
30     public static void main(String[] args) {
31         Scanner scanner = new Scanner(System.in);
32
33         // Solicitar datos al usuario
34         System.out.print("Ingrese el precio del producto: ");
35         double precioProducto = scanner.nextDouble();
36         System.out.print("Ingrese el peso del paquete en kg: ");
37         double peso = scanner.nextDouble();
38         System.out.print("Ingrese la zona de envío (Nacional/Internacional): ");
39         scanner.nextLine(); // Limpiar buffer
40         String zona = scanner.nextLine();
41
42         // Calcular costo de envío
43         double costoEnvio = calcularCostoEnvio(peso, zona);
44
45         // Validar zona y mostrar resultados
46         if (costoEnvio == -1) {
47             System.out.println("Error: Zona de envío inválida. Debe ser Nacional o Internacional.");
48         } else {
49             double totalCompra = calcularTotalCompra(precioProducto, costoEnvio);
50             System.out.println("El costo de envío es: " + costoEnvio);
51             System.out.println("El total a pagar es: " + totalCompra);
52         }
53
54         scanner.close();
55     }
56 }
```

Salidas:

```
Output - TP2_Ejercicio9 (run) x
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
BUILD SUCCESSFUL (total time: 16 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

```
5 package tp2_ejercicio10;
6 import java.util.Scanner;
7 /**
8  *
9  * @author BAD1110
10 */
11 public class TP2_Ejercicio10 {
12
13     // Método para actualizar el stock
14     public static int actualizarStock(int stockActual, int cantidadVendida, int cantidadRecibida) {
15         return stockActual - cantidadVendida + cantidadRecibida;
16     }
17
18     public static void main(String[] args) {
19         Scanner scanner = new Scanner(System.in);
20
21         // Solicitar datos al usuario
22         System.out.print("Ingrese el stock actual del producto: ");
23         int stockActual = scanner.nextInt();
24         System.out.print("Ingrese la cantidad vendida: ");
25         int cantidadVendida = scanner.nextInt();
26
27         int cantidadRecibida = scanner.nextInt();
28
29         // Calcular el nuevo stock
30         int nuevoStock = actualizarStock(stockActual, cantidadVendida, cantidadRecibida);
31
32         // Mostrar el resultado
33         System.out.println("El nuevo stock del producto es: " + nuevoStock);
34
35         scanner.close();
36     }
37
38 }
```

Salidas:

```
Output - TP2_Ejercicio10 (run) x
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: 10 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

```
5 package tp2_ejercicio11;
6 import java.util.Scanner;
7 /**
8  *
9  * @author BAD1110
10  */
11 public class TP2_Ejercicio11 {
12
13     // Variable global
14     private static final double DESCUENTO_FIJO = 0.10; // 10%
15
16     // Método para calcular el descuento especial
17     public static void calcularDescuentoEspecial(double precio) {
18         double descuentoAplicado = precio * DESCUENTO_FIJO; // Variable local
19         double precioFinal = precio - descuentoAplicado;
20
21         System.out.println("El descuento especial aplicado es: " + descuentoAplicado);
22         System.out.println("El precio final con descuento es: " + precioFinal);
23     }
24
25     public static void main(String[] args) {
26         Scanner scanner = new Scanner(System.in);
27     }
```

```
27
28      // Solicitar el precio al usuario
29      System.out.print("Ingrese el precio del producto: ");
30      double precio = scanner.nextDouble();
31
32      // Llamar al método para calcular y mostrar el descuento
33      calcularDescuentoEspecial(precio);
34
35      scanner.close();
36  }
37
38 }
```

Salidas:

```
Output - TP2_Ejercicio11 (run) x
run:
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: 7 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.

```
16 public static void main(String[] args) {  
17     // Declarar e inicializar el array de precios  
18     double[] precios = {199.99, 299.5, 149.75, 399.0, 89.99};  
19  
20     // Mostrar precios originales  
21     System.out.println("Precios originales:");  
22     for (double precio : precios) {  
23         System.out.println("Precio: $" + precio);  
24     }  
25  
26     // Modificar el precio en el índice 2 (tercer elemento)  
27     precios[2] = 129.99;  
28  
29     // Mostrar precios modificados  
30     System.out.println("Precios modificados:");  
31     for (double precio : precios) {  
32         System.out.println("Precio: $" + precio);  
33     }  
34 }  
35
```

Salidas:

```
Output - TP2_Ejercicio12 (run) x  
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)
```

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

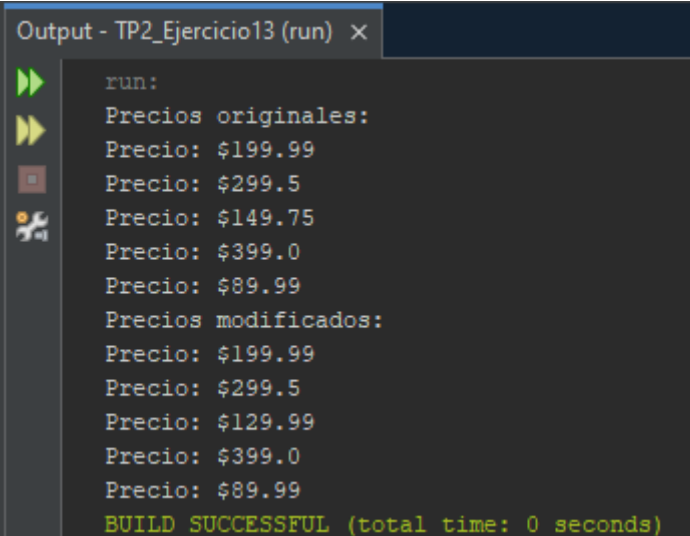
Precio: \$89.99

**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.

```
11 public class TP2_Ejercicio13 {
12
13     // Función recursiva para mostrar los precios
14     public static void mostrarPrecios(double[] precios, int indice) {
15         // Caso base: si el índice es igual o mayor a la longitud del array, detener
16         if (indice >= precios.length) {
17             return;
18         }
19         // Mostrar el precio actual
20         System.out.println("Precio: $" + precios[indice]);
21         // Llamada recursiva con el siguiente índice
22         mostrarPrecios(precios, indice + 1);
23     }
24
25     public static void main(String[] args) {
26         // Declarar e inicializar el array de precios
27         double[] precios = {199.99, 299.5, 149.75, 399.0, 89.99};
28
29         // Mostrar precios originales
30         System.out.println("Precios originales:");
31         mostrarPrecios(precios, 0);
32
33         // Modificar el precio en el índice 2
34         precios[2] = 129.99;
35
36         // Mostrar precios modificados
37         System.out.println("Precios modificados:");
38         mostrarPrecios(precios, 0);
39     }
40 }
```

Salidas:



```
Output - TP2_Ejercicio13 (run) ×
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)
```

## 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.