TP Programacion 2

Alumno: Castellini Gonzalo

Enlace a repositorio de GitHub:

https://github.com/gonzilla2021/UTN-TUP-P2-JAVA-.git

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.

```
package ej1condicionales;

import java.util.Scanner;

/**

 * Programa para verificar si un año es bisiesto
 *
 * Reglas para año bisiesto:
 * Divisible por 4 Y no divisible por 100, O
 * Divisible por 400
 *
 * @author Castellini Gonzalo
 */ public class Ej1Condicionales {

public static void main(String[] args) {

    // Crear objeto Scanner para leer entrada del usuario
    Scanner input = new Scanner(System.in);

    // Solicitar al usuario que ingrese un año
    System.out.print("Ingrese un año: ");
```

```
    // Leer el año ingresado y convertirlo a entero
    int anio = Integer.parseInt(input.nextLine());

    // Mostrar el año ingresado para confirmación
    System.out.println("El año ingresado es: " + anio);

    // Verificar si el año es bisiesto usando las reglas:
    // 1. Divisible por 4 Y no divisible por 100 (años
como 2004, 2008, etc.)
    // 2. O divisible por 400 (años como 1600, 2000, etc.)
    if ((anio % 4 == 0 && anio % 100 != 0) || anio % 400
== 0) {
        // Si cumple las condiciones, es bisiesto
        System.out.println("El año " + anio + " es
bisiesto.");

    } else {
        // Si no cumple las condiciones, no es bisiesto
        System.out.println("El año " + anio + " NO es
bisiesto.");
    }

    // Cerrar el scanner para liberar recursos
    input.close();
}


}
```

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.

```
package ej2elmayor;
```

```
import java.util.Scanner;

/**
```

- Programa para determinar el mayor de tres números
-
- Este programa solicita al usuario tres números enteros y determina cuál
- es el mayor de los tres utilizando un bucle for y comparaciones.
-
- Método utilizado: Comparación iterativa con variable auxiliar
- @author Gonzalo Castellini

```
*/

public class Ej2ElMayor {

public static void main(String[] args) {

    // Crear objeto Scanner para leer entrada del usuario
    Scanner input = new Scanner(System.in);

    // Inicializar variable para almacenar el número mayor
    // Se usa Integer.MIN_VALUE para asegurar que
cualquier número ingresado
    // sea mayor que este valor inicial (-2,147,483,648)
    int max = Integer.MIN_VALUE;

    System.out.println("=== PROGRAMA: MAYOR DE TRES
NÚMEROS ===");

    // Bucle para solicitar exactamente 3 números al
usuario
    for (int j = 0; j < 3; j++) {

        // Solicitar el número al usuario (j+1 para
mostrar "1°", "2°", "3°")
        System.out.print("Ingresa el " + (j + 1) + "°
```

```
número: ");

        // Leer y convertir la entrada del usuario a
entero
        int numero = Integer.parseInt(input.nextLine());

        // Comparar el número actual con el mayor
encontrado hasta ahora
        if (numero > max) {
            // Si el número actual es mayor, actualizamos
la variable max
            max = numero;
            System.out.println("  → Nuevo número mayor: "
+ max);
        } else {
            // Si no es mayor, informamos que se mantiene
el actual
            System.out.println("  → El mayor sigue siendo:
" + max);
        }
    }

    // Mostrar el resultado final
    System.out.println("\n=== RESULTADO ===");
    System.out.println("El mayor de los tres números es: "
+ max);

    // Cerrar el scanner para liberar recursos del sistema
    input.close();
}


}
```

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"

```
package ej3edades;

/**********************************************************
**********************
```

- **Programa para clasificar personas según su edad en diferentes categorías**
-
- **Categorías de edad:**
-
    - **Edad inválida: menor a 0 años**
-
    - **Niño: 0 a 11 años**
-
    - **Adolescente: 12 a 17 años**
-
    - **Adulto: 18 a 59 años**
-
    - **Adulto Mayor: 60 años en adelante**
-
- **@author Gonzalo**

```
**********************************************************
**********************/

import java.util.Scanner;

public class Ej3Edades {

public static void main(String[] args) {

    // Crear objeto Scanner para leer entrada del usuario
    Scanner input = new Scanner(System.in);

    // Solicitar la edad al usuario
    System.out.print("Ingresa tu edad: ");

    // Leer y convertir la entrada a número entero
    int edad = Integer.parseInt(input.nextLine());

    // ===== VALIDACIÓN Y CLASIFICACIÓN DE EDAD =====

    // Primera validación: verificar que la edad sea un
número válido
    if (edad < 0) {
        // Caso de error: edad negativa no es válida
        System.out.println("Edad no válida");

    } else {
        // Si la edad es válida, proceder con la
clasificación

        // Estructura de if-else if anidada para
clasificar por rangos de edad

        if (edad < 12) {
            // Rango de 0 a 11 años: Clasificado como niño
            System.out.println("Niño");

        } else if (edad < 18) {
```

```
            // Rango de 12 a 17 años: Clasificado como
adolescente
            System.out.println("Adolescente");

        } else if (edad < 60) {
            // Rango de 18 a 59 años: Clasificado como
adulto
            System.out.println("Adulto");

        } else {
            // Rango de 60 años en adelante: Clasificado
como adulto mayor
            System.out.println("Adulto Mayor");
        }
    }

    // Cerrar el scanner para liberar recursos del sistema
    input.close();
}


}
```

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

```
package ej4descuentos;

/**********************************************************
***********
```

- Calculadora de Descuento según categoría
-
- Este programa calcula descuentos basados en la categoría del producto:
-
    - Categoría A: 10% de descuento (precio final = precio * 0.90)
- Categoría B: 15% de descuento (precio final = precio * 0.85)
- Categoría C: 20% de descuento (precio final = precio * 0.80)
- El programa muestra: precio original, descuento aplicado y precio final
- @author Gonza

```
**********************************************************
**********************/

import java.util.Scanner;

public class Ej4Descuentos {

public static void main(String[] args) {

    // Crear objeto Scanner para leer entrada del usuario
    Scanner input = new Scanner(System.in);

    // Declarar variables para almacenar los datos del
producto
    double precio;              // Precio original del
producto
```

```
    char categoria;             // Categoría del producto
(A, B o C)
    double precioFinal = 0;     // Precio después de
aplicar descuento
    double montoDescuento = 0;  // Cantidad de dinero
descontada
    int porcentajeDescuento = 0; // Porcentaje de
descuento aplicado

    // ===== ENTRADA DE DATOS =====

    // Solicitar la categoría del producto
    System.out.print("Ingrese la categoría (A, B o C): ");
    // Leer entrada, convertir a mayúscula y tomar el
primer caracter
    categoria = input.nextLine().toUpperCase().charAt(0);

    // Solicitar el precio del producto
    System.out.print("Ingrese el precio del producto: $");
    // Leer y convertir la entrada a número decimal
    precio = Double.parseDouble(input.nextLine());

    // ===== VALIDACIÓN Y CÁLCULO DE DESCUENTOS =====

    // Estructura condicional para determinar descuento
según categoría

    if (categoria == 'A') {
        // Categoría A: 10% de descuento
        porcentajeDescuento = 10;
        precioFinal = precio * 0.90;  // Aplicar descuento
(mantener 90%)
        montoDescuento = precio * 0.10; // Calcular monto
descontado

    } else if (categoria == 'B') {
        // Categoría B: 15% de descuento
```

```
        porcentajeDescuento = 15;
        precioFinal = precio * 0.85;  // Aplicar descuento
(mantener 85%)
        montoDescuento = precio * 0.15; // Calcular monto
descontado

    } else if (categoria == 'C') {
        // Categoría C: 20% de descuento
        porcentajeDescuento = 20;
        precioFinal = precio * 0.80;  // Aplicar descuento
(mantener 80%)
        montoDescuento = precio * 0.20; // Calcular monto
descontado

    } else {
        // Categoría inválida: no existe descuento para
esta categoría
        System.out.println("\n ERROR: Categoría '" +
categoria + "' no válida.");
        System.out.println("Las categorías válidas son: A,
B, C");
        input.close(); // Cerrar scanner antes de salir
        return; // Terminar el programa si la categoría es
inválida
    }

    // ===== MOSTRAR RESULTADOS =====

    System.out.println("\n=== RESUMEN DE COMPRA ===");
    System.out.println("Categoría aplicada: " +
categoria);
    System.out.printf("Precio original: $%.2f%n", precio);
    System.out.println("Descuento aplicado: " +
porcentajeDescuento + "%");
    System.out.printf("Monto del descuento: $%.2f%n",
montoDescuento);
    System.out.printf("Precio final: $%.2f%n",
```

```
precioFinal);
    System.out.printf("Ahorro total: $%.2f%n",
montoDescuento);

    // Cerrar el scanner para liberar recursos del sistema
    input.close();
}


}
```

Otra forma de hacerlo es usando un Swich

```
import java.util.Scanner;

public class Main {

    public static void main(String[] args) {

        Scanner input = new Scanner(System.in);

        double precio;

        char categoria;

        double precioFinal = 0;

        String mostrarDescuento = "";

        System.out.print("Ingrese el precio del producto:
");

        precio = Double.parseDouble(input.nextLine());
```

```
        System.out.print("Ingrese la categoría A, B o C: 
");
        categoria = 
input.nextLine().toUpperCase().charAt(0); // Asegura 
mayúscula
        switch (categoria) {
            case 'A':
                precioFinal = precio * 0.90;
                mostrarDescuento = "10%";
                break;
            case 'B':
                precioFinal = precio * 0.85;
                mostrarDescuento = "15%";
                break;
            case 'C':
                precioFinal = precio * 0.80;
                mostrarDescuento = "20%";
                break;
            default:
                System.out.println("Categoría no 
válida.");
                input.close();
                return;
        }
```

```
        System.out.println("La categoría aplicada es: " +
categoria);

        System.out.println("El descuento aplicado es de: "
+ mostrarDescuento);

        System.out.printf("El precio final es: %.2f\n",
precioFinal);

        input.close();

    }

}
```

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 ej5ciclos;

/**************************************************
```

- Programa para sumar únicamente los números pares ingresados por el usuario
-
- Funcionamiento:
- Solicita números al usuario continuamente
- Solo suma los números que sean pares (divisibles por 2)
- Se detiene cuando el usuario ingresa 0
- Muestra la suma total de todos los números pares
-
- Ejemplo de uso:
- Entrada: 4, 7, 2, 0
- Números pares encontrados: 4, 2
- Suma total: 6
-
- @author Gonza

**********/

```
import java.util.Scanner;

public class Ej5Ciclos {

public static void main(String[] args) {

    // Crear objeto Scanner para leer entrada del usuario
    Scanner input = new Scanner(System.in);

    // Variables para el control del programa
    int num;           // Almacena el número actual
ingresado por el usuario
    int sumatoria = 0; // Acumulador para la suma de
números pares

    // ===== INSTRUCCIONES INICIALES =====
    System.out.println("=== CALCULADORA DE SUMA DE NÚMEROS
```

```
PARES ===");
    System.out.println("Ingrese números enteros. Solo se
sumarán los pares.");
    System.out.println("Para terminar, ingrese 0.\n");

    // ===== LECTURA DEL PRIMER NÚMERO =====
    // Se lee el primer número antes del bucle (patrón
"read-before-loop")
    System.out.print("Ingrese un número (0 para terminar):
");
    num = Integer.parseInt(input.nextLine());

    // ===== BUCLE PRINCIPAL CON WHILE =====
    // El bucle continúa mientras el número ingresado sea
diferente de 0
    while (num != 0) {

        // Verificar si el número actual es par
        // Un número es par si el residuo de dividirlo por
2 es igual a 0
        if (num % 2 == 0) {
            // Si es par, agregarlo a la sumatoria
            sumatoria += num;
            System.out.println("  ✓ Número par detectado:
" + num + " (suma actual: " + sumatoria + ")");
        } else {
            // Si es impar, informar que no se suma
            System.out.println("  ✗ Número impar: " + num
+ " (no se suma)");
        }

        // ===== LECTURA DEL SIGUIENTE NÚMERO =====
        // Solicitar el próximo número dentro del bucle
        // Esta lectura es crucial para evitar un bucle
infinito
        System.out.print("Ingrese otro número (0 para
```

```
terminar): ");
            num = Integer.parseInt(input.nextLine());
        }

        // ===== MOSTRAR RESULTADO FINAL =====
        System.out.println("\n=== RESULTADO ===");
        System.out.println("La suma de los números pares es: "
+ sumatoria);

        // Mensaje adicional si no se ingresaron números pares
        if (sumatoria == 0) {
            System.out.println("No se ingresaron números
pares.");
        }

        // Cerrar el scanner para liberar recursos del sistema
        input.close();
}


}
```

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.

```
package ej6ciclofor;

/**********************************************************
```

- Programa contador de números según diferentes criterios

- VERSIÓN 1: Contador de Positivos, Negativos y Ceros (según enunciado)

- VERSIÓN 2: Contador de Pares, Impares y Ceros (tu código actual)

- Ambas versiones solicitan números al usuario y los clasifican según
- diferentes criterios usando bucles for y estructuras condicionales.
-
- @author Gonza

**********/

```
import java.util.Scanner;

public class Ej6Ciclofor {

public static void main(String[] args) {

    // Crear objeto Scanner para leer entrada del usuario
    Scanner input = new Scanner(System.in);

    // ===== VERSIÓN 1: POSITIVOS, NEGATIVOS Y CEROS =====
    // (Según el enunciado del problema)

    System.out.println("=== CONTADOR DE POSITIVOS,
NEGATIVOS Y CEROS ===");

    // Contadores para cada categoría de números
    int contPositivos = 0;  // Contador de números mayores
a 0
    int contNegativos = 0;  // Contador de números menores
a 0
    int contCeros = 0;      // Contador de números igual a
0

    // Bucle for para solicitar exactamente 10 números
    for (int i = 0; i < 10; i++) {
        // Mostrar cuál número se está pidiendo (1°, 2°,
```

```
etc.)
        System.out.print("Ingrese el " + (i + 1) + "º
número: ");

        // Leer y convertir la entrada del usuario
        int numero = Integer.parseInt(input.nextLine());

        // Clasificar el número según su signo
        if (numero > 0) {
            // Número positivo (mayor que 0)
            contPositivos++;
            System.out.println("  → Positivo detectado");

        } else if (numero < 0) {
            // Número negativo (menor que 0)
            contNegativos++;
            System.out.println("  → Negativo detectado");

        } else {
            // Número igual a cero
            contCeros++;
            System.out.println("  → Cero detectado");
        }
    }

    // Mostrar resultados de la clasificación por signo
    System.out.println("\n=== RESULTADOS
(POSITIVOS/NEGATIVOS) ===");
    System.out.println("Números positivos: " +
contPositivos);
    System.out.println("Números negativos: " +
contNegativos);
    System.out.println("Números ceros: " + contCeros);
    System.out.println("Total de números ingresados: " +
(contPositivos + contNegativos + contCeros));
```

```
    // ===== VERSIÓN 2: PARES, IMPARES Y CEROS =====
    // (Tu código actual comentado)

    System.out.println("\n" + "=".repeat(60));
    System.out.println("=== CONTADOR DE PARES, IMPARES Y
CEROS ===");

    // Reiniciar contadores para la segunda clasificación
    int contPar = 0;     // Contador de números pares
(divisibles por 2)
    int contImpar = 0;   // Contador de números impares
(no divisibles por 2)
    contCeros = 0;       // Reiniciar contador de ceros

    // Bucle for para solicitar 5 números (como en tu
código original)
    for (int i = 0; i < 5; i++) {

        // Solicitar número al usuario
        System.out.print("Ingrese el " + (i + 1) + "º
número: ");
        int numero = Integer.parseInt(input.nextLine());

        // Clasificar el número según paridad
        if (numero == 0) {
            // Caso especial: el cero se cuenta
separadamente
            // Nota: matemáticamente 0 es par, pero aquí
se maneja aparte
            contCeros++;
            System.out.println("  → Cero detectado");

        } else if (numero % 2 != 0) {
            // Número impar: el residuo de dividir por 2
no es cero
            contImpar++;
            System.out.println("  → Impar detectado");
```

```
        } else {
            // Número par: el residuo de dividir por 2 es
cero
            contPar++;
            System.out.println("  → Par detectado");
        }
    }

    // Mostrar resultados de la clasificación por paridad
    System.out.println("\n=== RESULTADOS (PARES/IMPARES)
===");
    System.out.println("N° Pares: " + contPar);
    System.out.println("N° Impares: " + contImpar);
    System.out.println("N° Ceros: " + contCeros);
    System.out.println("Total de números ingresados: " +
(contPar + contImpar + contCeros));

    // Cerrar el scanner para liberar recursos del sistema
    input.close();
}


}
```

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 ej7dowhile;
/**********************************************************
********

   • Programa de validación de notas usando bucle do-while
   •
   • Este programa solicita una nota al usuario y valida
     que esté en el rango
   • válido de 0 a 10. Si la nota está fuera del rango,
     repite la solicitud
   • hasta que el usuario ingrese un valor correcto.
   •
   • Ventaja del do-while:
   • Garantiza que se ejecute al menos una vez la
     solicitud
   • Perfecto para validaciones donde necesitamos repetir
     hasta obtener
   • una entrada válida
   •
   • Rango válido: 0 ≤ nota ≤ 10
   •

 **************************************************/

import java.util.Scanner;

public class Ej7DoWhile {

public static void main(String[] args) {
    // Crear objeto Scanner para leer entrada del usuario
    Scanner input = new Scanner(System.in);

    // Variable para almacenar la nota ingresada
    int nota;
```

```
    // ===== PRESENTACIÓN DEL PROGRAMA =====
    System.out.println("=== VALIDADOR DE NOTAS ===");
    System.out.println("Ingrese una nota válida entre 0 y
10 (inclusive)\n");

    // ===== BUCLE DO-WHILE PARA VALIDACIÓN =====
    // El bucle do-while es ideal para validaciones
porque:
    // 1. Siempre ejecuta el bloque al menos una vez
    // 2. Evalúa la condición al final, después de cada
intento

    do {
        // Solicitar la nota al usuario
        System.out.print("Ingrese una nota (0-10): ");

        // Leer y convertir la entrada del usuario a
entero
        nota = Integer.parseInt(input.nextLine());

        // ===== VALIDACIÓN DEL RANGO =====
        // Verificar si la nota está fuera del rango
válido
        // Condición: nota < 0 OR nota > 10
        if (nota < 0 || nota > 10) {
            // Nota inválida: mostrar mensaje de error
específico
            System.out.println(" Error: Nota inválida.");

            // Dar feedback específico según el tipo de
error
            if (nota < 0) {
                System.out.println("La nota no puede ser
negativa.");
            } else {
                System.out.println("La nota no puede ser
```

```
mayor a 10.");
            }

            System.out.println("Ingrese una nota entre 0 y
10.\n");

        } else {
            // Nota válida: mostrar confirmación
            System.out.println("Nota válida ingresada: " +
nota);
        }

    } while (nota < 0 || nota > 10);
    // CONDICIÓN DE REPETICIÓN:
    // El bucle continúa mientras la nota esté fuera del
rango válido
    // Se detiene cuando: nota >= 0 AND nota <= 10

    // ===== CONFIRMACIÓN FINAL =====
    System.out.println("\n=== PROCESO COMPLETADO ===");
    System.out.println("Nota guardada correctamente: " +
nota);

    // Mostrar clasificación adicional de la nota
    if (nota >= 7) {
        System.out.println("Clasificación: Aprobado");
    } else if (nota >= 4) {
        System.out.println("Clasificación: Regular");
    } else {
        System.out.println("Clasificación: Reprobado");
    }

    // Cerrar el scanner para liberar recursos del sistema
    input.close();
}
```

```
}
```

Funciones:

8. Cálculo del Precio Final con impuesto y descuento.

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

La fórmula es: PrecioFinal = PrecioBase + (PrecioBase×Impuesto) − (PrecioBase×Descuento)

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

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

Ejemplo de entrada/salida: Ingrese el precio base del producto: 100 Ingrese el impuesto en porcentaje (Ejemplo: 10 para 10%): 10 Ingrese el descuento en porcentaje (Ejemplo: 5 para 5%): 5 El precio final del producto es: 105.0

```
package ej8funciones;

/**********************************************************
***********
```

- Programa para calcular el precio final de un producto en e-commerce
- aplicando impuestos y descuentos
-
- Fórmula utilizada:
- PrecioFinal = PrecioBase + (PrecioBase × Impuesto) - (PrecioBase × Descuento)

```
 *
 * Donde:
 * PrecioBase: precio original del producto
 * Impuesto: porcentaje de impuesto (ej: 10 = 10%)
 * Descuento: porcentaje de descuento (ej: 5 = 5%)
 *
 * Ejemplo:
 * PrecioBase = 100, Impuesto = 10%, Descuento = 5%
 * PrecioFinal = 100 + (100×0.10) - (100×0.05) = 100 +
   10 - 5 = 105
 *
 * @author Gonzalo
**********************************************************
********/

import java.util.Scanner;

public class Ej8Funciones {

/**
 * Método principal que controla la ejecución del programa
 *
 * @param args argumentos de línea de comandos (no
utilizados)
 */
public static void main(String[] args) {

    // Crear objeto Scanner para leer entrada del usuario
    Scanner input = new Scanner(System.in);

    // ===== PRESENTACIÓN DEL PROGRAMA =====
    System.out.println("=== CALCULADORA DE PRECIO FINAL E-
COMMERCE ===");
    System.out.println("Calcule el precio final aplicando
impuestos y descuentos\n");
```

```
    // ===== ENTRADA DE DATOS =====

    // Solicitar el precio base del producto
    System.out.print("Ingrese el precio base del producto:
$");
    double precioBase = input.nextDouble();

    // Solicitar el porcentaje de impuesto
    // Nota: se ingresa como número entero (10 para 10%)
    System.out.print("Ingrese el impuesto en porcentaje
(Ejemplo: 10 para 10%): ");
    double impuesto = input.nextDouble();

    // Solicitar el porcentaje de descuento
    // Nota: se ingresa como número entero (5 para 5%)
    System.out.print("Ingrese el descuento en porcentaje
(Ejemplo: 5 para 5%): ");
    double descuento = input.nextDouble();

    // ===== LLAMADA AL MÉTODO DE CÁLCULO =====

    // Llamar al método calcularPrecioFinal pasando los
tres parámetros
    // El método retorna el precio final calculado
    double precioFinal = calcularPrecioFinal(precioBase,
impuesto, descuento);

    // ===== MOSTRAR RESULTADOS DETALLADOS =====

    System.out.println("\n=== DESGLOSE DE CÁLCULO ===");
    System.out.printf("Precio base: $%.2f%n", precioBase);
    System.out.printf("Impuesto aplicado: %.1f%% =
$%.2f%n", impuesto, (precioBase * impuesto / 100));
    System.out.printf("Descuento aplicado: %.1f%% =
$%.2f%n", descuento, (precioBase * descuento / 100));
    System.out.println("-".repeat(40));
    System.out.printf("Precio final del producto:
```

```
$%.2f%n", precioFinal);

    // Mostrar el ahorro o costo adicional neto
    double diferencia = precioFinal - precioBase;
    if (diferencia > 0) {
        System.out.printf("Costo adicional: $%.2f%n",
diferencia);
    } else if (diferencia < 0) {
        System.out.printf("Ahorro neto: $%.2f%n",
Math.abs(diferencia));
    } else {
        System.out.println("Sin cambio en el precio");
    }

    // Cerrar el scanner para liberar recursos
    input.close();
}

/**
 * Método para calcular el precio final de un producto
aplicando impuesto y descuento
 *
 * Este método implementa la fórmula:
 * PrecioFinal = PrecioBase + (PrecioBase × Impuesto/100)
- (PrecioBase × Descuento/100)
 *
 * Proceso de cálculo:
 * 1. Calcular monto del impuesto: precioBase ×
(impuesto/100)
 * 2. Calcular monto del descuento: precioBase ×
(descuento/100)
 * 3. Sumar impuesto y restar descuento al precio base
 *
 * @param precioBase el precio original del producto (debe
ser positivo)
 * @param impuesto el porcentaje de impuesto a aplicar
(ej: 10.0 para 10%)
```

```
 * @param descuento el porcentaje de descuento a aplicar
(ej: 5.0 para 5%)
 * @return el precio final después de aplicar impuesto y
descuento
 */
public static double calcularPrecioFinal(double
precioBase, double impuesto, double descuento) {

    // Aplicar la fórmula completa en una sola línea
    // División por 100 convierte porcentajes a decimales
(10% = 0.10)

    // Desglose de la fórmula:
    // precioBase = precio original
    // (precioBase * impuesto / 100) = monto del impuesto
    // (precioBase * descuento / 100) = monto del
descuento

    return precioBase + (precioBase * impuesto / 100) -
(precioBase * descuento / 100);

    /* VERSIÓN DESGLOSADA PARA MEJOR COMPRENSIÓN:
    double montoImpuesto = precioBase * impuesto / 100;
    double montoDescuento = precioBase * descuento / 100;
    double precioConImpuesto = precioBase + montoImpuesto;
    double precioFinal = precioConImpuesto -
montoDescuento;
    return precioFinal;
    */
}


}
```

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 eTonvío 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

```
package ej9composicionfunciones;
/**********************************************************
```

- Sistema de cálculo de costos de envío y total de compra para e-commerce
-
- Este programa implementa un sistema modular para calcular:
- Costo de envío basado en peso y zona geográfica
- Total de compra sumando producto + envío
-
- Tarifas de envío:
- Nacional: $5.00 por kilogramo
- Internacional: $10.00 por kilogramo
-
- Arquitectura:
- Método calcularCostoEnvio(): Calcula solo el costo de envío
- Método calcularTotalCompra(): Suma producto + envío

- Composición de funciones: un método usa el resultado del otro

**********/

```
import java.util.Scanner;

public class Ej9composicionFunciones {

// ===== CONSTANTES GLOBALES =====
// Definir tarifas como constantes para facilitar mantenimiento y legibilidad

/** Tarifa de envío nacional: $5.00 por kilogramo */
public static final double PRECIO_NACIONAL = 5.0;

/** Tarifa de envío internacional: $10.00 por kilogramo */
public static final double PRECIO_INTERNACIONAL = 10.0;

/**
 * Método principal que controla el flujo del programa
 *
 * @param args argumentos de línea de comandos (no utilizados)
 */
public static void main(String[] args) {

    // Crear objeto Scanner para leer entrada del usuario
    Scanner input = new Scanner(System.in);

    // ===== PRESENTACIÓN DEL PROGRAMA =====
    System.out.println("=== CALCULADORA DE ENVÍO E-COMMERCE ===");
    System.out.println("Calcule el costo total incluyendo envío\n");

    // ===== ENTRADA DE DATOS =====
```

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

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

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

    // ===== CÁLCULO DEL COSTO DE ENVÍO =====

    // Llamar al método calcularCostoEnvio para obtener el
costo
    // Este método retorna -1 si la zona es inválida
    double costoEnvio = calcularCostoEnvio(peso, zona);

    // ===== VALIDACIÓN Y CÁLCULO FINAL =====

    // Verificar si la zona ingresada es válida
    if (costoEnvio != -1) {
        // Zona válida: proceder con los cálculos

        System.out.println("\n=== DESGLOSE DE COSTOS
===");
        System.out.printf("Precio del producto: $%.2f%n",
precioProducto);
        System.out.printf("Peso del paquete: %.2f kg%n",
peso);
        System.out.printf("Zona de envío: %s%n", zona);
        System.out.printf("Costo de envío: $%.2f%n",
costoEnvio);
```

```
        // ===== COMPOSICIÓN DE FUNCIONES =====
        // Usar el resultado de calcularCostoEnvio() como
parámetro
        // para calcularTotalCompra() - esto es
composición de funciones
        double total = calcularTotalCompra(precioProducto,
costoEnvio);

        // Mostrar resultado final
        System.out.println("-".repeat(35));
        System.out.printf("Total a pagar: $%.2f%n",
total);

    } else {
        // Zona inválida: mostrar mensaje de error
        System.out.println("\n❌ ERROR: Zona de envío no
válida");
        System.out.println("Las zonas válidas son:
Nacional o Internacional");
    }

    // Cerrar el scanner para liberar recursos del sistema
    input.close();
}

/**
 * Método para calcular el costo de envío según peso y
zona geográfica
 *
 * Este método implementa la lógica de tarifas
diferenciadas:
 * - Envío nacional: más económico ($5/kg)
 * - Envío internacional: más costoso ($10/kg)
 *
 * @param peso el peso del paquete en kilogramos (debe ser
positivo)
```

```
 * @param zona la zona de envío ("Nacional" o
"Internacional")
 * @return el costo total de envío, o -1 si la zona es
inválida
 */
public static double calcularCostoEnvio(double peso,
String zona) {

    // Variable para almacenar el costo calculado
    double costo;

    // ===== CLASIFICACIÓN POR ZONA =====
    // Usar equalsIgnoreCase() para hacer la comparación
insensible a mayúsculas

    if (zona.equalsIgnoreCase("Nacional")) {
        // Envío nacional: aplicar tarifa nacional
        costo = PRECIO_NACIONAL * peso;  // $5.00 × kg

    } else if (zona.equalsIgnoreCase("Internacional")) {
        // Envío internacional: aplicar tarifa
internacional
        costo = PRECIO_INTERNACIONAL * peso;  // $10.00 ×
kg

    } else {
        // Zona no reconocida: retornar código de error
        // -1 indica que la zona ingresada no es válida
        return -1;
    }

    // Retornar el costo calculado
    return costo;
}

/**
 * Método para calcular el total final de la compra
```

```
 *
 * Este método implementa la composición de funciones
sumando:
 * - El precio del producto
 * - El costo de envío (calculado por calcularCostoEnvio)
 *
 * Fórmula simple: Total = Producto + Envío
 *
 * @param precioProducto el precio base del producto (sin
envío)
 * @param costoEnvio el costo del envío (ya calculado)
 * @return el total final a pagar por el cliente
 */
public static double calcularTotalCompra(double
precioProducto, double costoEnvio) {

    // Sumar el precio del producto más el costo de envío
    // Esta es la operación final que combina ambos costos
    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:

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 ej10stock;
/**********************************************************
***********
```

- **Sistema de Gestión de Stock para control de inventario**
-
- **Este programa simula las operaciones básicas de un sistema de inventario:**
- **Ventas: reducen el stock disponible**
- **Recepciones: aumentan el stock disponible**
- **Cálculo automático: actualiza el stock final**
-
- **Fórmula utilizada:**
- **NuevoStock = StockActual - CantidadVendida + CantidadRecibida**
-
- **Ejemplo de funcionamiento:**
- **Stock inicial: 50 unidades**
- **Ventas: 20 unidades (reducen stock)**
- **Recepciones: 30 unidades (aumentan stock)**
- **Stock final: 50 - 20 + 30 = 60 unidades**
-
- **Características:**
- **Validación de entradas negativas**
- **Protección contra stock negativo**
- **Arquitectura modular con métodos separados**

```
**********************************************************
**********/

import java.util.Scanner;

public class Ej10Stock {

/**
 * Método principal que controla la interfaz de usuario y
validaciones
 *
 * @param args argumentos de línea de comandos (no
utilizados)
 */
public static void main(String[] args) {
    // Crear objeto Scanner para leer entrada del usuario
    Scanner input = new Scanner(System.in);

    // ===== PRESENTACIÓN DEL SISTEMA =====
    System.out.println("=== SISTEMA DE GESTIÓN DE STOCK
===");
    System.out.println("Actualice el inventario
registrando ventas y recepciones\n");

    // ===== ENTRADA DE DATOS =====

    // Solicitar stock actual del producto
    System.out.print("Ingrese el stock actual del
producto: ");
    int stockActual = Integer.parseInt(input.nextLine());

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

    // Solicitar cantidad recibida (entrada de stock)
```

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

    // ===== VALIDACIÓN DE ENTRADAS =====

    // Verificar que todos los valores sean no negativos
    // En un sistema real, las cantidades negativas no
tienen sentido
    if (stockActual < 0 || cantidadVendida < 0 ||
cantidadRecibida < 0) {
        // Error: mostrar mensaje y terminar programa
        System.out.println("\n ERROR: Los números no
pueden ser negativos");
        System.out.println("- Stock actual: debe ser ≥
0");
        System.out.println("- Cantidad vendida: debe ser ≥
0");
        System.out.println("- Cantidad recibida: debe ser
≥ 0");

    } else {
        // ===== CÁLCULO DEL NUEVO STOCK =====

        // Llamar al método actualizarStock para calcular
el nuevo inventario
        // Este método encapsula la lógica de cálculo y
validación
        int nuevoStock = actualizarStock(stockActual,
cantidadVendida, cantidadRecibida);

        // ===== MOSTRAR RESULTADOS DETALLADOS =====

        System.out.println("\n=== MOVIMIENTOS DE STOCK
===");
        System.out.println("Stock inicial: " + stockActual
+ " unidades");
```

```
        System.out.println("(-) Cantidad vendida: " +
cantidadVendida + " unidades");
        System.out.println("(+) Cantidad recibida: " +
cantidadRecibida + " unidades");
        System.out.println("-".repeat(40));
        System.out.println("Stock final: " + nuevoStock +
" unidades");

        // Análisis del resultad
        int diferencia = nuevoStock - stockActual;
        if (diferencia > 0) {
            System.out.println("Stock aumentó en: " +
diferencia + " unidades");
        } else if (diferencia < 0) {
            System.out.println("Stock disminuyó en: " +
Math.abs(diferencia) + " unidades");
        } else {
            System.out.println("Stock se mantiene igual");
        }

        // Alerta de stock bajo
        if (nuevoStock <= 5 && nuevoStock > 0) {
            System.out.println("ALERTA: Stock bajo -
considere reabastecer");
        } else if (nuevoStock == 0) {
            System.out.println(" CRÍTICO: Producto agotado
- reabastecimiento urgente");
        }
    }

    // Cerrar el scanner para liberar recursos del sistema
    input.close();
}

/**
 * Método para actualizar el stock después de operaciones
de venta y recepción
```

```
 *
 * Este método implementa la lógica central del sistema de
inventario:
 *
 * Proceso de cálculo:
 * 1. Partir del stock actual
 * 2. Restar las unidades vendidas (salida del inventario)
 * 3. Sumar las unidades recibidas (entrada al inventario)
 * 4. Aplicar protección contra stock negativo
 *
 * Fórmula: NuevoStock = StockActual - CantidadVendida +
CantidadRecibida
 *
 * Protección implementada:
 * - Si el cálculo resulta en stock negativo, se establece
en 0
 * - Esto evita tener inventarios "imposibles" en el
sistema
 *
 * @param stockActual el stock disponible antes de la
operación
 * @param cantidadVendida cuántas unidades se vendieron
(salida)
 * @param cantidadRecibida cuántas unidades se recibieron
(entrada)
 * @return el nuevo stock después de aplicar ambas
operaciones
 */
public static int actualizarStock(int stockActual, int
cantidadVendida, int cantidadRecibida) {

    // ===== APLICAR FÓRMULA DE ACTUALIZACIÓN =====

    // Calcular el nuevo stock usando la fórmula estándar
    // stockActual: punto de partida
    // -cantidadVendida: reduce el stock (operación de
```

```
salida)
    // +cantidadRecibida: aumenta el stock (operación de
entrada)
    int nuevoStock = stockActual - cantidadVendida +
cantidadRecibida;

    // ===== PROTECCIÓN CONTRA STOCK NEGATIVO =====

    // En un sistema real, el stock no puede ser negativo
    // Si el cálculo da negativo, establecer en 0 (stock
agotado)
    if (nuevoStock < 0) {
        System.out.println("Advertencia: La venta excede
el stock disponible.");
        System.out.println("Stock calculado: " +
nuevoStock + " → Ajustado a: 0");
        nuevoStock = 0;  // Establecer stock mínimo en 0
    }

    // Retornar el stock final calculado y validado
    return nuevoStock;
}


}
```

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 6

package ejercicio11funciones; import java.util.Scanner;

/*

- **Cálculo de descuento especial usando variable global.**
- **Declara una variable global descuento = 0.10 (10%)**
- **Crea un método que calcule el descuento y muestre el precio final**

***/ public class Ejercicio11Funciones {**

```
// Variable global - descuento fijo del 10%
public static double descuento = 0.10;

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

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

    // Llamar al método que calcula el descuento
    calcularDescuentoEspecial(precio);

    input.close();
}

/**
 * Método que calcula y muestra el descuento especial
 * @param precio - precio original del producto
 */
public static void calcularDescuentoEspecial(double
```

```
precio) {
    // Variable local - cantidad de dinero descontada
    double descuentoAplicado = precio * descuento;

    // Calcular precio final
    double precioFinal = precio - descuentoAplicado;

    // Mostrar resultados
    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

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 ejercicio12array;

/*
```

- **Modificación de un array de precios y visualización de resultados**
- **Declarar array con precios**
- **Mostrar precios originales**
- **Modificar un precio específico**
- **Mostrar precios modificados**

```
*/ public class Ejercicio12Array {

public static void main(String[] args) {

    // a. Declarar e inicializar array con precios de
productos
    double[] precios = {199.99, 299.5, 149.75, 399.0,
89.99};

    // b. Mostrar valores originales
    System.out.println("Precios originales:");
    mostrarPrecios(precios);
```

```
    // c. Modificar el precio de un producto específico
(índice 2)
    precios[2] = 129.99; // Cambiar 149.75 por 129.99

    // d. Mostrar valores modificados
    System.out.println("\nPrecios modificados:");
    mostrarPrecios(precios);
}

/**
 * Método para mostrar todos los precios del array
 * @param arrayPrecios - array con los precios a mostrar
 */
public static void mostrarPrecios(double[] arrayPrecios) {
    // Recorrer array con for-each para mostrar valores
    for (double precio : arrayPrecios) {
        System.out.println("Precio: $" + precio);
    }
}


}
```

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.

package ejercicio13recursividad;

/*

- **Impresión recursiva de arrays antes y después de modificar un elemento**
- **Declarar array con precios**
- **Función recursiva para mostrar precios originales**
- **Modificar un precio específico**
- **Función recursiva para mostrar precios modificados**

***/ public class Ejercicio13Recursividad {**

```
public static void main(String[] args) {

    // a. Declarar e inicializar array con precios de
productos
    double[] precios = {199.99, 299.5, 149.75, 399.0,
89.99};

    // b. Mostrar precios originales usando recursión
    System.out.println("Precios originales:");
    mostrarPreciosRecursivo(precios, 0); // Empezar desde
índice 0

    // c. Modificar el precio de un producto específico
(índice 2)
```

```
    precios[2] = 129.99; // Cambiar 149.75 por 129.99

    // d. Mostrar precios modificados usando recursión
    System.out.println("\nPrecios modificados:");
    mostrarPreciosRecursivo(precios, 0); // Empezar desde
índice 0 otra vez
}

/**
 * Función recursiva para mostrar todos los precios del
array
 * @param precios - array con los precios a mostrar
 * @param indice - posición actual en el array (para la
recursión)
 */
public static void mostrarPreciosRecursivo(double[]
precios, int indice) {

    // CASO BASE: Si llegamos al final del array, parar la
recursión
    if (indice >= precios.length) {
        return; // Salir del método (parar recursión)
    }

    // PASO RECURSIVO: Mostrar el precio actual
    System.out.println("Precio: $" + precios[indice]);

    // LLAMADA RECURSIVA: Llamarse a sí mismo con el
siguiente índice
    mostrarPreciosRecursivo(precios, indice + 1);
}

/**
 * Versión alternativa con explicación paso a paso
 * (solo para entender mejor cómo funciona)
 */
public static void
```

```
mostrarPreciosRecursivoConExplicacion(double[] precios,
int indice) {
    System.out.println("Llamada recursiva con índice: " +
indice);

    // CASO BASE: condición de parada
    if (indice >= precios.length) {
        System.out.println("Fin de la recursión (índice >=
longitud del array)");
        return;
    }

    // PASO ACTUAL: procesar elemento actual
    System.out.println("Procesando índice " + indice + ":
Precio: $" + precios[indice]);

    // LLAMADA RECURSIVA: continuar con el siguiente
elemento
    System.out.println("Llamando recursión para índice " +
(indice + 1));
    mostrarPreciosRecursivoConExplicacion(precios, indice
+ 1);

    System.out.println("Volviendo de la recursión del
índice " + indice);
}


}
```