UNIVERSIDAD TECNOLOGICA NACIONAL

Trabajo Práctico n° 2

Programación Estructurada

Comision 16

Macarena Marinoni <*marinonimacarena@gmail.com*>

Tecnicatura Universitaria en Programación - Universidad Tecnológica Nacional

Programación II

Profesor

Cortez, Alberto

Tutor

Bianchi, Neyén

15/09/2025

ÍNDICE

Objetivo general..... 3
Marco teórico..... 4
Estructuras condicionales.....5
Verificación de año bisiesto..... 5
Determinar mayor de tres números.....5
Clasificacion edad.....5
Calculadora de descuentos con categorías..... 6
Estructuras de Repetición.....6
Suma de numeros pares..... 6
Calculo precio final.....7
Envio total..... 7
Actualización de stock.....8
Calculo descuento especial..... 8
Arrays y Recursividad.....9
Modificación..... 9
Impresión recursiva.....9
Conclusiones.....10
ANEXO..... 11

Objetivo general

Este trabajo práctico reúne ejercicios básicos de programación estructurada en Java. El objetivo es 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

Este trabajo se apoya en cinco pilares de la programación estructurada: estructuras condicionales, ciclos, funciones, arrays y recursividad. A continuación se sintetiza cada concepto y su aplicación concreta dentro del proyecto.

Estructuras condicionales: Permiten decidir entre caminos de ejecución según una condición booleana. Usamos comparadores (<, <=, >, >=, ==, !=) y operadores lógicos (&&, ||, !). En el proyecto se aplican para la clasificación de edad por rangos y para la verificación de año bisiesto siguiendo la regla 400/100/4.

Ciclos (for, while, do-while): Sirven para repetir instrucciones. *for* es útil cuando conocemos la cantidad de iteraciones (p.ej., leer 10 números y contarlos por signo). *while* repite mientras una condición sea verdadera (acumular pares hasta ingresar 0). *do-while* garantiza al menos una ejecución antes de validar (pedir una nota hasta que esté entre 0 y 10). En todos los casos se usaron variables contadoras y acumuladores simples.

Funciones (métodos): Encapsulan cálculos y mejoran la modularidad y reutilización. En el trabajo se emplean funciones para el cálculo de precios con impuesto y descuento, para el costo de envío y para actualizar stock, separando la lectura de datos de la lógica de negocio. También se usa una constante de clase para evitar “números mágicos” (descuento especial).

Arrays: Estructura lineal que almacena elementos del mismo tipo en posiciones contiguas, indexadas desde 0. Se utilizaron para la gestión de precios de productos: recorrer con for, acceder por índice y modificar un valor puntual (tercer elemento).

Recursividad: Técnica donde una función se llama a sí misma definiendo un caso base (condición de corte) y un paso recursivo. Se aplicó para la impresión recursiva de arrays, mostrando cada elemento y avanzando en el índice hasta finalizar. Esta versión simple refuerza el concepto sin estructuras adicionales.

Estructuras condicionales

Verificación de año bisiesto

El ejercicio consiste en determinar si un año ingresado por teclado es bisiesto. Como entrada, el programa lee un número entero usando Scanner. La lógica implementa condicionales if/else y el operador módulo (%): si el año es divisible por 400, es bisiesto; en caso contrario, si es divisible por 100, no lo es; de no cumplirse, si es divisible por 4, es bisiesto; y en cualquier otro caso, no es bisiesto. Finalmente, como salida, muestra un mensaje indicando si el año “es bisiesto” o “no es bisiesto”.

El código utilizado puede visualizarse en el anexo de este documento o bien en el repositorio de GitHub https://github.com/MaquiMarinoni/TUPaD_P2-C22025.git

Determinar mayor de tres números

El ejercicio busca identificar el número mayor entre tres enteros ingresados por teclado. Como entrada, el programa solicita tres valores usando Scanner. La lógica asume inicialmente que el primero es el mayor y luego lo compara, de forma secuencial, con el segundo y el tercero mediante condicionales if; si encuentra un valor superior, actualiza la variable mayor. Finalmente, la salida muestra el mensaje “El mayor es: …” con el valor calculado.

El código utilizado puede visualizarse en el anexo de este documento o bien en el repositorio de GitHub https://github.com/MaquiMarinoni/TUPaD_P2-C22025.git

Clasificacion edad

El ejercicio clasifica a una persona en una categoría etaria a partir de su edad ingresada por teclado. Como entrada, el programa solicita un número entero con Scanner. La lógica utiliza condicionales if/else y comparaciones por rangos: menor de 12 “Niño”, entre 12 y 17 “Adolescente”, entre 18 y 59 “Adulto”, y desde 60 en adelante “Adulto mayor”. La salida muestra un mensaje con la categoría resultante.

El código utilizado puede visualizarse en el anexo de este documento o bien en el repositorio de GitHub https://github.com/MaquiMarinoni/TUPaD_P2-C22025.git

Calculadora de descuentos con categorías

El ejercicio calcula el precio final de un producto aplicando un descuento según su categoría ingresada por teclado. Como entrada, se toma el precio (número con decimales) y una letra de categoría: A, B o C. La lógica usa if/else para asignar el porcentaje de descuento (10%, 15% o 20%) y luego realiza operaciones aritméticas para obtener el monto descontado y el precio final. La salida muestra el porcentaje aplicado, el monto del descuento y el precio final.

El código utilizado puede visualizarse en el anexo de este documento o bien en el repositorio de GitHub https://github.com/MaquiMarinoni/TUPaD_P2-C22025.git

Estructuras de Repetición

Suma de numeros pares

El ejercicio solicita números enteros de forma repetida hasta que el usuario introduce 0, que actúa como señal de fin. La lógica usa un bucle while y una condición de corte cuando el valor leído es 0. En cada iteración, si el número es par (se verifica con n % 2 == 0), se acumula en una variable sumaPares. La salida muestra el total acumulado de todos los pares ingresados.

El código utilizado puede visualizarse en el anexo de este documento o bien en el repositorio de GitHub https://github.com/MaquiMarinoni/TUPaD_P2-C22025.git

Contador de positivos, negativos y ceros

El ejercicio solicita exactamente diez números enteros y clasifica cada uno como positivo, negativo o cero. La lógica usa un bucle for para repetir la lectura diez veces y condicionales if/else para actualizar los contadores positivos, negativos y ceros. Al finalizar, el programa muestra cuántos valores de cada tipo fueron ingresados

El código utilizado puede visualizarse en el anexo de este documento o bien en el repositorio de GitHub https://github.com/MaquiMarinoni/TUPaD_P2-C22025.git

Validación de nota

El ejercicio valida que una nota entera ingresada por teclado esté dentro del rango permitido de 0 a 10. La lógica utiliza un bucle do/while para forzar al menos una lectura y repetirla mientras el valor sea inválido. En cada iteración, un if verifica el rango y, si se sale de los límites, muestra un mensaje de error y vuelve a pedir la nota. Una vez obtenida una nota válida, el programa la muestra por pantalla.

El código utilizado puede visualizarse en el anexo de este documento o bien en el repositorio de GitHub https://github.com/MaquiMarinoni/TUPaD_P2-C22025.git

Calculo precio final

El ejercicio calcula el precio final de un producto aplicando un impuesto y un descuento ingresados por teclado. Como entrada, se leen el precio base y los porcentajes de impuesto y descuento; estos porcentajes se pasan a proporciones dividiéndolos por 100. La lógica se encapsula en una función calcularPrecioFinal que aplica la fórmula:

base + base*impuesto − base*descuento

La salida muestra el precio final formateado a dos decimales.

El código utilizado puede visualizarse en el anexo de este documento o bien en el repositorio de GitHub https://github.com/MaquiMarinoni/TUPaD_P2-C22025.git

Envio total

El ejercicio calcula el costo de envío de un producto en función del peso del paquete y la zona elegida (Nacional o Internacional), y luego obtiene el total a pagar sumando el precio del producto y del envío.

Como entrada, se lee el precio, el peso y la zona. La lógica usa condicionales para aplicar una tarifa por kilogramo ($5/kg para Nacional, $10/kg para Internacional) y dos funciones simples: una para el costo de envío y otra para el total.

La salida muestra el costo de envío y el total, formateados a dos decimales.

El código utilizado puede visualizarse en el anexo de este documento o bien en el repositorio de GitHub https://github.com/MaquiMarinoni/TUPaD_P2-C22025.git

Actualización de stock

El ejercicio actualiza el stock de un producto a partir de tres valores ingresados por teclado: stock actual, cantidad vendida y cantidad recibida. La lógica aplica una operación directa:

$$nuevo = actual - vendida + recibida$$

La salida informa el nuevo stock calculado.

El código utilizado puede visualizarse en el anexo de este documento o bien en el repositorio de GitHub https://github.com/MaquiMarinoni/TUPaD_P2-C22025.git

Calculo descuento especial

El ejercicio aplica un descuento fijo (10%) a un precio ingresado por teclado usando una constante de clase que actúa como “variable global” del programa. Como entrada, se recibe el precio del producto con Scanner. La lógica realiza operaciones aritméticas para calcular el monto descontado y el precio final con la ecuación:

$$precio - precio*DESCUENTO\_ESPECIAL$$

La salida muestra el descuento aplicado y el precio final formateado a dos decimales.

El código utilizado puede visualizarse en el anexo de este documento o bien en el repositorio de GitHub https://github.com/MaquiMarinoni/TUPaD_P2-C22025.git

Arrays y Recursividad

Modificación

El ejercicio trabaja con un arreglo de precios: primero muestra su contenido original y luego modifica un elemento específico accediendo por su índice. La entrada de datos no es necesaria porque se cargan valores de ejemplo directamente en el array. La lógica utiliza un bucle for para recorrer el arreglo y el acceso por índice (recordando que comienza en 0) para actualizar el tercer valor. No se emplean métodos auxiliares, sólo arrays, for y operaciones básicas de lectura/escritura en consola.

El código utilizado puede visualizarse en el anexo de este documento o bien en el repositorio de GitHub https://github.com/MaquiMarinoni/TUPaD_P2-C22025.git

Impresión recursiva

El ejercicio muestra cómo usar recursividad para recorrer e imprimir un arreglo. Se parte de un array de precios cargado en el código; primero se imprime completo, luego se modifica el tercer elemento y se vuelve a imprimir. La función `imprimirRec(arr, i)` usa un caso base (cuando `i` alcanza `arr.length`) para detenerse y un paso recursivo que muestra el elemento actual y llama a la misma función con `i + 1`.

El código utilizado puede visualizarse en el anexo de este documento o bien en el repositorio de GitHub https://github.com/MaquiMarinoni/TUPaD_P2-C22025.git

Conclusiones

Este trabajo permitió afianzar lectura por consola, operadores, condicionales, bucles y funciones simples en Java. Las soluciones priorizan claridad, validaciones básicas y salida legible. El uso de constantes y funciones ayudó a evitar duplicación, mientras que arrays y recursividad introdujeron estructuras y técnicas de recorrido con ejemplos sencillos.

ANEXO

01.Bisiesto

```
/*
                                    *                                    Click
nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt
to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Main.java to
edit this template
 */
package pkg01.bisiesto;
import java.util.Scanner;


/**
 *
 * @author marin
 */
public class Bisiesto {


    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // Creamos el Scanner para leer desde teclado
```

```
Scanner sc = new Scanner(System.in);


// Pedimos el año al usuario
System.out.print("Ingrese un año: ");
int anio = sc.nextInt();
// Variable booleana para guardar si es bisiesto o no
boolean esBisiesto;


// Reglas de año bisiesto (en este orden):
// 1) Si es divisible por 400 -> bisiesto
// 2) Si no, si es divisible por 100 -> NO bisiesto
// 3) Si no, si es divisible por 4 -> bisiesto
// 4) En cualquier otro caso -> NO bisiesto
if (anio %400 == 0) {
    esBisiesto = true;
} else if (anio %100 == 0) {
    esBisiesto = true;
} else esBisiesto = anio % 4 == 0;


// Mostramos el resultado por pantalla
if (esBisiesto) {
    System.out.println("El año " + anio + " es bisiesto.");
```

```
        } else {
            System.out.println("El año " + anio + " no es bisiesto.");
        }

        sc.close();
    }
}
```

02.DeterminarMayor

```
/*
                                    *                                    Click
nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt
to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Main.java to
edit this template
 */
package pkg02.determinarmayor;

import java.util.Scanner;

/**
 *
 * @author marin
```

```
 */
public class DeterminarMayor {


    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
         // Creamos el Scanner para leer desde teclado
        Scanner sc = new Scanner(System.in);


        // Leemos tres números enteros
        System.out.print("Ingrese el primer número: ");
        int a = sc.nextInt();


        System.out.print("Ingrese el segundo número: ");
        int b = sc.nextInt();


        System.out.print("Ingrese el tercer número: ");
        int c = sc.nextInt();


          // Suponemos que el mayor es el primero y comparamos con los
demás
```

```
        int mayor = a;
        if (b > mayor) {
            mayor = b;
        }
        if (c > mayor) {
            mayor = c;
        }

        // Mostramos el resultado
        System.out.println("El mayor es: " + mayor);

        // Cerramos el Scanner
        sc.close();
    }

}
```

03.ClasificarEdad

```
/*
                                     *                                     Click
nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt
to change this license
```

```
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Main.java to
edit this template
 */
package pkg03.clasificacionedad;

import java.util.Scanner;

/**
 *
 * @author marin
 */
public class ClasificacionEdad {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // Creamos el Scanner para leer desde teclado
        Scanner sc = new Scanner(System.in);

        // Leemos la edad (entero)
        System.out.print("Ingrese su edad: ");
```

```
        int edad = sc.nextInt();


        // Determinamos la categoría según rangos simples
        String clasificacion;
        if (edad < 12) {
            clasificacion = "Niño";
        } else if (edad <= 17) {
            clasificacion = "Adolescente";
        } else if (edad <= 59) {
            clasificacion = "Adulto";
        } else {
            clasificacion = "Adulto mayor";
        }


        // Mostramos el resultado
        System.out.println("Eres un " + clasificacion + ".");


        // Cerramos el Scanner
        sc.close();
    }

}
```

04.CalculadoraDescuentos

```
/*
                                    *                                   Click
nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt
to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Main.java to
edit this template
 */
package pkg04.calculadoradescuentos;


import java.util.Scanner;


/**
 *
 * @author marin
 */
public class CalculadoraDescuentos {


    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
```

```
        // Creamos el Scanner para leer desde teclado
        Scanner sc = new Scanner(System.in);

        // Leemos el precio (double) y la categoría (A/B/C)
        System.out.print("Ingrese el precio del producto: ");
        double precio = sc.nextDouble();

        System.out.print("Ingrese la categoría (A, B o C): ");
           String textoCat = sc.next(); // leemos una palabra (primer caracter alcanza)
        char categoria = Character.toUpperCase(textoCat.charAt(0));

        // Definimos el porcentaje de descuento según la categoría
        double porcentaje; // en proporción: 0.10 = 10%
        if (categoria == 'A') {
            porcentaje = 0.10;
        } else if (categoria == 'B') {
            porcentaje = 0.15;
        } else if (categoria == 'C') {
            porcentaje = 0.20;
        } else {
            // Si la categoría no es válida, informamos y terminamos
```

```
                System.out.println("Categoría inválida. Debe ser A, B o
C.");

            sc.close();

            return;

        }


        // Calculamos descuento y precio final

        double descuento = precio * porcentaje;

        double precioFinal = precio - descuento;


        // redondeo a 2 decimales (valor)

        precioFinal = Math.round(precioFinal * 100.0) / 100.0;


        // Mostramos resultados

         System.out.println("Descuento aplicado: " + (int)(porcentaje *
100) + "%");

        System.out.println("Monto de descuento: " + descuento);

        System.out.println("Precio final: " + precioFinal);


        // Cerramos el Scanner

        sc.close();

    }
```

```
}
```

05.SumaPares

```
/*
                                    *                                   Click
nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt
to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Main.java to
edit this template
 */
package pkg05.sumapares;

import java.util.Scanner;

/**
 *
 * @author marin
 */
public class SumaPares {

    /**
     * @param args the command line arguments
     */
```

```
public static void main(String[] args) {
    // Creamos el Scanner para leer desde teclado
    Scanner sc = new Scanner(System.in);

    // Acumulador para la suma de pares
    int sumaPares = 0;

    // Leemos números hasta que el usuario ingrese 0
    // Solo sumamos los que sean pares (n % 2 == 0)
    while (true) {
        System.out.print("Ingrese un número (0 para terminar): ");
        int n = sc.nextInt();

        if (n == 0) {
            // 0 es la señal para cortar el ciclo (no se suma)
            break;
        }

        // Si es par, lo agregamos a la suma
        if (n % 2 == 0) {
            sumaPares += n;
        }
```

```
        }


        // Mostramos el resultado

            System.out.println("La suma de los números pares es: " +
sumaPares);


        // Cerramos el Scanner

        sc.close();

    }


}
```

06.ContadorClasificado

```
/*

                                    *                                   Click
nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt
to change this license

 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Main.java to
edit this template

 */

package pkg06.contadorclasificado;


import java.util.Scanner;
```

```
/**
 *
 * @author marin
 */
public class ContadorClasificado {


    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
         // Creamos el Scanner para leer desde teclado
        Scanner sc = new Scanner(System.in);


        // Contadores para cada tipo
        int positivos = 0;
        int negativos = 0;
        int ceros = 0;


        // Leemos exactamente 10 números enteros
        for (int i = 1; i <= 10; i++) {
            System.out.print("Ingrese el número " + i + ": ");
            int n = sc.nextInt();
```

```
                // Clasificamos el número y actualizamos el contador
correspondiente
            if (n > 0) {
                positivos++;
            } else if (n < 0) {
                negativos++;
            } else {
                ceros++;
            }
        }

        // Mostramos los resultados
        System.out.println("Positivos: " + positivos);
        System.out.println("Negativos: " + negativos);
        System.out.println("Ceros: " + ceros);

        // Cerramos el Scanner
        sc.close();
    }

}
```

07.ValidarNota

```
/*
                                    *                                    Click
nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt
to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Main.java to
edit this template
 */
package pkg07.validarnota;


import java.util.Scanner;


/**
 *
 * @author marin
 */
public class ValidarNota {


    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
         // Creamos el Scanner para leer desde teclado
```

```
        Scanner sc = new Scanner(System.in);


        int nota; // guardará la nota válida


          // Repetimos la lectura hasta que el valor esté entre 0 y 10
(inclusive)
        do {
            System.out.print("Ingrese una nota (0-10): ");
            nota = sc.nextInt();


            // Si está fuera de rango, avisamos y volvemos a pedir
            if (nota < 0 || nota > 10) {
                  System.out.println("Valor inválido. Debe estar entre 0
y 10.");
            }
        } while (nota < 0 || nota > 10);


        // Mostramos el resultado cuando la nota es válida
        System.out.println("Nota válida: " + nota);


        // Cerramos el Scanner
        sc.close();
    }
```

```
}
```

08.PrecioFinal

```
/*
                                    *                                   Click
nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt
to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Main.java to
edit this template
 */
package pkg08.preciofinal;

import java.util.Scanner;

/**
 *
 * @author marin
 */
public class PrecioFinal {

    /**
     * @param args the command line arguments
```

```
     */

    // Función que calcula el precio final

     // impuesto y descuento se reciben como proporción (ej.: 0.10 para
10%)

         static double calcularPrecioFinal(double precioBase, double
impuesto, double descuento) {

        // Precio final = base + (base * impuesto) - (base * descuento)

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

    }



    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);



        // Leemos el precio base y los porcentajes

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

        double precioBase = sc.nextDouble();



        System.out.print("Impuesto % (ej: 10): ");

        double impPorc = sc.nextDouble();



        System.out.print("Descuento % (ej: 5): ");

        double descPorc = sc.nextDouble();
```

```
        // Convertimos a proporción (dividimos por 100)
        double impuesto = impPorc / 100.0;
        double descuento = descPorc / 100.0;

        // Llamamos a la función y obtenemos el resultado
         double precioFinal = calcularPrecioFinal(precioBase, impuesto,
descuento);

        // Mostramos el precio final (formateado a 2 decimales)
        System.out.printf("Precio final: %.2f%n", precioFinal);

        sc.close();
    }
}
```

09.EnvioTotal

```
/*
                                    *                                   Click
nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt
to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Main.java to
edit this template
 */
```

```
package pkg09.enviototal;


import java.util.Scanner;


/**
 *
 * @author marin
 */
public class EnvioTotal {


    /**
     * @param args the command line arguments
     */
    // Función simple para calcular el costo de envío según zona y peso
    static double calcularCostoEnvio(double pesoKg, String zona) {
        if (zona.equalsIgnoreCase("Nacional")) {
                return 5.0 * pesoKg;          // tarifa base por kg para
envíos nacionales
        } else if (zona.equalsIgnoreCase("Internacional")) {
                return 10.0 * pesoKg;         // tarifa base por kg para
envíos internacionales
        } else {
            return -1.0;                 // señal de zona inválida
```

```
        }
    }

    // Función simple para sumar producto + envío
      static double calcularTotalCompra(double precioProducto, double
costoEnvio) {
        return precioProducto + costoEnvio;
    }

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

        // Leemos precio del producto y peso del paquete
        System.out.print("Precio del producto: ");
        double precio = sc.nextDouble();

        System.out.print("Peso del paquete (kg): ");
        double peso = sc.nextDouble();

        // Leemos la zona como texto (Nacional/Internacional)
        System.out.print("Zona (Nacional/Internacional): ");
        String zona = sc.next(); // una palabra alcanza
```

```
        // Calculamos el costo de envío
        double envio = calcularCostoEnvio(peso, zona);
        if (envio < 0) {
               System.out.println("Zona inválida. Debe ser 'Nacional' o
'Internacional'.");
            sc.close();
            return;
        }

        // Calculamos el total a pagar
        double total = calcularTotalCompra(precio, envio);

        // Mostramos resultados (con dos decimales)
        System.out.printf("Costo de envío: %.2f%n", envio);
        System.out.printf("Total a pagar: %.2f%n", total);

        sc.close();
    }
}
```

10.ActualizarStock

```
/*
                                    *                                    Click
nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt
to change this license
  *  Click  nbfs://nbhost/SystemFileSystem/Templates/Classes/Main.java
to edit this template
 */
package pkg10.actualizarstock;

import java.util.Scanner;

/**
 *
 * @author marin
 */
public class ActualizarStock {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
// Creamos el Scanner para leer desde teclado
        Scanner sc = new Scanner(System.in);
```

```
        // Leemos los datos necesarios
        System.out.print("Stock actual: ");
        int stockActual = sc.nextInt();

        System.out.print("Cantidad vendida: ");
        int cantidadVendida = sc.nextInt();

        System.out.print("Cantidad recibida (reposición): ");
        int cantidadRecibida = sc.nextInt();

        // Fórmula: nuevo stock = stock actual - vendida + recibida
                int nuevoStock = stockActual - cantidadVendida +
cantidadRecibida;

        // Mostramos el resultado
            System.out.println("El nuevo stock del producto es: " +
nuevoStock);

        // Cerramos el Scanner
        sc.close();
    }
}
```

11.DescuentoEspecial

```
/*
                                    *                                    Click
nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt
to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Main.java to
edit this template
 */
package pkg11.descuentoespecial;


import java.util.Scanner;


/**
 *
 * @author marin
 */
public class DescuentoEspecial {


    /**
     * @param args the command line arguments
     */
    // Constante "global": define el % de descuento fijo del sistema.
```

```
      // Se declara a nivel de clase para evitar "números mágicos" y poder cambiarlo en un solo lugar.
    private static final double DESCUENTO_ESPECIAL = 0.10;


    public static void main(String[] args) {
        // Creamos el Scanner para leer desde teclado
        Scanner sc = new Scanner(System.in);


        // Leemos el precio del producto (double)
        System.out.print("Ingrese el precio del producto: ");
        double precio = sc.nextDouble();


        // Calculamos el monto descontado y el precio final
        double montoDescuento = precio * DESCUENTO_ESPECIAL;
        double precioFinal = precio - montoDescuento;


        // Mostramos resultados (con 2 decimales)
             System.out.printf("Descuento aplicado (%.0f%%): %.2f%n", DESCUENTO_ESPECIAL * 100, montoDescuento);
        System.out.printf("Precio final: %.2f%n", precioFinal);


        // Cerramos el Scanner
        sc.close();
```

```
    }
}
```

12.ModificarArrays

```
/*
                                    *                                    Click
nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt
to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Main.java to
edit this template
 */
package pkg12.modificararrays;


/**
 *
 * @author marin
 */
public class ModificarArrays {


    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
```

```
            // Creamos un arreglo (array) de precios con valores de
ejemplo
        double[] precios = {199.99, 299.50, 149.75, 399.00, 89.99};


        // Mostramos los precios originales
        System.out.println("Precios originales:");
        for (int i = 0; i < precios.length; i++) {
            System.out.println("precio[" + i + "] = " + precios[i]);
        }


          // Modificamos el tercer elemento (índice 2, porque el índice
empieza en 0)
        precios[2] = 129.99;


        // Mostramos los precios luego del cambio
        System.out.println("Precios modificados:");
        for (int i = 0; i < precios.length; i++) {
            System.out.println("precio[" + i + "] = " + precios[i]);
        }
    }
}
```

13.RecursividadArrays

```
/*
                                    *                                    Click
   nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.
   txt to change this license
  *  Click  nbfs://nbhost/SystemFileSystem/Templates/Classes/Main.java
   to edit this template
 */
package pkg13.recursividadarrays;


/**
 *
 * @author marin
 */
public class RecursividadArrays {


    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // Array de ejemplo con precios
        double[] precios = {199.99, 299.50, 149.75, 399.00, 89.99};


        // Imprimimos el array original usando recursividad
```

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

        // Modificamos el tercer elemento (índice 2)
        precios[2] = 129.99;

        // Imprimimos el array modificado usando recursividad
        System.out.println("Precios modificados:");
        imprimirRec(precios, 0);
    }

    // Función recursiva que imprime arr[i] y luego avanza
    static void imprimirRec(double[] arr, int i) {
        // Caso base: si i llegó al final, cortamos la recursión
        if (i >= arr.length) {
            return;
        }
        // Paso recursivo: mostramos el elemento actual y llamamos con i+1
        System.out.println("precio[" + i + "] = " + arr[i]);
        imprimirRec(arr, i + 1);
    }
}
```