PROGRAMACIÓN II
Trabajo Práctico 2: Programación Estructurada

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.

```
import java.util.Scanner;

public class ejercicio1 {

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

        System.out.print("Ingrese un año: ");
        int anio = sc.nextInt();

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

        sc.close();
    }

    static boolean esBisiesto(int anio) {
        if (anio % 400 == 0) {
            return true;
        }
        if (anio % 100 == 0) {
            return false;
        }
        return anio % 4 == 0;
    }
}
```

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

```
import java.util.Scanner;

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

        System.out.print("Ingrese el primer número: ");
        int n1 = sc.nextInt();

        System.out.print("Ingrese el segundo número: ");
        int n2 = sc.nextInt();

        System.out.print("Ingrese el tercer número: ");
        int n3 = sc.nextInt();

        int mayor = obtenerMayor(n1, n2, n3);

        System.out.println("El mayor es: " + mayor);

        sc.close();
    }

    static int obtenerMayor(int a, int b, int c) {
        int mayor = a
        if (b > mayor) mayor = b
        if (c > mayor) mayor = c
        return mayor;
    }
}
```

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.

```
import java.util.Scanner;

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

        System.out.print("Ingrese su edad: ");
        int edad = sc.nextInt();

        String categoria = clasificarEdad(edad);

        System.out.println("Eres un " + categoria + ".");

        sc.close();
    }

    static String clasificarEdad(int edad) {
        if (edad < 12) {
            return "Niño";
        } else if (edad <= 17) {
            return "Adolescente";
        } else if (edad <= 59) {
            return "Adulto";
        } else {
            return "Adulto mayor";
        }
    }
}
```

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

```
import java.util.Scanner;

public class Main {

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

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

        System.out.print("Ingrese la categoría del producto (A,
B o C): ");
        String categoria = sc.next().toUpperCase();

        double descuento =
obtenerDescuentoPorCategoria(categoria);

        if (descuento == -1) {
            System.out.println("No existe esa categoria.");
            return;
        }

        double precioFinal = calcularPrecioConDescuento(precio,
categoria);

        System.out.println("Descuento aplicado: " + (int)
(descuento * 100) + '%');
        System.out.println("Precio final: " + precioFinal);

        sc.close();
```

```
    }

    static double calcularPrecioConDescuento(double precio,
String categoria) {
        double descuento =
obtenerDescuentoPorCategoria(categoria);
        return precio - (precio * descuento);
    }

    static double obtenerDescuentoPorCategoria(String categoria)
{
        switch (categoria) {
            case "A":
                return 0.10;
            case "B":
                return 0.15;
            case "C":
                return 0.20;
            default:
                return -1;
        }
    }
}
```

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

```
import java.util.Scanner;

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

        int sumaPares = 0;
        int num;

        System.out.print("Ingrese un número (0 para terminar):
");
        num = sc.nextInt();

        // Repite mientras no sea 0
        while (num != 0) {
            if (esPar(num)) {
                sumaPares += num; // acumulo sólo los pares
            }

            System.out.print("Ingrese un número (0 para
terminar): ");
            num = sc.nextInt();
        }

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

    static boolean esPar(int n) {
        return n % 2 == 0;
    }
}
```

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

```
import java.util.Scanner;

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

        int positivos = 0;
        int negativos = 0;
        int ceros = 0;

        for (int i = 1; i <= 10; i++) {
            System.out.print("Ingrese el número " + i + ":
");
            int num = sc.nextInt();

            if (num > 0) {
                positivos++;
            } else if (num < 0) {
                negativos++;
            } else {
                ceros++;
            }
        }

        System.out.println("Resultados:");
        System.out.println("Positivos: " + positivos);
        System.out.println("Negativos: " + negativos);
        System.out.println("Ceros: " + ceros);

        sc.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.

```
import java.util.Scanner;

public class Main {

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

        do {
            System.out.print("Ingrese una nota (0-10): ");
            nota = sc.nextInt();

            if (nota < 0 || nota > 10) {
                System.out.println("Error: Nota inválida.
Ingrese una nota entre 0 y 10.");
            }

        } while (nota < 0 || nota > 10);

        System.out.println("Nota guardada
correctamente.");

        sc.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

```
import java.util.Scanner;

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

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

        System.out.print("Ingrese el impuesto en porcentaje (Ejemplo: 10
para 10%): ");
        double impuesto = sc.nextDouble();

        System.out.print("Ingrese el descuento en porcentaje (Ejemplo: 5
para 5%): ");
        double descuento = sc.nextDouble();

        double precioFinal = calcularPrecioFinal(precioBase, impuesto,
descuento);

        System.out.println("El precio final del producto es: " +
precioFinal);

        sc.close();
    }


    static double calcularPrecioFinal(double precioBase, double impuesto,
double descuento) {
        double imp = precioBase * (impuesto / 100);
        double desc = precioBase * (descuento / 100);
        return precioBase + imp - desc;
    }
}
```

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

a. **calcularCostoEnvio(double peso, String zona)**: Calcula el costo de envío basado en la zona de envío (Nacional o Internacional) y el peso del paquete.

Nacional: $5 por kg

Internacional: $10 por kg

b. **calcularTotalCompra(double precioProducto, double**

costoEnvio): Usa **calcularCostoEnvio** para sumar el costo del producto con el costo de envío.

Desde **main()**, solicita el peso del paquete, la zona de envío y el precio del producto. Luego, muestra el total a pagar.

Ejemplo de entrada/salida:
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

```
import java.util.Scanner;

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

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

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

        System.out.print("Ingrese la zona de envío
(Nacional/Internacional): ");
        String zona = sc.next();

        double costoEnvio = calcularCostoEnvio(peso,
zona);
        if (costoEnvio == -1) {
            System.out.println("Zona inválida.");
            sc.close();
            return;
        }

        double total = calcularTotalCompra(precioProducto,
costoEnvio);

        System.out.println("El costo de envío es: " +
costoEnvio);
        System.out.println("El total a pagar es: " +
total);

        sc.close();
    }
```

```
    static double calcularCostoEnvio(double peso, String
zona) {
        if (zona.equalsIgnoreCase("Nacional")) {
            return peso * 5;
        } else if (zona.equalsIgnoreCase("Internacional"))
{
            return peso * 10;
        } else {
            return -1;
        }
    }

    static double calcularTotalCompra(double
precioProducto, double costoEnvio) {
        return precioProducto + costoEnvio;
    }
}
```

10. Actualización de stock a partir de venta y recepción de productos. Crea un método **actualizarStock(int stockActual, int cantidadVendida, int cantidadRecibida)**, que calcule el nuevo stock después de una venta y recepción de productos:

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

```
import java.util.Scanner;

public class Main {

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

        System.out.print("Ingrese el stock actual del producto: ");
        int stockActual = sc.nextInt();

        System.out.print("Ingrese la cantidad vendida: ");
        int cantidadVendida = sc.nextInt();

        System.out.print("Ingrese la cantidad recibida: ");
        int cantidadRecibida = sc.nextInt();

        int nuevoStock = actualizarStock(stockActual, cantidadVendida,
cantidadRecibida);
        if (nuevoStock < 0) {
            nuevoStock = 0;
        }
        System.out.println("El nuevo stock del producto es: " +
nuevoStock);

        sc.close();
    }

    // NuevoStock = StockActual – CantidadVendida + CantidadRecibida
    static int actualizarStock(int stockActual, int cantidadVendida, int
cantidadRecibida) {
        return stockActual - cantidadVendida + cantidadRecibida;
    }
}
```

11. Cálculo de descuento especial usando variable global.

Declara una variable global **Ejemplo de entrada/salida:** = 0.10. Luego, crea un método **calcularDescuentoEspecial(double precio)** que use la variable global para calcular el descuento especial del 10%.

Dentro del método, declara una variable local **descuentoAplicado**, almacena el valor del descuento y muestra el precio final con descuento.

Ejemplo de entrada/salida:

Ingrese el precio del producto: 200

El descuento especial aplicado es: 20.0

El precio final con descuento es: 180.0

```
import java.util.Scanner;

public class Main {

    static final double DESCUENTO_ESPECIAL = 0.10; // 10%

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

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

        if (precio < 0) {
            System.out.println("El precio no puede ser negativo.");
            sc.close();
            return;
        }

        calcularDescuentoEspecial(precio);

        sc.close();
    }

    static double calcularDescuentoEspecial(double precio) {
        double descuentoAplicado = precio * DESCUENTO_ESPECIAL;
        double precioFinal = precio - descuentoAplicado;

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

        return 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.

```
public class Main {

    public static void main(String[] args) {

        double[] precios = {199.99, 299.5, 149.75, 399.0, 89.99};

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

        int indiceAModificar = 2;
        double nuevoPrecio = 129.99;
        precios[indiceAModificar] = nuevoPrecio;

        System.out.println("\nPrecios modificados:");
        mostrarPrecios(precios);
    }

    static void mostrarPrecios(double[] arr) {
        for (int i = 0; i < arr.length; i++) {
            System.out.println("Precio: $" + arr[i]);
        }
    }
}
```

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

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.

```
public class Main {

    public static void main(String[] args) {
        double[] precios = {199.99, 299.5, 149.75, 399.0, 89.99};

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

        int indiceAModificar = 2;
        precios[indiceAModificar] = 129.99;

        System.out.println("\nPrecios modificados:");
        imprimirRecursivo(precios, 0);
    }

    static void imprimirRecursivo(double[] arr, int indice) {
        if (indice == arr.length) {
            return;
        }

        System.out.println("Precio: $" + arr[indice]);
        imprimirRecursivo(arr, indice + 1);
    }
}
```