PROGRAMACIÓN II
Trabajo Práctico 2: Programación Estructurada

Alumno:

Franco Sarrú

Repositorio público – GitHub:

https://github.com/fsarru/Programacion2.git

OBJETIVO GENERAL

Desarrollar habilidades en programación estructurada en Java, abordando desde conceptos básicos como operadores y estructuras de control hasta temas avanzados como funciones, recursividad y estructuras de datos. Se busca fortalecer la capacidad de análisis y solución de problemas mediante un enfoque práctico,

MARCO TEÓRICO

Concepto	Aplicación en el proyecto
Estructuras condicionales	Clasificación de edad, verificación de año bisiesto
Ciclos (for, while, do-while)	Repetición de ingreso de datos y cálculos
Funciones	Cálculo modular de descuentos, envíos, stock
Arrays	Gestión de precios de productos
Recursividad	Impresión recursiva de arrays

Caso Práctico

Desarrollar los siguientes ejercicios en Java utilizando el paradigma de programación estructurada. Agrupados según el tipo de estructuras o conceptos aplicados:

Estructuras Condicionales:

1. Verificación de Año Bisiesto.

Escribe un programa en Java que solicite al usuario un año y determine si es bisiesto. Un año es bisiesto si es divisible por 4, pero no por 100, salvo que sea divisible por 400.

Ejemplo de entrada/salida:

Ingrese un año: 2024

El año 2024 es bisiesto.

Ingrese un año: 1900

El año 1900 no es bisiesto.

```
package sarru_tp2;


import java.util.Scanner;



public class Sarru_TP2_EJ1 {


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

```
        Scanner scanner = new Scanner(System.in);
        System.out.print("Ingresa un año: ");
        int año = scanner.nextInt();

        if ((año % 4 == 0 && año % 100 != 0) || (año % 400 == 0)) {
            System.out.println(año + " es un año bisiesto.");
        } else {
            System.out.println(año + " no es un año bisiesto.");
        }

        scanner.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.

Ejemplo de entrada/salida:

Ingrese el primer número: 8

Ingrese el segundo número: 12

Ingrese el tercer número: 5

El mayor es: 12

```
package sarru_tp2_ej2;

import java.util.Scanner;
```

```
public class Sarru_TP2_EJ2 {

/*Ejercicio 2: 2.          Determinar el Mayor de Tres Números.

Escribe un programa en Java que pida al usuario tres números enteros y
determine cuál es el mayor.  */


   public static void main(String[] args) {

      Scanner scanner = new Scanner(System.in);

      System.out.print("Ingresa el primer número: ");

      int num1 = scanner.nextInt();


      System.out.print("Ingresa el segundo número: ");

      int num2 = scanner.nextInt();


      System.out.print("Ingresa el tercer número: ");

      int num3 = scanner.nextInt();


      int mayor = num1;


      if (num2 > mayor) {

         mayor = num2;

      }


      if (num3 > mayor) {

         mayor = num3;

     }
```

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


        scanner.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"

Ejemplo de entrada/salida:

Ingrese su edad: 25

Eres un Adulto.

Ingrese su edad: 10

Eres un Niño.

```
package sarru_tp2_ej3;


import java.util.Scanner;


public class Sarru_TP2_EJ3 {


    /*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"

```
 */
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Ingresa su edad: ");
        int edad = scanner.nextInt();

        if (edad < 12){
            System.out.println("Eres un Niño");
        } else if (edad >= 12 && edad < 17) {
            System.out.println("Eres un Adolescente");
        } else if (edad >= 18 && edad < 59) {
            System.out.println("Eres un Adulto");
        } else {
            System.out.println("Eres un 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

```
package sarru_tp2_ej4;


import java.util.Scanner;


/*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
```

```
 */
public class Sarru_TP2_EJ4 {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Ingresa el precio de su producto: ");
        double precio = scanner.nextDouble();

        System.out.print("Ingresa la categoria (A,B o C): ");
        char categoria = scanner.next().charAt(0);;

        double descuento= 0.0;
        double precioFinal = 0.0;

        if (categoria == 'A' || categoria == 'a'){
            descuento = precio * 0.10;
        }
        else if (categoria == 'B' || categoria ==  'b'){
            descuento = precio * 0.20;
        }

        else if (categoria == 'C' || categoria ==  'c'){
            descuento = precio * 0.30;
        }
        else {
            System.out.println("No se reconoce la categoría");
```

```
        }

        precioFinal = precio - descuento;

        System.out.println("Precio original: $" + precio);
        System.out.println("Descuento aplicado: $" + descuento);
        System.out.println("Precio final: $" + precioFinal);

    }

}
```

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

```
public class Sarru_TP2_EJ5 {

    /*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.

     */
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int num = 0;
        int pares = 0;

        System.out.println("Ingresa un número (ingresa 0 para finalizar):");

        num = scanner.nextInt();

        while (num != 0) {
            if (num % 2 == 0) {
                pares += num;
            }

            System.out.println("Ingresa otro número (ingresa 0 para finalizar):");
            num = scanner.nextInt();
        }

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

        scanner.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.

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

```
package sarru_tp2_ej6;

import java.util.Scanner;

public class Sarru_TP2_EJ6 {
```

```
/*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. */

  public static void main(String[] args) {

    Scanner scanner = new Scanner(System.in);

      int num = 0;

      int positivos = 0;

      int negativos = 0;

      int ceros = 0;


      System.out.print("Ingresa 10 números enteros: ");

      for(int i = 1;i <= 10; i++){

         System.out.print("Ingrese el número " + i + ": ");

         num = scanner.nextInt();


         if (num > 0) {

            positivos++;

         } else if (num < 0) {

            negativos++;

         } else {

            ceros++;

         }

      }


      System.out.println("Hay " + positivos + " números positivos.");

      System.out.println("Hay " + negativos + " números negativos.");

      System.out.println("Hay " + ceros + " ceros");
```

```
        scanner.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 sarru_tp2_ej7;

import java.util.Scanner;

/*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.  */
public class Sarru_TP2_EJ7 {

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

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

        int nota;


        do {

            System.out.print("Ingresa una nota entre 0 y 10: ");

            nota = scanner.nextInt();


            if (nota < 0 || nota > 10) {

                System.out.println("Nota inválida. Por favor, ingrese un valor
entre 0 y 10.");

            }

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

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


        scanner.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 sarru_tp2_ej8;


import java.util.Scanner;


/*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.  */

public class Sarru_TP2_EJ8 {


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

        double precioFinal = precioBase + (precioBase * impuesto) - (precioBase * descuento);

        return precioFinal;

    }

    public static void main(String[] args) {

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

```
        System.out.print("Ingresa el precio base del producto: ");
        double precioBase = scanner.nextDouble();

        System.out.print("Ingresa el porcentaje de impuesto (ej. 10 para 10%): ");
        double porcentajeImpuesto = scanner.nextDouble();

        double impuesto = porcentajeImpuesto / 100;

        System.out.print("Ingresa el porcentaje de descuento (ej. 15 para 15%): ");
        double porcentajeDescuento = scanner.nextDouble();

        double descuento = porcentajeDescuento / 100;

        double precioFinal = calcularPrecioFinal(precioBase, impuesto, descuento);
        System.out.println("Precio final del producto: $" + precioFinal);

        scanner.close();
    }
}
```

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

```
package sarru_tp2_ej9;


import java.util.Scanner;


/*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

*/

```
public class Sarru_TP2_EJ9 {


    public static double calcularCostoEnvio(double peso, String zona) {
        double costo = 0.0;
        if (zona.equalsIgnoreCase("Nacional")) { //Encontré esta función
para evitar problemas con mayúsculas y mínusculas
            costo = peso * 5;
        } else if (zona.equalsIgnoreCase("Internacional")) {
            costo = peso * 10;
        }
        return costo;
    }



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


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

```
        // Solicitar datos al usuario
        System.out.print("Ingrese el precio del producto: ");
        double precioProducto = scanner.nextDouble();

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

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

        double costoEnvio = calcularCostoEnvio(peso, zona);

        double totalCompra = calcularTotalCompra(precioProducto,
costoEnvio);

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

        scanner.close();
    }
}
```

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 sarru_tp2_ej10;


import java.util.Scanner;


/*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

```
 */
public class Sarru_TP2_EJ10 {

    public static int actualizarStock(int stockActual, int cantidadVendida,
int cantidadRecibida) {
        int nuevoStock = stockActual - cantidadVendida +
cantidadRecibida;
        return nuevoStock;
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Ingrese el stoc actual: ");
        int stockActual = scanner.nextInt();
        System.out.print("Ingrese la cantidad de ventas: ");
        int cantidadVendida = scanner.nextInt();
        System.out.print("Ingrese la cantidad recibida: ");
        int cantidadRecibida = scanner.nextInt();

        int stock =
actualizarStock(stockActual,cantidadVendida,cantidadRecibida);

        System.out.println("El stock actual es: " + stock);
        scanner.close();

    }
```

```
}
```

---

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

```
package sarru_tp2_ej11;


import java.util.Scanner;


/*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
```

```
 */
public class Sarru_TP2_EJ11 {

    public static final double DESCUENTO_ESPECIAL = 0.10;

    public 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;
    }

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

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

        calcularDescuentoEspecial(valor);

        scanner.close();
    }
```

```
}
```

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 sarru_tp2_ej12;


/**
 *
 * @author franc
 */
public class Sarru_TP2_EJ12 {


    /*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 static void main(String[] args) {

    double[] precios = {199.99, 299.50, 149.75, 399.00, 89.99};

    System.out.println("Precios originales:");
    for (double precio : precios) {
        System.out.println("Precio: $" + precio);
    }

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

    System.out.println("Precios modificados:");
    for (double precio : precios) {
        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

Conceptos Clave Aplicados:

✔ Uso de arrays (double[]) para almacenar valores.

✔ Recorrido del array con una función recursiva en lugar de un bucle.

✔ Modificación de un valor en un array mediante un índice.

✔ Uso de un índice como parámetro en la recursión para recorrer el array.

CONCLUSIONES ESPERADAS

- Aplicar estructuras de control y decisión para resolver problemas.
- Diseñar soluciones usando estructuras iterativas y condicionales.
- Modularizar el código utilizando funciones con y sin retorno.
- Utilizar arrays para almacenamiento y manipulación de datos.
- Comprender y aplicar la recursividad en casos simples.
- Trabajar con variables locales y globales de forma adecuada.
- Fortalecer la capacidad de análisis lógico y la resolución de errores.
- Consolidar el uso del lenguaje Java mediante la práctica estructurada.