CASO PRÁCTICO 1 UNIDAD 2:
USO DE ESTRUCTURAS DE CONTROL EN JAVA

1. Introducción

Siguiendo las instrucciones de la tarea, en este trabajo se desarrollan cinco programas en Java que aplican diversas estructuras de control, abordando problemas prácticos propuestos en el caso práctico de la Unidad 2. Estas estructuras incluyen condicionales, bucles y validaciones, esenciales para la creación de aplicaciones funcionales y dinámicas.

El objetivo principal es demostrar el conocimiento de los conceptos fundamentales de programación, como el manejo de variables, la organización del código y la depuración, para resolver situaciones comunes en el desarrollo de software. A través del uso del entorno de desarrollo integrado (IDE) Eclipse, se ha priorizado la implementación de soluciones eficientes y funcionales.

Este informe no solo describe la lógica y el diseño de cada solución, sino que también incluye explicaciones detalladas sobre el uso de las estructuras de control y los resultados obtenidos, evidenciando un enfoque práctico y bien fundamentado.

2. Programa 1 - Ordenamiento de 3 números:

Este programa solicita tres números al usuario y un tipo de ordenamiento (ascendente o descendente). Utilizando estructuras condicionales y un enfoque basado en arreglos, los números se ordenan según la preferencia indicada y se muestran en pantalla.

```
import java.util.Arrays;
import java.util.Scanner;

public class OrdenarNumeros {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int[] numeros = new int[3];

        System.out.println("Introduce el primer número:");
        numeros[0] = scanner.nextInt();
        System.out.println("Introduce el segundo número:");
        numeros[1] = scanner.nextInt();
        System.out.println("Introduce el tercer número:");
        numeros[2] = scanner.nextInt();

        System.out.println("Introduce el orden (ascendente o descendente):");
        String orden = scanner.next();

        if (orden.equalsIgnoreCase("ascendente")) {
            Arrays.sort(numeros);
        } else if (orden.equalsIgnoreCase("descendente")) {
            Arrays.sort(numeros);
            for (int i = 0; i < numeros.length / 2; i++) {
                int temp = numeros[i];
                numeros[i] = numeros[numeros.length - 1 - i];
                numeros[numeros.length - 1 - i] = temp;
            }
        } else {
            System.out.println("Orden no válido. Inténtalo de nuevo.");
            return;
        }

        System.out.println("Números ordenados: " + Arrays.toString(numeros));
        scanner.close();
    }
}
```

Explicación:
El programa utiliza la clase Scanner para leer los números ingresados por el usuario y el tipo de ordenamiento. Los números se almacenan en un arreglo, lo que permite aplicar el método Arrays.sort() para el orden ascendente. En caso de que el usuario solicite un orden descendente, se invierte el arreglo. La validación asegura que el programa responda adecuadamente si se introduce un valor de orden no válido.

Resultados esperados:
El programa presenta los números ordenados en el formato indicado por el usuario. Por ejemplo, si los números ingresados son 4, 7, 2 y el orden es ascendente, el resultado será 2, 4, 7. Si el orden es descendente, el resultado será 7, 4, 2.

3. Programa 2 - Mostrar números del 10 al 1:

Este programa muestra los números del 10 al 1 en orden decreciente, utilizando un bucle for. Además, incluye la opción de repetir el programa si el usuario lo desea.

```
import java.util.Scanner;

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

        do {
            for (int numero = 10; numero >= 1; numero--) {
                System.out.println(numero);
            }
            System.out.println("¿Quieres repetir el programa? (sí/no):");
            respuesta = scanner.next();
        } while (respuesta.equalsIgnoreCase("sí"));

        scanner.close();
        System.out.println("Programa finalizado.");
    }
}
```

Explicación:
El programa utiliza un bucle for para iterar desde 10 hasta 1, imprimiendo cada número en la consola. Para mejorar la experiencia del usuario, se incluye una opción que permite repetir la ejecución preguntando al final si desea continuar, implementada con un bucle do-while. Esto hace que el programa sea más interactivo y útil en sesiones prácticas.

Resultados esperados:
El programa imprime los números en pantalla de forma decreciente. Por ejemplo:

```
10
9
8
7
...
1
```

Al finalizar, el usuario puede optar por reiniciar o salir del programa.

4. Programa 3 - Números sucesivos a un número positivo:

Este programa solicita un número positivo al usuario y, una vez validado, muestra los 20 números consecutivos a dicho valor utilizando un bucle for.

```
import java.util.Scanner;

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

        while (true) {
            System.out.println("Introduce un número positivo:");
            numero = scanner.nextInt();
            if (numero > 0) break;
            System.out.println("Número inválido. Debe ser positivo.");
        }

        System.out.println("Los primeros 20 números sucesivos son:");
        for (int i = numero + 1; i <= numero + 20; i++) {
            System.out.println(i);
        }

        scanner.close();
    }
}



            System.out.println(i);
        }

        scanner.close();
    }
}
```

Explicación:

Se emplea un bucle while para asegurar que el usuario introduzca un número positivo, mostrando un mensaje de error en caso contrario. Una vez que la entrada es válida, un bucle for genera e imprime los 20 números consecutivos al previamente ingresado. Este enfoque garantiza una interacción sencilla y eficiente.

Resultados esperados:

Si el usuario introduce el número 5, el programa mostrará:

```
6
7
8
...
25
```

5. Programa 4 - Cálculo del IMC:

Este programa calcula el Índice de Masa Corporal (IMC) de una persona con base en el peso y la altura ingresados. Ambos valores deben cumplir restricciones específicas, y el programa valida cada entrada antes de proceder con el cálculo.

```
import java.util.Scanner;

public class CalcularIMC {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        double peso = 0, altura = 0;

        while (true) {
```

```
            try {
                System.out.println("Introduce tu peso en kg (entre 30 y 300):");
                peso = scanner.nextDouble();
                if (peso >= 30 && peso <= 300) break;
                System.out.println("Peso inválido. Debe estar entre 30 y 300.");
            } catch (Exception e) {
                System.out.println("Entrada inválida. Por favor, introduce un número.");
                scanner.next(); // Limpiar el buffer
            }
        }

        while (true) {
            try {
                System.out.println("Introduce tu altura en metros (entre 1.30 y 2.00):");
                altura = scanner.nextDouble();
                if (altura >= 1.30 && altura <= 2.00) break;
                System.out.println("Altura inválida. Debe estar entre 1.30 y 2.00.");
            } catch (Exception e) {
                System.out.println("Entrada inválida. Por favor, introduce un número.");
                scanner.next(); // Limpiar el buffer
            }
        }

        double imc = peso / (altura * altura);
        System.out.println("Tu índice de masa corporal es: " + imc);

        if (imc < 18.5) {
            System.out.println("Nivel de peso: Bajo peso");
        } else if (imc < 25.0) {
            System.out.println("Nivel de peso: Normal");
        } else if (imc < 30.0) {
            System.out.println("Nivel de peso: Sobrepeso");
        } else {
            System.out.println("Nivel de peso: Obesidad");
        }

        scanner.close();
    }
}


        scanner.close();
    }
}
```

Explicación:
El programa solicita al usuario el peso y la altura, validándolos mediante bucles “while”. Si los valores están fuera de los rangos permitidos (peso entre 30 y 300 kg, altura entre 1.30 y 2.00 metros), se muestra un mensaje de error y se solicita nuevamente el dato incorrecto. Una vez validados, el programa calcula el IMC y determina el nivel de peso con base en la tabla proporcionada. El manejo de excepciones asegura robustez ante entradas no válidas.

Resultados esperados:
Si el usuario introduce 70 kg y 1.75 m, el programa calculará el IMC como 22.86 y mostrará:

```
Tu índice de masa corporal es: 22.86
Nivel de peso: Normal
```

6. Programa 5 - Múltiplos de 2 o 3:

Este programa encuentra y cuenta todos los números entre 1 y 100 que son múltiplos de 2 o 3, imprimiendo cada uno de ellos en la consola.

```
public class Multiplos {
    public static void main(String[] args) {
        int contador = 0;

        System.out.println("Números múltiplos de 2 o 3 entre 1 y 100:");
        for (int i = 1; i <= 100; i++) {
            if (i % 2 == 0 || i % 3 == 0) {
                System.out.println(i);
                contador++;
            }
        }

        System.out.println("Total: " + contador + " números múltiplos de 2 o 3.");
    }
}
```

Explicación:

Se utiliza un bucle for para recorrer los números del 1 al 100. Dentro del bucle, una condición (if) verifica si el número es múltiplo de 2 o de 3 utilizando los operadores de módulo (%). Cada número que cumple la condición se imprime, y un contador se incrementa para registrar la cantidad total. Esto demuestra un uso práctico de las estructuras de control.

Resultados esperados:

El programa imprime todos los múltiplos de 2 o 3 entre 1 y 100, terminando con un resumen del total:

```
2
3
4
...
96
98
99
Hay 67 números múltiplos de 2 o 3 entre 1 y 100.
```

7. Conclusión

En este trabajo, se han desarrollado cinco programas funcionales utilizando estructuras de control en Java para resolver problemas prácticos de programación. La implementación demuestra un conocimiento sólido en el uso de bucles, condicionales y validaciones, aplicados de manera eficiente para lograr los objetivos planteados. Las mejoras incorporadas han optimizado la legibilidad, escalabilidad y robustez del código, asegurando una experiencia clara tanto para el usuario final como para desarrolladores que deseen modificar o extender las soluciones.

8. Resultados de aprendizaje

A lo largo del desarrollo de este trabajo, se han alcanzado los siguientes resultados de aprendizaje relacionados con la programación en Java y el uso de estructuras de control:

1. **Comprensión de la estructura general de un programa Java:**
    - Se han creado programas funcionales con una estructura básica y correctamente definida, incluyendo clases principales, métodos y manejo de excepciones.
    - Los programas cumplen con las normas básicas de sintaxis y organización del código.

2. **Uso efectivo de estructuras de control:**
    - Se han implementado condicionales (if-else) y bucles (for, while, do-while) de manera eficiente para resolver problemas prácticos.
    - Las estructuras de control han sido combinadas y adaptadas según las necesidades específicas de cada programa.

3. **Validación y manejo de errores:**
    - Se han validado las entradas del usuario en todos los programas para asegurar datos correctos y evitar errores de ejecución.
    - Se ha añadido manejo de excepciones para controlar entradas no válidas en el programa del cálculo del IMC.

4. **Uso y manipulación de datos:**
    - Se han utilizado distintos tipos de variables (int, double, String) según las necesidades del problema.
    - Se ha trabajado con arreglos para simplificar operaciones como el ordenamiento de números.

5. **Pruebas y depuración de código:**
    - Cada programa se ha probado para garantizar que cumple los requerimientos sin errores de sintaxis o ejecución.
    - La estructura y comentarios del código facilitan su mantenimiento y comprensión.

6. **Orientación al usuario:**
    - Los programas han sido diseñados para ofrecer mensajes claros e interactivos que guían al usuario durante la ejecución.
    - Se ha incorporado retroalimentación inmediata en casos de entradas no válidas.

7. **Familiarización con el IDE Eclipse:**
    - El trabajo se ha desarrollado y ejecutado en el entorno de desarrollo Eclipse, consolidando habilidades prácticas en su uso.

9. Material consultado y referencias

- Material de la unidad
- Udemy
- https://www.youtube.com/watch?v=WEOcoIXI6gc
- https://www.youtube.com/playlist?list=PLSxN_97yth53uRiBTLcETdv0hdNDGwYch
- https://www.tutorialesprogramacionya.com/javaya/detalleconcepto.php?codigo=76&punto=&inicio=0