UNIVERSIDAD NACIONAL DE SAN AGUSTIN DE AREQUIPA

FACULTAD DE INGENIERIA DE PRODUCCION Y SERVICIOS

ESCUELA PROFESIONAL DE INGENIERIA DE SISTEMAS

Curso : **Estructura de Datos y Algoritmos**

Docente : **Mg. Ing. Rene Alonso Nieto Valencia.**

Informe de Entregable

Laboratorio E.D.A. E

Elaborado por : Quiñonez Delgado Aarón Fernando.

2025

Arequipa - Perú

Enunciado:
Se debe crear un programa en Java que permita al usuario ingresar las edades de varias personas. El usuario indicará cuántas edades desea ingresar. Una vez ingresadas todas las edades, el programa debe calcular y mostrar tres cosas:

1. La edad promedio del grupo.
2. La edad mayor.
3. La edad menor.

Explicación de la solución en lenguaje natural:

1. **Inicio del programa:**
   Se utiliza la clase `Scanner` para permitir que el usuario escriba datos por teclado.
2. **Cantidad de personas:**
   Se le pide al usuario que escriba cuántas personas hay en el grupo. Ese número se guarda en una variable.
3. **Lectura de edades:**
   Se crea un arreglo para guardar todas las edades. Luego, usando un bucle `for`, el programa le pide al usuario que ingrese una a una las edades de las personas.
4. **Cálculos:**
   - Se declara una variable para sumar todas las edades.
   - También se definen dos variables para guardar la edad mayor y la menor. Inicialmente, ambas toman el valor de la primera edad ingresada.
   - Luego, se recorre el arreglo de edades y:
     - Se va sumando cada edad a la variable `suma`.
     - Si una edad es mayor que la actual "edad mayor", se actualiza.
     - Si una edad es menor que la actual "edad menor", también se actualiza.
5. **Promedio:**
   Una vez terminado el recorrido, se calcula el promedio dividiendo la suma total de las edades entre la cantidad de personas.
6. **Resultados:**
   Finalmente, el programa muestra en pantalla:
   - La edad promedio.
   - La mayor edad.
   - La menor edad.
7. **Cierre:**
   Se cierra el objeto `Scanner` para liberar recursos.

Enunciado:

Se debe crear un programa en Java que calcule la **suma de los primeros N números naturales**. El usuario debe escribir el valor de **N**, y el programa debe usar un **bucle while** para realizar la suma.

- Un número natural N (ingresado por el usuario).
- La suma de los primeros N números naturales, es decir: 1 + 2 + 3 + ... + N.

Explicación de la solución en lenguaje natural:

1. **Inicio del programa:**
   Se importa la clase Scanner para permitir que el usuario escriba datos por teclado.
2. **Ingreso de N:**
   El programa muestra un mensaje pidiendo al usuario que ingrese un número natural N. Este valor se guarda en una variable.
3. **Declaración de variables para la suma:**
   Se declara una variable suma que comenzará en 0.
   También se declara una variable i que empezará en 1, porque se va a usar para sumar desde 1 hasta N.
4. **Bucle while:**
   Se usa un bucle while que se ejecuta mientras i sea menor o igual a N.
   Dentro del bucle:
   - Se suma el valor de i a la variable suma.
   - Luego, se incrementa i en 1 para pasar al siguiente número.
5. **Mostrar el resultado:**
   Cuando termina el bucle, se muestra en pantalla el resultado final de la suma.
6. **Cierre:**
   Se cierra el objeto Scanner.

Enunciado:

Se debe escribir un programa en Java que permita al usuario ingresar una lista de números y determine si esa lista está **ordenada de forma ascendente**.
Durante el proceso, se debe aplicar el **concepto de invariante**, es decir, una condición que se espera que se mantenga verdadera durante la ejecución del bucle.

- Una lista de números (ingresada por el usuario).
- Sí si la lista está ordenada ascendentemente.
- No si no lo está.

Explicación de la solución en lenguaje natural:

1. **Inicio del programa:**
   Se importa la clase Scanner para poder leer datos del teclado.
2. **Cantidad de elementos:**
   El programa pide al usuario que indique cuántos números quiere ingresar. Ese número se guarda en una variable n.
3. **Lectura de números:**
   Se crea un arreglo de tamaño n. Luego, usando un bucle for, se solicita al usuario que ingrese uno por uno los n números.
4. **Declaración de invariante:**
   Se crea una variable booleana llamada estaOrdenada y se inicializa en true.
   Esta variable representa la **invariante**, es decir, **la suposición de que la lista está ordenada**.
5. **Verificación de orden ascendente (bucle):**
   Usamos un bucle for que compara cada número con el anterior.
   - Si en algún momento encontramos que un número es menor que el anterior, se rompe la condición de orden.
   - En ese caso, se asigna false a la variable estaOrdenada y se interrumpe el bucle con break.
6. **Resultado final:**
   Al salir del bucle, el programa muestra el mensaje:
   - "Sí" si todos los números estaban en orden ascendente.
   - "No" si en algún punto se rompió el orden.
7. **Cierre:**
   Se cierra el objeto Scanner.