INFORME DE LABORATORIO

(formato estudiante)

INFORMACIÓN BÁSICA					
ASIGNATURA:	**ESTRUCTURA DE DATOS Y ALGORITMOS**				
TÍTULO DE LA PRÁCTICA:	*FUNDAMENTOS DE PROGRAMACIÓN*				
NÚMERO DE PRÁCTICA:	*01*	**AÑO LECTIVO:**	**2025 – A**	**NRO. SEMESTRE:**	**Tercero III**
FECHA DE PRESENTACIÓN		**HORA DE PRESENTACIÓN**			
INTEGRANTE (s): Yauli Merma Diego Raul				**NOTA:**	
DOCENTE(s): ● **Mg. Ing. Rene Alonso Nieto Valencia.**					

SOLUCIÓN Y RESULTADOS

I. SOLUCIÓN DE EJERCICIOS/PROBLEMAS

Problemas resueltos:

1. Arreglos Unidimensionales.

Enunciado: Escribe un programa en Java que permita ingresar las edades de un grupo de personas y determine la edad promedio, la mayor y la menor. El usuario debe poder especificar cuántas edades desea ingresar.

```java
public class EdadesGrupo {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        // Solicitar la cantidad de personas
        System.out.print(s:"Ingrese el número de personas: ");
        int n = scanner.nextInt();
        // Crear el arreglo para almacenar las edades
        int[] edades = new int[n];
        // Leer las edades desde la entrada
        System.out.println(x:"Ingrese las edades:");
        for (int i = 0; i < n; i++) {
            edades[i] = scanner.nextInt();
        }
        // Inicializar variables para el cálculo
        int suma = 0;
        int mayor = edades[0];
        int menor = edades[0];
        // Recorrer el arreglo para calcular suma, mayor y menor
        for (int edad : edades) {
            suma += edad;
```

```
            if (edad > mayor) {
                mayor = edad;
            }
            if (edad < menor) {
                menor = edad;
            }
        }
        // Calcular el promedio
        double promedio = (double) suma / n;
        // Mostrar los resultados
        System.out.println("Edad promedio: " + promedio);
        System.out.println("Edad mayor: " + mayor);
        System.out.println("Edad menor: " + menor);
        // Cerrar el escáner
        scanner.close();
    }
}
```

1. Pedir la cantidad de personas

El programa comienza solicitando al usuario cuántas edades va a ingresar. Esta cantidad define el tamaño del arreglo que almacenará las edades.

2. Crear un arreglo para las edades

Se declara un arreglo de enteros con el tamaño que el usuario indicó, donde se guardarán las edades ingresadas.

3. Leer las edades desde teclado

Usando un ciclo for, el programa lee las edades una por una desde la entrada del usuario y las guarda en el arreglo.

4. Inicializar variables para cálculos

Se inicializan tres variables:

- Una para la suma total de las edades.
- Una para la edad mayor (inicialmente con la primera edad).
- Una para la edad menor (también con la primera edad).

5. Recorrer el arreglo para procesar los datos

Se recorre el arreglo con un bucle mejorado (for-each) y en cada iteración:

- Se suma la edad al total.
- Se compara la edad actual con la mayor y menor para actualizarlas si es necesario.

6. Calcular el promedio

Se divide la suma total entre la cantidad de edades para obtener el promedio. Se convierte a double para obtener un resultado decimal si es necesario.

7. Mostrar resultados

Finalmente, se imprimen:

- El promedio de las edades.

- La edad mayor encontrada.
- La edad menor encontrada.

8. Cerrar el escáner

Se cierra el objeto Scanner para liberar recursos del sistema.

Resultado:

```
PS D:\UNSA\5° Semestre\EDA\Practicas de mi>  d:; cd 'd:\UNSA\5° Semestre\EDA\Practicas de mi'; & 'C:\Program Files\Java\jdk-13.0.2\bin\java.exe' '-cp' 'C:\Users\
Usuario\AppData\Roaming\Code\User\workspaceStorage\c3235d1a107b446fd323ef022363867a\redhat.java\jdt_ws\Practicas de mi_7b4876af\bin' 'EdadesGrupo'
Ingrese el número de personas: 3
Ingrese las edades:
25
12
27
Edad promedio: 21.333333333333332
Edad mayor: 27
Edad menor: 12
PS D:\UNSA\5° Semestre\EDA\Practicas de mi>
```

2. Iteraciones:

Enunciado: Escribe un programa en Java que permita calcular la suma de los primeros N números naturales usando un bucle while. El usuario debe ingresar el valor de N.

```java
import java.util.Scanner;
public class EdadesGrupo {
    Run | Debug | Run main | Debug main
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print(s:"Ingrese el número de personas: ");
        int n = scanner.nextInt();
        int[] edades = new int[n];
        System.out.println(x:"Ingrese las edades:");
        for (int i = 0; i < n; i++) {
            edades[i] = scanner.nextInt();
        }
        int suma = 0;
        int mayor = edades[0];
        int menor = edades[0];
        for (int edad : edades) {
            suma += edad;
            if (edad > mayor) {
                mayor = edad;
            }
            if (edad < menor) {
                menor = edad;
            }
        }
        double promedio = (double) suma / n;
        System.out.println("Edad promedio: " + promedio);
        System.out.println("Edad mayor: " + mayor);
        System.out.println("Edad menor: " + menor);
        scanner.close();
    }
}
```

1. Importa Scanner

Para poder leer entradas del usuario.

2. Inicio del método main

Punto de entrada del programa.

3. Crear objeto Scanner

Se usa para leer datos ingresados por el usuario.

4. Solicitar cuántas edades se ingresarán

El usuario indica cuántas personas hay en el grupo.

5. Declarar un arreglo del tamaño especificado

Se crea un arreglo de enteros para almacenar las edades.

6. Leer las edades una por una

Se usa un bucle para guardar cada edad en el arreglo.

7. Inicializar variables para la suma, mayor y menor edad

Estas variables permitirán calcular el promedio y encontrar los extremos.

8. Recorrer el arreglo para calcular:

- La suma total de edades.
- La edad mayor encontrada.
- La edad menor encontrada.

9. Calcular el promedio

Dividiendo la suma entre la cantidad de edades.

10. Mostrar los resultados al usuario

Se imprime el promedio, la mayor y la menor edad.

11. Cerrar el Scanner

Para liberar recursos del sistema.

Resultado:

```
PS D:\UNSA\5° Semestre\EDA\Practicas de mi>  d:; cd 'd:\UNSA\5° Semestre\EDA\Practicas de mi'; & 'C:\Program Files\Java\jdk-13.0.2\bin\java.exe' '-cp' 'C:\Users\
Usuario\AppData\Roaming\Code\User\workspaceStorage\c3235d1a107b446fd323ef022363867a\redhat.java\jdt_ws\Practicas de mi_7b4876af\bin' 'EdadesGrupo'
Ingrese el número de personas:
Usuario\AppData\Roaming\Code\User\workspaceStorage\c3235d1a107b446fd323ef022363867a\redhat.java\jdt_ws\Practicas de mi_7b4876af\bin' 'EdadesGrupo'
Usuario\AppData\Roaming\Code\User\workspaceStorage\c3235d1a107b446fd323ef022363867a\redhat.java\jdt_ws\Practicas de mi_7b4876af\bin' 'EdadesGrupo'
Ingrese el número de personas: 6
Ingrese las edades:
5
2
8
6
3
10
Edad promedio: 5.666666666666667
Edad mayor: 10
Edad menor: 2
PS D:\UNSA\5° Semestre\EDA\Practicas de mi>
```

3. Invariantes:

Enunciado: Implementa un algoritmo que determine si una lista de números ingresados por el usuario está ordenada de manera ascendente. Debes usar un concepto de invariante dentro del bucle para garantizar que la propiedad de orden se mantiene durante la ejecución..

```java
import java.util.Scanner;
public class ListaOrdenada {
    Run | Debug | Run main | Debug main
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print(s:"Ingrese el número de elementos: ");
        int n = scanner.nextInt();
        int[] numeros = new int[n];
        System.out.println(x:"Ingrese los números:");
        for (int i = 0; i < n; i++) {
            numeros[i] = scanner.nextInt();
        }
        boolean estaOrdenada = true; // Invariante: se supone que la lista está ordenada
        for (int i = 1; i < n; i++) {
            if (numeros[i] < numeros[i - 1]) {
                estaOrdenada = false; // Se rompe la invariante si encontramos un desorden
                break;
            }
        }
        System.out.println("¿Está ordenada la lista?: " + (estaOrdenada ? "Sí" : "No"));
        scanner.close();
    }
}
```

1. **Importa Scanner**
   Para leer datos ingresados por el usuario desde el teclado.

2. **Inicio del método main**
   Punto de inicio del programa.

3. **Crear objeto Scanner**
   Se usa para capturar la entrada del usuario.

4. **Solicitar al usuario la cantidad de elementos**
   El usuario indica cuántos números ingresará.

5. **Declarar un arreglo del tamaño especificado**
   Se crea un arreglo de enteros para guardar los números.

6. **Leer y almacenar los números ingresados**
   Se utiliza un bucle para ingresar los elementos en el arreglo.

7. **Inicializar una variable booleana para verificar si la lista está ordenada**
   Se asume inicialmente que la lista sí está ordenada.

8. **Recorrer el arreglo desde el segundo elemento**

- En cada iteración se compara el número actual con el anterior:
- Si algún número es menor que el anterior, se rompe la condición de orden ascendente.
- Se cambia la variable a false y se interrumpe el ciclo.

9. **Mostrar si la lista está ordenada**
   Se imprime un mensaje indicando si la lista está o no en orden ascendente.

10. **Cerrar el Scanner**
    Para liberar recursos del sistema.

Resultado:

```
PS D:\UNSA\5° Semestre\EDA\Practicas de mi>  & 'C:\Program Files\Java\jdk-13.0.2\bin\java.exe' '-cp' 'C:\Users\Usuario\AppData\Roaming\Code\User\workspaceStorage
\c3235d1a107b446fd323ef022363867a\redhat.java\jdt_ws\Practicas de mi_7b4876af\bin' 'ListaOrdenada'
Ingrese el número de elementos: 3
Ingrese los números:
10
5
15
¿Está ordenada la lista?: No
PS D:\UNSA\5° Semestre\EDA\Practicas de mi>  & 'C:\Program Files\Java\jdk-13.0.2\bin\java.exe' '-cp' 'C:\Users\Usuario\AppData\Roaming\Code\User\workspaceStorage
\c3235d1a107b446fd323ef022363867a\redhat.java\jdt_ws\Practicas de mi_7b4876af\bin' 'ListaOrdenada'
Ingrese el número de elementos: 3
Ingrese los números:
1
5
10
¿Está ordenada la lista?: Sí
PS D:\UNSA\5° Semestre\EDA\Practicas de mi>
```

Problemas propuestos:

1.- Desarrolla un programa en Java que implemente un sistema de gestión de calificaciones de estudiantes. El programa debe permitir al usuario ingresar las calificaciones de N estudiantes y calcular la mediana, moda y desviación estándar

Algoritmo en lenguaje natural:

1. Pedir al usuario cuántas calificaciones desea ingresar (N).

2. Crear un arreglo de tamaño N para guardar las calificaciones.

3. Leer N calificaciones y guardarlas en el arreglo.

4. Ordenar el arreglo de menor a mayor (para poder calcular la mediana).

5. Calcular la **mediana**:
   - Si la cantidad de calificaciones es impar, tomar la del centro.
   - Si es par, promediar las dos del centro.

6. Calcular la **moda**:
   - Contar cuántas veces aparece cada calificación.

- La moda es la que más se repite.

7. Calcular la **desviación estándar**:
   - Calcular la media (promedio) de las calificaciones.
   - Restar la media a cada valor y elevar al cuadrado.
   - Promediar esos cuadrados y sacar la raíz cuadrada del resultado.

8. Mostrar los resultados: mediana, moda y desviación estándar.

```java
import java.util.*;
public class CaliEstu {
    Run | Debug | Run main | Debug main
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        // Paso 1 y 2: Leer cantidad de estudiantes y declarar arreglo
        System.out.print(s:"Ingrese la cantidad de estudiantes: ");
        int n = sc.nextInt();
        int[] calificaciones = new int[n];
        // Paso 3: Leer calificaciones
        for (int i = 0; i < n; i++) {
            System.out.print("Ingrese la calificación del estudiante " + (i + 1) + ": ");
            calificaciones[i] = sc.nextInt();
        }
        // Paso 4: Ordenar el arreglo
        Arrays.sort(calificaciones);
        // Paso 5: Calcular mediana
        double mediana = calcularMediana(calificaciones);
        // Paso 6: Calcular moda
        int moda = calcularModa(calificaciones);
        // Paso 7: Calcular desviación estándar
        double desviacion = calcularDesviacion(calificaciones);
        // Paso 8: Mostrar resultados
        System.out.println("Mediana: " + mediana);
        System.out.println("Moda: " + moda);
        System.out.printf(format:"Desviación estándar: %.2f\n", desviacion);
        sc.close();
    }
    public static double calcularMediana(int[] arr) {
        int n = arr.length;
        if (n % 2 == 0) {
            return (arr[n / 2 - 1] + arr[n / 2]) / 2.0;
        } else {
            return arr[n / 2];
        }
    }
```

```java
    public static int calcularModa(int[] arr) {
        int moda = arr[0];
        int maxFrecuencia = 1;
        int actual = arr[0];
        int frecuencia = 1;

        for (int i = 1; i < arr.length; i++) {
            if (arr[i] == actual) {
                frecuencia++;
            } else {
                actual = arr[i];
                frecuencia = 1;
            }
            if (frecuencia > maxFrecuencia) {
                maxFrecuencia = frecuencia;
                moda = arr[i];
            }
        }
        return moda;
    }
    public static double calcularDesviacion(int[] arr) {
        int suma = 0;
        for (int num : arr) {
            suma += num;
        }
        double media = (double) suma / arr.length;
        double sumaCuadrados = 0;
        for (int num : arr) {
            sumaCuadrados += Math.pow(num - media, b:2);
        }
        return Math.sqrt(sumaCuadrados / arr.length);
    }
}
```

Resultados:

```
PS D:\UNSA\5° Semestre\EDA\Practicas de mi>  & 'C:\Program Files\Java\jdk-13.0.2\bin\java.exe' '-cp' 'C:\Users\Usuario\AppData\Roaming\Code\User\workspaceStorage
\c3235d1a107b446fd323ef022363867a\redhat.java\jdt_ws\Practicas de mi_7b4876af\bin' 'CaliEstu'
Ingrese la cantidad de estudiantes: 5
Ingrese la calificación del estudiante 1: 10
Ingrese la calificación del estudiante 2: 11
Ingrese la calificación del estudiante 3: 20
Ingrese la calificación del estudiante 4: 11
Ingrese la calificación del estudiante 5: 9
Mediana: 11.0
Moda: 11
Desviación estándar: 3.97
PS D:\UNSA\5° Semestre\EDA\Practicas de mi>
```

2. Implementa un programa en Java que encuentre todos los números primos en un rango definido por el usuario utilizando el algoritmo de la Criba de Eratóstenes

Pasos del algoritmo (lenguaje natural):

1. Pedir al usuario el límite superior del rango (por ejemplo, hasta qué número buscar los primos).
2. Crear un arreglo booleano de tamaño n+1, donde cada posición indica si ese número es primo (true = primo).
3. Inicializar todo el arreglo en true, excepto las posiciones 0 y 1 (que no son primos).

4. Empezar desde el número 2 hasta √n:
    - Si el número actual es primo:
        - Eliminar (marcar como false) todos sus múltiplos (2×n, 3×n, etc.).
5. Al finalizar, los índices que aún están en true son números primos.
6. Imprimir esos números primos.

```java
import java.util.Scanner;
public class CribaEra {
    Run | Debug | Run main | Debug main
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        // Paso 1: Pedir límite superior
        System.out.print(s:"Ingrese el número máximo del rango: ");
        int n = sc.nextInt();
        // Paso 2: Crear arreglo booleano para marcar si un número es primo
        boolean[] esPrimo = new boolean[n + 1];
        // Paso 3: Inicializar todos los valores como verdaderos (excepto 0 y 1)
        for (int i = 2; i <= n; i++) {
            esPrimo[i] = true;
        }
        // Paso 4: Aplicar Criba de Eratóstenes
        for (int i = 2; i * i <= n; i++) {
            if (esPrimo[i]) {
                // Marcar múltiplos de i como no primos
                for (int j = i * i; j <= n; j += i) {
                    esPrimo[j] = false;
                }
            }
        }
        // Paso 5 y 6: Imprimir los números que siguen siendo primos
        System.out.println("Números primos hasta " + n + ":");
        for (int i = 2; i <= n; i++) {
            if (esPrimo[i]) {
                System.out.print(i + " ");
            }
        }
        sc.close();
    }
}
```

Resultados:

```
PS D:\UNSA\5° Semestre\EDA\Practicas de mi>  d:; cd 'd:\UNSA\5° Semestre\EDA\Practicas de mi'; & 'C:\Program Files\Java\jdk-13.0.2\bin\java.exe' '-cp' 'C:\Users\
Usuario\AppData\Roaming\Code\User\workspaceStorage\c3235d1a107b446fd323ef022363867a\redhat.java\jdt_ws\Practicas de mi_7b4876af\bin' 'CribaEra'
Ingrese el número máximo del rango: 30
Números primos hasta 30:
2 3 5 7 11 13 17 19 23 29
PS D:\UNSA\5° Semestre\EDA\Practicas de mi>
```

3. Desarrolla un algoritmo que implemente el Ordenamiento por Inserción, asegurando que en cada paso del bucle el segmento procesado de la lista permanece ordenado (principio de invariante).

1. Crear una lista de números a ordenar.
2. Empezar desde el segundo elemento (índice 1) hasta el final de la lista.

3. Al inicio de cada iteración, considerar que la sublista desde el índice 0 hasta i-1 está ordenada (esto es la invariante del bucle).
4. Tomar el elemento actual como valorActual.
5. Compararlo con los elementos anteriores.
6. Mientras valorActual sea menor que los elementos anteriores, desplazarlos una posición hacia la derecha.
7. Insertar valorActual en su posición correcta.
8. Luego de cada paso, la sublista procesada (desde el inicio hasta la posición i) se mantiene ordenada, cumpliendo con el principio de invariante.
9. Continuar hasta ordenar toda la lista.

```java
public class InserCon {
    Run | Debug | Run main | Debug main
    public static void main(String[] args) {
        int[] arreglo = { 7, 3, 5, 2, 9, 1 };
        System.out.println(x:"Arreglo original:");
        imprimirArreglo(arreglo);
        ordenamientoPorInsercion(arreglo);
        System.out.println(x:"Arreglo ordenado:");
        imprimirArreglo(arreglo);
    }
    public static void ordenamientoPorInsercion(int[] arr) {
        for (int i = 1; i < arr.length; i++) {
            int valorActual = arr[i];
            int j = i - 1;
            // Invariante: arr[0..i-1] está ordenado antes de comenzar esta iteración
            while (j >= 0 && arr[j] > valorActual) {
                arr[j + 1] = arr[j]; // mover elementos mayores una posición a la derecha
                j--;
            }
            arr[j + 1] = valorActual; // insertar valorActual en su posición correcta
            // Invariante se mantiene: arr[0..i] está ordenado
            System.out.print("Paso " + i + " (sublista ordenada hasta índice " + i + "): ");
            imprimirArreglo(arr);
        }
    }
    public static void imprimirArreglo(int[] arr) {
        for (int num : arr) {
            System.out.print(num + " ");
        }
        System.out.println();
    }
}
```

Resultados:

```
PS D:\UNSA\5° Semestre\EDA\Practicas de mi>  & 'C:\Program Files\Java\jdk-13.0.2\bin\java.exe' '-cp' 'C:\Users\Usuario\AppData\Roaming\Code\User\worksp
\c3235d1a107b446fd323ef022363867a\redhat.java\jdt_ws\Practicas de mi_7b4876af\bin' 'InserCon'
Arreglo original:
7 3 5 2 9 1
Paso 1 (sublista ordenada hasta índice 1): 3 7 5 2 9 1
Paso 2 (sublista ordenada hasta índice 2): 3 5 7 2 9 1
Paso 3 (sublista ordenada hasta índice 3): 2 3 5 7 9 1
Paso 4 (sublista ordenada hasta índice 4): 2 3 5 7 9 1
Paso 5 (sublista ordenada hasta índice 5): 1 2 3 5 7 9
Arreglo ordenado:
1 2 3 5 7 9
PS D:\UNSA\5° Semestre\EDA\Practicas de mi>
```

II. SOLUCIÓN DEL CUESTIONARIO

1. ¿Cuáles fueron las dificultades que encontraste al desarrollar los ejercicios propuestos? por ejemplo, poca documentación, complejidad del lenguaje, etc.

Una de las principales dificultades que encontré al desarrollar los ejercicios fue entender bien algunos conceptos, como la moda o la desviación estándar, ya que no siempre hay ejemplos claros en español. También me costó un poco aplicar correctamente el principio de invariante en el ordenamiento por inserción, porque requiere pensar de forma lógica y no solo programar. Además, al usar Java, a veces se vuelve complejo manejar bien los arrays y estructuras si no se tiene mucha práctica.

III. CONCLUSIONES

Estos ejercicios me ayudaron a entender mejor cómo funcionan los algoritmos y aplicar conceptos como la mediana, moda y desviación estándar. También aprendí a usar el principio de invariante en el ordenamiento por inserción. Aunque tuve algunas dificultades con la lógica y la sintaxis de Java, practicar me permitió resolverlos y reforzar mis conocimientos en programación.

RETROALIMENTACIÓN GENERAL

El desarrollo de los ejercicios permitió aplicar de manera práctica conceptos fundamentales de programación, como estructuras de control, arreglos y algoritmos clásicos. Se logró implementar correctamente el ordenamiento por inserción respetando el principio de invariante, y se abordaron operaciones estadísticas con calificaciones, lo cual refuerza habilidades tanto lógicas como matemáticas. A pesar de algunas dificultades iniciales, como la comprensión de algunos conceptos y detalles del lenguaje Java, el trabajo fue completado satisfactoriamente y evidencia un avance sólido en la resolución de problemas computacionales.

REFERENCIAS Y BIBLIOGRAFÍA

Weiss M., Data Structures & Problem Solving Using Java, 2010, Addison-Wesley.
- *Weiss M., Data Structures and Algorithms Analysis in Java, 2012, Addison-Wesley.*
- *Cormen T., Leiserson C., Rivest R., Stein C., Introduction to Algorithms, 2022, The MIT Press*
- *The JavaTM Tutorials - https://docs.oracle.com/javase/tutorial/*