



	<p>UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p>Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLE-001</p>	<p>Página: 1</p>

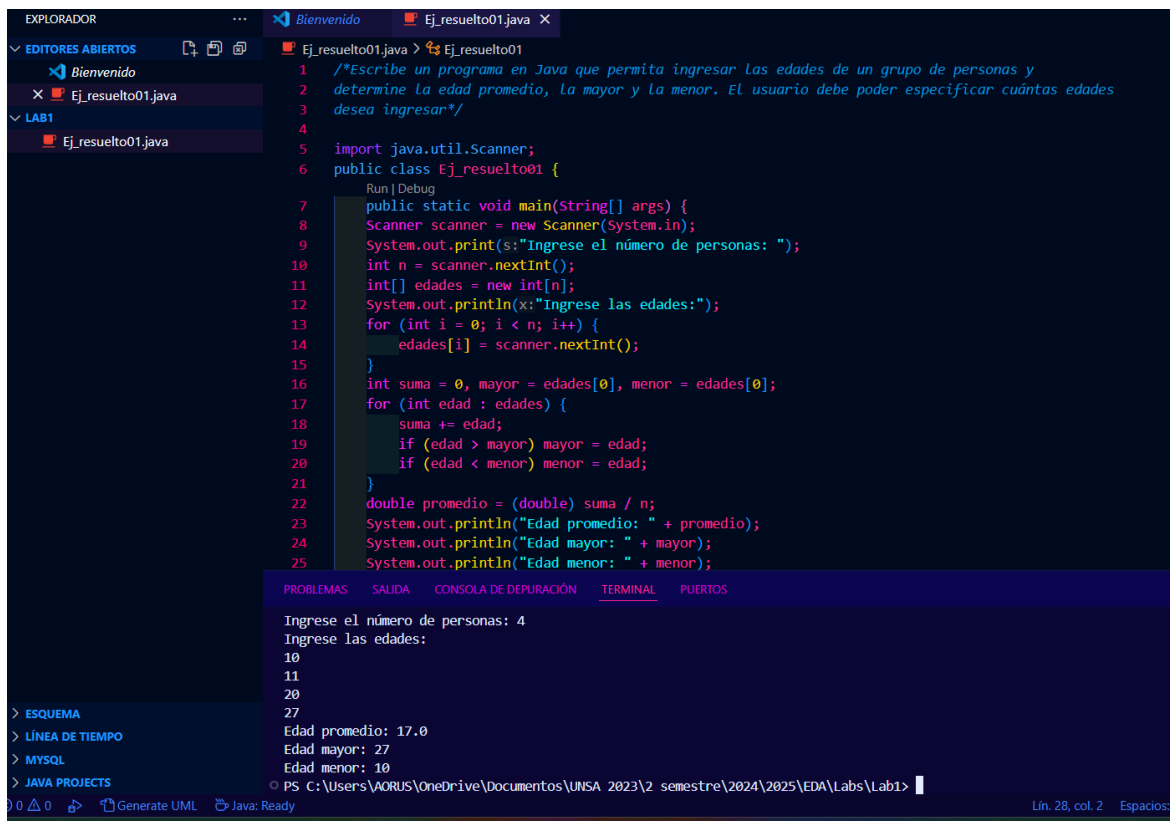
INFORME DE LABORATORIO

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	09/05/2025	HORA DE PRESENTACIÓN	11:59		
INTEGRANTE (s): Davila Flores Mathias Dario				NOTA:	
DOCENTE(s):					
<ul style="list-style-type: none"> Mg. Ing. Rene Alonso Nieto Valencia. ENLACE GITHUB: https://github.com/mathiasddf/LabsEDA 					

SOLUCIÓN Y RESULTADOS
<p>I. SOLUCIÓN DE EJERCICIOS/PROBLEMAS</p> <p>a. Ejercicios Resueltos:</p> <ol style="list-style-type: none"> 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 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. 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..

	<p style="text-align: center;">UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p style="text-align: center;">Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLE-001</p>	<p>Página: 2</p>

1)



```

1  /*Escribe un programa en Java que permita ingresar las edades de un grupo de personas y
2  determine la edad promedio, la mayor y la menor. El usuario debe poder especificar cuántas edades
3  desea ingresar*/
4
5  import java.util.Scanner;
6  public class Ej_resuelto01 {
7      Run | Debug
8      public static void main(String[] args) {
9          Scanner scanner = new Scanner(System.in);
10         System.out.print(s:"Ingrese el número de personas: ");
11         int n = scanner.nextInt();
12         int[] edades = new int[n];
13         System.out.println(x:"Ingrese las edades:");
14         for (int i = 0; i < n; i++) {
15             edades[i] = scanner.nextInt();
16         }
17         int suma = 0, mayor = edades[0], menor = edades[0];
18         for (int edad : edades) {
19             suma += edad;
20             if (edad > mayor) mayor = edad;
21             if (edad < menor) menor = edad;
22         }
23         double promedio = (double) suma / n;
24         System.out.println("Edad promedio: " + promedio);
25         System.out.println("Edad mayor: " + mayor);
26         System.out.println("Edad menor: " + menor);
27     }
28 }

```

Ingresa el número de personas: 4
Ingresa las edades:
10
11
20
27
Edad promedio: 17.0
Edad mayor: 27
Edad menor: 10

2)



```

1  /* Escribe un programa en Java que permita calcular la suma de los primeros N números
2  naturales usando un bucle while. El usuario debe ingresar el valor de N.*/
3
4  import java.util.Scanner;
5  public class Ej_resuelto02 {
6      Run | Debug
7      public static void main(String[] args) {
8          Scanner scanner = new Scanner(System.in);
9          System.out.print(s:"Ingrese el valor de N: ");
10         int N = scanner.nextInt();
11         int suma = 0;
12         int i = 1;
13         while (i <= N) {
14             suma += i;
15             i++;
16         }
17         System.out.println("La suma de los primeros " + N + " números naturales es: " + suma);
18         scanner.close();
19     }
20 }

```

Ingresa el valor de N: 10
La suma de los primeros 10 números naturales es: 55

	<p style="text-align: center;">UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p style="text-align: center;">Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLE-001</p>	<p>Página: 3</p>

3)



```

1  //Implementa un algoritmo que determine si una lista de números ingresados por el usuario
2  está ordenada de manera ascendente. Debes usar un concepto de invariante dentro del bucle para
3  garantizar que la propiedad de orden se mantiene durante la ejecución.. */
4
5  import java.util.Scanner;
6  public class Ej_resuelto03 {
7
8      public static void main(String[] args) {
9          Scanner scanner = new Scanner(System.in);
10         System.out.print("Ingrese el número de elementos: ");
11         int n = scanner.nextInt();
12         int[] numeros = new int[n];
13         System.out.println("Ingrese los números:");
14         for (int i = 0; i < n; i++) {
15             numeros[i] = scanner.nextInt();
16         }
17         boolean estaOrdenada = true; // Invariante: Se supone que la lista está ordenada
18         for (int i = 1; i < n; i++) {
19             if (numeros[i] < numeros[i - 1]) {
20                 estaOrdenada = false; // Se rompe la invariante si encontramos un desorden
21                 break;
22             }
23         }
24         System.out.println("¿Está ordenada la lista?: " + (estaOrdenada ? "Si" : "No"));
25         scanner.close();
26     }
27 }

```

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS

```

008be66\bin' 'Ej_resuelto03'
Ingrese el número de elementos: 5
Ingrese los números:
4
2
3
6
9
¿Está ordenada la lista?: No

```

b. Ejercicios Propuestos:

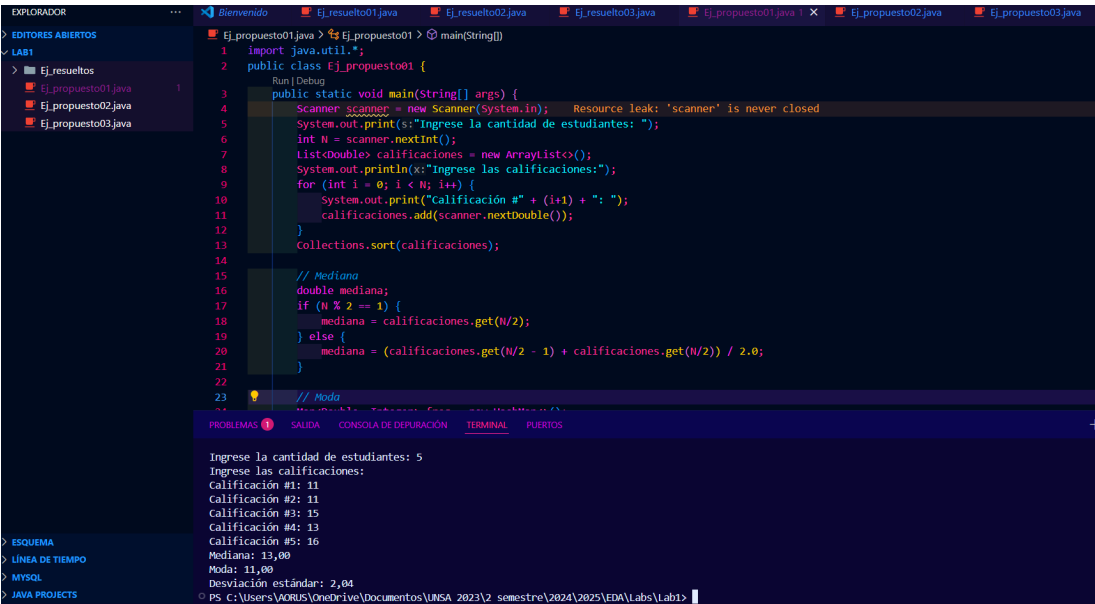
i. Desarrolla un programa en Java que implemente un sistema de gestión de calificaciones de estudiantes. El programa debe permitir al usuario ingresar las calificaciones de N estudiantes y calcular la mediana, moda y desviación estándar

● Algoritmo en lenguaje natural:

- Pedir al usuario la cantidad de estudiantes N
- Crear una lista vacía para almacenar las calificaciones
- Para i desde 1 hasta N:
 - Pedir al usuario la calificación del estudiante i
 - Agregar la calificación a la lista
- Ordenar la lista de calificaciones
- Calcular la mediana:
 - Si N es impar, mediana = valor en posición(N/2)
 - Si N es par, mediana = promedio de los valores en posiciones (N/2 - 1) y (N/2)

	<p style="text-align: center;">UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p style="text-align: center;">Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLE-001</p>	<p>Página: 4</p>

- Calcular la moda:
 - Recorrer la lista y contar la frecuencia de cada calificación usando un mapa (valor → frecuencia).
 - Identificar el valor(es) con mayor frecuencia.
- Calcular la desviación estándar :
 - Calcular el promedio (media) de las calificaciones.
 - Para cada calificación, calcular $(\text{calificación} - \text{media})^2$ y sumar todas.
 - Dividir la suma entre N.
 - Obtener la raíz cuadrada del resultado.
- Mostrar mediana, moda y desviación estándar.



```

1  import java.util.*;
2  public class Ej_propuesto01 {
3      public static void main(String[] args) {
4          Scanner scanner = new Scanner(System.in);
5          System.out.print(s:"Ingrese la cantidad de estudiantes: ");
6          int N = scanner.nextInt();
7          List<Double> calificaciones = new ArrayList<>();
8          System.out.println(s:"Ingrese las calificaciones:");
9          for (int i = 0; i < N; i++) {
10             System.out.print("Calificación #" + (i+1) + ": ");
11             calificaciones.add(scanner.nextDouble());
12         }
13         Collections.sort(calificaciones);
14
15         // Mediana
16         double mediana;
17         if (N % 2 == 1) {
18             mediana = calificaciones.get(N/2);
19         } else {
20             mediana = (calificaciones.get(N/2 - 1) + calificaciones.get(N/2)) / 2.0;
21         }
22
23         // Moda

```

PROBLEMAS 1 SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS

```

Ingrese la cantidad de estudiantes: 5
Ingrese las calificaciones:
Calificación #1: 11
Calificación #2: 11
Calificación #3: 15
Calificación #4: 13
Calificación #5: 16
Mediana: 13,00
Moda: 11,00
Desviación estándar: 2,04

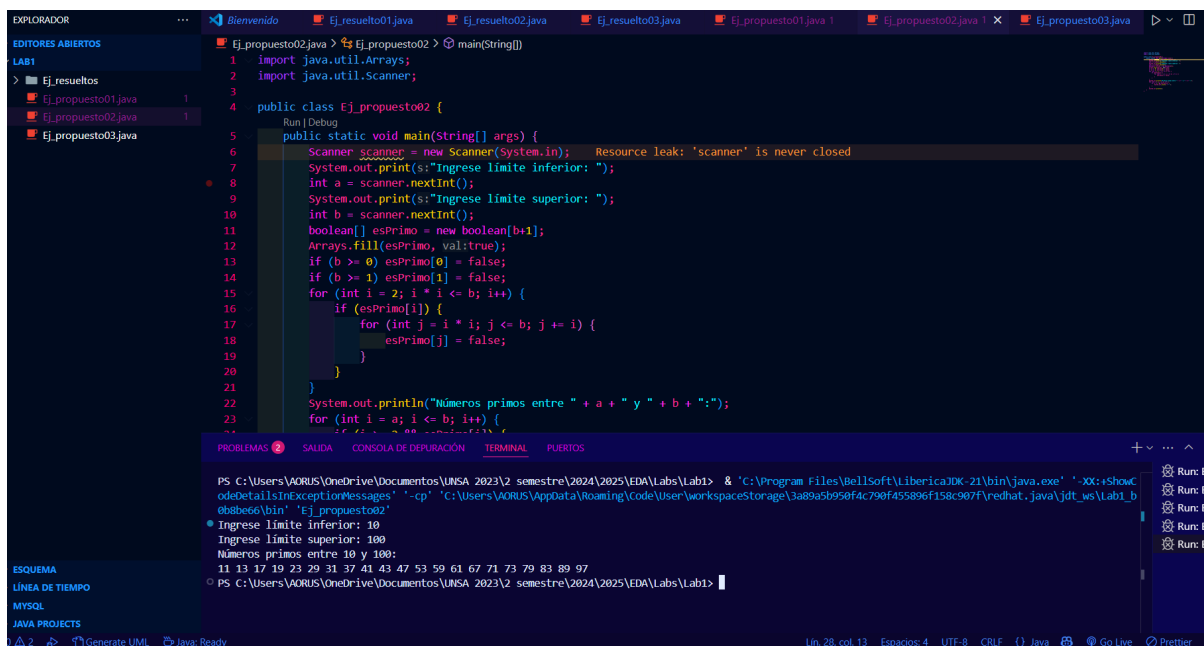
```

ii. 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

- **Algoritmo en lenguaje natural:**
 - Pedir al usuario el límite inferior a y el límite superior b.
 - Crear un arreglo booleano esPrimo de tamaño b+1, inicializado en true.
 - Marcar esPrimo[0] = false y esPrimo[1] = false.
 - Para i desde 2 hasta \sqrt{b} :

	<p style="text-align: center;">UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p style="text-align: center;">Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLE-001</p>	<p>Página: 5</p>

- Si esPrimo[i] es true, para j desde i*i hasta b en pasos de i:
 - Marcar esPrimo[j] = false.
- Recorrer i desde a hasta b y, si esPrimo[i] == true, imprimir i.



```

1 import java.util.Arrays;
2 import java.util.Scanner;
3
4 public class Ej_propuesto02 {
5     public static void main(String[] args) {
6         Scanner scanner = new Scanner(System.in);
7         System.out.print("Ingrese límite inferior: ");
8         int a = scanner.nextInt();
9         System.out.print("Ingrese límite superior: ");
10        int b = scanner.nextInt();
11        boolean[] esPrimo = new boolean[b+1];
12        Arrays.fill(esPrimo, val:true);
13        if (b >= 0) esPrimo[0] = false;
14        if (b >= 1) esPrimo[1] = false;
15        for (int i = 2; i * i <= b; i++) {
16            if (esPrimo[i]) {
17                for (int j = i * i; j <= b; j += i) {
18                    esPrimo[j] = false;
19                }
20            }
21        }
22        System.out.println("Números primos entre " + a + " y " + b + ":");
23        for (int i = a; i <= b; i++) {
24            if (esPrimo[i]) {
25                System.out.print(i + " ");
26            }
27        }
28    }
29 }

```

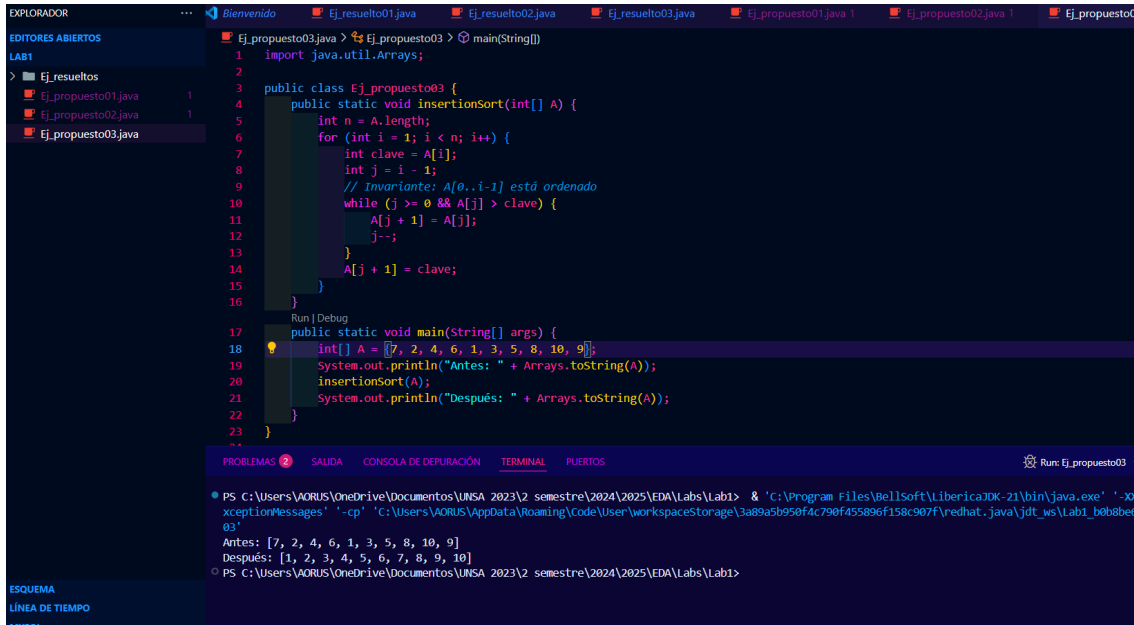
PS C:\Users\VAORUS\OneDrive\Documents\UNSA 2023\2 semestre\2024\2025\EDA\Labs\Lab1> & 'c:\Program Files\BellSoft\JibericaJDK-21\bin\java.exe' -XX:+ShowCodeDetailsInExceptionMessages -cp "C:\Users\VAORUS\AppData\Roaming\Code\User\workspaceStorage\3a89a5b950f4c790f455896f158c907F\redhat_java\jdt_ws\Lab1_b08b6e6\bin" "Ej_propuesto02"

● Ingrese límite inferior: 10
 Ingrese límite superior: 100
 Números primos entre 10 y 100:
 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97

○ PS C:\Users\VAORUS\OneDrive\Documents\UNSA 2023\2 semestre\2024\2025\EDA\Labs\Lab1>

iii. 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)

- **Algoritmo en lenguaje natural:**
 - Dada una lista A de longitud n.
 - El segmento A[0..i-1] está ordenado al inicio de cada iteración i.
 - Para i desde 1 hasta n-1:
 - Guardar clave = A[i].
 - Comparar con los elementos en A[0..i-1], desplazándolos a la derecha hasta encontrar la posición correcta.
 - Insertar clave en su lugar.
 - Al finalizar, todo el arreglo estará ordenado.



```
1 import java.util.Arrays;
2
3 public class Ej_propuesto03 {
4     public static void insertionSort(int[] A) {
5         int n = A.length;
6         for (int i = 1; i < n; i++) {
7             int clave = A[i];
8             int j = i - 1;
9             // Invariante: A[0..i-1] está ordenado
10            while (j >= 0 && A[j] > clave) {
11                A[j + 1] = A[j];
12                j--;
13            }
14            A[j + 1] = clave;
15        }
16    }
17
18    public static void main(String[] args) {
19        int[] A = {7, 2, 4, 6, 1, 3, 5, 8, 10, 9};
20        System.out.println("Antes: " + Arrays.toString(A));
21        insertionSort(A);
22        System.out.println("Después: " + Arrays.toString(A));
23    }
24 }
```

II. SOLUCIÓN DEL CUESTIONARIO

- a. ¿Cuáles fueron las dificultades que encontraste al desarrollar los ejercicios propuestos? por ejemplo, poca documentación, complejidad del lenguaje, etc.
- Manejar entradas no numéricas o valores fuera de rango requirió agregar comprobaciones o asumir datos correctos.
 - Calcular correctamente la mediana y la moda implicó ordenar y contar frecuencias, lo cual puede complicarse si hay muchas repeticiones.
 - Trabajar con double y operaciones de raíz cuadrada (Math.sqrt) puede producir ligeras inexactitudes en la desviación estándar.
 - Recordar la sintaxis (por ejemplo, List<Double>), manejo de colecciones y uso de Scanner para entrada.
 - Garantizar mentalmente la invariancia en Insertion Sort al desplazar elementos ayudó a entender y justificar la corrección del algoritmo.

III. CONCLUSIONES

- Implementar manualmente algoritmos estadísticos en Java mejora la comprensión de estructuras de datos y operaciones matemáticas básicas.

	<p align="center">UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		
Aprobación: 2022/03/01	Código: GUIA-PRLE-001	Página: 7

- El ejercicio de la Criba de Eratóstenes refuerza el uso de arrays y bucles anidados para optimizar la detección de números primos.
- Practicar el principio de invariante en Insertion Sort ayuda a desarrollar el pensamiento formal y la capacidad de razonar sobre la corrección del código

RETROALIMENTACIÓN GENERAL

- El informe presenta cada ejercicio con un enfoque progresivo, primero la formulación en lenguaje natural y luego el código en Java, lo cual facilita el entendimiento.
- Revisar espacios, numeración y estilos de encabezados para mantener uniformidad a lo largo de todo el informe.

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
- Oracle. The Java™ Tutorials. Disponible en: <https://docs.oracle.com/javase/tutorial/>.