


```
1 import java.util.Scanner;
2
3 public class EjercicioPropuesto1 {
4
5     public static void main(String[] args) {
6         Scanner sc = new Scanner(System.in);
7         System.out.println("Ingrese la cantidad de estudiantes:");
8         int n = sc.nextInt();
9         //Array de longitud "n"
10        int[] notas = new int[n];
11        //Haciendo un bucle que se repetirá "n" veces
12        for(int i=0; i<n; i++){
13            //Ingreso de notas
14            System.out.println("Ingrese la nota del estudiante " + (i+1) + ":");
15            notas[i] = sc.nextInt();
16        }
17        //Ordenamos el array usando el método "OrdenarArray"
18        OrdenarArray(notas);
19        //Imprimimos el array
20        ImprimirArray(notas);
21        //Usamos el método "Mediana"
22        System.out.println("La mediana es: " + Mediana(notas));
23        //Usamos el método "Moda"
24        System.out.println("La moda es: " + Moda(notas));
25        //Usamos el método "DesviacionEstandar"
26        System.out.println("La desviación estándar es: " + DesviacionEstandar(notas));
27
28    }
29    public static void OrdenarArray(int[] notas){
30        //Metodo Bubble Sort
31        for(int i=0; i<notas.length-1; i++){
32            for(int j=0; j<notas.length-1-i; j++){
33                //Si el numero actual es mayor que el siguiente, intercambian posiciones
34                if(notas[j] > notas[j+1]){
35                    int tmp = notas[j+1];
36                    notas[j+1] = notas[j];
37                    notas[j] = tmp;
38                }
39            }
40        }
41    }
42    public static void ImprimirArray(int[] notas) {
43        System.out.print("Notas: ");
44        for (int i = 0; i < notas.length; i++) {
45            System.out.print(notas[i]);
46            if (i < notas.length - 1) {
47                System.out.print(", ");
48            }
49        }
50        System.out.println();
51    }
52    public static double Mediana(int[] notas){
53        //Devolvemos la posición media del array si la longitud es par
54        if(notas.length % 2 == 0){
55            return (notas[notas.length/2]+notas[(notas.length/2)-1])/2.0;
56        }
57        return notas[notas.length/2];
58    }
59 }
```

```
1 public static double Mediana(int[] notas){
2     //Devolvemos la posición media del array si la longitud es par
3     if(notas.length % 2 == 0){
4         return (notas[notas.length/2]+notas[(notas.length/2)-1])/2.0;
5     }
6     return notas[notas.length/2];
7 }
8 public static int Moda(int[] notas){
9     //Suponemos que solo existe una única moda
10    int moda = notas[0];
11    int contadorMax = 0;
12    for(int i=0; i<notas.length; i++){
13        int contador = 0;
14        for(int j=0; j<notas.length; j++){
15            if(notas[j] == notas[i]){
16                contador++;
17            }
18            //Comparamos el valor de contador con contadorMax para definir la moda
19            if(contador > contadorMax){
20                contadorMax = contador;
21                moda = notas[i];
22            }
23        }
24    }
25    return moda;
26 }
27 public static double DesviacionEstandar(int[] notas){
28     double suma = 0;
29     double media = 0;
30     //Usamos for each para hallar la suma total
31     for(int nota : notas) {
32         suma += nota;
33     }
34     //Calculamos la media
35     media = suma / notas.length;
36
37     double sumaCuadrados = 0;
38     for(int nota : notas){
39         sumaCuadrados += Math.pow(nota - media, 2);
40     }
41     //Devolvemos el resultado
42     return Math.sqrt(sumaCuadrados / notas.length);
43 }
44 }
```



a. Ejecución del código.

```

Ingrese la cantidad de estudiantes:
10
Ingrese la nota del estudiante 1:
12
Ingrese la nota del estudiante 2:
5
Ingrese la nota del estudiante 3:
20
Ingrese la nota del estudiante 4:
20
Ingrese la nota del estudiante 5:
20
Ingrese la nota del estudiante 6:
7
Ingrese la nota del estudiante 7:
14
Ingrese la nota del estudiante 8:
16
Ingrese la nota del estudiante 9:
19
Ingrese la nota del estudiante 10:
19
Notas: 5, 7, 12, 14, 16, 19, 19, 20, 20, 20
La mediana es: 17.5
La moda es: 20
La desviación estándar es: 5.30659966456864

```

- Inicio del programa.
- Se solicita al usuario que ingrese la cantidad de estudiantes.
- Se crea un arreglo para almacenar las calificaciones, con una longitud igual a la cantidad de estudiantes ingresada.
- Se repite un proceso para cada estudiante:
 - Se pide al usuario que ingrese la calificación del estudiante.
 - La calificación se guarda en el arreglo correspondiente.
- Se ordenan las calificaciones de menor a mayor utilizando el método de ordenamiento burbuja:

	<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>

- Se comparan pares de elementos adyacentes.
- Si el primer elemento es mayor que el segundo, se intercambian.
- Este proceso se repite hasta que todo el arreglo esté ordenado.

- Se imprime en pantalla la lista de calificaciones ordenadas.

- Se calcula la **mediana**:
 - Si el número de calificaciones es impar, la mediana es el valor que se encuentra justo en el centro del arreglo.
 - Si el número de calificaciones es par, la mediana es el promedio de los dos valores centrales.

- Se calcula la **moda**:
 - Se recorre el arreglo para contar cuántas veces aparece cada calificación.
 - Se determina cuál es la calificación que más veces se repite (frecuencia máxima).
 - Se asume que solo hay una única moda.

- Se calcula la **desviación estándar**:
 - Se halla la media (promedio) de todas las calificaciones.
 - Se calcula la suma de los cuadrados de la diferencia entre cada calificación y la media.
 - Se divide esta suma entre el número total de calificaciones.
 - Finalmente, se obtiene la raíz cuadrada del resultado anterior.

- Se muestra por pantalla la mediana, la moda y la desviación estándar.

- **Fin del programa.**

Algoritmo GestionDeCalificaciones

Inicio

Escribir "Ingrese la cantidad de estudiantes:"

Leer n

Crear un arreglo Notas de tamaño n

Para i desde 0 hasta n - 1 hacer

Escribir "Ingrese la nota del estudiante ", i + 1

Leer Notas[i]

FinPara

Llamar a OrdenarArray(Notas)

Llamar a ImprimirArray(Notas)



mediana ← CalcularMediana(Notas)

Escribir "La mediana es: ", mediana

moda ← CalcularModa(Notas)

Escribir "La moda es: ", moda

desviacion ← CalcularDesviacionEstandar(Notas)

	<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: 6</p>

Escribir "La desviación estándar es: ", desviacion
Fin

Subalgoritmo OrdenarArray(Notas)

Para i desde 0 hasta longitud(Notas) - 2 **hacer**

Para j desde 0 hasta longitud(Notas) - 2 - i **hacer**

Si Notas[j] > Notas[j+1] **entonces**

Intercambiar Notas[j] con Notas[j+1]

FinSi

FinPara

FinPara

FinSubalgoritmo

Subalgoritmo ImprimirArray(Notas)

Escribir "Notas: "

Para i desde 0 hasta longitud(Notas) - 1 **hacer**

Escribir Notas[i] con coma si no es el último

FinPara

FinSubalgoritmo

Funcion CalcularMediana(Notas) → Real

Si longitud(Notas) es par **entonces**

return (Notas[n/2] + Notas[(n/2) - 1]) / 2.0

Sino

return Notas[n/2]

FinSi

FinFuncion

Funcion CalcularModa(Notas) → Entero

moda ← Notas[0]

contadorMax ← 0

Para i desde 0 hasta longitud(Notas) - 1 **hacer**

contador ← 0

Para j desde 0 hasta longitud(Notas) - 1 **hacer**

Si Notas[j] = Notas[i] **entonces**

contador ← contador + 1

FinSi

FinPara

Si contador > contadorMax **entonces**

contadorMax ← contador

moda ← Notas[i]

FinSi

FinPara

return moda

FinFuncion

Funcion CalcularDesviacionEstandar(Notas) → Real

suma ← 0

Para cada nota en Notas hacer

 suma ← suma + nota

FinPara

media ← suma / longitud(Notas)

sumaCuadrados ← 0

Para cada nota en Notas hacer

 sumaCuadrados ← sumaCuadrados + (nota - media)^2

FinPara

return RaízCuadrada(sumaCuadrados / longitud(Notas))

FinFuncion



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

```
1 public class EjercicioPropuesto2 {
2     public static void main(String[] args) {
3         Scanner sc = new Scanner(System.in);
4         //Pedimos al usuario que ingrese los limites
5         System.out.println("Ingrese el rango inferior");
6         int rangoInf = sc.nextInt();
7         System.out.println("Ingrese el rango superior");
8         int rangoSup = sc.nextInt();
9         //Comprobamos que los rangos son válidos y mayores que 1 para ver cuáles son primos
10        if(rangoInf < 2 && rangoInf < rangoSup){
11            System.out.println("Ingrese un rango válido");
12            return;
13        }
14        //Creamos un array de tipo boolean que empezará desde 0 hasta "rangoSup"
15        //A partir del número 2 serán True, con la criba de Eratóstenes se irán descartando los números no primos
16        boolean[] numPrimo = new boolean[rangoSup + 1];
17        for (int i=2; i<=rangoSup; i++){
18            numPrimo[i] = true;
19        }
20        // Criba de Eratóstenes
21        //Evaluamos con i*i porque no es necesario evaluar más allá de la raíz cuadrada del rangoSup
22        for (int i=2; i*i <= rangoSup; i++){
23            if (numPrimo[i]){
24                for (int j = i * i; j <= rangoSup; j += i){
25                    numPrimo[j] = false;
26                }
27            }
28        }
29        //Finalmente presentamos los números primos
30        System.out.println("Los numeros primos solicitados son:");
31        for (int i=rangoInf; i<=rangoSup; i++){
32            if (numPrimo[i]){
33                System.out.print(i + " ");
34            }
35        }
36    }
37 }
```

a. Ejecución

```
Ingrese el rango inferior
4
Ingrese el rango superior
31
Los números primos solicitados son:
5 7 11 13 17 19 23 29 31
```

- Iniciar el programa.
- Solicitar al usuario dos números: el rango inferior y el rango superior.

	<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: 9</p>

- Verificar que el rango inferior sea mayor o igual que 2 y que sea menor que el rango superior. Si no se cumple, mostrar un mensaje de error y finalizar el programa.
- Crear un arreglo booleano desde 0 hasta el rango superior, donde cada posición representa un número.
- Inicializar los valores del arreglo en `true` a partir del número 2, porque asumimos inicialmente que todos son primos.
- Aplicar la Criba de Eratóstenes:
 - Para cada número `i` desde 2 hasta la raíz cuadrada del rango superior:
 - Si el número `i` es primo (es decir, `true` en el arreglo):
 - Marcar como `false` todos sus múltiplos mayores o iguales a $i*i$, ya que no son primos.
- Imprimir todos los números dentro del rango que quedaron marcados como `true`, es decir, los que sí son primos.

Algoritmo Criba de Eratóstenes

Inicio

Escribir "Ingrese el rango inferior"

Leer rangoInf

Escribir "Ingrese el rango superior"

Leer rangoSup

Si rangoInf < 2 O rangoInf >= rangoSup Entonces

 Escribir "Ingrese un rango válido"

 Terminar programa

FinSi

Crear arreglo booleano numPrimo[rangoSup + 1]

Para `i` desde 2 hasta rangoSup hacer

 numPrimo[`i`] ← Verdadero

FinPara

Para `i` desde 2 hasta $i*i \leq$ rangoSup hacer

 Si numPrimo[`i`] = Verdadero Entonces

 Para `j` desde $i*i$ hasta rangoSup con paso `i` hacer

 numPrimo[`j`] ← Falso

 FinPara

 FinSi

FinPara

Escribir "Los números primos solicitados son:"

Para i desde rangoInf hasta rangoSup hacer
 Si numPrimo[i] = Verdadero Entonces
 Escribir i
 FinSi
FinPara

Fin

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 public class EjercicioPropuesto3 {
2     public static void main(String[] args) {
3         Scanner sc = new Scanner(System.in);
4         System.out.println("Ingrese la longitud de su array:");
5         int n = sc.nextInt();
6         int[] arr = new int[n];
7         for(int i=0; i<n; i++){
8             System.out.println("Ingrese el valor para la posición" + (i) + ":");
9             arr[i] = sc.nextInt();
10        }
11        System.out.println("Antes de ordenar:");
12        ImpArreglo(arr);
13        //Usamos el método de Insercion
14        Insercion(arr);
15        System.out.println("Después de ordenar:");
16        ImpArreglo(arr);
17    }
18    public static void ImpArreglo(int[] arr){
19        for(int num: arr){
20            System.out.print(num + " ");
21        }
22        System.out.println();
23    }
24    public static void Insercion(int[] arr){
25        for(int i=1; i<arr.length; i++){
26            int num = arr[i];
27            int j = i-1;
28            while (j >= 0 && arr[j] > num) {
29                arr[j+1] = arr[j];
30                j--;
31            }
32            arr[j+1] = num;
33        }
34    }
35 }
36
```



a. Ejecución

```

Ingrese la longitud de su array:
10
Ingrese el valor para la posición0:
8
Ingrese el valor para la posición1:
4
Ingrese el valor para la posición2:
1
Ingrese el valor para la posición3:
6
Ingrese el valor para la posición4:
15
Ingrese el valor para la posición5:
3
Ingrese el valor para la posición6:
2
Ingrese el valor para la posición7:
9
Ingrese el valor para la posición8:
17
Ingrese el valor para la posición9:
5
Antes de ordenar:
8 4 1 6 15 3 2 9 17 5
Después de ordenar:
1 2 3 4 5 6 8 9 15 17

```

- Iniciar el programa.
- Solicitar al usuario la longitud del arreglo a ordenar.
- Crear un arreglo de esa longitud.
- Solicitar al usuario que ingrese los valores del arreglo, uno por uno.
- Mostrar el arreglo antes de ordenar.
- Aplicar el algoritmo de **ordenamiento por inserción**:
 - Comenzar desde el segundo elemento (posición 1).

	<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: 13</p>

- Comparar ese elemento con los anteriores y mover los mayores una posición a la derecha.
- Insertar el elemento en su posición correcta.
- Repetir este proceso para cada elemento del arreglo, asegurando que el segmento ya procesado siempre quede ordenado.
- Mostrar el arreglo después de haberlo ordenado.

Algoritmo Ordenamiento por Inserción

Inicio

Escribir "Ingrese la longitud de su array:"

Leer n

Crear arreglo arr de tamaño n

Para i desde 0 hasta n-1 **hacer**

Escribir "Ingrese el valor para la posición ", i

Leer arr[i]

FinPara

Escribir "Antes de ordenar:"

Para cada elemento num en arr **hacer**

Escribir num

FinPara

Ordenamiento por inserción

Para i desde 1 hasta n-1 **hacer**

$num \leftarrow arr[i]$

$j \leftarrow i - 1$

Mientras $j \geq 0$ Y $arr[j] > num$ **hacer**

$arr[j + 1] \leftarrow arr[j]$

$j \leftarrow j - 1$

FinMientras

$arr[j + 1] \leftarrow num$

FinPara



Escribir "Después de ordenar:"

Para cada elemento num en arr **hacer**

Escribir num

FinPara

Fin

	<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: 14</p>

II. SOLUCIÓN DEL CUESTIONARIO

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

La dificultad fue al momento de decidir cómo diseñar el algoritmo para cada problema, por ejemplo, en el primer problema tenía que ver que algoritmo de ordenamiento podría usar, luego en el problema 2 para implementar un algoritmo en base a la Criba de Eratóstenes y en el problema 3 usando el ordenamiento por inserción. La elaboración de cada algoritmo fue compleja.

III. CONCLUSIONES

Los tres ejercicios permitieron aplicar y reforzar conceptos clave de programación en Java. Se trabajó el manejo de arreglos, estructuras de control y cálculos estadísticos con el sistema de calificaciones. Luego, con la Criba de Eratóstenes se implementó una forma eficiente de hallar números primos. Finalmente, el algoritmo de inserción ayudó a comprender cómo mantener ordenado un arreglo paso a paso.

RETROALIMENTACIÓN GENERAL

--

REFERENCIAS Y BIBLIOGRAFÍA

--