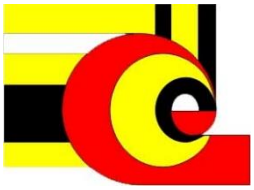
Instituto Politécnico Nacional

Centro de Investigación en Computo



Alumno

RODRÍGUEZ ARRENDONDO JACOBO

jacobo.rodriguez499@gmail.com

Diplomado en lenguaje Java

Profesor: Alan Badillo Salas

Módulo: 1

Práctica: 3

Fecha de entrega: 25 de abril de 2022

Desarrollo

En la presente práctica se implementó el método de ordenamiento “Burbuja”, el cual consiste en almacenar una “n” cantidad de datos en un arreglo de unidimensional de datos, posteriormente se procede a iterar sobre el arreglo haciendo uso de dos ciclos para que de esta forma con el segundo ciclo pueda ir comparando cada valor guardado con el valor que se encuentre en el primer ciclo; éste proceso es una analogía de cuando las burbujas de aire que se encuentran atrapadas en un líquido suben a la superficie, debido a que si se considera que las burbujas no se pueden unir, las burbujas más grandes serán las primeras en alcanzar la superficie, por lo tanto, si se acomodan las burbujas se vería como una lista descendente de mayor a menor, siendo la burbuja más grande la posición en el arreglo con el valor más grande y así sucesivamente.

Posteriormente se implemente el pseudo código en lenguaje java, en el cual se implemento dos una clase llamada “Fun” en la cual se escribieron las funciones utilizadas, las funciones utilizadas se crearon con el objetivo de leer los datos ingresados por el usuario, implementar el algoritmo de ordenamiento, imprimir un arreglo unidimensional de n elemento, y crear un arreglo unidimensional de tipo entero (Integer), a continuación, se agrega dicho código.

Clase Fun

```
1.  package com.company;
2.
3.  import org.jetbrains.annotations.NotNull;
4.
5.  import javax.lang.model.type.NullType;
6.  import java.util.Scanner;
7.
8.  public class Fun {//clase de funciones que implementan la lectura, de datos,
    impresión de arreglos de datos y el método
9.      //de ordenamiento "burbuja"
10.     public static String Leer(){// función que retorna una valor String para leer
    los datos provenientes del teclado del
11.         //usuario
12.         Scanner scn=new Scanner(System.in);/*se crea un objeto scanner para poder
    capturar los datos que dese ingresar
13.         el usuario*/
14.         String dato=scn.nextLine();/*con la variable String datos se almacenan
    el/los datos introducidos por el usuario*/
15.         return dato;// se regresa el valor capturado para poder hacer uso de el en
    otras partes del programa.
16.     }
17.     public static int[] LeerLista(int n){/*función la cual lee n-1 datos
    introducidos por el usuario y retorna una lista
18.     de tamaño n*/
19.         int[] lst=new int[n];//creación de la lista con capacidad para n elementos
20.         System.out.println("Ingrese los valores de la lista:");
21.         for(int i=0; i<n; i++)//ciclo for para asignar las n-1 posiciones con los
    datos que desee introduccir el usuario
22.         {
23.             System.out.print("lst["+i+"]: ");// impresión de la posición en la
    lista que va almacenar el dato
24.             lst[i]=Integer.parseInt(Leer());//parsing de los datos introducidos
    por el usuario a entero y asignación en
25.             //el espacio de memoria correspondiente.
26.         }
27.         return lst;//retorno de la lista creada por el usuario
```

```
28.
29.     }
30.     public static void Imp(int[] lst){/*función para imprimir todos los elementos
    que se encuentren en un arreglo
31.     tipo entero*/
32.         if(lst== null){//verificación que la lista no este vacía
33.             System.out.println("La lista que quiere imprimir esta vacia");
34.         }
35.         else{
36.             for(int elemt: lst)/*ciclo for que itera sobre todos los elementos de
    la lista para imprirlos*/
37.             {
38.                 System.out.println(elemt);
39.             }
40.         }
41.     }
42.     public static void Burbuja(int[] lst,int n){/*función para la implementación
    del método de ordenamiento burbuja
43.     el cual toma como parámetros un arreglo de entero un entero que termina la
    cantidad de datos que debe almacenar
44.     el arrego*/
45.         int opc=0;//variable para el almacenamiento de un dato de manera temporal
46.         for(int i=0; i<n; i++)/*ciclo for para iterar sobre cada elemento del
    arreglo*/
47.         {
48.             for(int j=0; j<n; j++)/*ciclo for anidado para la iteración de cada
    elemento del arreglo*/
49.             {
50.                 if(lst[j]>=lst[i])/*sentencia condicional para verificar que el
    valor actual del arreglo sea mayor o
51.                 menor del valor actual en la lista, en caso de que se cumpla la
    condición se realizará lo siguiente:*/
52.                 {
53.                     opc=lst[j];//el siguiente valor de la lista es guardado en la
    variable temporal
54.                     lst[j]=lst[i];//el valor actual se le asigna a la siguiente
    dirección de memoria
55.                     lst[i]=opc;//la dirección de memoria actual se le asigna el
    valor guardado en la variable temporal
56.                 }
57.             }//for 2
58.         }//for 1
59.     }
60.
61. }
```

Clase Main

```
1.  package com.company;
2.
3.  import static com.company.Fun.*;
4.
5.  public class Main {
6.
7.      public static void main(String[] args) {
8.          System.out.println("Dame el tamaño de la lista ordenada: ");
9.          int n = Integer.parseInt(Leer());//implementación de la función leer, el
    resultado regresado por la función se
10.         //convierte en entero para poder usar el dato como un número
```

```
11.         int[] lst=LeerLista(n);//implementación de la función leer lista para
    asignar los valores deseados por el usuario
12.         System.out.println("Lista antes de ordenar:");
13.         Imp(lst);// impresión de la lista sin ordenar
14.         Burbuja(lst,n);//implementación del método de ordenamiento burbuja
15.         System.out.println("Lista ordenada:");
16.         Imp(lst);// impresión de la lista ordenada
17.
18.     }
19. }
```

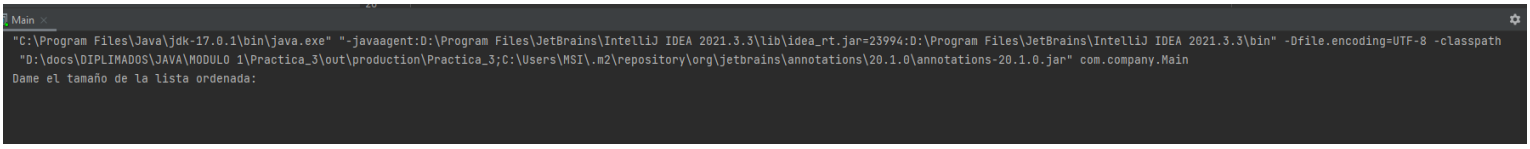


Ilustración 1 Compilación del programa

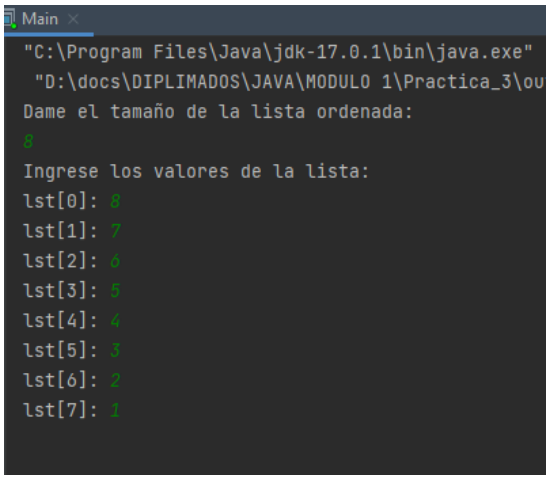


Ilustración 2 Ejecución del programa (captura de datos)

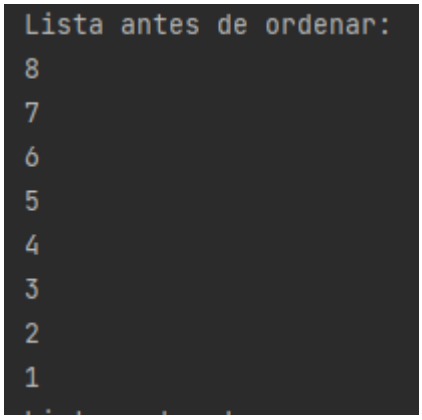


Ilustración 3 Impresión de la lista introducida por el usuario

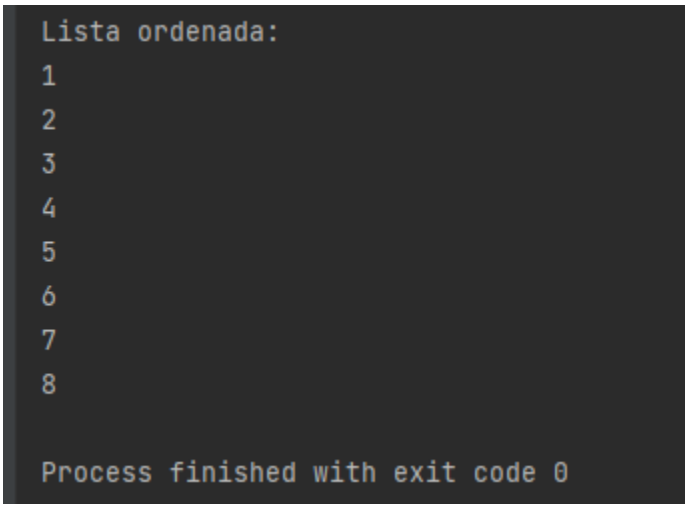


Ilustración 4 impresión de la lista después de ser ordenada y finalización del programa

Conclusiones

En esta práctica se pudo observar de una manera más practica la implementación de uno de los método ordenamiento vistos en las sesiones y del empleo de métodos de estructura, sentencias y ciclos utilizados en diversos lenguajes pero aplicando la sintaxis necesaria de Java, la cual es de fácil empleo ya que la estructura de este lenguaje resulta ser muy cómoda debido a que no hay que preocuparse en respetar a su totalidad las tabulaciones en ciclos, sentencias y funciones haciendo que la implantación de una programación estructurada sea más fácil.