Alumno—
Jesus Octavio Amarillas Amaya

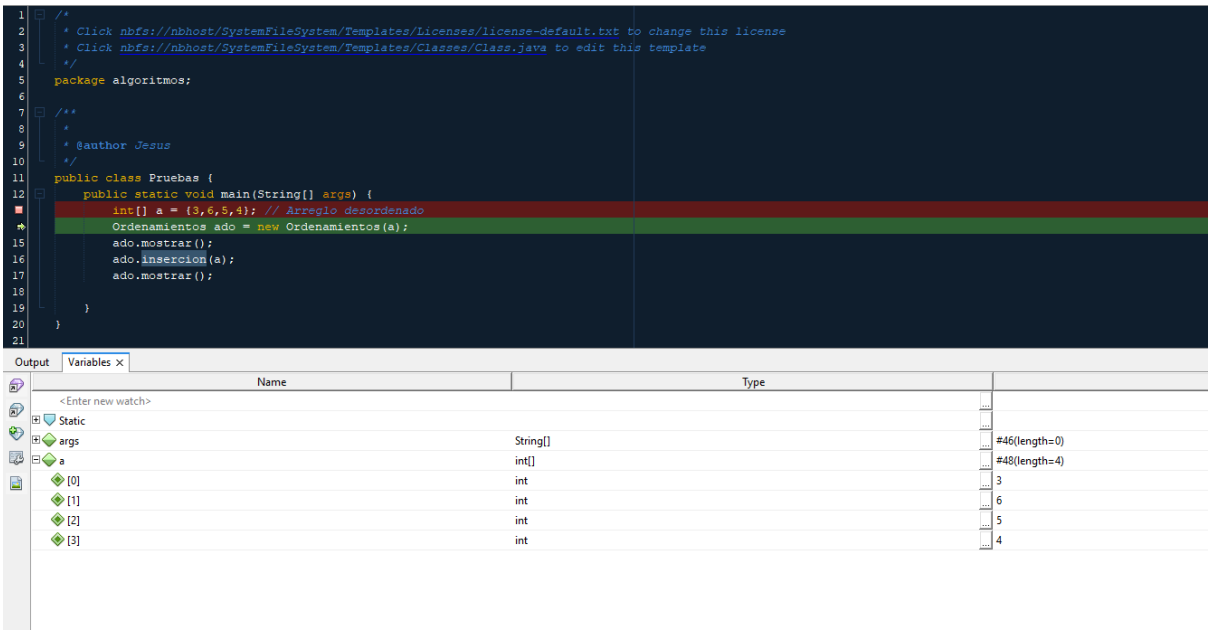
ID—
207653

Asignación—
Algoritmo inserción

Materia—
Análisis de Algoritmos

Profesor—
Sergio Castellanos Bustamante

1. **Se declara el arreglo y se accede al metodo de insercion.**

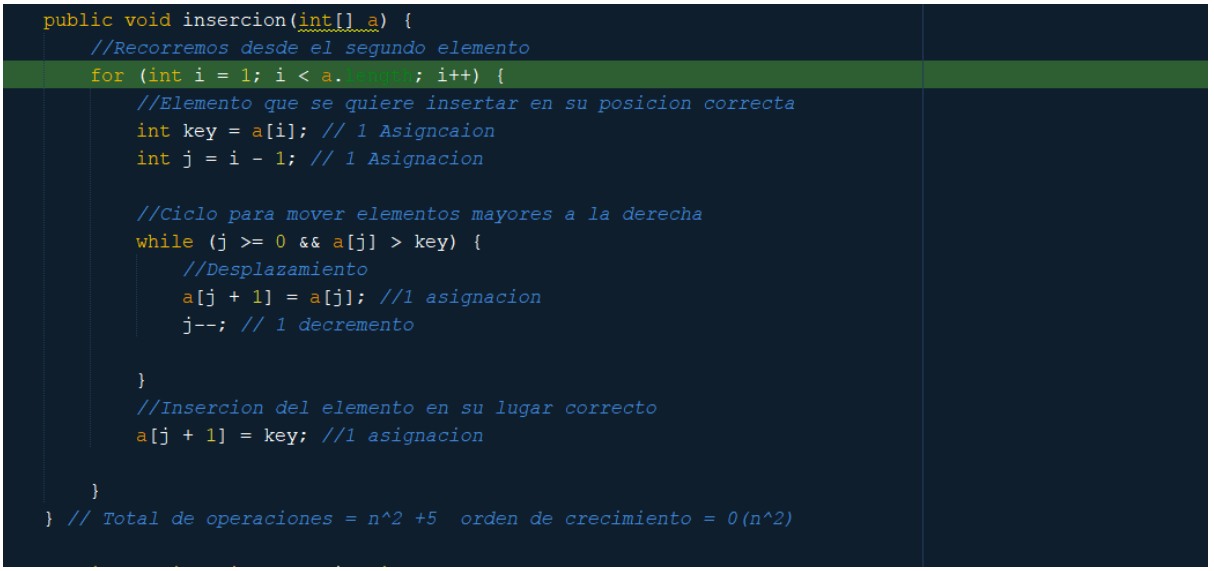


```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package algoritmos;

/**
 *
 * @author Jesus
 */
public class Pruebas {
    public static void main(String[] args) {
        int[] a = {3,6,5,4}; // Arreglo desordenado
        Ordenamientos ado = new Ordenamientos(a);
        ado.mostrar();
        ado.insercion(a);
        ado.mostrar();

    }
}
```

Output | Variables

Name	Type	
<Enter new watch>		
Static		
args	String[]	#46(length=0)
a	int[]	#48(length=4)
[0]	int	3
[1]	int	6
[2]	int	5
[3]	int	4

2. **Se inicializa la llave y el índice j, después se hace la inserción al segundo elemento por la llave.**



```
public void insercion(int[] a) {
    //Recorremos desde el segundo elemento
    for (int i = 1; i < a.length; i++) {
        //Elemento que se quiere insertar en su posicion correcta
        int key = a[i]; // 1 Asigncaion
        int j = i - 1; // 1 Asignacion

        //Ciclo para mover elementos mayores a la derecha
        while (j >= 0 && a[j] > key) {
            //Desplazamiento
            a[j + 1] = a[j]; //1 asignacion
            j--; // 1 decremento

        }
        //Insercion del elemento en su lugar correcto
        a[j + 1] = key; //1 asignacion

    }
} // Total de operaciones = n^2 +5  orden de crecimiento = 0(n^2)
```

3. **Se entra al while para pover todols los elemento mayores a la llave a la derecha y se hace la insercion de los elementos en los lugares**

correctos

```
    public void insercion(int[] a) {
        //Recorremos desde el segundo elemento
        for (int i = 1; i < a.length; i++) {
            //Elemento que se quiere insertar en su posicion correcta
            int key = a[i]; // 1 Asigncaion
            int j = i - 1; // 1 Asignacion

            //Ciclo para mover elementos mayores a la derecha
            while (j >= 0 && a[j] > key) {
                //Desplazamiento
                a[j + 1] = a[j]; //1 asignacion
                j--; // 1 decremento

            }
            //Insercion del elemento en su lugar correcto
            a[j + 1] = key; //1 asignacion

        }
    } // Total de operaciones = n^2 +5  orden de crecimiento = 0(n^2)

    public static void seleccion(int[] a) {
```

Variables

Name	Type	Value
<Enter new watch>		
this	Ordenamientos	#51
a	int[]	#48(length=4)
[0]	int	3
[1]	int	6
[2]	int	6
[3]	int	4
i	int	2
key	int	5
j	int	0

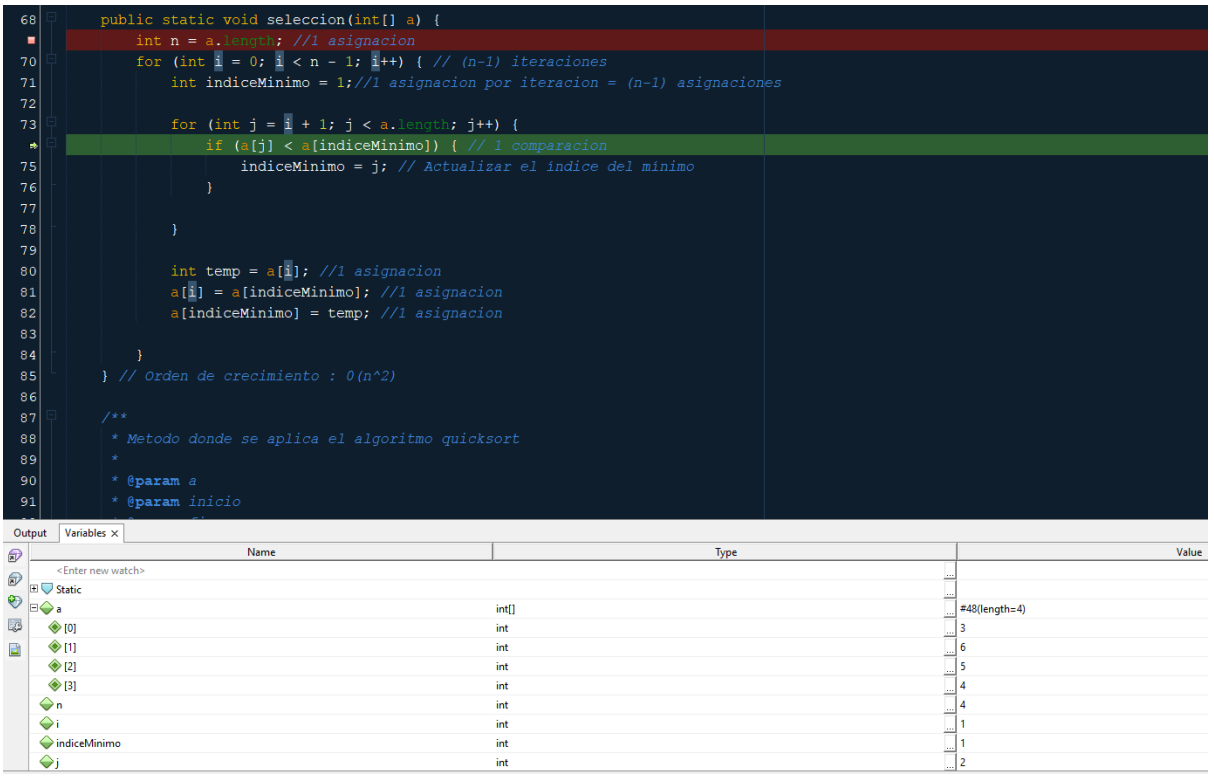
4. **Una vez finalizado los elementos en el orden correcto se termina el método y se sale al main y se accede el método mostrar.**

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package algoritmos;

/**
 *
 * @author Jesus
 */
public class Pruebas {
    public static void main(String[] args) {
        int[] a = {3,6,5,4}; // Arreglo desordenado
        Ordenamientos ado = new Ordenamientos(a);
        ado.mostrar();
        ado.insercion(a);
        ado.mostrar();

    }
}
```

Output | Variables

Name	Type	Value
Static		
args	String[]	#46(length=0)
a	int[]	#48(length=4)
[0]	int	3
[1]	int	4
[2]	int	5
[3]	int	6
ado	Ordenamientos	#51
a	int[]	#48(length=4)
[0]	int	3
[1]	int	4
[2]	int	5
[3]	int	6

5. **Se hace una comparacion entre enl indiceMinimo y la variable j si es menor el indiceMinimo sera intercambiado por .**

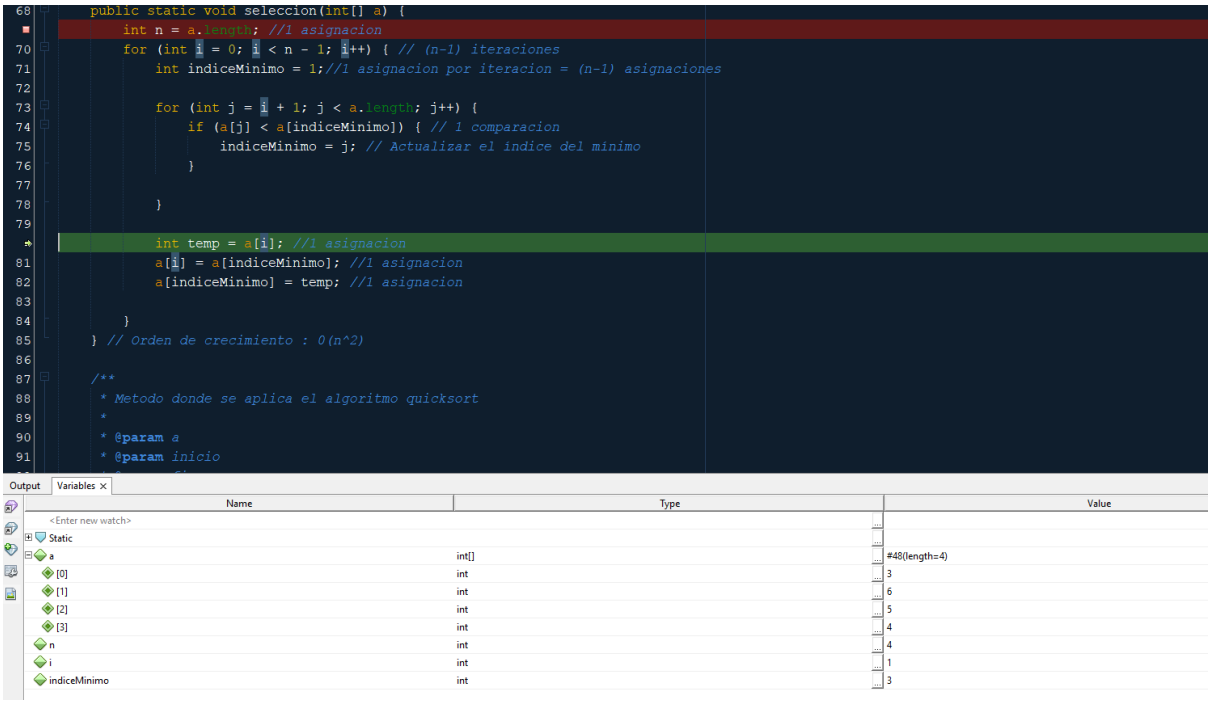


```
68      public static void seleccion(int[] a) {
            int n = a.length; //1 asignacion
70          for (int i = 0; i < n - 1; i++) { // (n-1) iteraciones
71              int indiceMinimo = 1;//1 asignacion por iteracion = (n-1) asignaciones
72
73              for (int j = i + 1; j < a.length; j++) {
                    if (a[j] < a[indiceMinimo]) { // 1 comparacion
75                      indiceMinimo = j; // Actualizar el indice del minimo
76                  }
77
78              }
79
80              int temp = a[i]; //1 asignacion
81              a[i] = a[indiceMinimo]; //1 asignacion
82              a[indiceMinimo] = temp; //1 asignacion
83
84          }
85      } // Orden de crecimiento : 0(n^2)
86
87      /**
88       * Metodo donde se aplica el algoritmo quicksort
89       *
90       * @param a
91       * @param inicio
```

Output | Variables

Name	Type	Value
<Enter new watch>		
Static		
a	int[]	#48(length=4)
[0]	int	3
[1]	int	6
[2]	int	5
[3]	int	4
n	int	4
i	int	1
indiceMinimo	int	1
j	int	2

6. **Se termina el for anidado por lo que se realiza el intercambio de variables. Se crea una variable auxiliar con valor del indice del arreglo índice Mínimo con valor al índice primero del arreglo .**

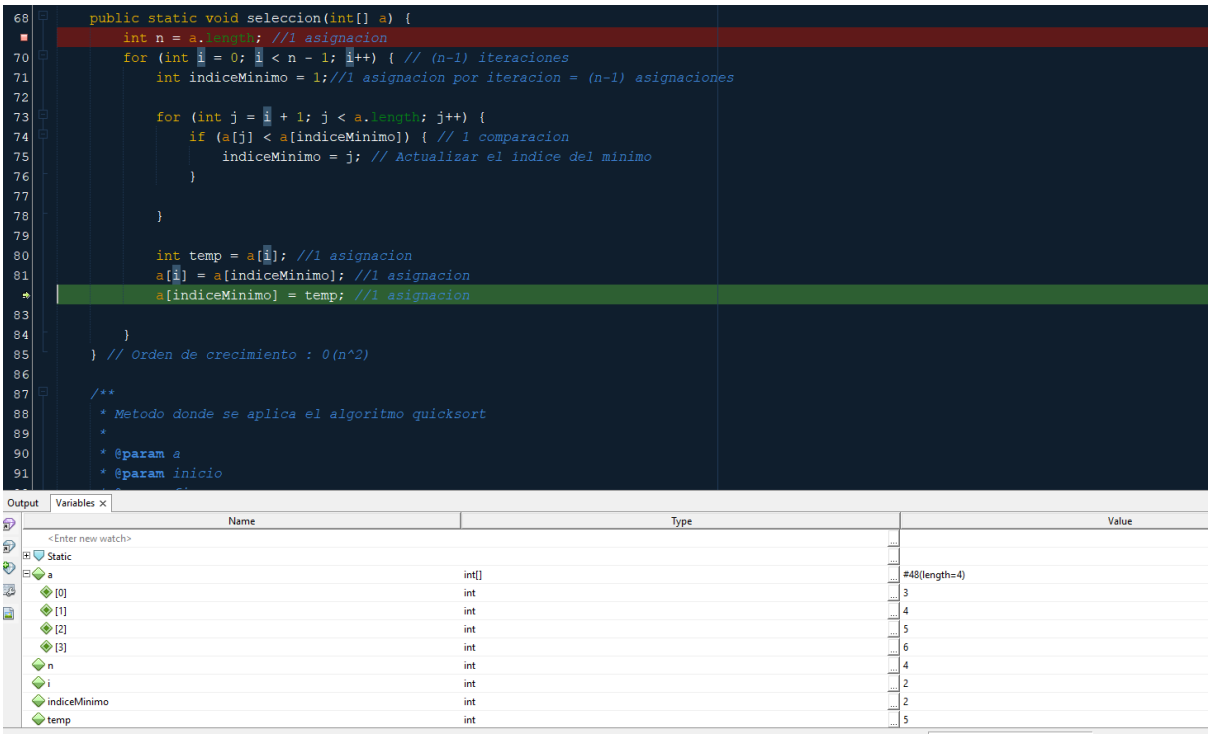


```
68      public static void seleccion(int[] a) {
            int n = a.length; //1 asignacion
70          for (int i = 0; i < n - 1; i++) { // (n-1) iteraciones
71              int indiceMinimo = 1;//1 asignacion por iteracion = (n-1) asignaciones
72
73              for (int j = i + 1; j < a.length; j++) {
74                  if (a[j] < a[indiceMinimo]) { // 1 comparacion
75                      indiceMinimo = j; // Actualizar el indice del minimo
76                  }
77
78              }
79
80              int temp = a[i]; //1 asignacion
81              a[i] = a[indiceMinimo]; //1 asignacion
82              a[indiceMinimo] = temp; //1 asignacion
83
84          }
85      } // Orden de crecimiento : 0(n^2)
86
87      /**
88       * Metodo donde se aplica el algoritmo quicksort
89       *
90       * @param a
91       * @param inicio
```

Output | Variables

Name	Type	Value
<Enter new watch>		
Static		
a	int[]	#48(length=4)
[0]	int	3
[1]	int	6
[2]	int	5
[3]	int	4
n	int	4
i	int	1
indiceMinimo	int	3

7. **Se repiten los pasos en caso de ser necesario para que el arreglo este ordenado**



```
68      public static void seleccion(int[] a) {
            int n = a.length; //1 asignacion
70          for (int i = 0; i < n - 1; i++) { // (n-1) iteraciones
71              int indiceMinimo = 1;//1 asignacion por iteracion = (n-1) asignaciones
72
73              for (int j = i + 1; j < a.length; j++) {
74                  if (a[j] < a[indiceMinimo]) { // 1 comparacion
75                      indiceMinimo = j; // Actualizar el indice del minimo
76                  }
77
78              }
79
80              int temp = a[i]; //1 asignacion
81              a[i] = a[indiceMinimo]; //1 asignacion
*               a[indiceMinimo] = temp; //1 asignacion
83
84          }
85      } // Orden de crecimiento : 0(n^2)
86
87      /**
88       * Metodo donde se aplica el algoritmo quicksort
89       *
90       * @param a
91       * @param inicio
```

Output | Variables

Name	Type	Value
<Enter new watch>		
Static		
a	int[]	#48(length=4)
[0]	int	3
[1]	int	4
[2]	int	5
[3]	int	6
n	int	4
i	int	2
indiceMinimo	int	2
temp	int	5