



## Carátula para entrega de prácticas

Facultad de Ingeniería

Laboratorio de docencia

# Laboratorios de computación salas A y B

Tista Garcia Edgar

*Profesor:*

Estructura de datos y algoritmos II

*Asignatura:*

5

*Grupo:*

4

*No de Práctica(s):*

Calzada Martinez Jonathan Omar

*Integrante(s):*

*No. de Equipo de  
cómputo empleado*

35

2019-2

*Semestre:*

*Fecha de entrega:*

*Obervaciones:*

**CALIFICACIÓN:**

**Objetivos:** Estudiante identificarà el comportamiento y características de los principales algoritmos de búsqueda por comparación de llaves.

## **Introduccion:**

Un algoritmo de búsqueda es aquel que está diseñado para localizar un elemento con ciertas propiedades dentro de una estructura de datos; por ejemplo, ubicar el registro correspondiente a cierta persona en una base de datos.

En esta ocasión estudiaremos el algoritmo de búsqueda lineal o secuencial y el algoritmo de búsqueda binaria.

## **Desarrollo:**

### **Ejercicio1. Listas en Java)**

Se hizo uso de la utilidad LinkedList así como de List que nos ayudó a poder hacer uso de listas. Las listas nos ayudan a manipular grandes volúmenes de datos.

Lo creamos con la lista con la instrucción con new LinkedList, crear listas en Java es mucho más sencillo que crearlas en C ya que en C se utilizan estructuras, y en Java también pero la manera en como se declaran es más sencilla. Java lo maneja algo así como de manera implícita.

A) -Explica la diferencia entre los métodos "set" y "add"

La diferencia entre set y add es que set nos ayuda a intercambiar valores que nosotros necesitemos pasándole el índice y el número a intercambiar.

"add" es un método que solo nos ayuda a agregar un valor en la última posición (cola).

B) -explica el funcionamiento del método "sub-list"

Este método elimina la necesidad de operaciones de rango explícitas. Cualquier operación que espere una lista se puede usar como una operación de rango pasando una vista de lista secundaria en lugar de una lista completa.

Esto quiere decir que para realizar una operacion en una lista se puede pasar solo una parte de la lista en dado caso de que solo se necesite realizar la accion o operacion en esa parte de la lista y de esa manera no realizar la accion en toda la lista completa.

```
61     System.out.println(lista1.equals(lista2));
62 }
63 public static void imprimirLista(List<Integer> listaPrint){
64     // for(Integer var: listaPrint){
65         System.out.println(listaPrint);
66     //}
67 }
68 }
69 }
70 }
71 }
72 }
73 }

Output - practica4 (run) x
run:
Estado punto 1
[15, 25, 45, 50, 26]
***
Estado punto 2
[15, 300, 25, 500, 45, 700, 50, 26]
***
false
-1
Estado punto 3
[4, 300, 500, 45, 700, 50, 8]
***
[45, 700, 50]
***
false
BUILD SUCCESSFUL (total time: 0 seconds)
```

Se implemento lo que se pide en el punto I) que es borrar un elemento de la lista y se uso la untruccion "remove"

```
System.out.println(listal);  
listal.remove(2);  
System.out.println(listal);
```

Para poder ver si un elelemto esta o no en la lista se utilizo la intruccion contains que nos devuelve un verdadero o un falso en caso de que este o no este

```
System.out.println(listal.contains(300));  
//System.out.println(listal.indexOf(59));
```

Y se agregaron las instrucciones necesarias para poder saber si la lista esta vacia o no.

```
if (!listal.isEmpty()) {  
    System.out.println("La lista no esta vacia");  
} else {  
    System.out.println("La lista esta vacia");  
}  
  
System.out.println("////////////////////////////////////////");  
}
```

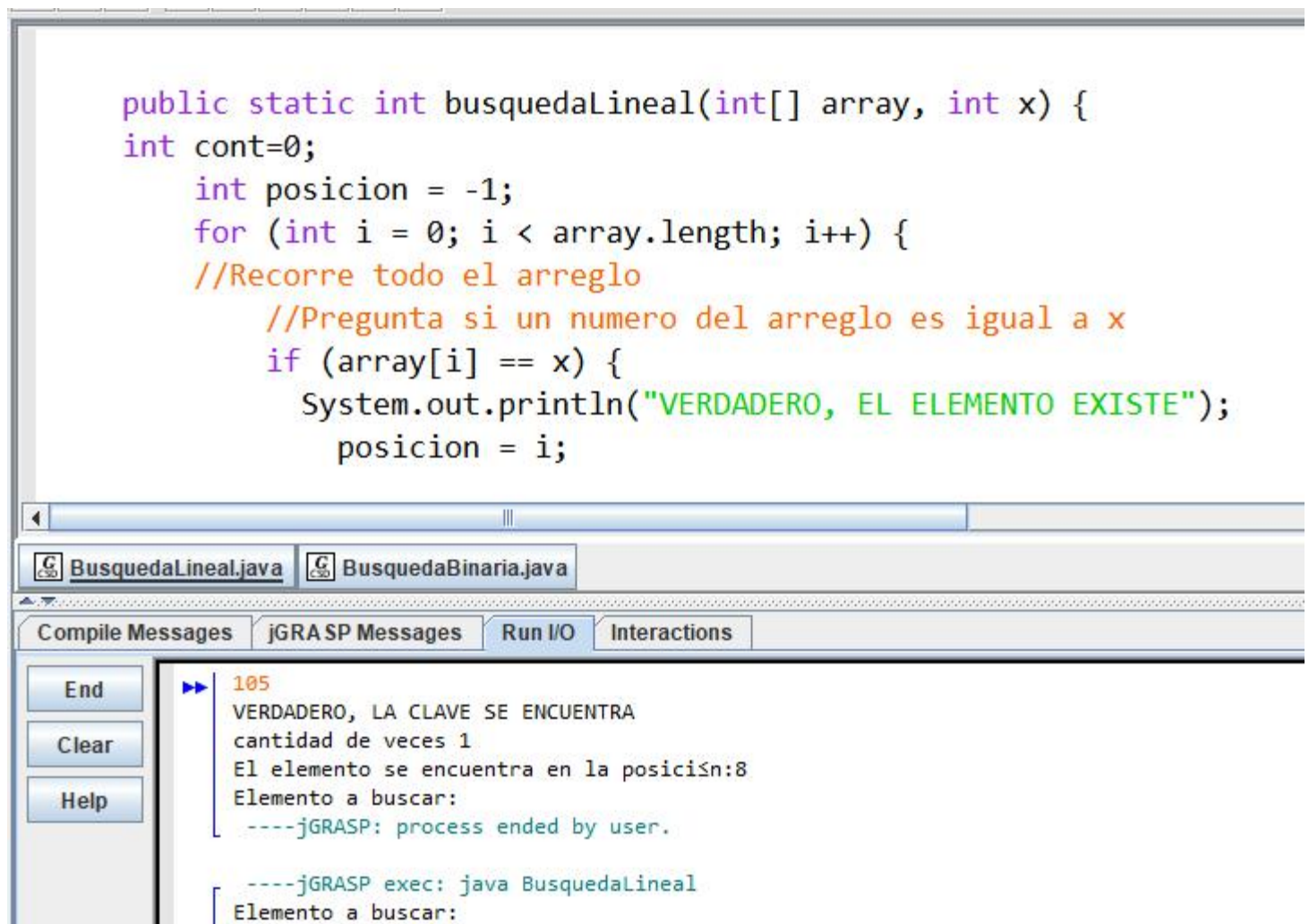
Y esto son los resultados.

```
////////////////////////////////////////  
[4, 300, 500, 60, 700, 50, 8]  
[4, 300, 60, 700, 50, 8]  
true  
La lista no esta vacia  
////////////////////////////////////////  
BUILD SUCCESSFUL (total time: 1 second)
```

## Ejercicio 2. busqueda Secuencial)

### Busqueda secuencial.

A decir verdad este algoritmo estuvo relativamente sencillo ya que prácticamente es lo mismo que ya había hecho en los anteriores lenguajes de programación que es ir recorriendo cada elemento del arreglo hasta encontrarlo. Lo que se me dificultó de este ejercicio fue que todavía ando aprendiendo java y no sabía como realizar las funciones. Intenté hacerlo con listas como en el ejercicio pasado que supongo que por eso mismo se realizó el ejercicio para que lo hagamos de esa manera pero por el motivo de que aun no estoy muy familiarizado con el lenguaje tuve muchos problemas al pasar listas por parametros en las funciones y tuve que hacer los ejercicios con arreglos. Al final se pudo realizar el ejercicio con éxito.



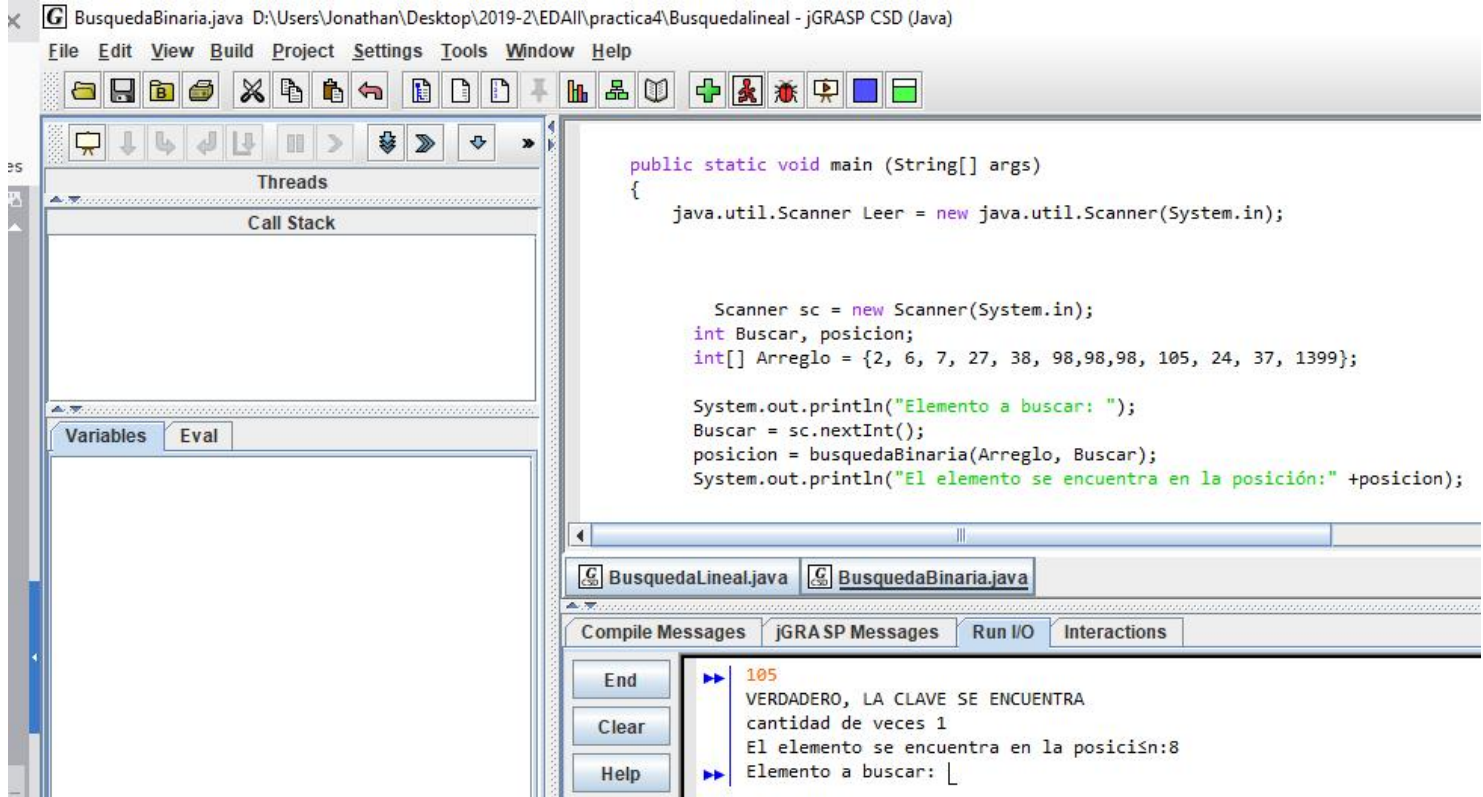
```
public static int busquedaLineal(int[] array, int x) {
    int cont=0;
    int posicion = -1;
    for (int i = 0; i < array.length; i++) {
        //Recorre todo el arreglo
        //Pregunta si un numero del arreglo es igual a x
        if (array[i] == x) {
            System.out.println("VERDADERO, EL ELEMENTO EXISTE");
            posicion = i;
        }
    }
}
```

The screenshot shows a Java IDE with two tabs: `BusquedaLineal.java` and `BusquedaBinaria.java`. The `BusquedaLineal.java` tab is active, displaying the code for a linear search algorithm. Below the code editor, there are tabs for `Compile Messages`, `jGRASP Messages`, `Run I/O`, and `Interactions`. The `Run I/O` tab is selected, showing the output of the program. The output includes the text "VERDADERO, LA CLAVE SE ENCUENTRA", "cantidad de veces 1", "El elemento se encuentra en la posición:8", and "Elemento a buscar:". Below this, there are two lines of text: "----jGRASP: process ended by user." and "----jGRASP exec: java BusquedaLineal", followed by "Elemento a buscar:".

## Ejercicio 3. Busqueda binaria.

Bueno a decir verdad este algoritmo si me costo más tiempo de lo que tenía pensado ya que tuve un poco de problemas al implementar el algoritmo ya que estaba un poco






confundido en la comparación del centro y por lo tanto me tarde mucho en este ejercicio. El arreglo lo implemente usando el mismo que el que utilice en el de Busqueda lineal para hacerlo más comodo.



## Conclusiones:

Se cumplió con el objetivo ya que se entendió la manera de trabajar de los algoritmos de búsqueda binaria y secuencial.

En general se tuvo problemas para unir los archivos o las clases al menú principal, ciertamente aun sigo aprendiendo java y en esa parte un me quedan un poco de dudas. Lo que hice para poder trabajar fue realizar los archivos por fuera utilizando un ide externo llamado JGRASP para poder probar el funcionamiento de los códigos del ejercicio 2 y 3 ya que como no pude implementarlo en el mismo proyecto el código me daba errores.

 build	3/2/2019 2:23 PM	jGRASP XML file	4 KB
 BusquedaAlgoritmo.class	3/2/2019 10:18 PM	CLASS File	1 KB
 BusquedaBinaria.class	3/2/2019 11:35 PM	CLASS File	3 KB
 BusquedaBinaria	3/2/2019 11:35 PM	jGRASP Java file	2 KB
 BusquedaLineal.class	3/2/2019 11:38 PM	CLASS File	2 KB
 BusquedaLineal	3/2/2019 10:16 PM	jGRASP Java file	2 KB
 CALZADAMARTINEZJONATHAN	3/3/2019 12:08 AM	WPS PDF Docume...	341 KB
 manifest.mf	3/2/2019 2:23 PM	MF File	1 KB

El algoritmo de busqueda lineal fue sencillo ya que ese algoritmo lo he echo muchas veces. En tonde si me tarde un poco fue en el de busqueda binaria por el echo de que estaba un poco confundido.

El ejercicio 4 por falta de tiempo no lo pude concluir.