UNIVERSIDAD DE CÓRDOBA
comprometida con el desarrollo regional

CENTRO DE INNOVACION EN TIC PARA APOYO A LA ACADEMIA
CINTIA

MÉTODOS DE BÚSQUEDA

DESARROLLO

CONCEPTO MÉTODOS DE BÚSQUEDA:

Los métodos de búsqueda permiten recuperar información de un vector o un archivo, que contenga una lista de datos. Por ejemplo se puede obtener el nombre y el número telefónico de nuestra agenda de contactos o la nota obtenida por un alumno en la lista de un curso.

Cuando se realizan búsquedas sobre vectores, se desea es encontrar la posición que ocupa el elemento buscado dentro de la lista de elementos que contiene el vector. Para la búsqueda de información en archivos es necesario realizar la búsqueda a partir de un campo clave dentro del archivo.

Existen diferentes métodos de búsqueda y se puede determinar con cual método trabajar dependiendo de la cantidad de elementos que existan en el vector o la organización de dichos elementos.

A continuación para determinar si un elemento pertenece a un conjunto de elementos e indicar su posición dentro de un vector, se utilizaran los métodos de búsqueda secuencial (lineal) y búsqueda binaria.

MÉTODOS DE BÚSQUEDA UTILIZADOS:

- **Búsqueda Secuencial o Lineal:**

En este método se recorre el vector desde el primer elemento hasta el último, comparando cada elemento del vector con el valor buscado, hasta que se encuentre el elemento o se llegue al final del vector. Este método es recomendado para realizar búsquedas con pocos datos.

Implementación del método que busca lineal o secuencial, y devuelve la posición del dato a buscar que se pasa como parámetro:

```
//Implementación del método que busca lineal o secuencialmente sobre los datos del vector.
    public int busquedaLineal(int dato){
        int i; //Variable para controlar el ciclo while.
        int posicion; //Variable que devuelve la posición en la que se encuentra el elemento buscado.
//Se Asigna el valor de -1 a la variable posición para devolver este valor en el supuesto caso de
//que no se encuentre el valor buscado dentro del vector.
        posicion = -1;
        i = 0;
//Mientras que no se llegue al final del vector y no se haya encontrado el dato buscado en el vector.
        while ((i <= getNumElementos()-1) && (posicion == -1)){
//Si el contenido del vector en la posición i-esima es igual al dato que se está buscando, entonces
```

```
//el dato si está en el vector y se devuelve la posición en donde se encuentra el dato dentro del
//vector. Si no, es porque el dato no se encuentra en esa posición, y se incrementa la posición (i)
//para una nueva comparación.
          if (getVectorDatos(i) == dato){
             posicion=i;
          }else{
             i=i+1;
          }
       }
       return posicion;
    }
```

- **Búsqueda Binaria:**

Este método es una técnica eficaz para realizar búsquedas en vectores o archivos que contengan un mayor número de datos. Este método divide el vector en mitades de manera sucesiva hasta que encuentra el dato buscado, es decir, el método divide el vector y se examina el elemento central del vector.

Si el elemento central es el dato que se busca, entonces la búsqueda finaliza, pero sino, se determina si el dato buscado está en la primera o la segunda mitad del vector y se repite el proceso en la nueva mitad, buscando su elemento central.

Para realizar la búsqueda binaria el vector debe estar ordenado, por esta razón hay que utilizar uno de los métodos ya estudiados; luego de tener el vector ordenado, se comienza comparando con el elemento central.

Implementación del método que busca de manera binaria, y devuelve la posición del dato a buscar que se pasa como parámetro:

```
public int bsuquedaBinaria(int dato){
       int posicion, izq, der, centro; //Estas 4 variables almacenan posiciones del vector.
       ordenarIntercambio(); //Para realizar la búsqueda el vector debe estar ordenado.
       izq = 0; //Primera posición del vector.
       der = getNumElementos()-1; //Ultima posición del vector.
//Se asigna el valor de -1 a la variable posición para devolver este valor en el supuesto caso de que
//no se encuentre dato buscado dentro del vector.
       posicion = -1;
//Mientras que no se llegue al final del vector y no se haya encontrado el dato buscado en el vector.
       while ((izq <= der) && (posicion == -1)){
//Se busca cual es la posición del dato que se encuentra en el centro del vector.
          centro = (izq + der) / 2;
//Si el dato buscado es igual a lo que tiene el vector en la posición del centro entonces ya se encontró el
//elemento buscado y se devuelve la posición en donde se encontró, pero sino, entonces
//se determina si el elemento está a la izquierda o a la derecha del vector. Y se procede a buscar el
//dato hacia el inicio del vector (izquierda) o hacia el final del vector (derecha), esto si el dato
//buscado es mayor o menor al elemento que se encuentra en el centro del vector.
          if (dato == (getVectorDatos(centro)) ){
             posicion = centro;
             }else{
                if (dato < (getVectorDatos(centro))){
```

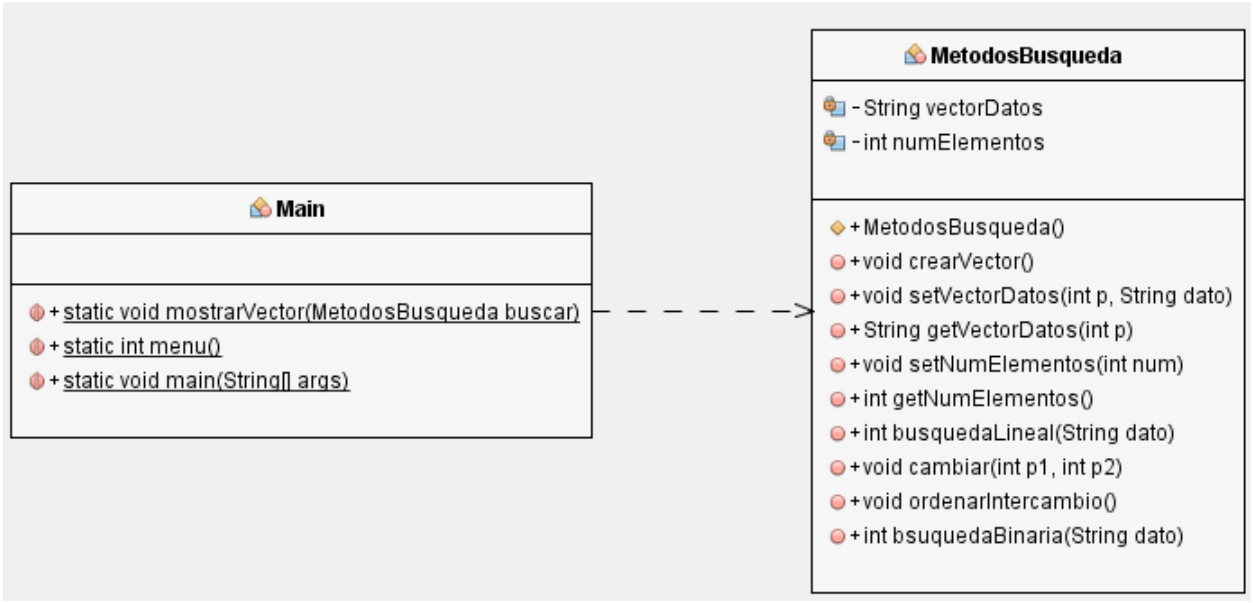
```
                der = centro-1;
            }else{
                izq = centro+1;
            }
        }
    }
    return posicion;
}
```

IMPLEMENTACIÓN DE LOS MÉTODOS DE ORDENAMIENTO EN JAVA:

- **Ejercicio propuesto implementado en Java**

A continuación se implementa un ejercicio que permite almacenar una cantidad específica de códigos en un vector y posteriormente se implementan los métodos de búsqueda descritos anteriormente, para determinar si un código pasado como dato de búsqueda se encuentra almacenado dentro del vector.

- **Diseño de clases UML de la solución**



- **Implementación de la clase MetodosBusqueda en el fichero MetodosBusqueda.java**

En esta clase se implementan los métodos de búsqueda binaria y búsqueda secuencial. Igualmente se implementa un método de ordenamiento, necesario para ordenar el vector y realizar la búsqueda binaria.

```
public class MetodosBusqueda {
    //Se declaran los atributos de la clase, en este caso se declara el vector (vectorDatos).
    //y un atributo para asignar el número de elementos que tendrá el vector (numElementos).
    private int vectorDatos[];
    private int numElementos;
    //posteriormente se implementa el método constructor de la clase para asignar los valores
    //iniciales de los atributos.
    public MetodosBusqueda(){
        vectorDatos = null;
        numElementos = 0;
    }
 //El siguiente método crea el vector en tiempo de ejecución, posteriormente se asignara su tamaño.
    public void crearVector(){
        vectorDatos = new int[numElementos];
    }
//Se implementa el método modificador para asignar los elementos en cada posición del vector.
    public void setVectorDatos(int p, int dato){
        vectorDatos[p] = dato;
    }
//Se implementa el método que permitirá obtener los elementos del vector.
    public int getVectorDatos(int p){
        return vectorDatos[p];
    }
//Método modificador para asignar el tamaño o número de elementos del vector.
    public void setNumElementos(int num){
        numElementos = num;
    }
//Método selector que permite obtener el tamaño o número de elementos que tiene el vector.
    public int getNumElementos(){
        return numElementos;
    }
//Implementación del método que busca lineal o secuencialmente los datos en el vector. Este
//método recorre el vector desde el primer elemento hasta el último, comparando cada elemento
//del vector con el valor buscado hasta que se encuentre el elemento o se llegue al final del vector.
//Este método es recomendado para realizar búsquedas con pocos datos.
    public int busquedaLineal(int dato){
        int i; //Variable para controlar el ciclo while.
        int posicion; //Variable que devuelve la posición en la que se encuentra el elemento buscado.
//Se Asigna el valor de -1 a la variable posición para devolver este valor en el supuesto caso de
//que no se encuentre el valor buscado dentro del vector.
        posicion = -1;
        i = 0;
//Mientras que no se llegue al final del vector y no se haya encontrado el dato buscado en el vector.
        while ((i <= getNumElementos()-1) && (posicion == -1)){
//Si el contenido del vector en la posición i-esima es igual al dato que se está buscando, entonces
//el dato si está en el vector y se devuelve la posición en donde se encuentra el dato dentro del
//vector. Sino, el dato no se encuentra en esa posición, y se incrementa la posición (i) para una
//nueva comparación.
            if (getVectorDatos(i) == dato){
```

```
            posicion=i;
         }else{
            i=i+1;
         }
      }
      return posicion;
   }

//Para poder implementar el método de búsqueda binaria, se requiere ordenar previamente el
//vector. Por esta razón se implementan los próximos dos métodos (cambiar y ordenarIntercambio).
   public void cambiar(int p1, int p2){
      int temp;
      temp = getVectorDatos(p1);
      setVectorDatos(p1, getVectorDatos(p2));
      setVectorDatos(p2, temp);
   }
   public void ordenarIntercambio(){
      int i, j;
      for (i=0; i<=getNumElementos()-1; i++){
         for (j=i+1; j<=getNumElementos()-1; j++){
            if (getVectorDatos(i) > getVectorDatos(j)){
               cambiar(i, j);
            }
         }
      }
   }

//Implementación del método de búsqueda binaria.
   public int bsuquedaBinaria(int dato){
      int posicion, izq, der, centro; //Estas 4 variables almacenan posiciones del vector.
      ordenarIntercambio(); //Para realizar la búsqueda el vector debe estar ordenado.
      izq = 0; //Primera posición del vector
      der = getNumElementos()-1; //Ultima posición del vector.
//Se asigna el valor de -1 a la variable posición para devolver este valor en el supuesto caso de que
//no se encuentre dato buscado dentro del vector.
      posicion = -1;
//Mientras que no se llegue al final del vector y no se haya encontrado el dato buscado en el vector.
      while ((izq <= der) && (posicion == -1)){
//Se busca cual es la posición del dato que se encuentra en el centro del vector.
         centro = (izq + der) / 2;
//Si el dato buscado es igual a lo que tiene el vector en la posición del centro entonces ya se
//encontró el elemento buscado y se devuelve la posición en donde se encontró, pero sino,
//entonces se determina si el elemento está a la izquierda o a la derecha del vector. Y se procede a
//buscar el dato hacia el inicio del vector (izquierda) o hacia el final del vector (derecha), esto si el
//dato buscado es mayor o menor al elemento que se encuentra en el centro del vector.
         if (dato == (getVectorDatos(centro)) ){
            posicion = centro;
            }else{
               if (dato < (getVectorDatos(centro))){
                  der = centro-1;
               }else{
                  izq = centro+1;
               }
            }
      }
      return posicion;
```

```
    }
}
```

- **Implementación de la clase Main en el fichero Main.java:**

En esta clase se almacena la información en el arreglo, se implementa el método que muestra los elementos de vector; y se implementa un menú de opciones para buscar en el arreglo según el método de búsqueda seleccionado.

```
import javax.swing.JOptionPane;


public class Main {
//Implementación del método que muestra los elementos del vector. Los almacena en una variable
//de tipo cadena y los visualiza en una ventana con la ayuda de JOptionPane y el método
//showMessageDialog.
    public static void mostrarVector(MetodosBusqueda buscar){
        String datosVector = ""; //Se declara una variable de tipo cadena, que esta inicialmente vacía.
//Se implementa un ciclo para recorrer en vector y se van almacenado los elementos en la variable
//de tipo cadena de nombre datosVector.
      for(int i=0; i<=buscar.getNumElementos()-1; i++){
          datosVector = datosVector+String.valueOf("Posición "+i+": "+buscar.getVectorDatos(i)+"\n");
       }
//Después de recorrer el vector y de tener todos los datos almacenados en la variable datosVector, se
//visualiza la información con la ayuda de JOptionPane y el método showMessageDialog.
     JOptionPane.showMessageDialog(null, "============ ELEMENTOS DEL VECTOR
============"+"\n"+datosVector);
    }

//Implementación del método que despliega el menú de opciones para escoger por cual método
//buscar los elementos.
    public static int menu(){
        int opcion = 0;
        do{
            opcion = Integer.parseInt(JOptionPane.showInputDialog("====== SELECCIONE EL METODO DE
BUSQUEDA ====== \n"+
                "1. Búsqueda Lineal o Secuencial \n"+"2. Búsqueda Binaria \n"+"3. Salir"+"\n \n Seleccione una
opción del 1 al 2"));
        }while(opcion <= 0 || opcion > 3);
        return opcion;
    }

    public static void main(String[] args) {
//Dentro del método estático main se crea un objeto de la case MetodosBusqueda, de nombre
//buscar para acceder a los métodos de búsqueda que se implementaron.
        MetodosBusqueda buscar = new MetodosBusqueda();
//Se pide en un showInputDialog el número de elementos que tendrá el vector.
        int numeroElementos = Integer.parseInt(JOptionPane.showInputDialog(null, "Digite el Número de
Elementos del Vector:"));
        //Se pasa el dato capturado al respectivo método modificador.
        buscar.setNumElementos(numeroElementos);
        //Se llama al método que crea el vector en tiempo de ejecución.
        buscar.crearVector();
```

```
//En el siguiente ciclo for, se piden los datos que se van a almacenar en el vector y se pasan al
//respectivo método modificador del vector (setVectorDatos).
      for(int i=0; i<=buscar.getNumElementos()-1; i++){
         int dato = Integer.parseInt(JOptionPane.showInputDialog(null, "Digitar Elemento de la Posición
"+i+": "));
         buscar.setVectorDatos(i, dato); //Se asigna en cada posición del vector el dato digitado.
      }
      JOptionPane.showMessageDialog(null,"Vector Lleno.... ");
      mostrarVector(buscar); //Se muestran los elementos del vector llamando al respectivo método.
      int datoBuscar; //Se almacena el dato que se quiere buscar en la variable datoBuscar.
      int posicionDato; //Variable para almacena la posición en donde se encuentra el dato buscado.

//Se implementa un do- while, que llama constantemente al menú de opciones y visualiza las
//diferentes opciones. Además, se implementa el código correspondiente a cada opción del menú
//según sea el caso.
      int opcion;
      do{
         opcion = menu();
         switch(opcion) {
         case 1:
            datoBuscar = Integer.parseInt(JOptionPane.showInputDialog(null, "Digite el dato que desea
buscar:"));
            posicionDato = buscar.busquedaLineal(datoBuscar);
            if(posicionDato != -1){
               JOptionPane.showMessageDialog(null, "El elemento
"+buscar.getVectorDatos(posicionDato)+
                     " se encuentra en la posicion "+posicionDato+" del vector");
            }else{
               JOptionPane.showMessageDialog(null, "El Elemento No se encuentra en el vector");
            }
            break;
         case 2:
            datoBuscar = Integer.parseInt(JOptionPane.showInputDialog(null, "Digite el dato que desea
buscar:"));
            posicionDato = buscar.bsuquedaBinaria(datoBuscar);
            if(posicionDato != -1){
               JOptionPane.showMessageDialog(null, "El elemento
"+buscar.getVectorDatos(posicionDato)+
                     " se encuentra en la posición "+posicionDato+" del vector");
            }else{
               JOptionPane.showMessageDialog(null, "El Elemento No se encuentra en el vector");
            }
            break;
         case 3:
            break;
         }
      }while(opcion != 3);
   }
}
```

EJERCICIOS/ACTIVIDADES

El coordinador del programa de Ingeniería de Sistemas, quiere tener un registro con la información de cada una de las asignaturas que se dictan en el programa, entre los datos relevantes que se quieren almacenar de cada asignatura están su código, nombre, semestre en donde se dicta y el número de créditos.

Implementar las clases necesarias en Java, que permitan registrar la información de cada materia en un vector de objetos (recomendado) o vectores en paralelo, luego de esto la aplicación permitirá al coordinador realizar operaciones sobre los datos.

Buscar la información de una asignatura, pasando como dato de búsqueda el código, para esto tienen que utilizar el método de búsqueda binaria, ordenando el vector con el método por intercambio.

Buscar el número de créditos correspondiente a una materia pasando como parámetro de búsqueda el nombre de la asignatura, pueden usar la búsqueda secuencial o binaria.

BIBLIOGRAFÍA

- Zahonero, I., y Joyanes Aguilar, L. (1999). Estructura de Datos - Algoritmos, Abstracción y Objetos. España: McGraw-Hill.

- Allen Weiss, M. (2004). Estructuras de datos en Java. España: Addison Wesley - Pearson. 776 pp.

- Guevara, P. Olascoaga, L. (n.d.). Métodos de Búsqueda. Universidad de Córdoba, Departamento de Ingeniería de Sistemas Telecomunicaciones. Disponible en: http://docplayer.es/1061327-Metodos-de-busqueda-pedro-guevara-salgado-luis-olascoaga.html