UNIVERSIDAD DE CÓRDOBA
comprometida con el desarrollo regional

CENTRO DE INNOVACION EN TIC PARA APOYO A LA ACADEMIA
CINTIA

MÉTODOS DE ORDENAMIENTO

DESARROLLO

CONCEPTO MÉTODOS DE ORDENAMIENTO:

Muchas veces es necesario además de buscar elementos dentro de en un vector, ordenarlos. Este ordenamiento puede ser de mayor a menor, si se está manejando números o en orden alfabético, si se trata de nombres o caracteres alfanuméricos.

Existe una gran variedad de métodos de ordenamiento que permiten organizar con rapidez los elementos que se encuentran dentro de un vector o estructura de datos. La elección de un determinado método de ordenamiento depende del tamaño del vector que se desea ordenar.

Entre los métodos de ordenamiento más populares y utilizados se encuentran: Burbuja, Intercambio, selección, shell, quick-sort, etc.

MÉTODOS DE ORDENAMIENTO UTILIZADOS:

- **Método de ordenamiento por Burbuja:**

Este método consiste en comparar los elementos del arreglo que se encuentran en posiciones adyacentes, empezando desde la primera posición y llegando hasta el final del arreglo. La comparación dependerá si están ordenando el arreglo en forma ascendente o descendente. Para el caso de ordenamiento ascendente (de menor a mayor) se compara cada elemento con el que le sigue inmediatamente, usando el operador mayor que (>).

Si el elemento de la posición de la izquierda es mayor que el elemento de la posición que le sigue a su derecha, entonces los dos elementos están mal ubicados y se procede a intercambiarlos de posición.

El proceso empieza entonces comparando los elementos de la posiciones 0 y 1 del vector, haciendo el intercambio si es preciso. Luego se comparan los elementos de las posiciones 1 y 2, y se procede de igual manera.

Seguidamente se comparan los elementos de las posiciones restantes del vector y se realiza el intercambio si es necesario. Se continua así hasta comparar los elementos de las últimas dos posiciones. Al llegar a este punto resulta que el valor mayor del arreglo quedara ubicado en la última posición del mismo, es decir estará en su posición correcta.

El proceso se repite de nuevo iniciando otra vez desde la posición 0 del vector, pero teniendo en cuenta que como ya se acomodó en la posición correcta el mayor elemento, la última comparación

que se hará; será entre los elementos de las posiciones penúltimas y antepenúltima. Es decir, que reducimos en uno el número de comparaciones hechas para acomodar el elemento.

A continuación se muestra la implementación en java de un método auxiliar para la implementación del método que intercambia los elementos de dos posiciones del vector. Este método se utilizara para mostrar la implementación de los otros dos métodos de ordenamiento, que se explicaran más adelante:

```
public void cambiar(int p1, int p2){
        int temp;
        temp = getVectorDatos(p1);
        setVectorDatos(p1, getVectorDatos(p2));
        setVectorDatos(p2, temp);
}
```

Implementación del método de ordenamiento Burbuja, este método compara elementos de la forma j, j+1. Es decir compara dos posiciones seguidas o continuas del vector:

```
    public void ordenarBurbuja(){
//Variables que controlan los ciclos que recorren el vector para comparar e intercambiar elementos.
        int i, j;
        for (i=0; i<=getNumElementos()-1; i++){
            for (j=0; j<=(getNumElementos()-i)-2; j++){
//Ordena de menor a mayor, si se quiere hacer de mayor a menor simplemente cambie el signo,
//aquí se compara si el elemento de la izquierda (j) es mayor que el elemento contiguo de la
//derecha (j+1). Si se cumple la condición los intercambia.
                if (getVectorDatos(j) > getVectorDatos(j+1)){
                    cambiar(j, j+1); //Se llama al método cambiar para el intercambio de los elementos.
                }
            }
        }
    }
```

- **Método de Ordenamiento Por Intercambio:**

El método de intercambio es uno de los más sencillos de comprender e implementar, pero así mismo, es uno de los menos eficientes. El trabajo que hace este método para ordenar un arreglo consiste en comparar cada elemento del arreglo con todos los que le siguen, es decir con aquellos que ocupan posiciones mayores que él, o sea los que están a su derecha.

Si los elementos que se comparan no están en el orden adecuado entonces se procede a intercambiarlos de posición. De esta manera el método empieza comparando el primer elemento del vector con todos los que le siguen, desde la posición uno hasta la última posición valida dentro del arreglo, haciendo intercambios en cada comparación cuando los elementos comparados no estén en el orden adecuado.

Al final de todas las comparaciones hechas con el primer elemento resulta que queda un valor acomodado en la primera posición del arreglo, que será el valor más pequeño, si ordenan de menor a mayor, o el valor más grande del arreglo si estamos ordenando de mayor a menor. Teniendo en cuenta esto se procede a comparar el segundo elemento del arreglo con todos los que le siguen, desde la posición del tercer elemento hasta el último.

El proceso se repite comparando los elementos siguientes con todos los que están a su derecha, hasta que en la última pasada solo se comparan los elementos de la penúltima y de la última posición del arreglo, momento en que ya queda ordenado el arreglo.

Implementación del método de ordenamiento por intercambio. Este método compara un elemento de la posición i con todos los que le siguen, es decir con los i+1, i+2, i+3:

```
   public void ordenarIntercambio(){
//Variables que controlan los ciclos que recorren el vector para comparar e intercambiar los
//elementos (i y j).
      int i, j;
      for (i=0; i<=getNumElementos()-1; i++){
         for (j=i+1; j<=getNumElementos()-1; j++){
//Se ordena de menor a mayor, si se quiere hacer de mayor a menor simplemente cambie el signo,
//aquí compara si el elemento de la posición (i) a la izquierda es mayor que el elemento de la
//posición (j) ubicado hacia la derecha.
            if (getVectorDatos(i) > getVectorDatos(j)){
               cambiar(i, j);
            }
         }
      }
   }
```

- **Método de ordenamiento por selección:**

Este método busca el elemento de menor valor dentro del vector y lo coloca en la primera posición, luego busca el segundo elemento más pequeño y procede a colocarlo en la segunda posición del vector, esto se repite hasta que se ordenan todos los elementos del vector.

Para la implementación de este método primero se busca la posición del elemento más pequeño del vector y se realiza el intercambio con el primer elemento del vector. Luego se buscan los elementos restantes hasta que solo quede el mayor.

Implementación del método auxiliar para buscar la posición del elemento más pequeño del vector y realizar el intercambio:

```
   public int posicionMenor(int inicio){
      int i; //Variable que controla el ciclo que recorre el vector para obtener el elemento menor.
      int posMenor; //Variable para establecer la posición en el vector del elemento menor.
      int menorElemento; //Variable para representar el elemento menor del vector.
      posMenor = inicio;
//A continuación se asume que el elemento más pequeño del vector está en la primera posición.
      menorElemento = getVectorDatos(inicio);
```

```
        for (i=inicio+1; i<=getNumElementos()-1; i++){
//Se compara si el elemento de la posición i es menor que el elemento menor.
            if (getVectorDatos(i) < menorElemento){
                menorElemento = getVectorDatos(i);
                posMenor = i;
            }
        }
//Se devuelve la posición en donde está el elemento menor del vector para realizar el intercambio
//hacia la primera posición.
        return posMenor;
    }
```

Cada vez que se obtiene la posición del menor elemento del vector, se implementa el método que intercambia dicho elemento a la primera posición del vector y así sucesivamente:

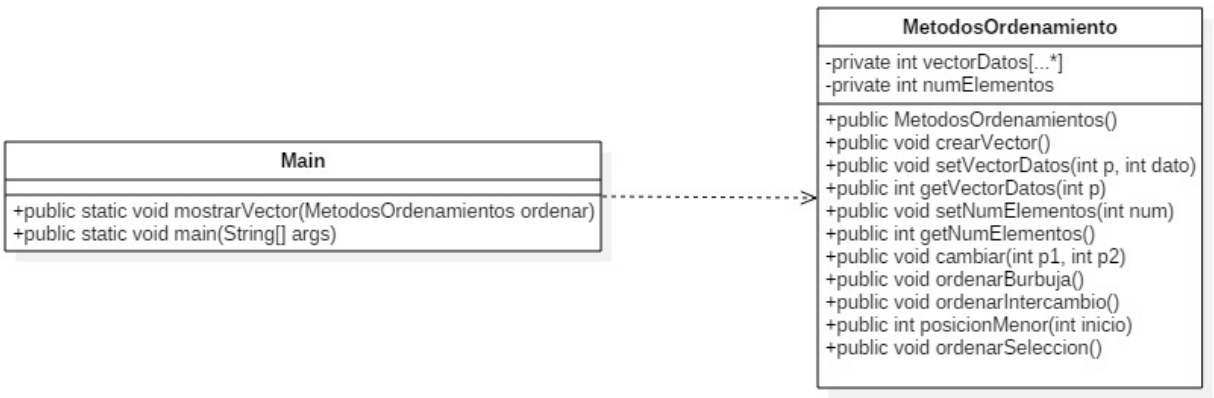
```
//Implementación del método que ordena el vector, llamando al método auxiliar posicionMenor para
//realizar el intercambio.
    public void ordenarSeleccion(){
        for (int i=0; i<=getNumElementos()-1; i++){
            cambiar(i, posicionMenor(i));
        }
    }
```

IMPLEMENTACIÓN DE LOS MÉTODOS DE ORDENAMIENTO EN JAVA:

- **Ejercicio propuesto implementado en Java**

Para entender el concepto de los métodos de ordenamiento, a continuación se implementa un ejercicio, que permite almacenar números enteros en un vector, y posteriormente se implementan los métodos descritos anteriormente, para ordenar los elementos almacenados.

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



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

En esta clase se implementan los siguientes métodos: el método que permite cambiar los elementos en dos posiciones en el arreglo; y los métodos de ordenamiento burbuja, intercambio y selección.

```
package metodosordenamiento;
public class MetodosOrdenamientos {
//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 MetodosOrdenamientos(){
      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];
   }
//Implementación del método modificador para asignar los elementos en cada una de las
//posiciones 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];
   }
//Implementación del método modificador para asignar el tamaño 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 permite intercambiar los elementos de dos posiciones del vector.
   public void cambiar(int p1, int p2){
      int temp;
      temp = getVectorDatos(p1);
      setVectorDatos(p1, getVectorDatos(p2));
      setVectorDatos(p2, temp);
   }

//Implementación del método de ordenamiento Burbuja, este método compara elementos de la
//posición j con la posición j+1, es decir compara dos posiciones seguidas o continuas del vector.
   public void ordenarBurbuja(){
//Variables que controlan los ciclos que recorren el vector para comparar e intercambiar los
//elementos.
      int i, j;
      for (i=0; i<=getNumElementos()-1; i++){
```

```
            for (j=0; j<=(getNumElementos()-i)-2; j++){
//Ordeno de menor a mayor, si se quiere hacer de mayor a menor simplemente cambie el signo,
//aquí se compara si el elemento de la izquierda (j) es mayor que el elemento contiguo de la
//derecha (j+1). Si se cumple la condición los intercambia.
                if (getVectorDatos(j) > getVectorDatos(j+1)){
                    cambiar(j, j+1);
//Para hacer el intercambio en los elementos se llama al método cambiar.
                }
            }
        }
    }


//Implementación del método de ordenamiento por intercambio. Este método compara un elemento
//de la posición i con todos los que le siguen, es decir con los de la posición i+1, i+2, i+3...
    public void ordenarIntercambio(){
//Variables que controlan los ciclos que recorren el vector para comparar e intercambiar los
//elementos (i y j).
        int i, j;
        for (i=0; i<=getNumElementos()-1; i++){
            for (j=i+1; j<=getNumElementos()-1; j++){
//Ordeno de menor a mayor, si se quiere hacer de mayor a menor simplemente cambie el signo,
//aquí compara si el elemento de la posición (i) a la izquierda es mayor que el elemento de la
//posición (j) ubicado hacia la derecha.
                if (getVectorDatos(i) > getVectorDatos(j)){
                    cambiar(i, j);
                }
            }
        }
    }

//Implementación del método auxiliar para buscar la posición del elemento más pequeño del vector
//y realizar el intercambio.
    public int posicionMenor(int inicio){
        int i; //Variable que controla el ciclo que recorre el vector para obtener el elemento menor.
        int posMenor; //Variable para establecer la posición en el vector del elemento menor.
        int menorElemento; //Variable para representar el elemento menor del vector.
        posMenor = inicio;
//A continuación se asume que el elemento más pequeño del vector está en la primera posición.
        menorElemento = getVectorDatos(inicio);
        for (i=inicio+1; i<=getNumElementos()-1; i++){
//Se compara si el elemento de la posición i es menor que el elemento menor.
            if (getVectorDatos(i) < menorElemento){
                menorElemento = getVectorDatos(i);
                posMenor = i;
            }
        }
//Se devuelve la posición en donde está el elemento menor del vector para realizar el intercambio
//hacia la primera posición.
        return posMenor;
    }

//Implementación del método que ordena el vector, llamando al método auxiliar posicionMenor para
//realizar el intercambio.
    public void ordenarSeleccion(){
        for (int i=0; i<=getNumElementos()-1; i++){
```

```
            cambiar(i, posicionMenor(i));
        }
    }
}
```

- **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 ordenar el arreglo según el método de ordenamiento seleccionado.

```
package metodosordenamiento;
//import javax.swing.JOptionPane;
import javax.swing.*;
/**
 *
 * @author DELL
 */
public class Main {
    /**
     */
//Se crea un método estático que permita mostrar los elementos del vector, dentro del método
//estático main.
        public static void mostrarVector(MetodosOrdenamientos ordenar){
            String datosVector = ""; //Se declara una variable de tipo cadena para almacenar los datos.
//En el siguiente ciclo se recorre el vector y se van almacenando los elementos en la variable
//anterior (datosVector).
             for(int i=0; i<=ordenar.getNumElementos()-1; i++){
                datosVector = datosVector+String.valueOf("Posicion "+i+": "+ordenar.getVectorDatos(i)+"\n");
            }
//Después de tener almacenados todos los datos del vector, en la variable de tipo cadena
//(datosVector), se procede a visualizar los elementos con la ayuda de JOptionPane y el método
//showMessageDialog.
            JOptionPane.showMessageDialog(null, "============ ELEMENTOS DEL VECTOR
============"+"\n"+datosVector);
        }

    public static void main(String[] args) {
//Se crea un objeto de la clase MetodosOrdenamientos llamado ordenar para acceder a los
//métodos púbicos implementados en la clase.
        MetodosOrdenamientos ordenar = new MetodosOrdenamientos();
//Las variables para capturar el número de elementos, controlar el ciclo for y capturar los datos del
//vector, son: numeroElementos, i y dato, respectivamente.

//Se captura en un showInputDialog el número entero correspondiente a la cantidad de elementos
//del vector. Como el dato que se captura está en un tipo de dato String, se debe pasar a un tipo de
//dato entero con la ayuda de Integer.parseInt.
        int numeroElementos = Integer.parseInt(JOptionPane.showInputDialog(null, "Digite el Número de
Elementos del Vector:"));
//Se pasa el dato capturado al respectivo método modificador.
        ordenar.setNumElementos(numeroElementos);
//Se llama al método que crea el vector en tiempo de ejecución.
        ordenar.crearVector();
```

```
      for(int i=0; i<=ordenar.getNumElementos()-1; i++){
         int dato = Integer.parseInt(JOptionPane.showInputDialog(null, "Digitar Elemento de la Posición "+i+": "));
         ordenar.setVectorDatos(i, dato);
      }
      JOptionPane.showMessageDialog(null,"Vector Lleno.... ");
      mostrarVector(ordenar); //Se llama al método para visualizar los elementos del vector.


//Se implementa un menú de opciones para escoger con que método se quiere ordenar el vector.
int opcion = Integer.parseInt(JOptionPane.showInputDialog("====== Seleccione el Método de Ordenamiento ====== \n"+
            "1. Ordenamiento Burbuja \n"+"2. Ordenamiento por Intercambio \n"+"3. Ordenamiento por Seleccion \n"));
      switch(opcion) {
         case 1:
            ordenar.ordenarBurbuja();
            mostrarVector(ordenar);
            break;
         case 2:
            ordenar.ordenarIntercambio();
            mostrarVector(ordenar);
            break;
         case 3:
            ordenar.ordenarSeleccion();
            mostrarVector(ordenar);
            break;
         default:
            JOptionPane.showMessageDialog(null, "¡No selecciono una opción válida entre 1 y 3!");
            break;
      }
   }
}
```

EJERCICIOS/ACTIVIDADES

- Siguiendo el ejercicio anterior implementar los métodos Shell y QuickSort, utilizar el método cambiar en su implementación; además incluir dos nuevas opciones en el menú para ordenar por estos métodos.

- 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.

  La aplicación debe tener las siguientes opciones:

  - Generar un listado ordenando las asignaturas según el número de créditos de mayor a menor, para esto deben usar el método por selección.

  - Listado con la información de las asignaturas, ordenadas alfabéticamente, utilizar el método de ordenamiento Shell.

  - Generar un listado ordenando las asignaturas según su código de menor a mayor, para esto deben usar el método burbuja.

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 Ordenamiento. Universidad de Córdoba, Departamento de Ingeniería de Sistemas Telecomunicaciones. Disponible en: https://es.scribd.com/document/106328137/Metodos-ordenamiento