Programación Básica 1 (JAVA)

Apunte de Teoría

Arrays- Vectores y Matrices en Java

Docentes:

BORGEAT, Andrés
MONTEAGUDO, Juan Manuel
PARDO, Sebastián
GOITEA, Alejandro

UNIVERSIDAD NACIONAL DE LA MATANZA

Contenido

Arrays Unidimensionales - Vectores

Definición

Los vectores son estructuras que nos permiten agrupar valores numéricos ordenados linealmente, desde el concepto gráfico decimos que tienen dirección, sentido y módulo.

Repasando el concepto

Geométricamente un vector es un segmento orientado que posee dirección y sentido, cuya representación en componentes podría ser:

v = (v1, v2, v3, v4, …, vn)

El uso de esta array unidimensional o vector o matriz degenerada (1 columna o 1 fila) es muy común muchos lenguajes, normalmente se lo denomina array.

En el caso de Java lo más apropiado es decir que es una colección de variables del mismo tipo.

Por el modo en el que están organizados los datos es que nos permite manipularlos fácilmente.

Otra ventaja importante es que en Java los arrays se utilizan como objetos, esto potencia la implementación de este tipo de estructura.

Pero mientras un arreglo es de cierto tamaño dado (manejo estático), un objeto de tipo Vector puede dinámicamente crecer y decrecer conforme se vaya necesitando.

La clase Vector se conforma como parte del paquete java.util de la librería estándar de clases de Java.

Nos permite manipular el mismo en forma similar a un arreglo, ya que se pueden almacenar, buscar, ordenar y acceder a valores y referencias a través de un índice.

Una de las maneras de implementar el vector es usando la declaración:

```
tipoDeDatoPrimitivo nombreVector [];
```

o

```
tipoDeDatoPrimitivo [] nombreVector;
```

Los corchetes indican que estamos en presencia de un array o vector.

Este tipo de declaración es la que se va a utilizar cuando queremos trabajar con los tipos primitivos del lenguaje (int, float, long, etc).

Si ahora lo que se desea hacer es crear una instancia de esta declaración lo que debemos hacer es escribir la siguiente sentencia:

```
nombreVector = new tipoDeDatoPrimitivo [TAMAÑO];
```

- tipoDeDatoPrimitivo: con él se especifica el tipo de dato que se va a usar.
- TAMAÑO: se le está dando la cantidad de elementos de ese tipo.
- nombreVector: definimos el nombre de la variable que va a ser del tipo array.

Por lo tanto, para generar una instancia del mismo usamos la palabra reservada **new** para asignar un array, debe especificar el tipo y la cantidad de elementos a asignar.

A diferencia de un array (arreglo), un objeto Vector no está declarado para ser de un tipo particular.

Un elemento puede insertarse y eliminarse de una posición específica a través de la invocación de un sólo método.

Un objeto de tipo Vector maneja una lista de referencias a la clase Object, así no pueden almacenarse tipos de datos primitivos.

Para usar la clase Vector tenemos que poner al principio del archivo del código fuente la siguiente sentencia import:

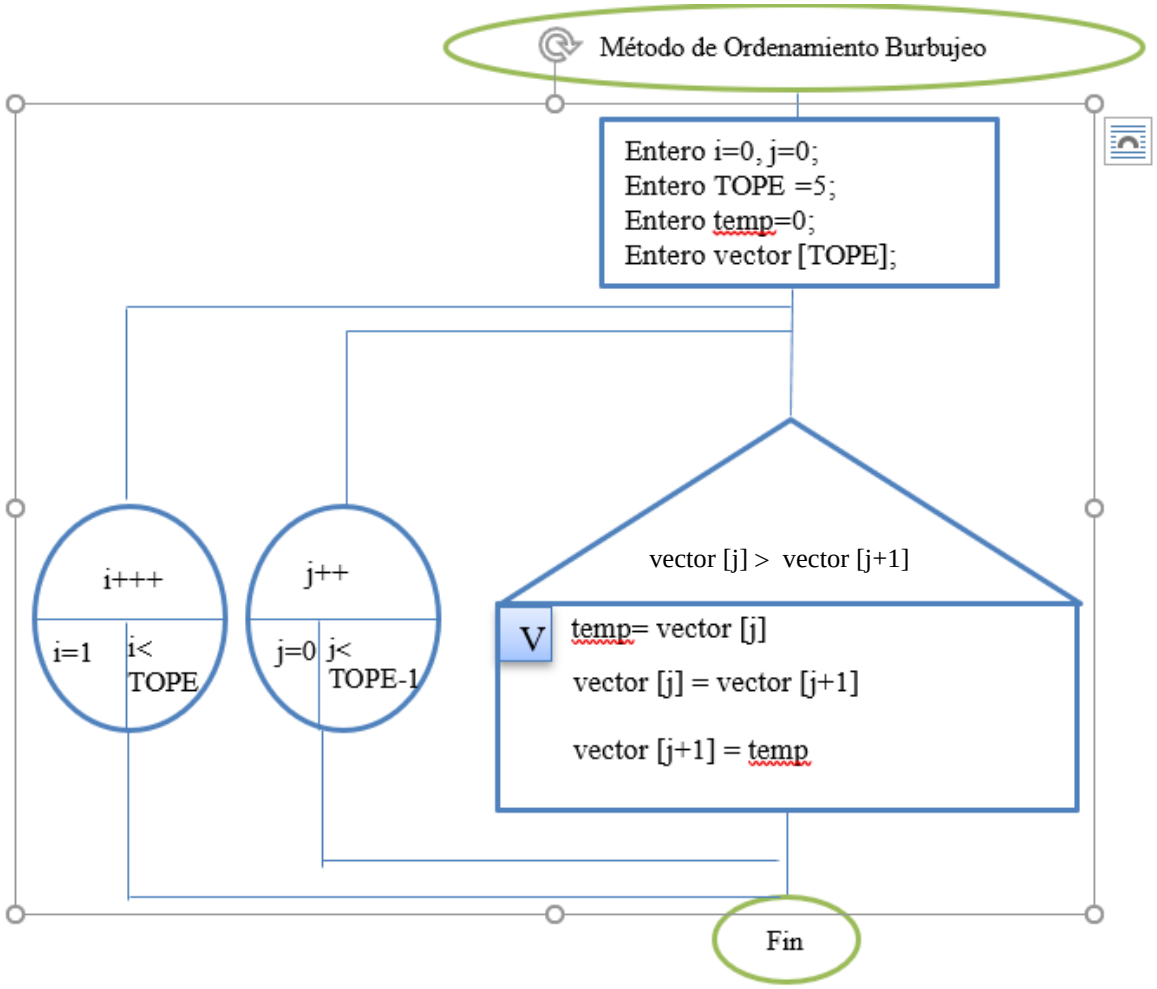
•import java.util.Vector;

No vamos a utilizar esta clase en el presente curso.

Ejemplo de vector estático en Java

```
public class VectorComun {

	public static void main(String[] args) {

			final int TOPE = 5;
			int i=0;
				// declara un vector de tipo primitivo enteros.
			 int[] vectorEstatico;
				// El tamaño del vector va a ser de 5
			      vectorEstatico = new int[TOPE];
				// Inicializamos el vector con 0 en cada posición
			for (i = 0; i < vectorEstatico.length; i++) {
			   vectorEstatico[i] = 0;
			}
				// Almacenar un dato en la primera posición del vector
			vectorEstatico[0] = 11;

				// Asignamos otro elemento a la segunda posición
			vectorEstatico[1] = 8;

				// Lo hacemos con todas las posiciciones
			vectorEstatico[2] = 33;
			vectorEstatico[3] = 14;
			vectorEstatico[4] = 5;

			 // Mostrando los elementos del array
			 for (i = 0; i < vectorEstatico.length; i++) {
	System.out.println("Elemento en el índice " + (i+1) + " : "+ vectorEstatico[i]);
			 }
	}
}
```

Ordenamiento de un vector por el método de la burbuja mejorada



Uno de los métodos más usado para el ordenamiento de vectores o arrays es el denominado de la burbuja. Este método no es el mejor, pero debido a su simplicidad y facilidad de entendimiento es muy usado para realizar esta operación.

La versión que usamos es la mejorada, esto quiere decir que la cantidad de iteraciones que hacen se redujeron, quitando aquellas que no son necesarias.

El método va recorriendo el vector y compara dos elementos del mismo, uno en la posición j y el otro en la posición siguiente, o sea j+1, de esta manera si el elemento en la posición j es más grande que el elemento en la posición j+1 se ingresa a la rutina de intercambio donde se usa una variable auxiliar para intercambiar la posición de los elementos.

Tomemos los elementos usados en el ejemplo anterior para realizar la prueba de escritorio.

```
vectorEstatico[0] = 11;
vectorEstatico[1] = 8;
vectorEstatico[2] = 33;
vectorEstatico[3] = 14;
vectorEstatico[4] = 5;
```

Prueba de escritorio usando el método de la burbuja y el vector anterior

	Posición en el vector											
	v[0]	v[1]	v[2]	v[3]	v[4]		contador i	contador j	vector[j]		vector[j+1]	temp
vector original	11	8	33	14	5		1	0	11	>	8	
	8	11	33	14	5				8		11	11
								1	11	>	33	
								2	33		14	33
	8	11	14	33	5				14	>	33	
								3	33	>	5	33
	8	11	14	5	33				5		33	
							2	0	8	>	11	
								1	11	>	14	
								2	14	>	5	
	8	11	5	14	33				5		14	14
								3	14	>	33	
							3	0	8	>	11	
								1	11	>	5	
	8	5	11	14	33				5		11	11
								2	11	>	14	
								3	14	>	33	
							4	0	8	>	5	
vector ordenado	5	8	11	14	33				5		8	8
								1	8	>	11	
								2	11	>	14	
								3	14	>	33	

La prueba de escritorio muestra como se va produciendo el intercambio a medida que se recorre el vector por medio del algoritmo.

A continuación, el código del diagrama de ordenamiento:

```
public class OrdenamientoVector {

	public static void main(String[] args) {

	final int TOPE = 5;
		int i=0;
		int j=0;
		int temp=0;
		int vectorEstatico [];

		vectorEstatico = new int[TOPE];
		vectorEstatico[0] = 11;
		vectorEstatico[1] = 8;
		vectorEstatico[2] = 33;
		vectorEstatico[3] = 14;
		vectorEstatico[4] = 5;

	for (i = 1; i < TOPE; i++) {

		for (j = 0; j < TOPE -1; j++) {
```

```
                    if (vectorEstatico[j] > vectorEstatico[j+1]){
                          temp=vectorEstatico[j];
                          vectorEstatico[j] = vectorEstatico[j+1];
                          vectorEstatico[j+1] = temp;
                          }
                    }
              }
      for (i = 0; i < vectorEstatico.length; i++) {
      System.out.println("El vector ordenado en " + (i+1) + " : "+ vectorEstatico[i]);
                    }
      }
}
```

Búsqueda del mayor en un vector

Para encontrar el valor almacenado más grande dentro de un vector existen dos estrategias a plantear :

- lo que hacemos es almacenar en una variable auxiliar un valor muy pequeño que al empezar a recorrer el vector se cambie por un elemento más grande
- asignamos como el valor más grande al elemento que se encuentra en la primera posición del vector, para luego recorrer el resto del array y encontrar el valor más grande,

Como sería el código necesario para resolver esta búsqueda:

```
      final int TOPE = 5;
            int i=0;
            int numeroMayor=0;
           int vectorEstatico [];
            vectorEstatico = new int[TOPE];
            vectorEstatico[0] = 11;
            vectorEstatico[1] = 8;
            vectorEstatico[2] = 33;
            vectorEstatico[3] = 14;
            vectorEstatico[4] = 5;
            numeroMayor = -1;  //no aparece en la otra versión
    for (i = 0; i < TOPE; i++) {
      if (vectorEstatico[i] > numeroMayor ){
            numeroMayor = vectorEstatico[i];
      }
}
```

La otra forma de iniciar la búsqueda sería:

```
   for (i = 0; i < TOPE; i++) {
      if (vectorEstatico[i] > numeroMayor ){
            if (i==0){
            numeroMayor = vectorEstatico[i];
            }
            numeroMayor = vectorEstatico[i];
      }
```

Búsqueda del menor en un vector

De manera similar podemos realizar la búsqueda de el menor elemento dentro de un vector.
Podemos usar las dos versiones anteriores, lo único que debemos tener en cuenta que al darle un valor por defecto ese número debe ser lo suficientemente grande como para que sea reemplazado por los que se encuentran almacenados en el vector.
A continuación, el código que resuelve esta búsqueda:

```
        final int TOPE = 5;
              int i=0;
              int numeroMenor=0;
             int vectorEstatico [];
              vectorEstatico = new int[TOPE];
              vectorEstatico[0] = 11;
              vectorEstatico[1] = 8;
              vectorEstatico[2] = 33;
              vectorEstatico[3] = 14;
              vectorEstatico[4] = 5;

      for (i = 0; i < TOPE; i++) {
        if (vectorEstatico[i] < numeroMenor){
              if (i==0){
              numeroMenor = vectorEstatico[i];
              }
        if (vectorEstatico[i] < numeroMenor
              numeroMenor = vectorEstatico[i];
        }
}
```

Búsqueda de un número dentro de un vector

Para realizar la búsqueda de un número dentro de un vector almacenado lo que cambia con respecto a las dos búsquedas anteriores es que el valor se ingresa por teclado o se coloca en una variable auxiliar, la que se va a usar para comparar con cada uno de los elementos del vector.
Veamos a continuación como queda el código:

```
        final int TOPE = 5;
              int i=0;
              int numeroABuscar=0;
              boolean valorEncontrado= false;
             int vectorEstatico [];
              vectorEstatico = new int[TOPE];
              vectorEstatico[0] = 11;
              vectorEstatico[1] = 8;
              vectorEstatico[2] = 33;
              vectorEstatico[3] = 14;
              vectorEstatico[4] = 5;
              numeroABuscar=33;
      for (i = 0; i < TOPE; i++) {
        if (vectorEstatico[i] ==numeroABuscar){
              valorEncontrado = true;
        }
}
```

Una variante de esta búsqueda sería no solamente encontrar el valor, en el caso que el mismo se repita dentro del vector habría que contabilizarlo por medio de un contador y almacenar en un nuevo vector las posiciones donde se lo encontró.

Arrays Multidimensionales - Matrices

Un array en Java puede tener más de una dimensión. El caso más general son los arrays bidimensionales también llamados matrices o tablas.
La dimensión de un array la determina el número de índices necesarios para acceder a sus elementos. Los vectores que hemos visto en otra entrada anterior son arrays unidimensionales porque solo utilizan un índice para acceder a cada elemento.
Una matriz necesita dos índices para acceder a sus elementos. Gráficamente podemos representar una matriz como una tabla de n filas y m columnas cuyos elementos son todos del mismo tipo.

Por ejemplo podemos crear la matriz nombres en la que almacenarnos en filas y columnas distintos Strings para luego recorrerla y mostrarlos

```
   int filas = 2, columnas = 2;
        String[][] nombres = new String[filas][columnas];
        nombres[0][0] = "Pedro";
        nombres[0][1] = "Juan";
        nombres[1][0] = "Alberto";
        nombres[1][1] = "Oscar";
        int i, j;

        for (i = 0; i < nombres.length; i++) {
            for (j = 0; j < nombres[i].length; j++) {
                System.out.print(nombres[i][j] +" ");
            }
            System.out.print("\n");
        }
```

También podemos realizar operaciones matemáticas como por ejemplo la suma de matrices

```
Integer filas = 2, columnas = 2;
        Integer[][] a = new Integer[filas][columnas];
        Integer[][] b = new Integer[filas][columnas];
        a[0][0] = 1;
        a[0][1] = 4;
        a[1][0] = 6;
        a[1][1] = 10;

        b[0][0] = 11;
        b[0][1] = 5;
        b[1][0] = 2;
```

```
        b[1][1] = 4;

        Integer i, j;

        System.out.println("Dada la matriz A");
        for (i = 0; i < a.length; i++) {
            for (j = 0; j < a[i].length; j++) {
                System.out.print(a[i][j] + " ");
            }
            System.out.print("\n");
        }
        System.out.print("\n");
        System.out.println("y la matriz B");
        for (i = 0; i < b.length; i++) {
            for (j = 0; j < b[i].length; j++) {
                System.out.print(b[i][j] + " ");
            }
            System.out.print("\n");
        }
        System.out.print("\n");
        System.out.println("La suma de A + B es");

        for (i = 0; i < a.length; i++) {
            for (j = 0; j < a[i].length; j++) {
                Integer r = a[i][j]+ b[i][j];
                System.out.print( r + " ");
            }
            System.out.print("\n");
        }
```