Juan Gabriel Gómez Vargas

Estructura de los vectores y las matrices en Java

A criterio personal, no se puede aprender correctamente el uso de las matrices sin haber previamente comprendido el manejo de los vectores, por lo tanto, se realizará una breve explicación de estos.
Los vectores básicamente son un contenedor de elementos que comparten un tipo en común (mismo), ejemplo: [int, int]. Estos proveen una gran ventaja a la hora de desarrollar software, ya que se pueden crear "n" instancias de un mismo tipo sin necesidad de definir un repositorio de variables cuando es innecesario hacerlo.
Un ejemplo sencillo de una declaración e instanciación de un vector es el siguiente:

tipo[] nombre = new tipo[tamaño];

Otra forma alternativa (y más rápida cuando se conocen los elementos que se van a incluir) es la que asigna los valores directamente al instanciarlo:

tipo[] nombre = { elemento, elemento, elemento };
donde cada elemento debe corresponder a un tipo en común, ejemplo:
String[] palabrasConA = { "abuela", "abeja", "adelante" };
Note que se hace uso de corchetes para establecer los elementos dentro de estos.
Un uso incorrecto sería:
String[] palabrasConA = { "abuela", "abeja", 15 };

Si se declarara (y asigna) manualmente el vector anterior sería:
String[] palabrasConA = new String[3];
palabrasConA[0] = "abuela";
palabrasConA[1] = "abeja";
palabrasConA[2] = "adelante";

Otros detalles que se deben conocer sobre los vectores son:

- El **índice** de un vector **inicia siempre en cero** y es de **tipo int**, ejemplo: vector[0] = "abuela".
- Se declara con el **símbolo []** (corchetes cuadrado) porque esto **especifica** que el elemento definido va a ser un vector.
- El tipo de elemento especificado en la declaración debe corresponder a la hora de instanciarlo, un **ejemplo incorrecto** es entonces: String[] vector = new int[10];.
- El **tamaño** se especifica para **definir cuántos elementos** debe incluir el vector a utilizar, ejemplo: new String[**3**]. En este caso el vector es de tamaño **3**.
- El **tamaño máximo** de un vector está dado por el **atributo length** (por ejemplo: palabrasConA.length). Debe diferenciarse entre el atributo length del vector
- La última posición del vector es nombreDelVector.length - 1 (es decir, que si se desea acceder al penúltimo, sería vector.length - 2 y así sucesivamente).

Podemos ver un vector como un contenedor de elementos que comparten un mismo tipo. Es decir, visualmente el vector palabrasConA tendría la siguiente forma:

posición	0	1	2
Posición inversa	length - 3	length - 2	length - 1
valor	"abuela"	"abeja"	"adelante"

Para el ejercicio anterior se define un vector de tres elementos, por lo que es equivalente decir:
palabrasConA[0] equivale a palabrasConA[palabrasConA.length - 3];
Ya que palabrasConA.length es de tamaño 3.
Es decir, 3 (length del vector) - 3 es 0.
Y finalmente palabrasConA[0] = palabrasConA[3-3];

Para el caso del índice 1 ocurre lo mismo:
palabrasConA[1] equivale a palabrasConA[palabrasConA.length - 2];
= palabrasConA[0] equivale a palabrasConA[3 (length) - 2];
Y así sucesivamente...

Para acceder a algún elemento del vector simplemente debemos definir el índice, es decir que se desea consultar:
vector[indice]

Por ejemplo:

palabrasConA[0];//el vector retorna el elemento en la primera posición (abuela).
String elemento1 = palabrasConA[1];//el vector retorna el elemento en la primera posición (abeja) y lo asigna a la variable elemento1.

Una de las cosas como programador que me interesa siempre saber es la ventaja que me provee utilizar uno u otro comando a la hora de escribir código y esto no es indiferente en el caso de los vectores. Una ventaja de usar vectores se encuentra a continuación:

Primero, explicamos la forma en la que **no se usan vectores**, como es el caso a siguiente:

```
char a = 'a';
char b = 'b';
char c = 'c';
...
...
char z = 'z';

public boolean contieneLetra(char letra) {
```

```
        return a == letra || b == letra || c == letra ..... .... ... || z == letra;
}
```

Como la situación anterior, necesitaríamos validar tantas variables como letras del abecedario tengamos, en ese caso habría que hacer 27 validaciones; esto se simplifica a la hora de implementar un vector, como lo muestra la contraparte del código explicado anteriormente:

```
char[] letrasAlfabeto = {'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'ñ', 'o', 'p', 'q', 'r', 's', 't',
'u', 'v', 'w', 'x', 'y', 'z'};

/**cuando se desea solo conocer el valor de cada elemento del vector sin importar su
posición se puede optar por usar for-each*/
public boolean contieneLetraEnVector(char letra) {
        for (char letraAlfabeto : letrasAlfabeto) {
                if (letraAlfabeto == letra) {
                        return true;
                }
        }
        return false;
}
```

Ambas porciones de código hacen exactamente lo mismo, pero una lo hace más adecuado, ordenado y entendible que el otro, ¿por qué? porque solo ocupamos hacer una validación (en este caso con for) para todas las variables que almacena el vector, sin saber el nombre o valor de cada una de ellas por separado (como la primera porción mostrada).

Cuando se debe acceder al índice del vector, es recomendado hacer uso de un for, para así tener control sobre el índice que queremos acceder. Como lo muestra el siguiente ejemplo:

```
public int retornarIndiceVector(char letra) {
      for (int i = 0; i < letrasAlfabeto.length; i++) {
         if(letrasAlfabeto[i] == letra) {
            return i;
         }
      }
      return -1;
   }
```

EN CONSTRUCCION...