Estructuras de datos

Pedro Ulises Cervantes González

Universidad Nacional Autónoma de México

confundeme@ciencias.unam.mx

Estructuras de datos

Cuando manejamos muchos datos, es necesario usar alguna estructura que nos provea herramientas para poder manipularlos. Dependiendo de nuestras necesidades será la herramienta que decidamos usar.

Una forma de clasificar estas estructuras es en **homogéneas** y **heterogéneas**. Las homogéneas permiten datos de un solo tipo, mientras que las heterogéneas permiten varios tipos al mismo tiempo.

Estructuras de datos

Otra clasificación es en **dinámicas** y **estáticas**. Las primeras son estructuras que ajustan su tamaño respecto a la cantidad de datos que poseen, mientras que las segundas siempre poseen el mismo tamaño, independientemente del número de datos que contengan.

Existen varias estructuras de datos distintas. En esta presentación se revisarán solamente dos: arreglos y listas.

Arreglos

Java provee una estructura de datos homogénea y estática llamada arreglo. Un arreglo es un objeto de tipo contenedor, cuyo tamaño es establecido cuando es creado.

La declaración de un arreglo es con la siguiente sintaxis:

```
<tipo>[] <nombre>;
<tipo> <nombre>[];
```

Ejemplo:

```
double[] reales;
int enteros[];
```

Arreglos

Existen dos formas de inicializar arreglos. La primera es como sigue:

```
<nombre> = new <tipo>[<cantidad de elementos>];
```

Ejemplo:

```
reales = new double[15];
```

En este caso el tamaño del arreglo es de 15.

Al declarar el arreglo de esta forma, todos los elementos tendrán el valor por defecto que le corresponde a su tipo, es decir, un arreglo de booleanos inicializa a todos en false, uno numérico los inicializa en 0 y uno de objetos los inicializa en *null*.

Arreglos

La segunda forma consiste en darle valores a los elementos al inicializar:

```
<nombre> = {<valores separados por coma>};
```

Ejemplo:

```
reales = {1, 3.14, -3.3, 15};
```

Con esta forma, el tamaño queda fijo como el número de elementos de la inicialización, en este caso 4.

Arreglos

Los arreglos tienen un atributo llamado *length* que indica el tamaño de estos.

La forma de acceder a los elementos del arreglo es usando entre corchetes un índice que puede ir desde 0 hasta el tamaño del arreglo menos 1.

Ejemplo:

```
System.out.println(reales.length);
reales[3]=100; //Actualiza el elemento 3
System.out.println(reales[0]); //Obtiene el elemento 0
```

Arreglos

Existen arreglos de más de una dimensión que son prácticamente arreglos de arreglos. En este caso la dimensión con mayor prioridad será la primera, de tal forma que el atributo *length* se referirá al tamaño de la primera dimensión.

Para inicializar de forma explícita cada elemento en uno multidimensional se deben de anidar llaves. Ejemplo:

```
boolean[][] varios={{true,false},{false,false}
                   ,{true,true,false,false}};
```

El número de elementos de una misma dimensión no necesariamente debe ser el mismo, ya que cada subarreglo es independiente de los otros.

Arreglos

También se pueden inicializar arreglos multidimensionales reservando en memoria el espacio de cada dimensión. Ejemplo:

```
char[][][] letras=new char [10][2][5];
```

Para obtener el valor de sus dimensiones es como sigue:

```
int x=letras.length; //10
int y=letras[0].length; //2
int z=letras[0][0].length; //5
```

No es necesario tomar el elemento 0 de una dimensión para saber acceder al valor de la siguiente dimensión. Basta con tomar un elemento que sepamos que existe.

Arreglos

Como cada subarreglo es independiente a los demás, cada uno puede tener distinto tamaño. Si esto ocurre, se dice que es *escalonado*.

Para crearlo, se reserva memoria comenzando con las primeras dimensiones. Ejemplo:

```
String[][] cadenas=new String [2][];
cadenas[0]=new String[3];
cadenas[1]=new String[7];
```

Entonces en las dimensiones obtendríamos distintos valores:

```
int x=cadenas.length; //2
int y=cadenas[0].length; //3
int z=cadenas[1].length; //7
```

Listas ligadas

Las listas ligadas son estructuras de datos homogénas y dinámicas. En ellas se almacenan elementos de tal modo que cada elemento tiene una referencia al siguiente elemento de la lista.

En las listas debe haber una referencia al primer elemento de la lista, lo que hace que sea sencillo hacer operaciones con el principio de la lista como insertar, eliminar o consultar elementos.

Si en la lista se harán muchas operaciones al final de esta, conviene también tener una referencia al último elemento de la lista.

Listas ligadas

Las listas manejan muchas referencias, por lo que hay que tener cuidado de actualizar todas cuando sea necesario (la cabeza, las referencias a los elementos siguientes, el último elemento si es que existe, etc).

Una lista *doblemente ligada* consiste en una lista en que cada elemento tiene dos referencias: una al siguiente elemento y una al elemento anterior.

Paquetes

Uno de los tipos de acceso para variables y métodos es el de paquete, que es el acceso por defecto en Java, por lo que no hay una palabra reservada para este tipo de acceso. Si no se quiere un acceso de paquete, entonces hay que escribir **public**, **private** o **protected**.

Un paquete es un conjunto de elementos relacionados que proveen protección de acceso y administración del espacio de nombres. Esos elementos pueden ser de distintos tipos, pero en principio solo consideraremos que son clases e interfaces.

Paquetes

Para crear un paquete, es necesario que cada elemento del paquete esté en un mismo directorio que tenga el nombre de ese paquete. Además de eso, la **primera instrucción** en cada archivo tiene que ser la siguiente:

```
package <nombre>;
```

Las convenciones del nombre del paquete son varias, pero por el momento consideraremos que son las mismas que para nombrar variables. Ejemplo:

```
package numeros;
```

Paquetes

Para usar elementos de un paquete externo, hay que importar esos elementos al principio de cada archivo como sigue:

```
import <nombre del paquete>.<elemento1>;
import <nombre del paquete>.<elemento2>;
...
import <nombre del paquete>.<elementoN>;
```

Ejemplo:

```
import java.util.Scanner;
import java.util.Random;
```

En el ejemplo se hacen importaciones de un solo paquete, pero pueden hacerse desde varios.

Paquetes

En lugar de importar elemento por elemento de un mismo paquete, se pueden importar todos los elementos de este como sigue:

```
import <nombre del paquete>.*;
```

Ejemplo:

```
import java.util.*;
```

Al hacer este tipo de importaciones **no** se importan los elementos en subpaquetes del paquete que se importó.

Referencias

Java Oracle (2015)

Arrays
https://docs.oracle.com/javase/tutorial/java/nutsandbolts/arrays.html
▸ Link

Packages
https://docs.oracle.com/javase/tutorial/java/package/index.html
▸ Link

Elisa Viso G. y Canek Peláez V. (2012)

Introducción a las Ciencias de la Computación con Java.
Segunda edición.

Carlos Gerardo Morales García (2016)

Material de apoyo para la enseñanza de programación con Java.