Taller de preparación para la OCI - PUC

Arreglos

1. Introducción

Los arreglos son fundamentales en todo lenguaje de programación. Hasta ahora, sólo se podía guardar un dato por variable pero un arreglo nos permite almacenar un conjunto de datos o elementos del mismo tipo bajo una misma variable para luego acceder a ellos.

En vez de declarar variables individualmente como edad0, edad1, edad2, etc. se puede declarar un arreglo *edades* el cual almacenará estas edades de forma ordenada.

Por ejemplo, el arreglo a continuación tiene nueve números:

10	1	-3	13	4	-6	20	17	10

Esta es una colección de números enteros. Como vemos, los datos no necesariamente tienen que estar ordenados, además

2. Conceptos básicos

2.1. Índices

Los datos dentro de un arreglo estan indexados, es decir, tienen un índice que indica en que posición se encuentran. Este índice comienza con el valor de cero para el primer elemento, 1 para el segundo y así en adelante. Gráficamente se puede ver de la siguiente manera:

Dato	10	1	-3	13	4	-6	20	17	10
Indice	0	1	2	3	4	5	6	7	8

Así, si se quiere acceder al primer elemento del arreglo habría que mirar el índice 0 de éste, para acceder al segundo habría que mirar el índice 1 y así en adelante (se parte contando desde el cero).

2.2. Largo fijo

Los arreglos poseen un largo fijo, es decir, si se le quiere agregar nuevos datos a un arreglo una vez creado, habría que crear un nuevo arreglo.

2.3. Mismo tipo

Todos los datos dentro de un arreglo deben ser del mismo tipo (`int`, `double`, `String`, etc). Es decir, no se puede tener un arreglo que, por ejemplo, tenga `ints` y `doubles` como datos.

2.4. Modificación de un arreglo

Como se mencionó anteriormente, un arreglo tiene un largo fijo y para modificar el largo habría que crear un nuevo arreglo. Pero los datos dentro de estos arreglos si son modificables/reemplazables siempre que el nuevo dato sea del mismo tipo.

3. Estructura

3.1. Declarando arreglos

Para usar un arreglo, primero hay que declarar una variable que pueda referenciarlo. A modo general se declara de esta forma:

```
tipoDeDato[] nombreDeVariable;

// Como por ejemplo, se declara una variable que
// es un array de doubles:

double[] miLista;
```

3.2. Creando arreglos

El modo general de crear un arreglo y asignarlo a una variable **ya creada** es el siguiente:

```
nombreDeVariable = new tipoDeDato[tamañoDelArray];

// Si queremos crear un array que  contenga 8
// doubles y asignarlo a la variable ya creada
// se haría de la siguiente forma:

double[] miLista;
miLista = new double[8];
```

Pero no es necesario hacer esto en pasos separados. Se puede crear una variable y asignarle un arreglo en un solo paso siguiendo la siguiente regla general:

```
tipoDeDato[] nombreDeVariable = new tipoDeDato[tamañoDelArray];

// Como ejemplo, en este caso queremos crear un
// arreglo con los nombres de 20 personas:

String[] miListaDeNombres = new String[20];
```

Existe una manera de crear un arreglo introduciendo inmediatamente los datos que tendrá. En este caso no es necesario especificarle el largo que tendrá ya que se inferirá de los datos introducidos. A modo general se hace de la siguiente manera:

```
tipoDeDato[] nombreDeVariable = {valor 0, valor 1 , ..., valor n};


// Por ejemplo, queremos crear un arreglo que contenga
// los nombres de los siguientes animales: pato, perro,
// gato, hamster y vaca:

String[] miListaDeAnimales = {"pato", "perro", "gato", "hamster", "vaca"};
```

3.3. Accediendo a los datos del arreglo

Ya sabemos crear arreglos, pero tambien necesitamos poder acceder a ellos para obtener un dato en cierto índice o para modificarlo.
A modo general, para poder acceder a un dato ubicado en el **índice** i, se hace lo siguiente:

```
// Utilicemos el mismo arreglo de nombres
// de animales:
String[] miListaDeAnimales = {"pato", "perro", "gato", "hamster", "vaca"};


// Queremos acceder cuarto dato del arreglo,
// es decir, al que le corresponde el índice 3:
miListaDeAnimales[3];

// Si queremos asignarlo a una variable:
String miAnimalEnCuestion = miListaDeAnimales[3];

//Y si lo imprimimos, obtendremos "hamster":
System.out.println(miAnimalEnCuestion);
```

3.4. Modificando los datos del arreglo

Para modificar un dato de un arreglo, simplemente tenemos que acceder a éste y asignarle un nuevo valor siguiendo la siguiente forma general:

```
nombreDeVariableDelArray[índiceACambiar] = nuevoValor;

// Si en el ejemplo anterior queremos que el
//dato que se encuentra en el índice 3 ahora
//tenga el valor de "paloma":
miListaDeAnimales[3] = "paloma";
```

3.5. Obteniendo el largo de un arreglo

Para obtener el largo (length en inglés) de un arreglo, hay que hacerlo de la siguiente manera:

```
nombreDeVariable.length;

// Siguiendo el ejemplo de la lista de animales,
// si quisiésemos imprimir el largo del array:
System.out.println(miListaDeAnimales.length);
```

3.6. Ejemplo consolidador de conocimientos

Se propone el siguiente ejemplo:

```
1  class miClase
2  {
3    public static void main (String[] args)
4    { // Se crea un array de Strings
5      String[] miListaDeAnimales = {"pato", "perro", "gato", "hamster", "vaca"};
6
7      // Se crea una variable que tendrá el valor "hamster"
8      String miAnimalEnCuestion = miListaDeAnimales[3];
9      System.out.println(miAnimalEnCuestion);
10
11     // Se modifica el valor del dato en el tercer índice
12     miListaDeAnimales[3] = "paloma";
13     System.out.println(miListaDeAnimales[3]);
14
15     // Pero la variable miAnimalEnCuestion todavía vale "hamster"
16     System.out.println(miAnimalEnCuestion);
17
18     // Se imprime el largo del array
19     System.out.println(miListaDeAnimales.length);
20
21     // Se imprime el último elemento del array
22     System.out.println(miListaDeAnimales[miListaDeAnimales.length - 1]);
23   }
24 }
```

4. Operaciones sobre arreglos

4.1. Iterando en un arreglo

Para iterar sobre un arreglo, es necesario recorrer todos sus elementos utilizando un for (o un while que termine cuando se llegue al último elemento del arreglo). A continuación se entrega un código que imprime todos los elementos de un arreglo:

```
1  class ArregloDePrueba {
2
3     public static void main(String[] args) {
4        double[] miLista = {1.9, 2.9, 3.4, 3.5};
5
6        // Imprime todos los elementos del array
7        for (int i = 0; i < miLista.length; i++) {
8           System.out.println(miLista[i]);
9        }
10    }
11 }
```

4.2. Sumando todos los elementos de un arreglo

En este caso, es necesario recorrer todo el arreglo e ir incrementando una variable aparte con el valor de los datos del arreglo:

```
1  class ArregloDePrueba {
2
3     public static void main(String[] args) {
4        double[] miLista = {1.9, 2.9, 3.4, 3.5};
5
6        // Obtiene la suma de todos los elementos
7        double total = 0;
8        for (int i = 0; i < miLista.length; i++) {
9           total += miLista[i];
10       }
11       System.out.println("El total es " + total);
12    }
13 }
```

4.3. Buscando elementos en un arreglo

Para saber si un elemento está en un arreglo, hay que recorrerlo comparando valores:

```
1  class ArregloDePrueba {
2
3     public static void main(String[] args) {
4        double[] miLista = {1.9, 2.9, 3.4, 3.5};
5
6        // Revisa si 2.9 está en el arreglo
7        double datoABuscar = 2.9;
8        boolean encontrado = false;
9        for (int i = 0; i < miLista.length; i++) {
10          if (miLista[i] == datoABuscar) {
11           System.out.println("El dato se encuentra en el array");
12           encontrado = true;
13           break;
14          }
15       }
16       if (!encontrado) {
17           System.out.println("El dato no se encuentra en el array");
18       }
19    }
20 }
```

4.4. Comparando dos arreglos

En este caso, hay que iterar sobre ambos arreglos y comparar los datos que se encuentran en el mismo índice:

```
1  class ArregloDePrueba {
2
3     public static void main(String[] args) {
4        double[] miLista1 = {1.9, 2.9, 3.4, 3.5};
5        double[] miLista2 = {1.9, 2.9, 3.4, 3.6};
6
7        // Primero revisar si son del mismo largo
8        if (miLista1.length != miLista2.length) {
9          System.out.println("No son iguales");
10         return;
11       }
12
13       // Revisa dato por dato
14       for (int i = 0; i < miLista1.length; i++) {
15          if (miLista1[i] != miLista2[i]) {
16           System.out.println("No son iguales");
17           return;
18          }
19       }
20
21     System.out.println("Son iguales");
22    }
23 }
```

4.5. Ordenar un arreglo

Hay muchas formas distintas de ordenar arreglos. Para efectos de esta guía, se ofrecerá un método sencillo pero poco eficiente que ordena un arreglo. Queda para ejercicio del lector (si lo desea) investigar e implementar mejores algoritmos de ordenación.
En este algoritmo lo que se hace es extraer el menor elemento del arreglo inicial y colocarlo como primer elemento de un arreglo nuevo. El número de iteraciones de esta operación es el mismo que el largo del arreglo.

```
1  // Este import es para poder imprimir arreglos fácilemente
2  import java.util.Arrays;
3
4  class ArregloDePrueba {
5
6     public static void main(String[] args) {
7        double[] miLista = {1.9, 3.5, 3.4, 2.9};
8        int largo = miLista.length;
9        double[] ListaOrdenada = new double[largo];
10
11     for (int j = 0; j < largo; j++) {
12         double minimo = Double.MAX_VALUE;
13         int indiceMinimo = 0;
14         for (int i = 0; i < largo; i++) {
15            if (miLista[i] < minimo) {
16              minimo = miLista[i];
17              indiceMinimo = i;
18            }
19         }
20         ListaOrdenada[j] = minimo;
21         miLista[indiceMinimo] = Double.MAX_VALUE;
22       }
23    System.out.println(Arrays.toString(ListaOrdenada));
24    }
25 }
```

4.6. Agregar un elemento al final de un arreglo

Como los arreglos son de largo fijo, para agregar un elemento a un arreglo, hay que crear uno nuevo con mayor tamaño, copiar el arreglo inicial e insertarle el dato:

```
1  import java.util.Arrays;
2
3  class ArregloDePrueba {
4
5     public static void main(String[] args) {
6        double[] miLista = {1.9, 3.5, 3.4, 2.9};
7        double datoInsertar = 5.5;
8        double[] nuevaLista = new double[miLista.length + 1];
9
10
11     for (int j = 0; j < miLista.length; j++) {
12         nuevaLista[j] = miLista[j];
13       }
14       nuevaLista[nuevaLista.length - 1] = datoInsertar;
```

```
15         System.out.println(Arrays.toString(nuevaLista));
16     }
17 }
```

4.7. Eliminar un elemento de un arreglo

Igual que como para agregar un elemento hay que crear un nuevo arreglo, lo mismo sucede para eliminar un elemento:

```
1  import java.util.Arrays;
2
3  class ArregloDePrueba {
4
5     public static void main(String[] args) {
6        double[] miLista = {1.9, 3.5, 3.4, 2.9};
7        int indiceEliminar = 2;
8        double[] nuevaLista = new double[miLista.length - 1];
9        int k = 0;
10
11      for (int j = 0; j < miLista.length; j++) {
12          if (j == indiceEliminar) {
13            continue;
14          }
15          nuevaLista[k] = miLista[j];
16          k ++;
17       }
18       System.out.println(Arrays.toString(nuevaLista));
19    }
20 }
```

5. Relación entre strings y arreglos

Un string se podría ver como un arreglo de caracteres en donde cada caracter tiene un índice con el cual se le puede acceder. Por ejemplo:

```
1  class StringArray {
2
3     public static void main(String[] args) {
4        String miString = "Hola alumno!"
5
6        // Para acceder al caracter en el índice i,
7        // podemos usar charAt (notar el parecido con arrays).
8        // Queremos acceder al cuarto caracter, es decir,
9        // aquel que se encuentra en el índice 3
10       System.out.println( miString.charAt(3) );
11
12       // Podemos obtener el largo de un string
13       // de forma similar a como obtenemos el
14       // largo de un array:
15        System.out.println( miString.length() );
16
17    }
```

```
18 }
```

5.1. Uso del split

Split es una función particular que ' separa' un string siguiendo un criterio de separación y retorna un array de strings con los strings separados. Por ejemplo:

```
1  import java.util.Arrays;
2
3  class Split {
4
5          public static void main(String[] args) {
6             String miString = "Hola-alumno! como-esta?";
7
8             // Si queremos separar por el caracter "-"
9             String[] separados1 = miString.split("-");
10
11            // Tambien podemos separar por el caracter o
12             String[] separados2 = miString.split("o");
13
14            // O separar por el espacio " "
15             String[] separados3 = miString.split(" ");
16
17             System.out.println(Arrays.toString(separados1));
18             System.out.println(Arrays.toString(separados2));
19             System.out.println(Arrays.toString(separados3));
20         }
21      }
```