UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

Facultad de Ciencias

Introducción a las Ciencias de la Computación

Práctica 8: Arreglos

Profesora:
Amparo López Gaona
Ayudante:
Adrián Aguilera Moreno
Ayudante de Laboratorio:
Kevin Jair Torres Valencia

Objetivos

El objetivo de esta práctica es que el alumno refuerce sus conocimientos acerca de arreglos y matrices de datos de tipo primitivo. Ejercitando la inicialización, llenado y consulta de arreglos, así como el trabajo con instrucciones de iteración.

Introducción

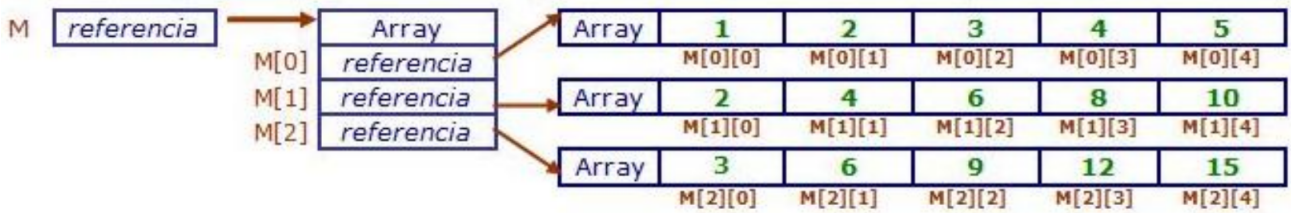
Un arreglo en Java puede tener más de una dimensión. El caso más general son los arreglos bidimensionales también llamados **matrices** o **tablas**.

La dimensión de un arreglo la determina el número de índices necesarios para acceder a sus elementos. Los arreglos que hemos visto han sido 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.

La siguiente figura representa un arreglo M de 3 filas y 5 columnas:

	0	**1**	**2**	**3**	**4**
0	1	2	3	4	5
1	2	4	6	8	10
2	3	6	9	12	15

Pero en realidad una matriz en Java es un arreglo de arreglos. Gráficamente podemos representar la disposición real en memoria del arreglo anterior así:



La longitud del array M (M.length) es 3.
La longitud de cada fila del array (M[i].length) es 5.

Para acceder a cada elemento de la matriz se utilizan dos índices. El primero indica la fila y el segundo la columna.

Se crean de forma similar a los arreglos unidimensionales, añadiendo un índice. Por ejemplo, una matriz de datos de tipo `int` llamado ventas de 4 filas y 6 columnas:

```
int [][] nombreDeArreglo = new int[4][6];
```

Para recorrer una matriz se anidan dos instrucciones de iteración. En general para recorrer un arreglo. multidimensional se anidan tantas instrucciones de iteración como dimensiones tenga el arreglo.

Por ejemplo:

```
// Mostramos en pantalla los valores que contiene la matriz
System.out.println("Valores introducidos:");

for (int i = 0; i < A.length; i++) {
    for (int j = 0; j < A[i].length; j++) {
        System.out.print(A[i][j]);
    }
}
```

Por otro lado y con frecuencia se tiene que los métodos requieren de algunos datos para realizar su tarea, estos datos se conocen como parámetros. Los parámetros formales se especifican, después del nombre del método, entre paréntesis, como una lista de parejas separadas por comas. Cada pareja incluye el tipo y el nombre de cada parámetro. Los paréntesis son parte de la sintaxis, así que deben estar presentes aunque el método no requiera parámetros. Existen diferencias entre la declaración de una variable y la de un parámetro. En los parámetros no se especifica la visibilidad, cada parámetro se precede de su tipo y la definición no termina con punto y coma. El dato con el que se llama a ejecutar el método se conoce como parámetro real o bien como parámetro actual. Al llamar a ejecución un método, el valor del parámetro real se asigna como valor inicial del parámetro formal y termina la relación entre ambos parámetros; es decir, si en el cuerpo del método se modifica el valor del parámetro formal no cambia el valor del parámetro real; esto se conoce como paso de parámetros por valor.

El método `main` tiene como parámetro un arreglo de cadenas cuyos valores se proporcionan al llamar a ejecutar el programa. Éstos son cadenas separadas por espacios en blanco. Si se desea que alguna cadena incluya un espacio debe encerrarse ésta entre comillas. Una vez en ejecución el método `main`, su parámetro puede usarse como cualquier otro arreglo de cadenas.

Desarrollo

Instrucciones: Para esta práctica, de ninguna manera se permite el uso de estructuras de datos distintas a los arreglos, es decir, no deben usar las bibliotecas: ArrayList, LinkedList, entre otras.

1. (Arreglos primitivos) Escribe una clase que modele el objeto `String` como un arreglo de tipo `char`. La clase debe cumplir con los siguientes requisitos:

1. Llama a tu clase de la siguiente manera: `MyString`.
2. Crea constructores que tenga la siguientes firmas:
   (a) `MyString(char[] arregloCaracteres)`.
   (b) `MyString(int longitud)`.
3. Crea un método `cambiarEn(int posicion, char caracter)` que sustituya el caracter en la posición ingresada.
4. Crea un método llmado:
   (a) `substraer(int inicio)` tal que devuelva un arreglo de elementos tipo `char`.
   (b) `substraer(int inicio, int fin)` tal que devuelva un arreglo de elementos tipo `char`.

   que simule el comportamiento del método `substring` de la clase `String` en sus dos versiones.
5. Implementa un método llamado: `obtenerNumeroBytes()` tal que devuelva el número de bytes que suman los elementos tipo `char` en el objeto `MyString`.
6. Por último: Deben escribir un método `obtenerRepresentacion()` tal que devuelva una representación del obtejo `MyString` sin utilizar objetos `String`. Este método es, escencialmente, un método `toString`

   Hint. Puedes utilizar la instrucción `for-each`.

2. (Paso de parámetros en el main) Escribir programas cuyo método main reciba una serie de parámetros para poder trabajar dando difrentes resultados de acuerdo con dichos parámetros.

1. Escribir un programa Nombre que reciba n parametros que se supone son el nombre completo de una persona y los escriba empezando por los apellidos.

   Ejemplos:

   ```
   >java Nombre Juan Paco Pedro "de la Mar" Perez
   El nombre completo es: de la Mar Perez Juan Paco Pedro.
   El nombre es: Juan Paco Pedro.

   >java Nombre
   Debes ingresar al menos tres cadenas para el nombre.

   >java Nombre Ana Maria
   Debes ingresar al menos tres cadenas para el nombre.

   >java Nombre Ana Patricia Maria
   El nombre completo es: Patricia Maria Ana
   El nombre es: Ana
   ```

2. Escribir un programa Numeros que lea 4 cadenas, la primera indica qué hacer y las otras tres deben tratarse como números.

   Los posibles valores para la primera cadena son un guión seguido de alguna de las siguientes letras M, m, p,o, t. El significado es:

   - M determina el valor mayor.
   - m determina el valor menor.
   - p calcula el promedio de los tres valores.
   - o muestra los tres valores en orden descreciente.
   - t realiza las cuatro funciones anteriores.

   Ejemplos de ejecución del programa son los siguientes:

```
> java Numeros -M 45 78 90
El mayor de 45, 78 y 90 es 90

> java Numeros -p 45 345 90
El promedio de 45, 345 y 90 es 160

> java Numeros -t 45 345 90
El mayor de 45, 345 y 90 es 345
El menor de 45, 345 y 90 es 45
El promedio de 45, 345 y 90 es 160
Los números ordenados son 345, 90, 45

> java Numeros -x 45 345 90
Opcion incorrecta !!!

> java Numeros -t 45
Cantidad incorrecta de datos!!!
```

Formato de Entrega

1. Las prácticas se entregarán en parejas.
2. NO incluir los archivos .class dentro de la carpeta.
3. Los archivos de código fuente deben estar documentados.
4. Se pueden discutir y resolver dudas entre los integrantes del grupo. Pero cualquier práctica plagiada total o parcialmente será penalizada con cero para los involucrados.
5. La práctica se debe subir al Github Classroom correspondiente.
6. La entrega en classroom debe contener el link HTTPS de su repositorio y es lo único que se debe entregar.
7. **Nota: Para alcanzar una calificación superior a 5, la práctica debe compilar correctamente y sin advertencias, en caso contrario tendrán 0 en automático**