UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

Facultad de Ciencias

Introducción a las Ciencias de la Computación

Práctica 10: Arreglos III

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

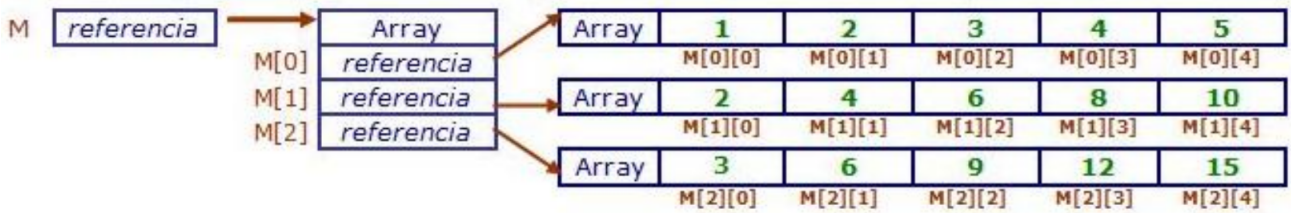
Un arreglo en Java puede tener más de una dimensión. El caso más general son los arreglos bidimensionales también llamados **matrices** o **tablas**.

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

En la Facultad de Ciencias se está organizando una feria de juegos. Para lograrlo, se ha asignado la tarea de desarrollar el siguiente juego a los alumnos de Introducción a las Ciencias de la Computación (ICC) asegurándose de que cumpla con todos los requerimientos planteados.

Las Torres de Hanoi es un rompecabezas matemático que fue inventado en 1883 por el matemático francés Édouard Lucas. Este juego se juega con tres postes y un número variable de discos de diferentes tamaños, que se apilan en uno de los postes, para que fuese medianamente complicado jugar a la comunidad de la facultad, el juego será utilizando 6 discos y 3 postes.

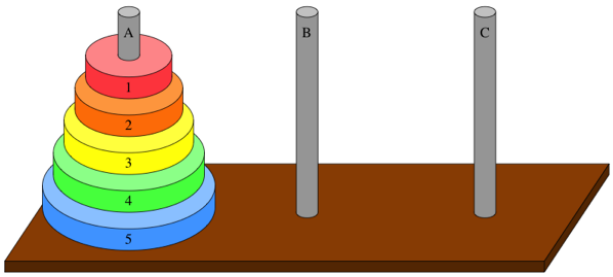
El juego consiste en mover una torre de 6 discos desde un poste de origen hasta un poste destino, utilizando un tercer poste auxiliar, y cumpliendo las siguientes reglas:

- Solo se puede mover un disco a la vez.
- Solo se puede mover el disco superior de una torre.
- Ningún disco puede colocarse sobre otro más pequeño.

El programa debe permitir visualizar o mostrar los movimientos realizados hasta completar la transferencia de todos los discos al poste destino. También este debe permitir al usuario seleccionar el poste de origen y el poste de destino para cada movimiento.

Consideraciones

- Validar que los movimientos sean válidos (no se puede colocar un disco grande sobre uno más pequeño).
- Detectar y mostrar un mensaje cuando el jugador haya completado el juego o permitirle abandonar el juego en cualquier momento.
- 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.



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**