UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

Facultad de Ciencias

Introducción a las Ciencias de la Computación

Práctica 9: Arreglos II.

Profesora: Amparo López Gaona
Ayudante: Ramsés Antonio López Soto
Ayudantes de Laboratorio:
Adrián Aguilera Moreno
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

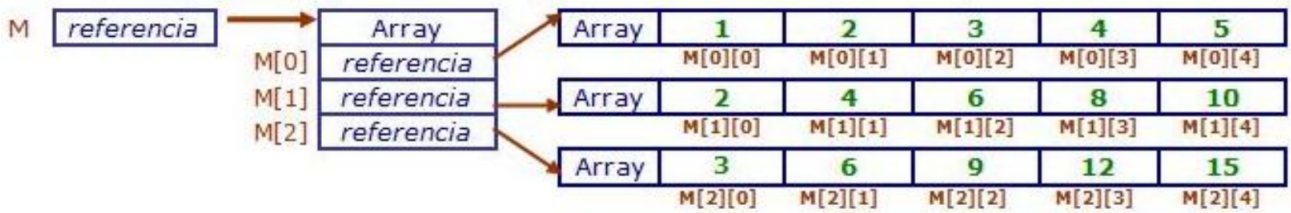
Un arreglo en Java puede tener más de una dimensión. El caso más general son los arreglos bidimensionales también llamados **matrices** o **tablas**.

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.

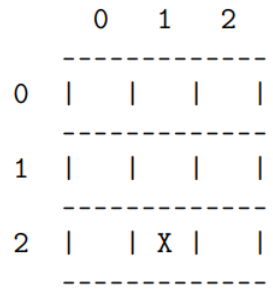
Desarrollo

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. (Gato): Se deberá programar el juego de tres en línea, es decir, el "Gato".

Para ello deberán generar la clase `Gato` con los siguientes métodos:

1. Un atributo que reperesente un tablero de $3x3$.
2. Un constructor por omisión.
3. Método que permita colocar una figura ('X', 'O') en el tablero por medio de coordenadas y este deberá regresar si es valida o no la jugada. Por ejemplo:
   - Se tira 'X' en la coordenada $(1, 2)$ regresa **true**.



   - Se tira 'O' en la coordenada $(1, 2)$ regresa **false**, dado que 'X' ya ocupaba esa posición.
4. Método que regresa un boolean indicando si termino el juego, dependiendo del estado del tablero.
   - Regresa **True** si:
     - **Caso 1:** Existe un jugador.
       (Si un jugador junta tres figuras iguales en diagonal, horizontal o vertical).
     - **Caso 2:** Las 9 casillas se llenan (Empate).
   - En caso contrario **False.**
5. Método que muestre por pantalla el jugador que gano ('X' o 'O') o en otro caso el mensaje de **"Empate"**

Ejemplo de ejecución

```
-------------
|   |   |   |
-------------
|   |   |   |
-------------
|   |   |   |
-------------

Jugador X :
coordenada x,y :
1,1
-------------
|   |   |   |
-------------
|   | X |   |
-------------
|   |   |   |
-------------


Jugador O :
coordenada x,y :
2,0
-------------
|   |   | O |
-------------
|   | X |   |
-------------
|   |   |   |
-------------


Jugador X :
coordenada x,y :
0,0
-------------
| X |   | O |
-------------
|   | X |   |
-------------
|   |   |   |
-------------
```

```
Jugador O :
coordenada x,y :
2,1
-------------
| X |   | O |
-------------
|   | X | O |
-------------
|   |   |   |
-------------
```

```
Jugador X :
coordenada x,y :
2,2
-------------
| X |   | O |
-------------
|   | X | O |
-------------
|   |   | X |
-------------

Felicidades jugador X has ganado!!!!!
```

2. Tener una opción para jugar contra la computadora.
Hint: Usen random para generar una 'X' o 'O' en alguna coordenada aleatoria.

Formato de Entrega

1. Las prácticas se entregarán en parejas.
2. Cada práctica (sus archivos y directorios) deberá estar contenida en un directorio llamado `EquipoX_pY`, donde:
   (a) X es el número de equipo correspondiente.
   (b) Y es el número de la práctica.

   Por ejemplo: `Equipo08_p01`
3. NO incluir los archivos .class dentro de la carpeta.
4. Los archivos de código fuente deben estar documentados.
5. 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.
6. La práctica se debe subir al Github Classroom correspondiente.
7. La entrega en classroom debe contener el link HTTPS y SSH de su repositorio y es lo único que se debe entregar.
8. El horario y día de entrega se acordará en la clase de laboratorio y no deberá sobrepasar 2 clases de laboratorio.