UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

Facultad de Ciencias

Introducción a las Ciencias de la Computación

Práctica 08: Arreglos en varias dimensiones.

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

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

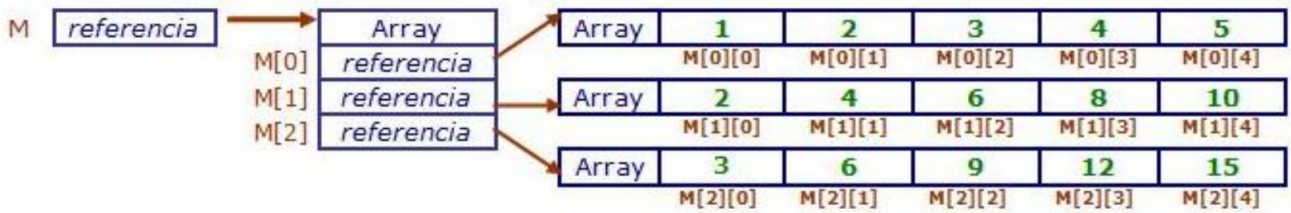
Un arreglo en Java puede tener más de una dimensión. El caso más general son los arreglos bidimensionales también llamados **matrices** o **tablas**.

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]);
    }
}
```

Desarrollo

Instrucciones: Resuelve 2 de los 3 problemas y sus respectivas preguntas.

1. Permutaciones de Josephus: Supongamos que n personas están sentadas alrededor de una mesa circular con n sillas, y que tenemos un entero positivo $m \geq n$. Comenzando con la persona con etiqueta 1, (moviendonos siempre el la dirección de las manecillas del reloj) comenzamos a remover los ocupantes de las sillas como sigue: Primero eliminamos la persona con etiqueta m. Recursivamente, eliminamos al m-ésimo elemento de los elementos restantes. Este proceso continua hasta que las n personas han sido eliminadas. El orden en que las personas han sido eliminadas, se le conoce como la (n, m)-permutación de Josepus. Por ejemplo sin $n = 7$ y $m = 3$, la $(7, 3)$-permutación de Josephus es: $\{3, 6, 2, 7, 5, 1, 4\}$.

Escribe un programa *recursivo* que genere la (n, m)-permutación de Josephus.

Hint. Utiliza un arreglo recorriendolo de manera "circular".

Responde las siguientes preguntas respecto al problema anterior:

1. ¿Cómo clasificarías el problema anterior respecto a una dificultad de entre 0 y 2? Donde 0 es equivalente a mencionar que el problema fue fácil y 2 es equivalente a mencionar que el problema fue difícil. ¿Por qué?
2. ¿Qué fue lo más difícil del problema?
3. ¿Crees que el problema anterior aporta conocimiento a tu formación como programador?
4. ¿Te gustarían más problemas de este estilo? ¿Por qué?
5. ¿Cuál crees que fue la intención de este problema?

   Obs. Estas preguntas son con respecto a tu opinión y se realizan para apoyar tú retroalimentación.

2. Escribe una clase que modele el objeto `String` como un arreglo de tipo `char`. La clase debe cumplir con los siguientes puntos:

1. Llama a tu clase de la siguiente manera: `MyString`.
2. Crea constructores que tenga la siguientes firma: `MyString(char[] arregloCaracteres)` y `MyString(int longitud)`.
3. Crea un método `ingresarEn(int posicion, char caracter)` que sustituya el caracter en la posición ingresada.
4. Crea un método que simule el método `substring` de la clase `String` en sus dos versiones.

5. Implementa algún otro método de la clase `String` en tu clase `MyString`. Puedes consultar la documentación de la clase String.

Al terminar este problema responde las siguientes preguntas de manera detallada:

1. ¿Qué se te dificultó más en este problema?
2. ¿Qué hubieses quitado de los incisos?
3. ¿Qué hubieses anexado en este problema?
4. ¿Qué te hubiese ayudado a resolver este problema de manera más sencilla?

3. Crear una clase que modele el comportamiento de una matriz de números. Para esta clase debes implementar los siguientes métodos:

1. Un método toString que visualice la matriz de números.
2. Crea un método que aplique la regla de *Sarrus* a matrices de $3 \cdot 3$.
3. Escribe un método que realice el producto de dos matrices si estas cumplen las propiedades necesarias, *i.e.* Para las matrices A, B en el producto $A \cdot B$ el número de columnas de A es el mismo que el número de filas de B (coincidencia de orden).

A partir de los incisos anteriores, contesta las siguientes preguntas:

1. De los incisos anteriores ¿Cuál ha sido el más complicado? ¿Por qué?
2. Tu implementación ¿Fue realizada con *recursión* o con *iteración*?
3. ¿Cuánto tiempo le dedicaste a la elaboración de cada inciso?
4. Explica con tus palabras ¿Qué son los arreglos y cómo los puedes utilizar? ¿En qué ámbitos los puedes utilizar?
5. ¿Cómo puedes obtener la diagonal de un arreglo 2-dimensional? Explica con detalle como lo harías. ¿Cómo obtendrías la diagonal de un arreglo 3-dimensional? Detalla tú respuesta.
6. ¿Cuál crees que fue la intención de este problema?

   Obs. Estas preguntas son con respecto a tu opinión y se realizan para apoyar tú retroalimentación.

Formato de Entrega

1. Las prácticas serán entregadas de forma individual.
2. Cada práctica (sus archivos y directorios) deberá estar contenida en un directorio llamado `apellidoPaterno_nombre_pX`, donde X es el número de la práctica.
   Por ejemplo: `aguilera_adrian_p8`
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.