Práctica 3
Polinomio de direccionamiento

Estructuras de datos

Meta

Que el alumno domine el manejo de información almacenada en arreglos multidimensionales.

Objetivos

Al finalizar la práctica el alumno será capaz de:

- Almacenar un arreglo de n dimensiones en uno de una sola dimensión.

Antecedentes

Vectores de Iliffe

Los arreglos multidimensionales son aquellos que tienen más de una dimensión, los más comunes son de dos dimensiones, conocidos como matrices. Este tipo de arreglos en Java se ven como arreglos de arreglos, a los cuales se llama *vectores de Iliffe*.

Existen dos momentos fundamentales en la creación de arreglos:

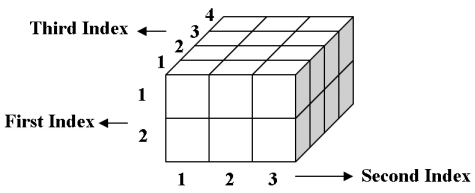
1. Declaración: en esta parte no se reserva memoria, solo se crea una referencia.

```
int [][] arreglo;
float[][][] b;
```

2. Reservación de la memoria y la especificación del número de filas y columnas.

```
//matriz con 10 filas y 5 columnas
arreglo = new int[10][5];

//cubo con 13 filas, 25 columnas y 4 planos
b = new float [13][25][4];
```



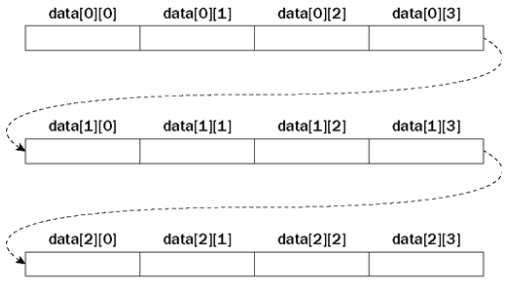
Three-dimensional array with twenty four elements

Nota: Se puede declarar una dimensión primero, pero siempre debe de ser en orden, por ejemplo `arreglo = new int[][10];` es erróneo pues las filas quedan indeterminadas. Esta libertad permite crear arreglos de forma irregular, como:

```
int[][][] arreglo = new int[3][][];
arreglo[0] = new int[2][];
arreglo[0][1] = {1,2,3};
arreglo[2] = new int[1][2];
// Se ve como:
// {{{1,2,3}, null}, null, {0,0}}
```

Polinomio de direccionamiento

Dado que la memoria de la computadora es esencialmente lineal es natural pensar en almacenar los elementos de un arreglo multidmensional en un arreglo de unidimensional. Por ejemplo, consideren un arreglo de tres dimensiones:



The array elements are stored in contiguous locations in memory.

Generalicemos el ejemplo anterior a uno de n-dimensiones, donde el tamaño de cada dimensión i esta dada por δ_i, entonces tenemos un arreglo n-D: $\delta_0 \times \delta_1 \times \delta_2 \times ... \times \delta_{n-1}$.

Entonces la posición del elemento $a[i_0][i_1][...][i_{n-1}]$ esta dada por:

$$p(i_0, i_1, ..., i_{n-1}) = \sum_{j=0}^{n-1} f_j i_j \tag{1}$$

donde

$$f_j = \begin{cases} 1 & \text{si } j{=}n-1 \\ \prod_{k=j+1}^{n-1} \delta_k & \text{si } 0 \leqslant j < n-1 \end{cases} \tag{2}$$

Desarrollo

La práctica consiste en implementar los métodos definidos en la interfaz `IArreglo`, la cual convierte un arreglo de enteros de n-dimensiones en uno de una dimensión. El constructor de la clase que implemente la interfaz deberá tener como parámetro un arreglo de `int`s que representen las dimensiones del arreglo. Por ejemplo para crear un arreglo tridimensional ($3 \times 10 \times 5$). Observa que entonces el número de dimensiones en la matriz estará dada por la longitud de este primer arreglo. Invocamos el constructor de la siguiente forma:

```
1  Arreglo a = new Arreglo(new int [] {3,10,5});
```

Observa que todas las dimensiones deben ser mayores que cero.

Preguntas

1. Explica la estructura de tu código, explica en más detalle tu implementación del método `obtenerIndice`.
2. ¿Cuál es el orden de complejidad de cada método?

Forma de entrega

- Asegúrense de borrar todos los archivos generados, incluyendo archivos de respaldo usando `ant clean`.
- Copien el directorio `Practica3` dentro de un directorio llamado `<apellido1_apellido2>` donde `apellido1` es el primer apellido de un miembro del equipo y `apellido2` es el primer apellido del otro miembro. Por ejemplo: `marquez_vazquez`.
- Borren el directorio libs en la copia.
- Agreguen un archivo `reporte.pdf` dentro del directorio `Practica3` de la copia, con el nombre completo de los integrantes del equipo, las repuestas a las preguntas de la sección anterior y cualquier comentario que quieran hacer sobre la práctica.
- Compriman el directorio `<apellido1_apellido2>` creando `<apellido1_apellido2>.tar.gz`. Por ejemplo: `marquez_vazquez.tar.gz`.
- Suban este archivo en la sección correspondiente del aula virtual. Basta una entrega por equipo.

Forma de evaluación

Para su calificación final se tomarán en cuenta los aspectos siguientes:

60 % Calificación generada por las pruebas automáticas. Usaremos nuestros archivos, por lo que si realizan modificaciones a sus copias, éstas no tendrán efecto al momento de calificar.

10 % Reporte con respuestas a las preguntas.

15 % Documentación completa y adecuada. Entrega en el formato requerido, sin archivos `.class`, respaldos, bibliotecas de `JUnit` u otros no requeridos.

15 % Revisión manual del código, para verificar que se cumpla con las especificaciones, que no se haya copiado, etcétera.