



Introducción a la Programación. Curso 2021-2022

EJERCICIO OBLIGATORIO DE SEMINARIO

Fecha tope para subir el proyecto al campus virtual es el día 16 de diciembre


PARTE 1: Nota máxima 7 puntos (80% implementación 20% pruebas unitarias)

Construir un proyecto en blueJ con una clase `Matrix` con tenga como propiedad una matriz bidimensional de números enteros de dimensión máxima 20x20.

Se deben implementar las siguientes operaciones:

- 1. Constructor con un parámetro entero.** Crea una matriz **cuadrada** cuya dimensión se corresponde con el valor del parámetro. La matriz se rellenará de forma aleatoria con números enteros, de tipo `int`, entre 0 y 255. Si el parámetro no es válido se debe lanzar una excepción "Invalid dimension for the matrix".
- 2. Constructor con un parámetro array.** Se le pasa en la llamada una matriz bidimensional con unos valores concretos. Crea una matriz **copia** de la matriz pasada como parámetro. Este constructor es el que se debe usar para las pruebas unitarias. Si la matriz no es cuadrada, se debe lanzar una excepción "The matrix must be square". Si el tamaño de la matriz excede el indicado, debe lanzarse la excepción indicada en el apartado anterior.
- 3. Implementar el método** `public int getAverage()` Devuelve la media aritmética de todos los elementos de la matriz.
- 4. Implementar el método** `public int[] addByColumns()` Devuelve un vector con la suma de los elementos por columnas. Consiste en sumar los elementos de cada columna almacenándolos en un vector según se muestra en la figura. Suponiendo una matriz 6x6 el resultado sería:

0	1	2	3	4	5
5	10	8	9	45	23
16	30	25	41	12	49
7	17	50	12	36	27
45	8	22	34	18	9
6	21	13	5	11	41
23	45	16	18	29	25




0	1	2	3	4	5
102	131	134	119	151	174



5. **Implementar el método** `public void swapColumns()`. Consiste en intercambiar los elementos de las columnas de manera que se intercambien las columnas 0-1, 1-2, 2-3, ... según se muestra en la figura:

5	10	3	9	23	12
16	30	25	41	50	9
2	17	50	12	46	34
45	8	22	34	8	48
6	21	13	5	11	41
23	45	16	18	29	25

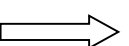


10	3	9	23	12	5
30	25	41	50	9	16
17	50	12	46	34	2
8	22	34	8	48	45
21	13	5	11	41	6
45	16	18	29	25	23

En caso de que el número de columnas sea impar, la última columna será la que se quede como está.

6. **Implementar el método** `public void rotateMatrix()` Consiste en girar la matriz 90° hacia la izquierda según se muestra en la figura. Suponiendo una matriz 4x4 el resultado sería:

5	10	8	9
16	30	25	41
7	17	50	12
45	8	22	34



9	41	12	34
8	25	50	22
10	30	17	8
5	16	7	45

7. **Implementar una clase de prueba unitaria** para probar los métodos de los apartados 3, 4, 5 y 6.

IMPORTANTE:

1. Para realizar algunas operaciones se puede utilizar un vector o una matriz auxiliar pero siempre debe estar declarada de forma local en el método.
2. En las pruebas unitarias se deben realizar un mínimo de 3 casos de prueba por método, utilizando matrices de diferentes dimensiones en cada caso. Hay que tener en cuenta, donde proceda, que hay contemplar casos de prueba positivos y negativos.
3. La signatura de los métodos tiene que ser exactamente la indicada en el documento, dado que los profesores vamos a construir nuestras propias pruebas. Si no podemos probar uno de sus métodos porque su signatura es incorrecta, se verá afectada la nota.
4. Existen asertos para arrays:

```
assertArrayEquals(int[][] expecteds, int[][] actuals)
```

Este aserto se cumple (es verdadero) si dos arrays de enteros son iguales. Se puede ver la documentación en el siguiente enlace:

[http://junit.org/junit4/javadoc/4.12/org/junit/Assert.html#assertArrayEquals\(int\[\],%20int\[\]\)](http://junit.org/junit4/javadoc/4.12/org/junit/Assert.html#assertArrayEquals(int[],%20int[]))



Para utilizarlo, se puede implementar un método `getMatrix()`, que devuelva **una copia** del atributo `matrix`. **No se debe implementar un método que devuelva una referencia a la propia matriz.**

DESAFÍO. Parte II (Junto con la parte I se obtendría la nota máxima 10 puntos)

- 8. Implementar el método** `public int getMaxAdjacentValue(int row, int column)`. Devuelve el valor mayor adyacente a una posición dada. Cada posición tiene 8 adyacentes, exceptuando las posiciones de los extremos que tienen 5 adyacentes o de las esquinas que tienen 3 adyacentes. Por ejemplo, el valor mayor de `matriz[1][1]` es 50 y el valor mayor de `matriz[1][3]` es 41

5	10	3	9
16	30	25	41
2	17	50	12
45	8	22	34

⇒ **50**

- 9. Implementar el método** `public int[][] smoothMatrix()`. Devuelve una matriz obtenida a partir del atributo, reemplazando cada elemento por la media aritmética de los vecinos (incluyendo el propio elemento) de la matriz original:

5	10	3	9
16	30	25	41
2	17	50	12
45	8	22	34

⇒

15	14	19	19
13	17	21	23
19	23	26	30
18	24	23	29

- 10. Añadir a la clase de prueba unitaria** los tests de los métodos de los apartados 8 y 9.

NOTA: Para realizar algunas operaciones se puede utilizar un vector o una matriz auxiliar pero siempre debe estar declarada de forma local en el método.