OOP using JAVA – Lab 2

This lab is concerned with manipulating arrays of two dimensions. This set of exercises is ensure you can use loops to navigate and manipulate a 2D array in many different ways.

In each case you need to write a method (function) that performs some calculations on a parameter array. Put all these methods into a *utility class* called `Array2DExercises`. Test each function in the main method.

1. Write a method `public static int max(int[][] a)` that returns the maximum value in the 2d parameter array `a`.

2. Write a method `public static int rowSum(int[][] a, int x)` that returns the sum of the elements in Row `x` of `a`.

3. Write a method `public static int columnSum(int[][] a, int x)` that returns the sum of the elements in Column `x` of `a` (careful with rows of different lengths!).

4. Write a method `public static int[] allRowSums(int[][] a)` that calculates the row sum for *every* row and returns *each* of the values in an array. Index i of the return array contains the sum of elements in row i.

5. Write a method `public static boolean isRowMagic(int[][] a)` that checks if the array is row-magic (this means that every row has the same row sum).

6. Write a method `public static boolean isColumnMagic(int[][] a)` that checks if the array is column-magic (this means that every column has the same column sum).

7. Write a method `public static boolean isSquare(int[][] a)` that checks if the array is square (i.e. every row has the same length as `a` itself).

8. Write a method `public static boolean isMagic(int[][] a)` that checks if the array is a *magic square*. This means that it must be square, and that all row sums, all column sums, and the two diagonal-sums must all be equal.

9. Write a method `public static boolean isLatin(int[][] a)` that checks to see if the array is a *Latin square*. This means that it must be square (suppose it is n x n), and that each row and each column must contain the values 1, 2, ..., n with no repeats.

10. Write a methodc `public static boolean isSequence(int[][] a)` that checks to see if the array is square (suppose it is n x n), and contains each of the digits from 1 to n*n, eg. 1, 2, ..., 16 for a 4 x 4 array.