# Lab assignment 17.1

**MagicSquare**

**Create a new project, TwoDimensionalArrays.**

**Place the starter code from Schoology called, MagicSquare.java and MagicSquareTester.java, into this new project.**

**Explanation**

Write a program that creates odd-sized **Magic Squares**. A magic square is a square matrix of consecutive numbers, such that the numbers in the rows, the numbers in the columns, and the numbers in the diagonals add up to the same sum. For this program you will only be concerned with **“odd-sized”** magic squares. Odd-sized refers to the size of the matrix, such as a 3 X 3, a 5 X 5, and 9 X 9 or 13 X 13 sized matrix.

Examples of a 3 X 3 magic square and a 5 X 5 magic square are shown below:

|  |  |  |
| --- | --- | --- |
| **8** | **1** | **6** |
| **3** | **5** | **7** |
| **4** | **9** | **2** |

Every row, column and diagonal in the 3 X 3 matrix adds up to 15.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **17** | **24** | **1** | **8** | **15** |
| **23** | **5** | **7** | **14** | **16** |
| **4** | **6** | **13** | **20** | **22** |
| **10** | **12** | **19** | **21** | **3** |
| **11** | **18** | **25** | **2** | **9** |

Every row, column and diagonal in the 5 X 5 matrix adds up to 65.

The magic square program is quite challenging, and comes in two different levels of difficulty. It is very important that you first are able to create magic squares, of any odd size, yourself on paper. On the next page is a five-step algorithm to create an odd magic square. Study these steps carefully and use them to create magic squares on paper. Only after you are confident that you understand the magic square algorithm, are you ready to proceed to write a program.

**Magic Square Algorithm**

The creation of a magic square involves using the following algorithm. It will be your assignment to translate this algorithm into Java program code.

|  |
| --- |
| **Odd-Sized Magic Square Algorithm** |
| * Start number **1** in the **top row**, **middle column**. * Place consecutive integers in a **diagonally-up-to-the-right** pattern. * Any number that goes outside the matrix **above row 1** is moved to the **bottom row**. * Any number that goes outside the matrix past the **right column** is moved to the **left column**. * Any number, which follows the **multiple of the matrix size**, is moved **down 1 row**. |

**Required Methods**

**Note:** Numbers for all versions of this assignment need to use a three digits "000" decimal format.

**public void computeMagicSquare()**

Your program must place consecutive integers in the proper matrix locations.

**public void displayMagicSquare()**

Your program must displays the values of the matrix.

**Important Hint:** Place each consecutive number, starting at 1, in the correct matrix location, not the other way around. Think what you do when you create a magic square on paper. You start by placing number 1 in the right place, then number 2, number 3 and so on.

**public void checkRows()**

Your program must prove the magic square is truly a magic square by adding up and displaying the sums of all of the numbers in every row. This method should only display the sum.

**public void checkColumns()**

Your program must prove the magic square is truly a magic square by adding up and displaying the sums of all of the numbers in every column. This method should only display the sum.

**public void checkDiagonals()**

Your program must prove the magic square is truly a magic square by adding up and displaying the sums of all of the numbers in every diagonal. This method should only display the sum.

**Note:** When you observe the magic squares you will note that the *median* number in the consecutive integer sequence is in the center of the square. If you take the median number and multiply it times the size of the magic square, you will get the sum of each row, column and diagonal.

|  |
| --- |
| **Odd-Sized Magic Square Sum Rule** |
| The sum of the rows, columns and diagonals in an odd magic square will always be the size of the square times the median number in the square.  In a 3 X 3 that means **3 \* 5 = 15**.  In a 5 X 5 that means **5 \* 13 = 65**. |

**Output for a 5x5 matrix**

|  |
| --- |
| 5x5 Magic Square  ==================  017 024 001 008 015  023 005 007 014 016  004 006 013 020 022  010 012 019 021 003  011 018 025 002 009  Checking Rows  =============  65 65 65 65 65  Checking Columns  ================  65 65 65 65 65  Checking Diagonals  ==================  65 65 |