**Design: Module/Function/Method List**

Module:

* MagicSquare: A class that utilizes an Integer Array as a private member to represent a perfect square matrix. The class contains a main (driver) function and other functions to assist in the creation of the matrix. The class also contains supporting getter and setter and methods, as well as methods to test the Magic Square requirements for the NxN matrix.

Functions:

* main (driver): The main function is the driver for the program. It contains both the directions for the user and will use both functions and class methods to accept input from the user, and determine if the user entered matrix is a Magic Square.
* initiate: The function begins our program processes by requesting inputs from the user in string format and then separating these inputs in a String Array.
* perfectSquare: This function represents the first test of the Magic Square. It returns a Boolean value of true if the number of inputs provided by the user is a perfect square, allowing the matrix to be generated. If false, the user will be notified that their inputs did not meet requirements of the first test, and the program closes.

Methods:

* constructor(default): Default constructor for the class. Sets the size of the Integer array to 0.
* constructor: Takes the String array of perfect square numbers and parses it into an integer array which will serve as the representation of the matrix.
* setMatrix: A setter method for the matrix. Can edit each individual index of the array.
* getMAtrix: Getter method for the matrix. Returns the Integer array.
* testOccurence: This method serves as the second test for the Magic Square. It checks if every number between 1 & n, where n is the number of inputs provided by the user, appear at least once in the matrix. Returns a Boolean value of true if it does, and false otherwise (exiting the program).
* calcSum: Sums up the values in the first column of the matrix. The sum of the column will serve as a benchmark for all other rows, columns and diagonals to equal. Returns the value of the sum.
* testRows: First part of the third test. Cycles through the rows of the matrix and compares their sums to the benchmark value. If any of them fail to equal the sum, the method will return false, exiting the program, otherwise the method returns true.
* testColumns: Second part of the third test. Cycles through each column, comparing their sums to the benchmark value. All of them must equal the benchmark for the method to return true.
* testDiagonals: Last part of the third test. Compares the sum of both diagonal paths of the matrix to the benchmark value. If they are both equivalent, then the method returns true, or false otherwise once again exiting the program. If the method returns true, then all tests have been passed successfully and the matrix is a perfect square.
* toString: Returns a display of the Integer Array in matrix form.