CSGE601021 Foundations of Programming 2 (Dasar-Dasar Pemrograman 2)

Programming Assignment 2:

Java Arrays and Magic Squares

- Deadline for submitting the final result of your work to SCeLE: Tuesday 14 March 2023 (11:55 PM SCeLE Time). Use github for storing your partial works.
- Zip all files of your Java project (1 folder) and its runnable JAR file, into one file with name: TP02_<class>_<TAcode>_<StudentName>_<NPM>.zip, for example: TP02_A_XY_Unyil_1234567890.zip
- Arrange a demo session with your Asdos as soon as possible, while everything about this assignment is still fresh in your memory.
- Start working on this assignment immediately. You will need enough time to understand the problem.
- We learn best by not doing plagiarism.

Evaluation scheme:

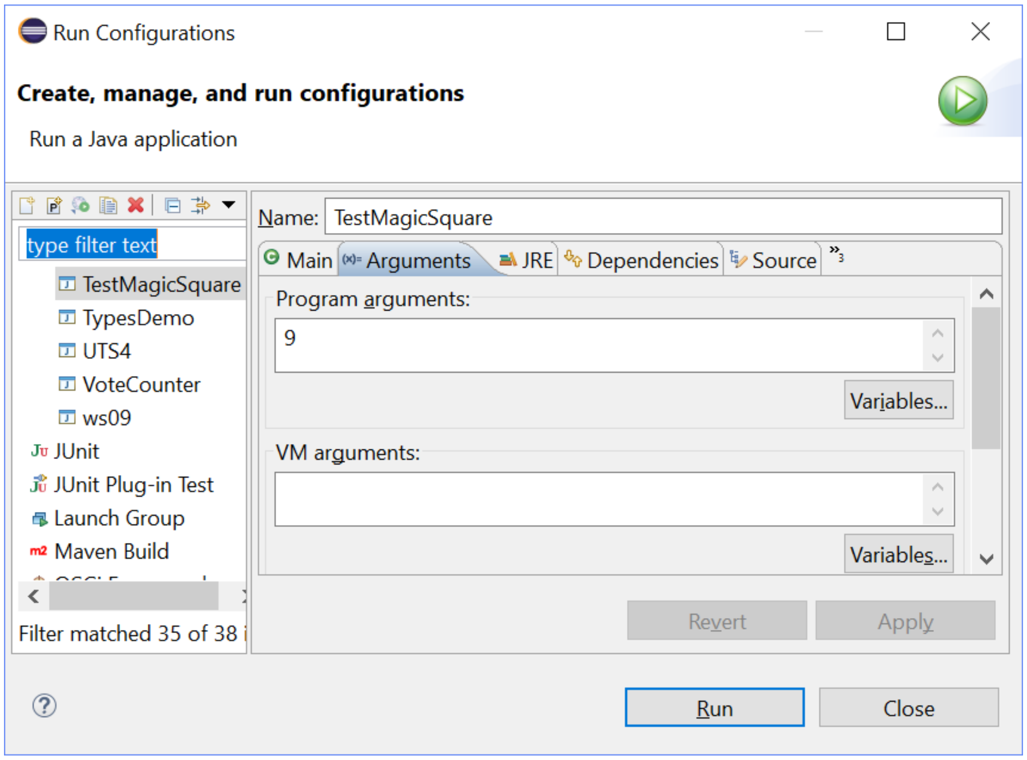
60 % correctness
30 % explanation in demo session
10 % program documentation (comments, neatness)

Task Description

Build a *Java project* using your favorite IDE. Your project should consist of two classes: MagicSquare and TestMagicSquare. Export your project as a *runnable JAR file* which can be executed in command shell or terminal and takes an argument for the method main from *command line*.
See the example at the end of this document which shows the passing of the argument "9" to the method main:

```
> java -jar magicSquare.jar 9
```

Passing arguments for the method main can also be done in an IDE such as Eclipse using the "Run Configurations" menu: Run → Run Configurations → Arguments …
For instance:


Run Configurations

Create, manage, and run configurations

Run a Java application

type filter text

TestMagicSquare
TypesDemo
UTS4
VoteCounter
ws09
JUnit
JUnit Plug-in Test
Launch Group
Maven Build

Filter matched 35 of 38 i

Name: TestMagicSquare

Main | Arguments | JRE | Dependencies | Source

Program arguments:
9

Variables...

VM arguments:

Variables...

Revert | Apply

Run | Close

Other IDEs works similarly.

Magic Squares:

An $n \times n$ matrix that is filled with the numbers 1, 2, 3, . . . , n^2 is a magic square if the sum of the elements in each row, in each column, and in the two diagonals is the same value. For example,

```
Magic Square of size 5x5:

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

Diagonal sum: 65
Column sum: 65
Row sum: 65
```

Following is an algorithm to construct magic n-by-n squares; it works only if n is odd. Place a 1 in the middle of the bottom row. After k has been placed in the (i, j) square, place $k + 1$ into the square to the right and down, wrapping around the borders. However, if the square to the right and down has already been filled, or if you are in the lower-right corner, then you must move to the square straight up instead.

Write a program whose input from command line is an odd number n and whose output is the magic square of order n. Implement a class MagicSquare with a constructor that constructs the square and a toString method that returns a string representation of the square. In Java, the method toString will be automatically executed when an object is converted into a string. This string is then printed nicely on the screen.

Program template:

//MagicSquare.java

```
package ddp2.tp02;


/**
A magic square that is constructed by a the "right and down" algorithm.
*/
public class MagicSquare
{
	private int[][] square;
	private int size;

	 /**
	Construct a MagicSquare object
	(precondition: s is odd).
	@param s the size of the square
	 */
	public MagicSquare(int s)
	{
		size = s;
		square = ....................................................;
```

```
        //your code here for filling in the square


    }

    /**
   Find the sum of the diagonal.
   @return sum: the sum of the diagonal
   */
    public int diagonalSum()
    {
        int sum = 0;

        // your code here

        return sum;
    }

    /**
     Add the numbers in a column of the square.
     @param i the column index
     @return the sum of the column
    */
    public int columnSum(int i)
    {
        int sum = 0;

        // your code here

        return ………………….;
    }

    /**
     Add the numbers in a row of the square.
     @param i the row index
     @return the sum of the row
    */
    public int rowSum(int i)
    {


        // your code here


    }
```

```
	/**
	 Gets a string representation of the contents of this square.
	 @return a string represenation of the square
	*/
	public String toString()
	{
		String r = "";

		// your code here: nested for-each loops

		return r;
	}
}


//TestMagicSquare.java

package ddp2.tp02;

public class TestMagicSquare {

    public static void main(String[] args) {

        int n = 0; //size of magic square

        //Process input from command-line
        if (args.length == 1) {

            // your code here

        }
        else {
            System.out.println("Usage: java -jar <jarFile> <odd size of square>");
            System.exit(1);
        }


        //Create an object of MagicSquare
        MagicSquare ms = new MagicSquare(n);

        // Print the results

        System.out.println("Magic Square of size " + n + "x" + n + ":\n\n" + ms +
            "\nMain diagonal sum: " + ms.diagonalSum() +
            "\nColumn sum: " +  ms.columnSum(0) +
            "\nRow sum: " + ms.rowSum(n-1) + "\n");

    }
}
```

Examples:

1)

```
C:\Kuliah\DDP2_2020> java -jar magicSquare.jar 9
Magic Square of size 9x9:

   37   48   59   70   81    2   13   24   35
   36   38   49   60   71   73    3   14   25
   26   28   39   50   61   72   74    4   15
   16   27   29   40   51   62   64   75    5
    6   17   19   30   41   52   63   65   76
   77    7   18   20   31   42   53   55   66
   67   78    8   10   21   32   43   54   56
   57   68   79    9   11   22   33   44   46
   47   58   69   80    1   12   23   34   45

Main diagonal sum: 369
Column sum: 369
Row sum: 369
```

2)

```
C:\Kuliah\DDP2_2020> java -jar magicSquare.jar
Usage: java -jar <jarFile> <odd size of square>
```

3)

```
C:\Kuliah\DDP2_2020> java -jar magicSquare.jar 5
Magic Square of size 5x5:

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

Main diagonal sum: 65
Column sum: 65
Row sum: 65
```

Challenge

You will get a **bonus** of **25** if your program can also construct magic squares of **even** order such as order 4, 6, 8, 10, 12, etc. You can refer to the algorithms in the paper “ON THE CONSTRUCTION OF EVEN ORDER MAGIC SQUARES” by ABDULLAHI UMAR.

For example, here is a magic square of order 4:

```
 1  5 12 16
15 11  6  2
14  8  9  3
 4 10  7 13
```

Happy Programming!
Selamat Mengerjakan!
'Met Ngoding! ☺
L.Y.Stefanus & the Asdos Team