Magic Squares

One interesting application of two-dimensional arrays is *magic squares*. A magic square is a square matrix in which the sum of every row, every column, and both diagonals is the same. Magic squares have been studied for many years, and there are some particularly famous magic squares. In this exercise you will write code to determine whether a square is magic.

File *Square.java* contains the shell for a class that represents a square matrix. It contains headers for a constructor that gives the size of the square and methods to read values into the square, print the square, find the sum of a given row, find the sum of a given column, find the sum of the main (or other) diagonal, and determine whether the square is magic. The read method is given for you; you will need to write the others. Note that the read method takes a Scanner object as a parameter.

File *SquareTest.java* contains the shell for a program that reads input for squares from a file named *magicData* and tells whether each is a magic square. Following the comments, fill in the remaining code. Note that the main method reads the size of a square, then after constructing the square of that size, it calls the *readSquare* method to read the square in. The readSquare method must be sent the Scanner object as a parameter.

You should find that the first, second, and third squares in the input are magic, and that the rest (fourth through seventh) are not. Note that the -1 at the bottom tells the test program to stop reading.

```
// ****************************************************************

// Square.java

//

// Define a Square class with methods to create and read in

// info for a square matrix and to compute the sum of a row,

// a col, either diagonal, and whether it is magic.

//

// ****************************************************************


import java.util.Scanner;


public class Square

{

    int[][] square;


    //--------------------------------------
```

```
//create new square of given size
//--------------------------------------
public Square(int size)
{

}

//--------------------------------------
//return the sum of the values in the given row
//--------------------------------------
public int sumRow(int row)
{

}

//--------------------------------------
//return the sum of the values in the given column
//--------------------------------------
public int sumCol(int col)
{

}

//--------------------------------------
//return the sum of the values in the main diagonal
//--------------------------------------
public int sumMainDiag()
{

}
```

```
//--------------------------------------
//return the sum of the values in the other ("reverse") diagonal
//--------------------------------------
public int sumOtherDiag()
{

}


//--------------------------------------
//return true if the square is magic (all rows, cols, and diags have
//same sum), false otherwise
//--------------------------------------
public boolean magic()
{

}


//--------------------------------------
//read info into the square from the input stream associated with the
//Scanner parameter
//--------------------------------------
public void readSquare(Scanner scan)
{
  for (int row = 0; row < square.length; row++)
    for (int col = 0; col < square.length; col ++)
      square[row][col] = scan.nextInt();
}


//--------------------------------------
//print the contents of the square, neatly formatted
//--------------------------------------
```

```
    public void printSquare()

    {



    }



}
```

```
// ****************************************************************
// SquareTest.java
//
// Uses the Square class to read in square data and tell if
// each square is magic.
//
// ****************************************************************

import java.util.Scanner;

public class SquareTest
{
    public static void main(String[] args) throws IOException
    {
      Scanner scan = new Scanner(new File("magicData"));

      int count = 1;                 //count which square we're on
      int size = scan.nextInt();     //size of next square

      //Expecting -1 at bottom of input file
      while (size != -1)
          {

            //create a new Square of the given size

            //call its read method to read the values of the square

            System.out.println("\n******** Square " + count + " ********");
            //print the square

            //print the sums of its rows
            //print the sums of its columns


            //print the sum of the main diagonal


            //print the sum of the other diagonal


            //determine and print whether it is a magic square


            //get size of next square

            size = scan.nextInt();


          }


    }

}
```

magicData

```
3

8  1  6

3  5  7

4  9  2

7

30  39  48   1  10  19  28

38  47   7   9  18  27  29

46   6   8  17  26  35  37

5   14  16  25  34  36  45

13  15  24  33  42  44   4

21  23  32  41  43   3  12

22  31  40  49   2  11  20

4

48   9   6   39

27  18  21   36

15  30  33   24

12  45  42    3

3

6  2  7

1  5  3

2  9  4

4

3  16   2  13

6   9   7  12

10  5  11   8

15  4  14   1

5

17  24  15   8   1

23   5  16  14   7
```

```
 4   6  22  13  20

10  12   3  21  19

11  18   9   2  25

7

30  39  48   1  10  28  19

38  47   7   9  18  29  27

46   6   8  17  26  37  35

5   14  16  25  34  45  36

13  15  24  33  42   4  44

21  23  32  41  43  12   3

22  31  40  49   2  20  11

-1
```