Basis of Computer Programming (Java A)
Lab Exercise 6

[Experimental Objective]

- Learn how to use static method
- Learn how to use static method from other class
- Learn how to use method overloading
- Learn how to use two dimensional arrays
- Learn to develop and invoke methods with array arguments and return values.

[Exercises]

1. Create a class named ***MyTriangle*** that contains two static methods
   - **a)** **public static double area(double a, double b, double c)**
   - **b)** **public static double perimeter(double a, double b, double c)**

   to compute area and perimeter of a triangle respectively given three *valid* sides ***a***, ***b*** and ***c***.

   And add a static method

   ```
   /** Return true if the sum of any two sides is greater than the third side.
   **/
   ```

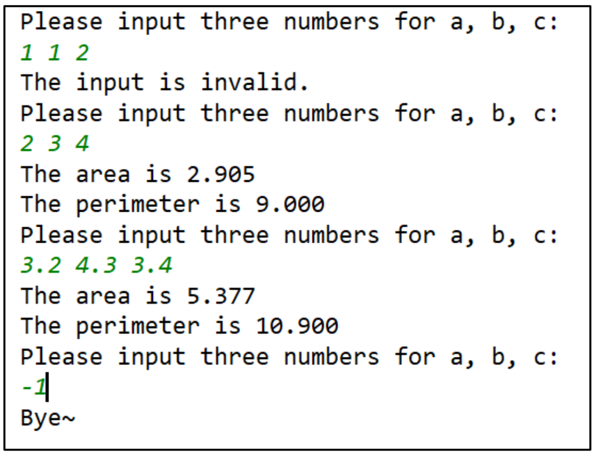
   - **c)** **public static boolean isValid(double a, double b, double c)**

   In the main method of ***MyTriangle***, test the three methods you write.

   1) Get ***a***, ***b*** and ***c*** from the Console
   2) If ***a*** is -1, exit your program and print **"Bye~"**
   3) If ***a*** is not -1, use ***isValid*** to check the input
   4) If the input is valid, compute the area and perimeter and print them
   5) If the input is not valid, return false and print **"The input is invalid."**
   6) Go to **1)**

   *Tips: To call a method in the same class, you can try **method_name**( ).*

   Sample:



```
Please input three numbers for a, b, c:
1 1 2
The input is invalid.
Please input three numbers for a, b, c:
2 3 4
The area is 2.905
The perimeter is 9.000
Please input three numbers for a, b, c:
3.2 4.3 3.4
The area is 5.377
The perimeter is 10.900
Please input three numbers for a, b, c:
-1
Bye~
```

2. In the ***MyTriangle*** class created in Exercise 1, add two another static overloaded methods

   a) **public static double area(double bottom, double height)**

   b) **public static double area(double a, double b, int angleOfAandB)**

   to compute the area.

   The **a)** method is to compute area by ***bottom*** and ***height***:

   area = 1/2 * ***bottom*** * ***height***

   And the **b)** method is to compute area by two sides ***a***, ***b*** and the angle between the two sides(***angleOfAandB***)

   area = 1/2 * ***a*** * ***b*** * sin(***angleOfAandB***)

   Then create another class ***Lab6E2*** that contains the main method.
   In the main method:
   1) Read ***bottom*** and ***height*** from the Console to compute area by calling the corresponding method you created in ***MyTriangle***;
   2) Read two sides ***a***, ***b*** and ***angleOfAandB*** from the Console to compute area by calling the corresponding method you created in ***MyTriangle***.

   Tips: To call a **static** *method in another class* ***class_name*** *under the same file directory, you can try* ***class_name.method_name****( ).*

   Sample:

```
Please input two numbers for bottom and height:
4 5.6
The area is 11.200
Please input two numbers for a and b:
3 5.6
Please input a number in (0, 180) for angle (angle is a float variable):
55
The area is 6.881
```

3. Write a program to get students' grades of many courses and print scores and average scores in a grade table.

   1) Get the number of students ***sNum***(less than 10) and the number of courses ***cNum***(less than 10) from the console

   2) Get scores from console. The scores are inputted by ***cNum*** lines. In each line, there are ***sNum*** students' scores of one course in order.

   3) Print a grade table of ***cNum***+2 columns and ***sNum***+2 rows , of which the first row are course names, the first column are student names, the last row are the average scores of each course and the last column are the average scores of each students.

   Sample:

```
Please enter the number of subjects: 3
Please enter the number of students: 4
32 44 52 32
89 92 80 94
11 22 32 23

          Course1    Course2    Course3    Average
Student1     32         89         11       44.00
Student2     44         92         22       52.67
Student3     52         80         32       54.67
Student4     32         94         23       49.67
Average     40.00      88.75      22.00
```

4. Write a program to calculate the product of n matrixes.

   1) Get ***n*** (number of matrixes, less than 5) from console

   2) Get ***n*** matrixes from console. For each matrix, the first line is ***Rows*** (number of rows, less than 10) & ***Cols*** (number of columns, less than 10) of the matrix, and then entries of the matrix

   3) Print the product (Each entry of the product should be mod 100)

   Sample:

```
3
3 5
6 0 8 4 9
0 1 0 7 6
9 6 9 9 2
5 1
0
4
9
2
6
1 3
4 2 7

36 68 38
16 8  78
40 70 45
```

```
3
3 3
7 2 0
4 2 3
4 5 5
3 4
3 4 3 3
3 2 1 3
7 6 3 0
4 3
2 6 0
2 5 2
6 0 6
1 8 0

83 38 2
10 68 14
55 68 4
```

5. Sudoku is a famous mathematical game in which players fill numbers 1-9 in a 9×9 square. The square satisfies that every row and every column contain 1-9 only once. Specially, the square is divided into 9 sub-squares (as shown below), and every sub-squares also contains 1-9 only once. Following is a valid Sudoku square (Notice that the sub-squares are separated by red lines).

2	9	3	7	1	5	4	8	6
8	6	1	2	4	9	5	3	7
7	4	5	8	6	3	1	9	2
6	7	8	9	2	1	3	4	5
1	3	9	5	7	4	2	6	8
4	5	2	6	3	8	7	1	9
9	2	4	3	8	7	6	5	1
3	8	6	1	5	2	9	7	4
5	1	7	4	9	6	8	2	3

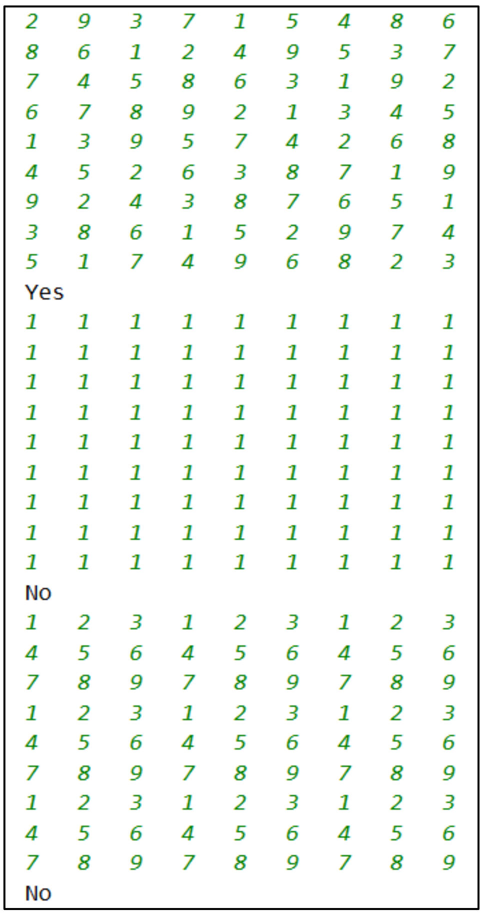
Write a program to judge whether a 9×9 square is a Sudoku square.

1) Get a 9×9 square from console

2) If it is a Sudoku square, print **"Yes"**

3) If it is not a Sudoku square, print **"No"**

Sample:



```
2    9    3    7    1    5    4    8    6
8    6    1    2    4    9    5    3    7
7    4    5    8    6    3    1    9    2
6    7    8    9    2    1    3    4    5
1    3    9    5    7    4    2    6    8
4    5    2    6    3    8    7    1    9
9    2    4    3    8    7    6    5    1
3    8    6    1    5    2    9    7    4
5    1    7    4    9    6    8    2    3
Yes
1    1    1    1    1    1    1    1    1
1    1    1    1    1    1    1    1    1
1    1    1    1    1    1    1    1    1
1    1    1    1    1    1    1    1    1
1    1    1    1    1    1    1    1    1
1    1    1    1    1    1    1    1    1
1    1    1    1    1    1    1    1    1
1    1    1    1    1    1    1    1    1
1    1    1    1    1    1    1    1    1
No
1    2    3    1    2    3    1    2    3
4    5    6    4    5    6    4    5    6
7    8    9    7    8    9    7    8    9
1    2    3    1    2    3    1    2    3
4    5    6    4    5    6    4    5    6
7    8    9    7    8    9    7    8    9
1    2    3    1    2    3    1    2    3
4    5    6    4    5    6    4    5    6
7    8    9    7    8    9    7    8    9
No
```