# 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)

Tip: Given three valid sides *a*, *b*, *c* of a triangle, the area of the triangle can be computed using the Heron's formula:

$$A = \sqrt{s(s-a)(s-b)(s-c)}$$

In the formula, s is the semi perimeter of the triangle, i.e., s = (a + b + c)/2.

In MyTriangle.java, you can see that we can directly call the static method **perimeter()** from the method **main()** just by using the method name. This is because **perimeter()** and **main()** are declared in the same class.

## 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()** (here please put MyTriangle.java and Lab6E2.java in the same folder).

#### 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 an int variable):
55
The area is 6.881
```

3. Enter an integer *n*, please output the *n*th term of Fibonacci sequence. (starting from 0, the 0th term is 0)

The Fibonacci sequence is defined as follows:

$$f(0) = 0$$
  
 $f(1) = 1$   
 $f(n) = f(n-1) + f(n-2)$ , when  $n >= 2$ 

Sample:

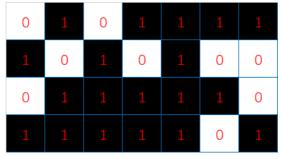
*30* 832040

4. Given a chessboard, 1 represents a black grid and 0 represents a white grid. If a grid is white and the top, bottom, left, and right grids of it are black, we call this grid a "bingo" grid. Please write a method:

public static boolean check(int[][] board, int row, int column)

\*board is the chessboard, board[row][column] is the target grid

to determine whether a grid is a bingo grid. Use this method to calculate how many bingo grids are on the board and output the result.



Sample: