CS102A Introduction to Computer Programming

Fall 2020

Lab 6

Objectives

1. Learn how to use static methods.
2. Learn how to overload methods.
3. Learn how to use two-dimensional (2D) arrays.
4. Learn how to invoke methods with array arguments and get the returned values.

1 Exercises

1.1 Exercise 1

Create a class named `MyTriangle` that contains the following two static methods:

```
public static double area(double a, double b, double c)
```

```
public static double perimeter(double a, double b, double c)
```

These will be used to compute the area and perimeter of a triangle, respectively, given three valid sides `a`, `b`, and `c`. Next, add the following static method:

```
/** Return true if the sum of any two sides is greater than the
   third side. **/
public static boolean isValid(double a, double b, double c)
```

In the `main` method of `MyTriangle`, test the three methods you wrote as follows:

1. Get `a`, `b`, and `c` from the console.
2. If `a` is `-1`, your program should print *Bye!* and exit.
3. If `a` is not `-1`, use `isValid` to verify the input.
4. If the input is valid, compute and print the area and perimeter.
5. If the input is not valid, print *Invalid input!* and return `false`.
6. Go to step 1.

> **Tip**
> To invoke a method in the same class, you can simply call `method_name()`.

Sample input and output:

```
Please input three numbers for a, b, c:
1 1 2
Invalid input!
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!
```

1.2 Exercise 2

In the `MyTriangle` class created in Exercise 1, add two more static overloaded methods:

```
public static double area(double bottom, double height)
```

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

These will be used to compute the area of a triangle; the first method will do so using the bottom and height as: `area` $= 1/2 \times$ `bottom` $\times$ `height`. Instead, the second method will compute the area using the two sides `a`, `b` and the angle between them (`angleOfAandB`): `area` $= 1/2 \times$ `a` $\times$ `b` $\times \sin($`angleOfAandB`$)$.

Next, create another class named `Lab6E2` that contains a `main` method. In the main method:

1. Read `bottom` and `height` from the console to compute the area of a triangle by calling the corresponding method you created in `MyTriangle`.

2. Read `a`, `b`, and `angleOfAandB` from the console to compute the area of a triangle by calling the corresponding method you created in `MyTriangle`.

> **Tip**
> To invoke a static method in another class `class_name` under the same file directory, you can use `class_name.method_name()`.

Sample input and output:

```
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
```

1.3 Exercise 3

Write a program to get students' grades from their courses and then print the individual and average scores for each student in a grade table. Specifically:

1. Prompt the user to enter the number of students and courses (both less than 10).
2. Prompt the user to enter the course scores. Read one line per course, with each line containing the students' space-separated scores for that course.
3. Print a grade table. The first row shows the course names, while the first column shows the student names. The last row shows the average scores for each course and the last column shows the average scores for each student.

Sample input and output:

```
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
```

1.4 Exercise 4

Write a program that calculates the product of n matrices.

1. Read the number of matrices as input from the user.
2. Read the elements of all the matrices as input from the user. Before the elements of each matrix, the user should input the number of rows and columns for that matrix.

3. Print the result.

Sample input and output:

```
Please enter the number of matrices: 3
Enter the number of row and column of matrix 1: 3 5
Enter the elements of the matrix:
6 -7 3 -5 1
0  4 8  2 3
3 -2 1 -7 2
Enter the number of row and column of matrix 1: 5 1
Enter the elements of the matrix:
0
9
-3
4
1
Enter the number of row and column of matrix 1: 1 3
Enter the elements of the matrix:
-1 3 9
The results:
91   -273 -819
-23  69   207
47   -141 -423
```

1.5 Exercise 5

Sudoku is a famous mathematical game in which players fill in a 9×9 square with integers in $\{1, \ldots, 9\}$. Every row and column must contain every integer in $\{1, \ldots, 9\}$ exactly once. The square is divided into nine 3×3 subsquares, where each subsquare also contains every integer in $\{1, \ldots, 9\}$ exactly once. Write a program that asserts whether a square is a Sudoku square.

1. Read a 9×9 square from the console.

2. If it is a Sudoku square, print *Yes*.

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

Sample input and output:

```
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
```