Tutorial of multi-dimensional array and static methods

> Based on the tutorial of "2020F-Java-A" designed by teaching group in SUSTech
>
> Modified (mainly change to markdown file) by ZHU Yueming in 2021. March. 22th
>
> Add "before exercise" by ZHU Yueming in 2021. Oct. 18th
>
> Minor changes by Yida Tao, Oct. 14, 2022

Experimental Objectives

1. Learn how to use static methods and method overloading.
2. Learn how to use two-dimensional arrays.

Before Exercise

Run the following code and observe the results.

```
int[][] arr = new int[3][];
arr[0] = new int[]{1, 2, 3};
arr[1] = arr[0];
arr[2] = new int[]{3, 4, 5, 6};

System.out.println(Arrays.toString(arr[1]));
System.out.println(Arrays.toString(arr));
System.out.println(Arrays.deepToString(arr));
```

Try the following three `foreach` methods and think: whether the values change or not.

```
for (int e : arr[1]) {
    System.out.println(e);
    e = 0;
    System.out.println(e);
}
for (int[] a : arr) {
    System.out.println(a);
    a = new int[2];
}
System.out.println(Arrays.deepToString(arr));

for (int[] a : arr) {
    for (int i = 0; i < a.length; i++) {
        a[i]++;
    }
}
System.out.println(Arrays.deepToString(arr));
```

Exercises

Exercise 1:

Create a class named `MyTriangle`. The class contains two static methods `area` and `perimeter`, which compute the area and perimeter of a triangle respectively, given three valid sides `a`, `b` and `c`.

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

The class also contains a static method `isValid`, which checks the validity of input `a`, `b`, `c`.

```
// return true if the sum of any two sides is greater than the third side.
otherwise return false
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 input and output

```
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~
```

Exercise 2:

In the `MyTriangle` class created in previous exercise, add another two static overloaded methods to compute the area.

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

There are two ways to compute the area:

1. compute area by bottom and height: area = 1/2 * bottom * height
2. 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, which

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 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
```

Exercise 3:

Write a program to get students' grades from their courses and then print the scores and average scores in a grade table.

1. Prompt the user to enter the number of students (< 10) and the number of courses (< 10).
2. Prompt the user to enter the course scores for each student. The scores from different courses are entered on separate lines. On each line, there are scores for a course for each student.
3. Print a grade table. The first row shows the course names and the first column shows the student names. The last row shows the average scores of each course and the last column shows the average scores of each students.

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
```

Exercise 4:

Write a program to calculate the product of n matrices.

1. Read the number of matrices from user.
2. Read the elements of all the matrices from user. Before inputing the elements of each matrix, the user should input the rows and columns of that matrix.
3. Print the result.

```
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
```

Exercise 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. In addition, the square is divided into 9 subsquares, and every subsquares also contains 1–9 only once. Write a program to judge whether a 1–9 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 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
```

Advanced problem: Given an incomplete board, write a program to solve a Sudoku puzzle by filling the empty cells. Please check here.