CSC1003 Assignment 1

## Access Code: 67tmo#c7Xz

## Important Notes:

1. The assignment is an individual project, to be finished on one’s own effort.
2. The work must be submitted before 6pm Oct. 17, 2023 (Tuesday), Beijing Time. This is a firm deadline. No late submissions are accepted.
3. Plagiarism is strictly forbidden, regardless of the role in the process. Notably, ten consecutive lines of identical codes are treated as plagiarism. Depending on the seriousness of the plagiarism, 30%-100% marks will be deducted.
4. This assignment is supposed to be finished after learning arrays. Meanwhile, a draft may be released earlier for students’ preparation purpose.
5. Please let the teaching team know for any ambiguity or incorrectness in this draft.

## Marking Criterion:

1. The full score of the assignment is 100 marks.
2. Two java programs are to be submitted. Each program will be evaluated with several unseen test cases. A submission obtains the full score if and only if both programs pass all test cases.

## Running Environment:

1. The submissions will be evaluated in the OJ system running Java SDK. It is the students’ responsibility to make sure that his/her submissions are compatible with the OJ system.
2. The submission is only allowed to import four packages of (java.lang.\*; java.util.\*; java.math.\*; java.io.\*) included in Java SDK. No other packages are allowed.
3. All students will have an opportunity to test their programs in the OJ platform prior to the official submission.

## Submission Guidelines:

1. Inconsistency with or violation from the guideline leads to marks deduction.
2. All students are reminded to read this assignment document carefully and in detail. No argument will be accepted on issues that have been specified in this document.

## Program One:

Fibonacci series is defined by:

1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, …

Represented by an array, it would be: a[0] = 1, a[1] = 1, a[2] = 2, …, a[n] = a[n-1] + a[n-2].

Write a Java program named “**TestFibonacci.java**” which reads two non-negative numbers from the **console by using System.in**. Denote the two numbers by *n* and *d*, and the program is expected to output *d* elements of the series from **n-th** element (i.e. *a[n-1]*) in the reverse order to the **console** by using **System.out**. An example is given below. (You can assume that the numbers are no larger than *100000*).

|  |  |
| --- | --- |
| An example of console input | Expected console output |
| 3  5, 3  7, 0  7, 7 | 5, 3, 2  13, 8, 5, 3, 2, 1, 1 |

Note:

1. Each line of the input file consists of a positive integer, a comma, a space, and a non-negative number. The output is expected to be a series of numbers, separated by a comma and a space. If there is no number to output, the output is an empty line. We guarantee that the input is valid.
2. The first line of each test case input will be an integer N (In the example above, N equals to 3). This integer N represents the number of lines of data to be input next. This part of the code is already given in the template, please refer to the program template section below.

## Program Two:

Write a java program named “TestMathExpr.java” to evaluate mathematical expressions with two operands (nonnegative integers) and one operator (+, -, \*, or /).

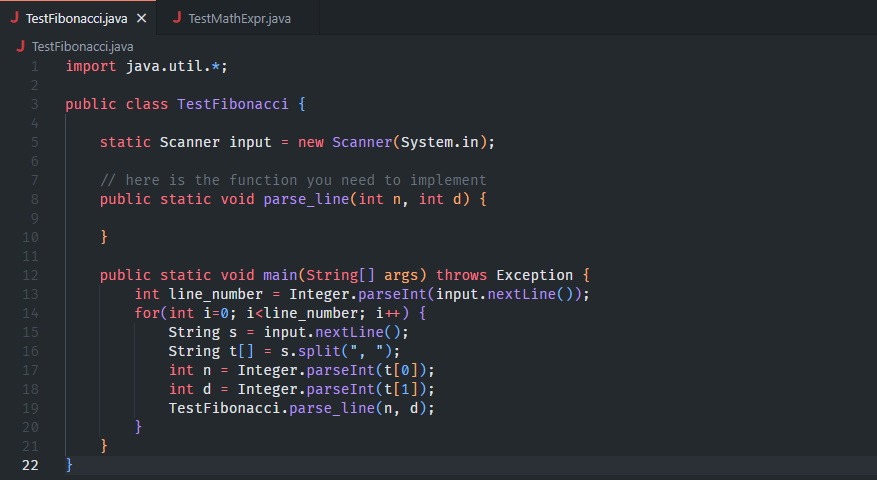
|  |  |
| --- | --- |
| An example of console input | Expected console output |
| 7  3 + 2  4 \* 4  3 - 5  155 / 5  157 / 5  158 / 5  0 + 7 | 5  16  -2  31  31  31  7 |

Note:

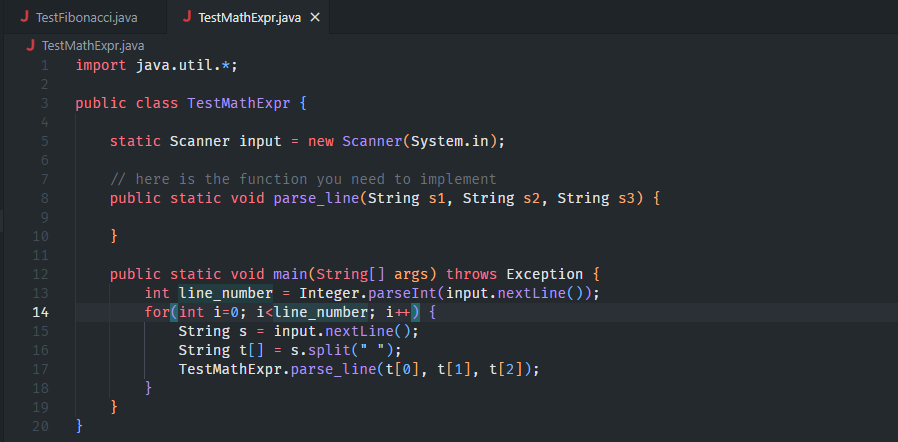
1. Each row of the console input consists of a non-negative integer, a space, an operator, a space, and another non-negative integer. The output is an integer (**round down**).
2. The first line of each test case input will be an integer N (In the example above, N equals to 7). This integer N represents the number of lines of data to be input next. This part of the code is already given in the template, please refer to the program template section below.

## Program Template:

**Program 1:**



**Program 2:**



1. Each test case on OJ may have multiple lines of input, but in these templates, you only need to complete the implementation of the method “**parse\_line()**” , which processes each line of input. Please refer to the template of “TestFibonacci.java” and “TestMathExpr.java” above carefully.
2. These templates are just references, which means you don't need to write code based entirely on them. You can change any line of code in the sample file, or even write a new file from scratch, but you have to meet the required functions defined above.
3. Students can submit their code to the OJ system at most ten times, but only the last submission is used for scoring. This means that if your first submission is already excellent, then the next submission will potentially pull your score down.