ARITHMETIC-SOLVER

**SUMMARY**

**Time Taken** : Around 6 – 8 Hours (Total cumulative).

**Link** : <https://github.com/cryofoton/arithmetic-solver>

**Problem Statement**

Given the following method:

public static double calculate(String sum) {

// Your code starts here

}

The code must return correct result for below expression: -

1. 1 + 1
2. 2 \* 2
3. 1 + 2 + 3
4. 6 / 2
5. 11 + 23
6. 11.1 + 23
7. 1 + 1 \* 3
8. ( 11.5 + 15.4 ) + 10.1
9. 23 – ( 29.3 – 12.5 )
10. 10 – ( 2 + 3 \* (7 – 5 ) )

**Condition**

1. String parameter always consists of numbers and operators, and optionally brackets.
2. Each number, operator and bracket are separated by spaces.

**Discussion**

This program is coded with Java 11 as compiler, although it can be compiled and run with Java 8 without any issues. It consists of two methods; calculate methods to process the expression string, and main method for invoking the calculate method for each expression and print the outputs.

The flow for the calculate method is as follows;

* Trim spaces, leading and trailing brackets if exist.
* Convert the expression string into array by splitting it with spaces as delimiter.
* Handle the brackets first as it’s the highest priority.   
  In order to process it, we will iterate through the expression array and add each item into a List, while at the same time, we take note of the opening bracket index and push it into a Stack. When we encounter a closing bracket, we will pull back the last opening index from the Stack, create a substring from the List, based of those opening and closing index and recursively call calculate method to process it. The result of recursive call will replace the bracketed expression in the List. In the end, we will get a List of string that need to be processed clear of all the bracketed expression.
* Begin processing the expression by handling the multiplication and division.  
  The List of string resulted from previous step will be iterated and calculated if it’s a multiplication and division operation. The result of calculation, along with other uncalculated string will be pushed into a Queue for next processing.
* Process the remaining expression by handling addition and subtraction.  
  The Queue from previous step will be processed one by one and calculated.
* Return the final calculated result back to the method caller.

**Limitation**

* This program supports +, -, \*, and / operator only.
* This program doesn’t check for validity of given expression, and may throw Exception or return unexpected results if the expression is not spaced out properly, unbalanced opening and closing brackets, or found out to be invalid.

By: Osman Sulaiman

Last Update: 11/09/2022