

Project: Postfix Notation

Introduction: In this assignment, you will use java pre-defined data structure Stack to implement (1) an infix to postfix converter; and (2) a postfix expression calculator.

Files information:

- **input.txt** contains the infix expressions.
 - ❖ Each line contains one infix expression.
 - ❖ There could be white spaces between tokens (operators, numbers, and parentheses), before or after an expression.
 - ❖ To simply the problem, only integer operands are allowed in these expressions.
 - ❖ The acceptable operators are '+', '-', '*', and '/'. Parentheses '(' and ')' are also allowed. Note (1) '*' and '/' have higher precedence than '+' and '-'; (2) '*' and '/' have the same level precedence; and (3) '+' and '-' have the same level precedence.
- **Project1.java** contains the structure of the program. It has four functions
 - ❖ `public static void main(String[] args)`, which controls the flow of the program. This function has been written for you. Do not make changes to it.
 - ❖ `public static void addSpaces(String inputfile, String outputfile)`, which reads infix expression from inputfile and add white space before the after each operator. The new expressions are saved in outputfile. The reason to add spaces to infix expressions is to make it easy to separate tokens later. This function has been written for you. Do not make changes to it.
 - ❖ `public static void toPostFix(String inputfile, String outputfile)`, which reads infix expression from inputfile and converts it to postfix expression and saves it to outputfile.
 - ❖ `public static void evaluateExp(String inputfile)`, which reads and evaluates postfix expressions from inputfile and displays the result on screen.

Your task: Complete the missing code in two functions: `toPostFix(String, String)` and `evaluateExp(String)`

Note:

- No error checking is needed. Assume all infix expressions are legal.
- You only need to deal with integer operands. No floating-point number will be given. But the intermediate and final result could be a floating point number.
- If you need to create additional functions, declare and implement them in Project1.java.
- Please use java pre-defined Stack class to implement these two functions.
- You might want to comment your code clearly to get partial credit if your program could not work correctly.
- This is a team-homework. At most three students could work together. Please add your team member names at the beginning of the Project1.java and only one submission is needed for the team.

Submission: (1) **Project1.java**, (2) **output.txt**, and (3) **screenshot** of the output running the program.

Your output.txt file should look like	Your screen display should look like
12 8 + 10 -	10.0
10 4 2 * /	1.25
12 8 + 10 - 5 / 2 *	4.0
12 8 + 10 - 5 2 * /	1.0
12 8 10 5 / - 2 * +	24.0
3 3 + 3 / 3 4 * + 28 /	0.5
2 5 * 8 +	18.0
4 2 3 + 8 16 / * +	6.5
300 6 5 * + 8 5 - 2 * -	324.0
121 6 2 * 4 - + 82 -	47.0