Evaluation of Postfix Expressions

Do Now

What is a prefix expression?

What is an infix expression?

What is a postfix expression?

Infix expression

An operator is written in between two operands.

Example: 4 * 10

Prefix Expression

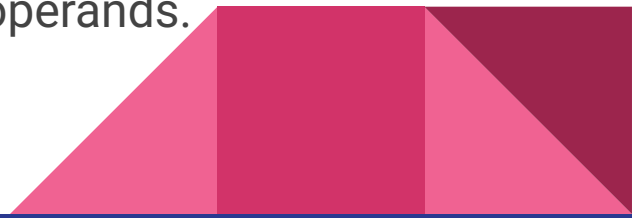
It requires that all operators precede the two operands that they work on.

Example: * 4 10

Postfix Expression

In this type of expression an operator is written after its operands.

Example: 4 10 *



Specifications to design your calculator to evaluate postfix expressions

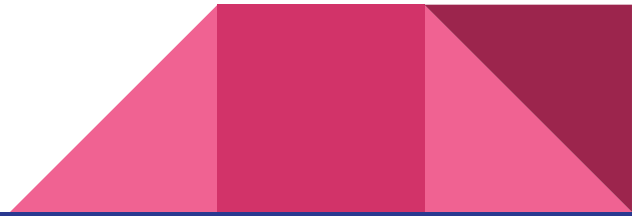
- Operands could be valid numbers (int or double). For our calculator let's use double numbers.

- Valid operators:

  Add (+) Subtract (-)

  Multiply (*) Divide (/)

  Remainder (%)



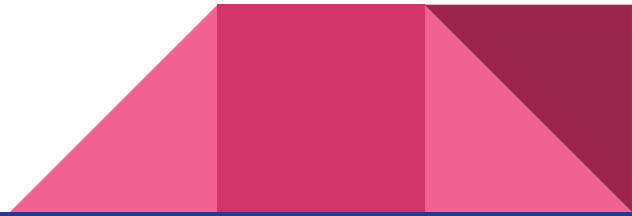
Specifications to design your calculator to evaluate postfix expressions

- The operators work on 2 values (4 10 *)
- The expressions we are going to evaluate are strings and all operands and operators are separated by a single space.

  Examples:

  “10 2.5 /” “8 2 + 99 9 - * 2 + 9 -”

  “4 5 - 2 + 1.5” “2 4 6 8 10 + * - -”

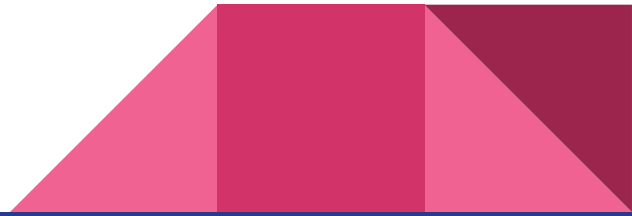


Group discussion

Let’s think on an algorithm that can help us evaluate a postfix expression.

Consider the following questions:

- How can you read postfix expression (string)?
- You must use a data structure to evaluate the expression, which one would you use?
- Explain how your algorithm would work.

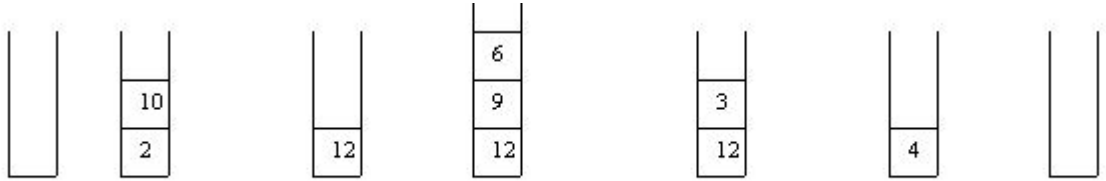


Would your algorithm work for the following postfix expressions?

Postfix Expression	Infix Equivalent	Result
4 5 7 2 + − ×	4 × (5 − (7 + 2))	−16
3 4 + 2 × 7 /	((3 + 4) × 2)/7	2
5 7 + 6 2 − ×	(5 + 7) × (6 − 2)	48
4 2 3 5 1 − + × + ×	? × (4 + (2 × (3 + (5 − 1))))	not enough operands
4 2 + 3 5 1 − × +	(4 + 2) + (3 × (5 − 1))	18
5 3 7 9 + +	(3 + (7 + 9)) ... 5???	too many operands

Could a stack work?

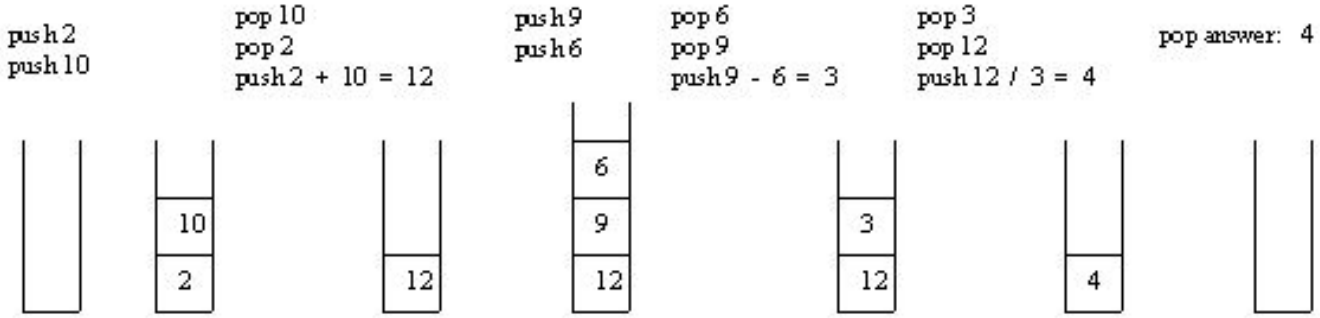
Postfix Expression: **2 10 + 9 6 - /**



- What happens when there is an operand in the expression?
- What happens when there is an operator in the expression?
- Is there a way to know if there are too many or too few operands/operators?

Postfix expression using a stack

Expression: **2 10 + 9 6 - /**



Stack Calculator - Implementation

- Save your work here:
  .../APCSA_1/apcsa-assignments-YourUsername/classwork/42_stack_calculator/Calculator.java
- Write the method eval(String expression)

```
public class Calculator{

   // Evaluate a postfix expression stored in expression
   public static double eval(String expression){
     return 0.0;
   }

}
```

- String expression: Contains ints, doubles, and operators (+ - / * and %) separated by one space
- You must use an ArrayDeque<Double> to store the values. It will act as a stack.
- Return a double value
- Throw an IllegalArgumentException when there are too many or too few operands/operators.

Some Examples

https://github.com/novillo-cs/apcsa_material/blob/main/classwork/33_postfix_expressions/examples.md