Airwallex

Payments re-imagined

A good piece of code works, but a brilliant one is like poetry, innovative, logical, well structured and crafted with passion.

We'd appreciate your effort if you could show us the aesthetics of programming, and we are expecting to see:

1. A well designed **OO model**;
2. **Production level** software engineering practices with tests and build script included;
3. The correct usages of **software design patterns**;
4. Your understanding of **SOLID principles**.

Airwallex

Background

Some of the best calculators in the world have an ‘RPN’ (reverse polish notation) mode.

Please implement a command-line RPN calculator as your code challenge.

However, We still feel it is worth to give you a big picture of the future requirments and hoping you can take them into consideration when design your solution.

Ultimately, we woud like to build an online tool which is powerful and highly flexiable. The work has been divided into 3 stages and **ONLY STAGE 1** is your code challenge. Detailed requirement can be found in next page. Below is to give you some ideas of the future requirement only

Stage 1: A command-line based RPN

- **Basic math operators (+, - ,*, /, sqrt)**
- **Undo and clear functionality**

Stage 2: Enhanced functionalities

- More math operators, such as n!, COS, ATAN, etc
- Other operators such as redo, swap, etc.

Stage 3: Online & UI

- Bring it online
- Customize the colour of each number in the stack depends on it position.

Note: please only implement requirements in **STAGE 1** as your code challenge. No need to implemente stage 2 and 3.

Please ensure your solution is maintainable, extensiable and testable.

Airwallex

Programming Exercise – RPN Calculator

We would like you to write a command-line based RPN calculator.

Requirements

The calculator has a stack that can contain real numbers.

• The calculator waits for user input and expects to receive strings containing whitespace separated lists of numbers and operators.

• Numbers are pushed on to the stack. Operators operate on numbers that are on the stack.

• Available operators are +, -, *, /, sqrt, undo, clear.

• Operators pop their parameters off the stack, and push their results back onto the stack.

• The 'clear' operator removes all items from the stack.

• The 'undo' operator undoes the previous operation. "undo undo" will undo the previo us two operations.

• sqrt performs a square root on the top item from the stack.

• The '+', '-', '*', '/' operators perform addition, subtraction, multiplication and division respectively on the top two items from the stack.

• After processing an input string, the calculator displays the current contents of the stack as a space-separated list.

• Numbers should be stored on the stack to at least 15 decimal places of precision, but displayed to 10 decimal places (or less if it causes no loss of precision).

• All numbers should be formatted as plain decimal strings (ie. no engineering formatting).

• If an operator cannot find a sufficient number of parameters on the stack, a warning is displayed:

operator <operator> (position: <pos>): insufficient parameters

• After displaying the warning, all further processing of the string terminates and the current state of the stack is displayed.

Deliverables

Please solve the problem in Java or Kotlin. The solution submitted should be in PRODUCTION quality, which contains

- Source code
- Unit tests
- Build script (prefer gradle or maven)

In order to get around firewall issues we recommend the solution be packaged as a

password protected zip file or Github repository link.password protected zip file or Github repository link.

NO need to include artifacts or packaged class files.

NO need to provide any graphical interface.

Airwallex

Examples

Example 1
5 2
stack: 5 2

Example 2
2 sqrt
stack: 1.4142135623
clear 9 sqrt
stack: 3

Example 3
5 2 -
stack: 3
3 -
stack: 0
clear
stack:

Example 4
5 4 3 2
stack: 5 4 3 2
undo undo *
stack: 20
5 *
stack: 100
undo
stack: 20 5

Example 5
7 12 2 /
stack: 7 6
*
stack: 42
4 /
stack: 10.5

Example 6
1 2 3 4 5
stack: 1 2 3 4 5
*
stack: 1 2 3 20
clear 3 4 -
stack: - 1

Example 7
1 2 3 4 5
stack: 1 2 3 4 5
* * * *
stack: 120

Example 8
1 2 3 * 5 + * * 6 5
operator * (position: 15): insucient parameters
stack: 11
(the 6 and 5 were not pushed on to the stack due to the previous error)

Airwallex About

Airwallex is...

Payments Re-imagined

Using advanced technology to deliver seamless end to end solutions that transcend borders and industries; we relentlessly challenge the industry for the better of our customers, creating opportunity without exception.

Supported by top-tier investors

Tencent 腾讯 SEQUOIA SquarePeg capital mastercard

BCA HILLHOUSE Horizons Ventures 維港投資 GOBI PARTNERS