CALCULATOR

Alican Balık

Abstract

There are many calculator programs using Reverse Polish Notation (RPN). RPN represents expressions in which the operator symbol is placed after the arguments being operated on. Polish notation, in which the operator comes before the operands, was invented in the 1920s by the Polish mathematician Jan Lukasiewicz. In the late 1950s, Australian philosopher and computer scientist Charles L. Hamblin suggested placing the operator after the operands and hence created reverse polish notation. It is a console application where program reads from input, line by line, interprets commands and prints the result on the console.

Introduction

A calculator is a device that performs arithmetic operations on numbers. The simplest calculators can do only addition, subtraction, multiplication, and division. There are many calculator programs using Reverse Polish Notation (RPN). RPN represents expressions in which the operator symbol is placed after the arguments being operated on. Reverse Polish notation (RPN), also known as Polish postfix notation or simply postfix notation, is a mathematical notation in which operators follow their operands, in contrast to Polish notation (PN), in which operators precede their operands. It does not need any parentheses as long as each operator has a fixed number of operands. The first computers to implement architectures enabling reverse Polish notation were the English Electric Company's KDF9 machine, which was announced in 1960 and delivered (i.e. made available commercially) in 1963, and the American Burroughs B5000, announced in 1961 and also delivered in 1963. One of the designers of the B5000, Robert S. Barton, later wrote that he developed reverse Polish notation independently of Hamblin sometime in 1958 after reading a 1954 textbook on symbolic logic by Irving Copi, where he found a reference to Polish notation, which made him read the works of Jan Lukasiewicz as well, and before he was aware of Hamblin's work. Designed by Robert "Bob" Appleby Ragen, Friden introduced reverse Polish notation to the desktop calculator market with the EC-130 supporting a four-level stack in June 1963. The successor EC-132 added a square root function in April 1965. Around 1966, the Monroe Epic calculator supported an unnamed input scheme resembling RPN as well.

Our calculator is a console application which is implemented on Java. It is a string calculator where users are allowed to type arithmetic signs in letter instead of symbols. Available arithmetic signs are written in Lexemes section.

Alphabet

A digit is an element of a set that, taken as a whole, comprises a system of numeration. Thus, a digit is a number in a specific context. In the decimal (base-10) Arabic numbering system, the digits are the elements of the set {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}. The term "digit" is often used to represent a number in a specific position in a larger number, or in a number with a fraction expressed using a radix point. For example, in the decimal number 2609.37, the number 2 is the thousands digit, the number 6 is the hundreds digit, the number 0 is the tens digit, the number 9 is the ones digit, the number 3 is the tenths digit, and the number 7 is the hundredths digit. Users are allowed to enter numbers in operations in order to let the system calculate their input. Each arithmetic syntax has to have delimiters in order to detect numbers. Allowed delimiters are “(”, “)”, “,”, “ , ”.

Available alphabet samples:

- digits (0,9) and period (.),
- letters (a-z, A-Z),
- delimiters (“(”, “)”, “,”, “ , ”)

Lexemes

A lexeme is a unit of lexical meaning that exists regardless of the number of inflectional endings it may have or the number of words it may contain. It is a basic unit of meaning, and the headwords of a dictionary are all lexemes.

There are 8 lexemes that are supported by the application.

- **get**: reads variable from console.
- **put**: set value to specified variable.
- **print**: displays all stored operations.
- **remove**: removes variable from memory.
- **move**: moves variable value to specified variable.
- **save**: stores saved operations into a text file.
- **load**: loads a text file.
- **add**: addition.
- **sub**: subtraction.
- **mult**: multiplication.
- **div**: division.

Syntax

variable ::= letter { letter | digit }
filename ::= variable
number ::= digit { digit } { period } { digit }
statement ::= operator (operand, operand)
operator ::= add | sub | mult | div
operand ::= variable | statement
get ::= get(variable)
print ::= print
set ::= put(variable)
load source code ::= load(filename)
save ::= save
terminate interpreter ::= end | exit

Example:

put(a,add(8,sub(16,mult(3,4))))
put(b,add(22,sub(16,mult(3,4))))
put(c,div(add(2,3), sub(mult(3,4),add(2,2))))

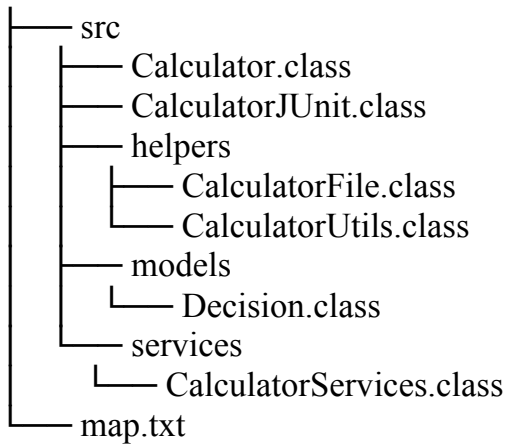
get(a)
move(add(b,c),a)
get(a)
remove(a)

print

Example with result:

put(a,add(8,sub(16,mult(3,4)))) creates variable a in the memory and its value is 12.
put(b,add(22,sub(16,mult(3,4)))) creates variable b in the memory and its value is 26.
put(c,div(add(2,3), sub(mult(3,4),add(2,2)))) creates variable c in the memory and its value is 0.625.
If we type get(a), it returns value of the variable a which is 12.
move(add(b,c),a) first adds variable b and c, then puts the result in variable a.
If we again type get(a), it will show updated value which is 26.625.
remove(a) removes the specified variable from the memory.
print operation shows all declared variables in the memory in a table format.

Application Structure



Calculator.class is the main class where main function of the application is implemented. The main() method in the Java language is similar to the main() function in C and C++. When you execute a C or C++ program, the runtime system starts your program by calling its main() function first. The main() function then calls all the other functions required to run your program. Similarly, in the Java language, when you execute a class with the Java interpreter, the runtime system starts by calling the class's main() method. The main() method then calls all the other methods required to run your application.

CalculatorJUnit.class is not important for the project, but it is important to test every input possibility. Since we have saving method which is “put”, we need to take care of input validation. Junit is a unit testing framework which is important in the development of the test-driven development. We use Junit version 4 in this application. One of the important project configuration is to configure the right Junit version which is 4. In this class, we tested user inputs, because when we take user input, we put it in a filter to separate numbers and arithmetic lexemes to detect what the input is entered for.

There are three helper classes for the application. They are lesser known code smell we identified some miscellaneous, commonly used operations and attempted to make them reusable by lumping them together in an unnatural grouping.

CalculatorFile.class is responsible both to save saved operations into a file, and to load a file where previous operations are saved. We store saved operations by serializing the object with Serialization e.g.: a=["(add(5,5)",10.0] where “a” is variable name. Right side of the equivalent symbol which is = has 2 indexes. First index is the user input and second index is sum of the first index. In above example, user wants the application to add two numbers which are 5 and 5. Sum of the user input is 10 which is stored in second index.

CalculatorUtils.class is a class that contains utilities such as checking if input is number, converting String input to float primitive data type, or setting empty indexes to null in an array. These functionalities help us to detect user's wish by detecting arithmetic syntaxes or decision.

Decision.class is an enum type where arithmetic syntaxes and decision prefixes are stored. An enum type is a special data type that enables for a variable to be a set of predefined constants. The variable must be equal to one of the values that have been predefined for it.

GlobalServices.class is the service function of the application. It is unit of the application functionalities that are automatically available when they are called.

How It Works

User enters an input. The application detects starting keyword from the input. It checks following keywords: move, put, load, save, get, print and remove. Each keyword has their own body content except print, load and save keywords. They do not have body content. If the input contains any calculation method, algorithm takes that part from the input and converts it into an array by splitting the input with defined delimiter. In some cases, there might be multiple closing parentheses one after the other. Due to the fact that, array index contains empty index. In order to get rid of this, the algorithm detects and sets them null. Then the algorithm reverses the array beause ordering mathematical operations. Then the array indexes are checked through a for loop. The loop first detects variable name from the input and first checks that variable name if it exists in memory. If so, the application throws an exception to warn user that entered variable name cannot be used. If the variable name does not exist in the memory, the algorithm then checks arithmetic operation. Regarding to arithmetic letters which are add, mult, sub and div, the algorithm calculates it and sets value of entered variable name to result and sends the variable to memory.

How to Run

In order to run the application, we first need to install a java integrated development environment. After we import our project, we need to configure build path. If we assume that we use Eclipse IDE, right click to project name and find Build Path and click Configure Build Path… You will then see Java Build Path window. First, we need to Add necessary jars. To do that, we need to click Add Jars… button, then find two necessary jars inside "bin" folder in the project. Necessary jars are "commons-lang3-3.0.jar" and "gson-2.8.2.jar". Then click OK to import them. Second, we need to import Junit library. To do that, we need to click Add Library… in Java Build Path window. Then find JUnit. Then we need to select version 4 as known as JUnit 4. After importing necessary things, we need to make sure that default java version is 8, because we use Java SE 8 in our application. After everything is done, we are good to go.

Conclusion

This application is so called string calculator. Instead of using arithmetic symbols, we define them with letters. The application is an interpreter for calculator. It is an interpreter program using polish notation where program reads from input, line by line, interprets commands and prints the result on the console.