**AC12001**

Name: …Miklos Mayer………………………………………………………………………..

Matric number: …190017778 …………………………………………………….……….

Lab Title: …AC12001 assignment 1: Stacks………………………………..……….

Test number/date/version: 03/02/20 ……………………………………….……..

Test Notes: …Tests run on submitted assignment ……………………………

Summary:

A program is needed to calculate Reverse Polish Notation expressions.

Requirements:

* The program shall be able to solve Reverse Polish Notation expressions with the 4 basic binary operator (+, -, \*, /).
* The calculator shall use a Stack data structure for calculating the result
* The program shall include a Menu options for the following tasks:
  + Show help
  + Enter and calculate a Reverse Polish Notation Expression
  + Exit
* The program shall handle all the errors and exceptions which could occur during the calculation and inform the user about them.
* The program should transform the Reverse Polish Notation into Infix Notation

Class designs:

* RPN\_Calculator:
  + Handle user inputs
  + Calculate expressions
  + Informs the user about errors
  + Fields: none
  + Methods:
    - static main(String[]): void – Entry point of the program, operate the calculator and the user inputs.
    - private static printMenu(Scanner): String – Print the menu of the game to the user and return their commands.
    - calculate(String): double – Calculate the result of the given expression.
    - private operate(String, Stack): double – Executes one operation between two numbers and handle Exceptions. Returns the result of the operation
* ListNode:
  + A node in a Linked List data structure for double numbers
  + Fields:
    - number: double – The stored number in the node
    - next: ListNode – The next node in the list
  + Methods:
    - getNext(): ListNode
    - setNext(ListNode): void
    - getNumber(): double
    - setNumber(double): void
* LinkedList:
  + A Linked List data structure implementation for double numbers
  + Fields: head: The first node in the Linked List
  + Methods:
    - isListEmpty(): boolean
    - getHead(): ListNode
    - addToList(double): void – Add a new number to the beginning of the list
    - deleteFromStart(): void – Delete the first number from the list
    - findInList(double): ListNode – Find a number in the list. Return null if it does not contain the number.
* Stack:
  + A Stack data structure implementation.
  + Fields: stack: LinkedList – The stack uses a LinkedList as its core data structure
  + Methods:
    - push(double): void – Add a new number to the top of the stack
    - pop(): double – Return and delete the number from the top of the stack
    - isEmpty(): boolean
* EmptyStackExpression: Thrown when you try to pop an empty stack
* InvalidRPNExpressionException: Thrown when an invalid expression is provided to the RPN calculator.

Pseudocodes:

The calculator:

1. Split the expression into numbers and operators
2. Push numbers into the stack
3. When an operator reached execute it on the last two numbers from the stack
4. Push the result back into the stack
5. Continue from step 2. until the expression run out of numbers and/or operators
6. Return the end-result

Possible errors:

* The expression is ending with a number
* Too much number in the expression, there’s no operators left to operate with them during the calculation
* Too few numbers in the expression, there are unused operators left during the calculation
* Invalid operator provided by the user
* A division by zero occurs during the calculation
* The provided expression is empty
* A space is missing between a number and an operator

Test results:

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Description** | **Test Data** | **Expected result** | **Worked?** |
| Add one node to a list | 55 | Code completes, node is outputted with correct data (55). | Y |
| Add multiple nodes to a list beginning | 55, 8, -89.6897 | Code completes, node is outputted with correct data (-89.6897, 8, 55). | Y |
| Check if a list is empty | none | true | Y |
| Delete from start of a list | 55, 8, -89.6897 | Code completes, node is outputted with correct data (8, 55). | Y |
| Delete from start of an empty list | none | Nothing happens | Y |
| Push one node onto an empty stack. | 55 | Code completes, node is outputted with correct data (55). | Y |
| Push multiple nodes to a stack | Three items pushed on stack [55, 8, -89.6897] | Code completes, node is outputted with correct data (-89.6897, 8, 55). | Y |
| Check if the stack is empty | none | true | Y |
| Pop node from a not empty stack | Three items pushed on stack than popped once [55, 8, -89.6897] | Code completes, node is outputted with correct data (8, 55). | Y |
| Pop node from an empty stack | none | Throws empty stack exception | Y |
| RPN with empty expression | none | Throws RPN invalid format error. |  |
| RPN addition | 4 6 + | 10 | Y |
| RPN with negative number | 4 -6 + | -2 | Y |
| RPN calculation with too many numbers | 4 6 2 + | Outputs error as stack is not empty after answer is popped | Y |
| RPN calculation with too few numbers | 4 + | Outputs RPN invalid format error. | Y |
| RPN subtraction with positive result | 6 4 - | 2 | Y |
| RPN subtraction with negative result | 4 6 - | -2 | Y |
| RPN multiplication | 4 6 \* | 24 | Y |
| RPN calculation with missing space before operator | 4 6+ | Outputs RPN invalid format error. | Y |
| RPN calculation with expression ending with a number | 15 7 1 1 + - / 3 \* 2 1 1 + + - 5 | Outputs RPN invalid format error. | Y |
| RPN dividing with whole number result | 6 3 / | 2 | Y |
| RPN dividing with not whole number result | 3 6 / | 0.5 | Y |
| RPN dividing with 0 | 6 0 / | Outputs error because dividing with 0 | Y |
| RPN calculation with invalid operation character | 4 6 @ | Outputs RPN invalid format error. | Y |
| RPN complex calculation with whole numbers | 15 7 1 1 + - / 3 \* 2 1 1 + + - | 5 | Y |
| RPN complex calculation with not whole numbers | 15 7 1 4 + - / 3 \* 2 1 1 + + - | 18.5 | Y |
| RPN complex calculation with negative numbers | 15 7 1 1 + - / 3 \* 2 1 -1 + + - | 7 | Y |
| RPN complex calculation with dividing 0 | 15 7 1 6 + - / 3 \* 2 1 1 + + - | Outputs error because dividing with 0 | Y |
| RPN complex calculation with too much numbers | 15 7 1 1 + - / 3 \* 2 1 1 1 + + - | Outputs error as stack is not empty after answer is popped | Y |
| RPN complex calculation with too few numbers | 15 7 1 1 + - / 3 \* 2 1 + + - | Outputs RPN invalid format error. | Y |

Self-evaluation:

I enjoyed this assignment. I am pretty familiar with Linked Lists and Stacks, so my challenge for the assignment was to run the test in Junit5. I really enjoyed figuring out all the edge-cases and write a bunch of tests for them.