Name: Jamie Fergus

Matric number: 1900018054

Lab Title: AC12001 assignment 1: Stacks

# Introduction

The aim of this assignment is to learn how to design and code a Java program that uses a stack data structure, implemented from first principles, using a linked list. You will also need to understand the concept of reverse polish notation

# Requirements

Tackled most of the requirements excluding requirement 6 & 7. I achieved requirements 1 to 4. And I partially achieved 5.

# Design

## Class Diagram

|  |
| --- |
| Menu |
| RPNCalculator rpn; |
| main()  runMenu()  displayInstuctions()  displayMenu()  checkInt() |

|  |
| --- |
| Tester |
| Stack testStack;  RPNCalculator rpn; |
| main()  init()  process()  rpnTests()  stackTests() |

|  |
| --- |
| ListNode |
| int number;  ListNode next; |
| ListNode()  getNext()  SetNext()  getNumber() |

|  |
| --- |
| RPNCalculator |
| Stack rpn; |
| RPNCalculator()  evaluatePostfix()  checkInt()  checkOperator()  getString() |

|  |
| --- |
| Stack |
| List aStack; |
| InitialiseStack()  push()  pop()  isStackEmpty()  printStack() |

|  |
| --- |
| List |
| Listnode top; |
| List()  isListEmpty()  addToList()  printList()  deleteFromStart() |

## Pseudocode

evaluatePostfix() - Prompts user to enter postfix notation calculation and delimiting factor then calculates and displays the answer.

// initialise rpn

// local variables

// do until error handling is not triggered

// initialise variables

// do until valid Postfix notation length is entered

// if quit is selected return to menu

// for each token

// check if token numerical or operator

// if numerical push

// if operator and 2 numbers are on stack pop twice and apply operator and push

// a series of error handling checking if correct Postfix notation is used

// more error handling

// pop and display answer

# Test Plan & Results

Test number/date/version: 01/02/20

Test Notes: Tests run on submitted assignment

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Description** | **Test Data** | **Expected result** | **Worked?** |
| Initialise Stack, Push node onto an empty stack, then print node. | number = 42, | Code completes, node is outputted with correct data (42). | Y |
| Push multiple nodes to a stack | 17, 50, -3 | [output results] | Y |
| Pop node from an empty stack | none | Outputs empty stack message | Y |
| Push 2 nodes then pop | 67, 32, | 67 | Y |
| Test isStackEmpty() method with empty stack |  | true | Y |
| Test isStackEmpty() method with non-empty stack | 1,2,3 | false | Y |
| Print empty stack |  | The list is empty | Y |
|  |  |  |  |
| RPN calculation | 4,6,+ | 10 | Y |
| RPN calculation | 2,10,- | 8 | Y |
| RPN calculation | 7,8,\* | 56 | Y |
| RPN calculation | 5,20,/ | 4 | Y |
| RPN calculation | 4,6,2,+ | Outputs error as stack is not empty after answer is popped | Y |
| RPN calculation | 4,+ | Outputs RPN invalid format error. | Y |
| Quit rpn calculation | q | return | Y |
| Invalid rpn input | cheese | Invalid input error | Y |
| Invalid rpn input | b,b,q,+,+ | Invalid input error | Y |
|  |  |  |  |
|  |  |  |  |

# Self-Evaluation

Was not able to get all the optional extras done but I achieved the other requirements. Could have probably tidied up my RPNCalculator code and made it with more methods and make it more efficient.