**AC12001**

Name: Gerda Ugne Pupelyte

Matric number: 190013631

Lab Title: AC12001 assignment 1: Stacks

Test number/date/version: 31/01/20

Test Notes: Tests run on submitted assignment

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Description** | **Test Data** | **Expected result** | **Worked?** |
| Push node onto an empty stack, then print node. | 55 | Code completes, node is outputted with correct data (55) | Y |
| Push multiple nodes to a stack | Three items pushed on stack (89, 52, 430, 72, 59) | Results are printed | Y |
| Pop node from an empty stack | none | Outputs empty stack message | Y |
| RPN calculation | 5 10 + | 15 | Y |
| RPN calculation | 5 10 5 + | Recognizes invalid input and prompts user to re-enter | Y |
| RPN calculation | 7 + | Recognizes invalid input and prompts user to re-enter | Y |
| RPN calculation | 20 3 \* | 60 | Y |
| RPN calculation | 2 3 + 7 \* | 35 | Y |
| RPN calculation | 10 2 / | 5 | Y |
| RPN calculation | 9 0 / | Informs user about invalid input and prompts to re-enter | Y |
| RPN calculation | 9 15 2 + 7 \* | Informs user about invalid input and prompts to re-enter | Y |
| RPN calculation | 2 10 5 \* + | 52 | Y |
| RPN calculation | 90 9 - | 81 | Y |
| RPN calculation | 10 5 + + | Informs user about invalid input and prompts to re-enter | Y |
| RPN calculation | 5 / / | Informs user about invalid input and prompts to re-enter | Y |
| RPN calculation | - 12 3 + | Informs user about invalid input and prompts to re-enter | Y |
| RPN calculation | \* | Informs user about invalid input and prompts to re-enter | Y |
| RPN calculation | + + | Informs user about invalid input and prompts to re-enter | Y |

**Class design**

EmptyStackException

A custom exception for dealing with invalid input

Tester

Tester contains the main method. Contains RPN Calculator.

RPN Calculator

RPN Calculator contains Stack.

Stack

Stack contains the list with list nodes.

List

List contains List Nodes.

Contains head.

List Node

List node is an element contained in the list.

Contains number and next node fields

**Pseudocode for RPN calculator:**

User’s input is received as a parameter.

User’s input is split and put into a String array.

Initiating a new array, which contains all possible operators which can be found in user’s input.

**For the length of user’s String, every element is processed:**

System checks whether the element is a number:

*If the element is a number (calls a method to check that), it is converted into a real number from String and pushed onto stack.*

*Else, the system tries to determine whether the value is a possible operator:*

**For the length of previously made operator array:**

The element is checked if it is a possible operator.

*If the element is an operator, calculations are made (calculate method is called). Else, the user is prompted to re-enter, since input is invalid.*

After calculations, number of elements in the stack is checked*. If it’s more than one, then user is prompted to re-enter the invalid input.*

**Pseudocode for determining if String is a number:**

System receives the string as a parameter.

The system tries to parse the String element into a double.

*If parsed successfully, returns true.*

*If an exception is thrown, it is known that the String element is not a number – returns false.*

**Pseudocode for calculations:**

System receives the operator as a parameter.

Two numbers from the stack are popped.

*In case of an error, custom empty stack exception is caught and user is prompted to re-enter.*

Switch cases are used depending on the operator.

There is each case for +,-,\* and /.

*Note for /: if number 1 contains 0, an exception is thrown and user is prompted to re-enter their data.*

If operation was successful, result is pushed onto the stack.

**Self-evaluation:**

Personally, I found the assignment to be really hard and very time consuming. It’s very hard to understand why stacks are used in the first place when it’s mostly calling methods from the List class. I managed to complete the main goals of the program and implemented data validation, did my best with exception handling and also added a division operator. Despite the struggles, I managed to remember some of the things from last semester and understood the very fundamentals of stacks. I am satisfied with the outcome of the program, although I think I could have done better if there was more time given for the assignment.