Assignment 1 – RPN Calculator – Documentation (Max Fyall)

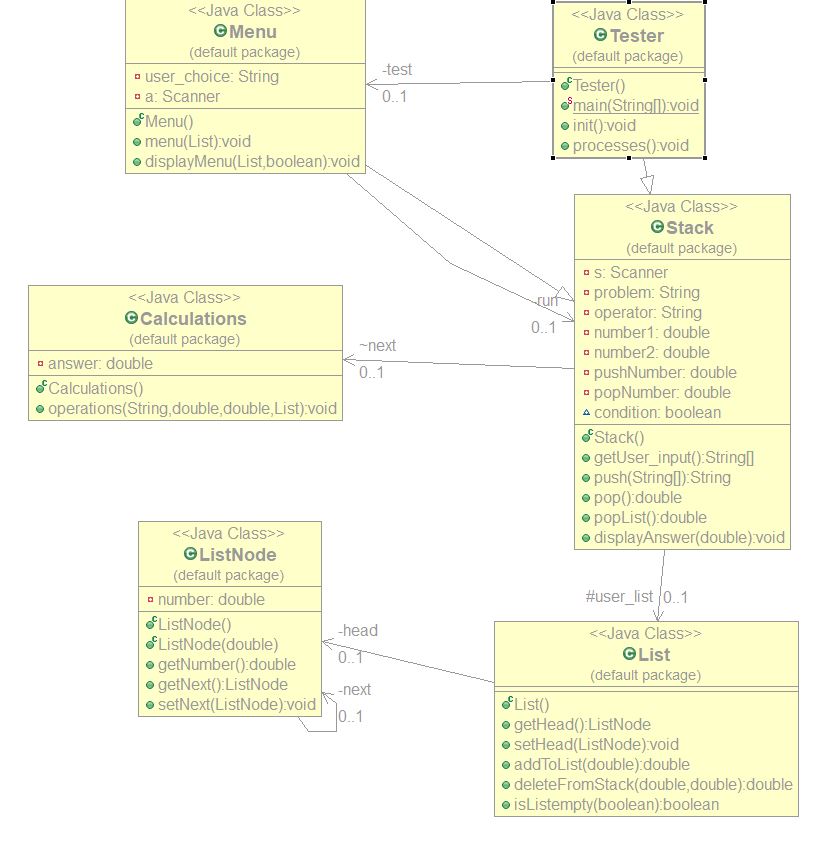
Introduction to problem:

The aim of this assignment is to implement a Reverse Polish Notation calculator with ‘+’, ‘-‘ and ‘\*’ as operators using a stacks data structure, implemented from first principles, using a linked list.

Requirements:

Requirements 1-6 were satisfied when coding this program. The implementation of these 6 requirements was successful. Requirement 7 was not completed due to the sophistication and time constraints that were met while coding. It was attempted but, the code was deleted due to problems it may cause when running the program normally.

Class Design:



Pseudo Code:

getUser\_Input()

* Receive input from keyboard
* Split input into array of strings
* Validate input by checking for operators, surround in try and catch to catch any exceptions
* Call push method

push()

* Start for loop, set condition in for loop to length of the array
* Checks for operator, if not operator then converts to double and adds to stack
* If operator found then pop class is called and operator is stored in field
* Validation to check if still no operator
* Calls poplist method to pop answer off the list once answer has been found
* Checks if list is empty
* Displays answer if list is empty.

pop()

* Call deleteFromStack method and store data returned from this method in a variable. Do this again to have the two numbers
* Call the operations method and pass in the fields that hold the two numbers and the operator stored earlier.
* Surround this in a try catch to prevent any crashes.
* Return the number fields

operations()

* Use switch statement and have 4 cases for the 4 operators, addition, subtraction, multiplication and division.
* Add if statement to check if user had divided by 0.
* Push the answer back onto the stack.

popList()

* Call deleteFromStack method and store the data returned in a variable. This variable will be the one to display as it should be the last item on the stack meaning it is the answer.
* Surround in a try catch to prevent any exceptions.
* Return field.

deleteFromStack()

* Store data stored in stack in a field.
* Set new head of the stack to remove the previous head.
* Return the field storing the number.

isListempty()

* Create instance of ListNode marker, set to null
* Set to head.
* If marker is null then list is clear, set found to true
* If not then the list has data in it
* If list has data in it then display message and delete the data from the stack.
* Return found

Test Plan:

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Description** | **Test Data** | **Expected result** | **Worked?** |
| Push multiple nodes to a stack | Three items pushed on stack [enter data] | Data pushed onto stack | Y |
| Pop node from an empty stack | none | Outputs Error Message and asks for new equation | Y |
| RPN calculation | 6,6,+ | 12 | Y |
| RPN calculation | 3,5,6,+ | Outputs error message ask user to press enter and bring them back to menu | Y |
| RPN calculation | 4,+ | Outputs Exception error | Y |
| RPN calculation | 2,4,- | 2 | Y |
| RPN calculation | 2,3,-,1,+ | 2 | Y |
| Invalid input | 2+4 | Error message, re-enter equation | Y |
| RPN calculation | 3,5,\* | 15 | Y |
| Invalid input | 3/1 | Error message, re-enter equation | Y |
| RPN calculation | 3,1,/ | 3 | Y |
| RPN calculation | 5,5,\*,2,+,3,/ | 9 | N |
| RPN calculation | 5,5,\*,2,+,3,/ | 9 | Y |
| Invalid input | 5\*5/5 | Error message, re-enter equation | Y |
| Invalid input | 5,5,\*,/ | Error message, re-enter equation | Y |
| RPN calculation | 23,5,- | 18 | Y |
| RPN calculation | 6,8,\*,2,4,+,-,6,/ | 7 |  |

Evaluation:

I found this assignment to be another very good source of learning new ways to code in java. The assignments allow me to try out new coding methods to reach the final working program. This can allow me to try multiple solutions in future projects I will be carrying out. As ever, this assignment did have areas which were very puzzling. Most of these problems, I found solutions to. These problems I am referring to occurred when I was trying to implement code for requirements 1-6. Errors would show up, some data would not display or more data than I intended to display, displayed. Solutions to these were found and I believe the program works as intended. I did try and find an algorithm that would allow me to implement requirement 7 but due to the time constraints and the difficulty of the challenge I decided to leave it out. The rest of the requirements were met in the coding of this project.