COURSE: SOFTWARE ENGINEERING

MODULE: DATA STRUCTURES & ALGORITHMS & DATA STRUCTURES

CRN: 34126

MODULE LEADER: TAREK GABER

Name: O'Diepriye Graham-Douglas

Student ID: AGD626

Student Number: 00592983

Assignment 1 (DS)

This is a summary of the project given to create an infix to post fix converter using abstract data types of personal choice.

CONTENT

Justification of Using my chosen ADTs:

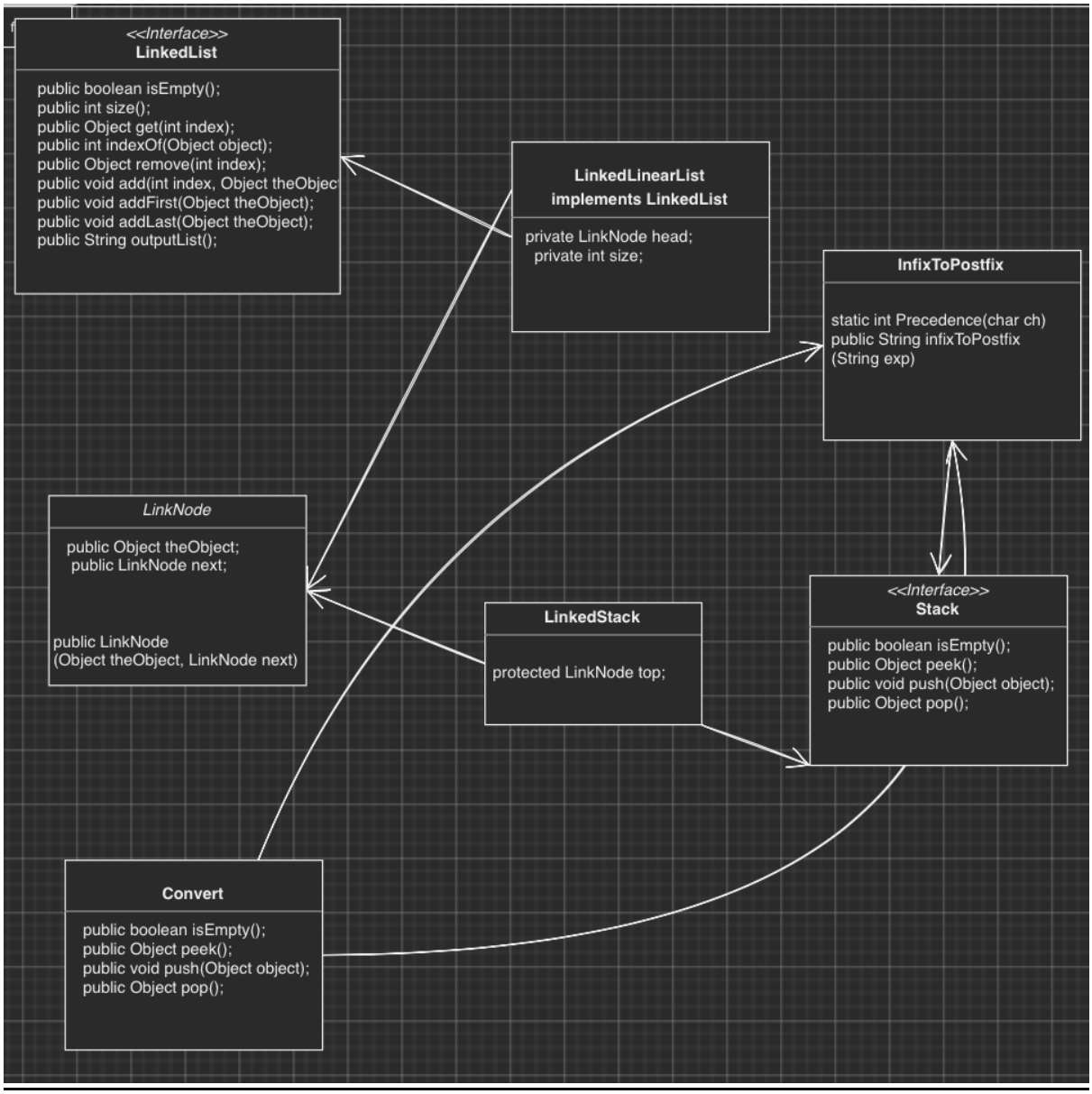
I believe the best way to tackle the infix and postfix converter is by using a Stack ADT and implementing it using a linked list. I knew Stack would be perfect for this problem because it is typically used for parenthesis matching. Stack is perfect for doing this conversion because the last in first out method is suits the converters need for operator precedence. Stack is perfect for this converter because it can work parallel to it.

It would scan the operators and operands in the expression from left to right and put the operators in a stack from left to right. In which the left is the bottom of the stack and right is the top of it. It is also helps that the Stacks only maintain one pointer to access the list because only one is needed. The converter scans the expression from left to right and pops the operators straight into a stack. After stating what the precedence is through code it puts the operators in a stack and the output will be the result. It would output all operands after the left bracket then put the operators into a stack. When there is a right bracket every operator in the stack gets popped to the output. For the operators Stack if an operator with a lower precedence is added into the stack when there is a higher one the higher one is popped out.

The Queue ADT would be less efficient because its used more for sequential processing and the data transferred is not necessarily received at same rate as sent between two processes. Queue uses first in first out and would insert and remove from two different points of the list making it less efficient.

Then I used linked list to implement the Stack ADT. I believe the best way to implement the Stack ADT is using a single linked list because there is no risk of expanding the array, adding, and removing items in a LinkedList is simple and quick. Linked list will work best because I will be adding and removing items from the stack quite frequently to get the postfix output. Array List needs to resize and copy content to a new array in this case therefore I feel it will be more efficient using the linked list over the array list. Linked List is also more advantageous to use than array list because the items in the Stack need to be linked. The converter works based on operator precedence and to know if the item on top of the one under it is of higher or lower precedence it needs to be linked. The linked list has data on the next node on every node making this much easier for my implementation in comparison to the array list.

Class Diagram:



List of the expressions:

Expression 1:

Input: (P*Q)+(R/(S*(T+U)))

Output: PQ*RSTU+*/+

Expression 2:

Input: (P+Q)*(R/(S+(T*U)))
Output: PQ+RSTU*+/*

```
(P+Q)*(R/(S+(T*U)))
PQ+RSTU*+/*
```

```
Can only enter input while your programming is running
```

Expression 3:

Input: (P+Q*L)/(R*S+(T*U)*S)
Output: No output given because there are more than 20 characters.
Explanation: Basically the code was not meant to run anything above 20 characters. So the output of the code says too many characters so the user knows to input let characters.

```
(P+Q*L)/(R*S+(T*U)*S)
Too many characters.
```

```
Can only enter input while your programming is running
```

Expression 4:

Input: (A*B+C/D)/(E*(F*G))
Output: AB*CD/+EFG**/

```
(A*B+C/D)/(E*(F*G))
AB*CD/+EFG**/
```

```
Can only enter input while your programming is running
```

Bibliography:

1. The Call Stack. (2014). *YouTube*. Available at: https://www.youtube.com/watch?v=Q2sFmqvpBe0.

2. Jenny's lectures CS/IT NET&JRF. (2019, October 10). *3.6 Infix to Postfix using stack | Data structures*. Www.youtube.com. https://www.youtube.com/watch?v=PAceaOSnxQs

3. *Infix to Postfix Converter | Dynamic Step-By-Step Stack Tutorial*. (n.d.). Free-Online-Calculator-Use.com. Retrieved November 30, 2021, from https://www.free-online-calculator-use.com/infix-to-postfix-converter.html