Comp 2000, Data Structures – Fall, 2021

Stack ADT and App

Luis Gjuraj

Assignment: Stack ADT/Application: Infix Expression Evaluator

Instructions:

* Rename this file: replace ‘username’ with your username – delete the date that follows the username
* Replace ‘Your Name’ with your actual name above
* Replace '#' with your group number
* Replace 'App Name' with the name of your group's application
* Answer all questions fully
* Save this file (must keep it as a Word document) in the admin folder

**ADT Analysis**

These questions apply to the book's implementation of ArrayStack.java (Please be specific)

1. Given the functional definition of a Stack ADT (ordered/LIFO, allows duplicates, restricts access to the elements in it/top only), is this an appropriate implementation? Why or why not?

Yes, it has implemented all its methods and attributes.

1. Can the Stack ADT accept/handle null entries? Why or why not? What functionality, if any, would break if the application could push( null )?

The Stack ADT does accept null entries and adds them inside the stack. It does not break the application.

1. In general, how useful is the Stack data structure? Explain your answer. Can you think of other uses aside from those discussed in the book?

The Stack ADT could be useful when we are interested in storing data in a certain order and retrieving it in reverse order from the insertion. You can not go into the structure to check for data. The only way to find a certain value is to take them out one by one.

1. If we didn't have the Stack data structure, can you think of another way to implement the LIFO (last in/first out) concept?

Not really. The Stack ADT is the only one that has hat ordered kind of implementation. Queries do give a similar idea, but the first inserted element in the array gets taken out instead of the last one inserted. Bag ADTs are unordered, so there is no chance for those to work.

**Infix Expression Evaluator Analysis**

These questions apply to InfixExpressionEvaluator.java (Please be specific)

1. Briefly describe the Infix Expression Evaluator application (in ‘real world’ terms). Include relevant decision points (e.g., how to implement operator precedence?). What issues/problems arose during the design, implementation, and testing phases and how did you resolve them?

The Infix Expression Evaluator is simply a calculator. You put values in one stack, operations in another, and it executes them according to the PEMDAS method. The hardest part was implementing it into code.

Operators are ranked by precedence. We gave each operator a value. If according to the PEMDAS method it is more important or of the same, it has a bigger value, so it is computed before the next one.

I had different cases for operators, for both brackets, and for numbers. I had trouble with negative numbers, with exceptions, and I was basically looking at the text log of the tests and looking at specific incorrect answers, and I fixed them one by one.

1. Would you say that the Stack ADT is useful for evaluating arithmetic expressions? Why or why not?

The stack ADT does operations correctly and efficiently. I am not aware of other abstract data types that do the same thing better.

1. If you implemented multi-digit support, how did you approach the task/what did you need to change from your single-digit implementation?

I didn’t.

1. If you implemented invalid expression handling, how did you approach the task/what did you need to change from your multi-digit implementation?

I didn’t.

**Assignment Summary**

These questions apply to the assignment as a whole (ADT and Application) (Please be specific)

1. What did you learn from this assignment?

How to manage Stack ADT and use it for specific purposes. I was able to debug with every mistake, and I got to have fun with a little of Junit testing.

1. Where did you have trouble with this assignment? How did you move forward? What topics still confuse you?

Yes, the hardest part is implementing the logic behind the Stack ADT. But once I visualized it and I read the pseudocode in the book, I just went through the assignments step by step. I have made a habit of doing that now with every code I write.

1. How could this assignment be improved in the future?

Its great, don’t change it. I have learned more here than in the past two years.