Nick Cheatwood Monday, April 19, 2021

CAC 310 | Final

Report

1. **What language did you choose to implement the program in and why?**
    1. To write this program, I opted for Java. Although there is a bit more overhead than Python, I do like Java's static typing. I typically lean toward object-oriented programming when I'm able and I find Java to be the superior language for that schema. I also find that Java has neat packages that can be imported, works well with VS Code (my IDE of choice), and includes support for Reflection, which is super powerful when abstracting functionality in the program. Additionally, I did find that getting memory addresses for objects is somewhat easily done in Java (in my limit experience). There seemed to be a lot of discussion on the web around that topic, so reading various arguments for a certain approach was very interesting.
2. **Formally describe the language (generation, paradigm, compiled vs. interpreted, etc.). Did any of these features influence your language choice?**
    1. Generation: 3rd
    2. Paradigm: Object-oriented
    3. Compiled
    4. Declarative
    5. The only feature offered here that really swayed my decision was the paradigm being object-oriented. As someone who cut their teeth using languages such as Swift and Objective C, I find that the somewhat explicit nature of object-oriented programming lends more closely with my thought process when approaching puzzles like this. I also like how Java is statically typed. This reduces the odds of type errors as the project scales forward and can lend more closely to test-driven development in my experience.

3. **Briefly describe the logic behind your program. How did you approach the problem? Highlight any areas that were particularly difficult or that you are proud of.**

    1. Basically, my program consists of 3 main files and a helper file. I have a Doubly Linked List class file that contains all methods and attributes of a Doubly Linked List. I have a Node file that contains the definition of a Node and its various methods. I have a main file that executes a single method, which prompts the user for various inputs and prints the results of those inputs. The helper file contains some code I found in a Stack Overflow thread, which manages the getting of an object's memory address.

    2. Essentially the Doubly Linked List (DLL) methods perform various operations on the list and its subsequent Nodes. These range from insert methods, so deletions and lookups. The Node class structure is also very simple, containing essentially data, a previous and first node object, and various getters and setters. I split what I could into classes and went from there.

    3. I approached the problem by first doing my research into what a DLL is and how various methods work within the list. I then created a simple project file structure and connected my files to one another.

    4. Some areas I found difficult were deleting nodes from the list, as well as inserting into the list at a given position. In theory, these are simple tasks, but the added complexity of each node having a next and previous node, as well as the list having a head and tail made this a very convoluted puzzle to solve. When I got stuck, I was sure to check Stack Overflow and other programming sites for guidance.

    5. Some areas I am particularly proud of are the separation of concerns and abstracting what I can into getters and setters. Also, although I try and keep comments to a minimum, their inclusion in this program did help me better troubleshoot odd errors, as well as help me better explain the problem to myself.

4. **Reflection about the course: do you think you learned something new in this course? Did the course content push you to think more critically? Did any portion of the course aid in your ability to complete the final project? Constructive criticism is always welcome.**

    1. I definitely learned a lot from this class. For one I learned not to take our modern programming languages for granted. I found it very challenging to learn things like grammars, automata, and intermediate Prolog.

    2. The course definitely made me think more critically and I think it even made me think more practically as well. I'm definitely used to creative problem solving and things like grammars and automata really stretched my limits. Honestly my only suggestion to offer is that the concepts of grammars and especially automata felt like too much to unpack in a short amount of time. In the case of automata I think working through more examples in class would have helped me personally understand how to approach a problem using that tool. Aside from that I think this was one of the most developmental classes for me personally. I feel like I have a better understanding and appreciation for programming as a whole and I definitely think I could tackle programming interview questions better.

    3. I think the repeated exposure to Java helped me complete this project. Also thinking about the toolset of concepts and problem-solving methods I had learned helped me approach the daunting task of a Doubly Linked List a bit more easily.