Assignment 2 Analysis

Brayden Wright
u0895942

I didn't find this assignment to be too bad. Getting some of the loops right took a little thinking sometimes, and some of the comparisons using the 'compareTo' method took a little tinkering, but other than those two aspects, everything went fairly smoothly. This seems like it was a good assignment to slowly progress further into the intricacies of Java.

1. **This assignment is traditionally done with Pair Programming. Were there times you wish you had a partner to bounce ideas off of? We will ask you to compare your experiences with these first, "single" to assignments to the paired assignments later on in the semester. Please use your answer here for reference later.**

I can't really think of any specific aspect of this assignment I would have wished to have a partner to bounce ideas off of. The project specifications were given in a fairly clear manner through the assignment's page, which were relatively easy to follow. I also more experience programming than just getting out of CS1410, in this case meaning I knew most of the stuff already. If there was something I was not completely sure on, it was usually one of the methods that could be completed by glancing at one of the other pre-written methods. Looking at those pre-written methods made writing the other methods easier.

2. **Java's built-in classes Comparable and Comparator are both interfaces for doing comparisons among objects. What is the difference between the two interfaces? Give a situation when it is best to use each. Is it possible to change the extra features in LibraryGeneric such that the Comparable interface is used instead of Comparator? Why or why not?**

As far as I'm aware there are a couple notable differences between the Comparable and Comparator classes. Comparable generally seems to use the 'compareTo' method and has a predefined (by Java) way to compare objects. For example, Java's 'String' object is predefined to compare strings alphabetically, that is, strings that are alphabetically before the string being compared to will return a specific value for every instance where that is true. Comparator generally is used for generics and makes use of a 'compare' method that compares two objects in the manner the programmer has specified. Say we wanted to compare strings based on the amount of capital letters, that is when we would use Comparable.

Could we instead replace the use of Comparable in the project with uses of Comparator? Yes, though it would make things a little more complicated for the programmer and there is no need to reinvent the wheel if the current Comparable methods accomplish our goals.

3. **Comment about the efficiency of your programming time. Did you utilize the time spent on this assignment effectively? How might it be improved?**

I'm gonna be completely honest here, I procrastinated a good amount with this assignment and started it the Monday before it was due. Which means, I did not use my time efficiently. Since I did get it done though, I may have used the little bit of time I gave myself effectively. The situation could be greatly improved by me starting the assignment the night it is assigned and working on it at least an hour or two every night until it's due.

4. **Reiterate why writing Generic code is important for this course. Phrase your answer in terms of Data Structures and Algorithms.**

Writing code that uses generics is generally important for any programming where bits of the code may be used in the future. Specific to this class, we will later be writing sorting code and other code that may be useful later in the semester. Using generics allows us to reuse those methods with minimal (none if done right) rewriting of code. In terms of Data Structures and Algorithms as a topic, generics allow one to reuse structures and sorting algorithms without having to rewrite the code in order to accommodate a new type of data.

5. **How many hours did you spend on this assignment?**

I'm not entirely sure, though a good estimate for how long I spent on the programming portion of this assignment was somewhere around 5 to 10 hours. If I had to estimate what that time was specifically spent on, it was probably about 70% writing Java, 15% debugging small issues, and 15% creating some tests to make sure everything worked as expected.