Ben Fullenkamp
CSC6301 - Project 5

Coding project 4 with code reuse in mind made this maintenance task very simple. I was able to fully reuse my previous code with minor refactoring. When I wrote my code for the LinkedList implementation, I took advantage of the fact that the LinkedList and Stack classes implement the List interface. Therefore, I ensured to use methods that are defined in the List interface as both classes will use the same names. This allowed me to replace all instances of LinkedList with Stack in the code without any additional changes. The two methods I used are **add()** and **sort().** Similar to the **push()** method, **add()** appends to the end of the list, which corresponds to the top of the Stack, then the **sort()** method will sort the Stack in ascending order.

The following list of changes were needed:

1. I had to import **java.util.Stack** instead of **java.util.LinkedList.**
2. I changed the return type of the **parseAndSortInput()** method to **Stack<Integer>.**
3. When instantiating the data structure, I now needed to define it as:
    a. **Stack<Integer> integerList = new Stack<Integer>();**
4. Finally, I needed to clean up the Javadoc comments for clarity as I referenced the specific data structures in my comments.

When examining both projects in totality, I wrote this small program with a lot of code reuse. I used the Java standard library **Scanner** class to read the user's input using the **nextLine()** method. I parsed the input using the **String** class method, **split()**, which accepts a regular expression. Finally, I used the **LinkedList** and **Stack** classes from the Java Collections Framework to store the data using the **add()** method and sort the integers in ascending order using the **sort()** method. Utilizing built-in methods from the Java Standard Library made the initial program much simpler to implement. Then, writing my own code to allow for easy maintenance to switch from LinkedList to Stack shows the importance of code reuse in programming.