Ben Fullenkamp
CSC6301 - Module 4 Project

Code reuse is the practice of using existing source code instead of writing new code. It can come in the form of utilizing functions, libraries, frameworks, or even copy-paste. It is best thought of as “don’t re-invent the wheel” so you can spend more of your time developing custom code that has not been written before.

In my code, I used built in methods and the Java standard library to accomplish the task. First, I utilized the Scanner class (java.util.Scanner) to allow me to accept user input from the keyboard and store it in a variable. This is done by creating a Scanner object and calling the nextLine() method. Secondly, I utilized the String class to parse the user’s input. I was able to use the split() method, which accepts a regular expression. I could easily detect patterns in the input to parse out each individual integer. This is incredibly useful to create a short regex pattern (\s*,+\s*|\s+) to add flexibility to the user input and limit errors. Lastly, I imported the LinkedList class (java.util.LinkedList), which is a part of the Java Collections Framework. This framework contains a large list of interfaces and classes for data structures that hold collections of items. The LinkedList class implements interfaces higher in the hierarchy, such as List and Deque, to have its own implementation for methods. In my code, I used the add() method, which adds items to the end of the list, and the sort() method, which uses an adaptive mergesort algorithm, to add the user’s integers to the LinkedList and sort them. From my experience in Merrimack’s data structures and algorithms courses, creating the basic structure for the LinkedList data structure and the method to add elements takes quite a bit of time. Then creating an efficient sorting algorithm is incredibly complicated, so code reuse is very important to spend more time thinking about the code design.

In terms of designing my own code for easy maintenance, I wrote my code with project 5 in mind. This project asks us to adapt the code to using a Stack data structure rather than a LinkedList. Since LinkedList and Stack both implement the List interface, I ensured to only use methods that both classes share. The use of interfaces allows each class to have a different implementation with the same method name. Therefore, I’ll easily be able to change each use of LinkedList with Stack with no other changes needed.