Tony Diep
CS 2420
Assignment 4 Design
2/9/17

The purpose of this project is to work with an anagram utilization program using the insertion sort algorithm, the comparator interface, and the generics mechanism. This project also introduces the application of "big-O" complexity and how it is used to efficiently process any given list of words to determine if they are anagrams. Although insertion sort is an inefficient algorithm for finding anagrams, we are investigating the efficiency behind the algorithm depending on the size of N, where N denotes the number of words in a list.

Three classes are to be made in order to succeed in this project plan. First is "AnagramUtil," which is the main workhorse of the program. It contains 5 methods that will be used to sort a list of words as well as determining if two given words are anagrams of each other. The project specifications allows us, however, to add as many helper methods and helper classes as desired. There is an opportunity to get more credit if we implement JUnit Tests for those helper methods. The second class is the JUnit Test Case class for the AnagramUtil class, called "AnagramUtilTest." Finally, the third class is the Main, which contains timing experiments to test for the generic insertion sort method and the getLargestAnagramGroup method. The Main will time the complexity of those two methods and we will observe their efficiency based on the algorithm complexity.

It can be expected that with an increased number of words, there will be a longer processing time. At this point it can be assumed to be a linear relationship, though faster computers may result in a quadratic result. It is approximated that this project will take 8-12 hours in order to complete the code in a thorough manner, as well as test the methods with JUNIT testing. This time will also be spent commenting the program and insuring high quality style of writing so that future users may easily understand the algorithm and process.

When approaching this assignment at first glance, we will construct our JUnit Tests first to get a good idea of we should expect for the methods in the AnagramUtil class should output. Then we will implement the 5 empty methods in the AnagramUtil class. Next, we will implement the timing code from Monday's lab session. Finally, we will begin the timing experiments for the two methods we are required to test and construct the required documents (i.e. analysis document, partner evaluation, README, and i_worked_with).