Homework 1 Analysis — Josh Ross
CIS 350 — January 2017

1. First I read through the entire assignment to make sure I had a clear understanding of what was required for my program. Then, in order to decide what classes were necessary or could be useful, I parsed out what I thought was important and would be used multiple times. The first class I created was the Main class as we were instructed to do in the assignment write up. The Main class is responsible for running the program and would be responsible for storing all of the tweets and states information. Next, I realized it would be good to have a Tweet class so that I could treat each tweet as an object of the same type. The three important fields the Tweet class needs are state, time, text. Then I created subsequent methods that allow the user to get the values of each of those fields. The main class would use the Tweet class to create an ArrayList of all of the tweets received from the input file. Then, I made the Location class which allowed me to add all of the information regarding the states to a Map which mapped state to a specific coordinate. Within that class I wanted to simplify the latitude and longitude coordinates so I made a inner class called LatLong which stored both coordinates in one object and allowed them to be individually retrieved through simple get methods. The two methods that I made for the Location class were addState, which allowed me to add all of the states and their coordinates to the Map of states, and findState, which allowed me to find what state was closest to a given latitude and longitude. In addition, I made a Menu class which the Main class would use to perform the 3 different operations specified by the user. In this class, I have a Rank method, which takes in all the tweets and states and outputs the list of states ranked by count of tweets, a Hashtags method, which takes in all the tweets and a given state and finds all the tweets for a given state while adding the hahstags to a HashMap that is keeping track of how many times that hashtag has occurred, and lastly there is a PerHour method, which takes in all the tweets and a phrase and then checks all the tweets for that phrase and then stores them in a HashMap according to the time the phrase occurred. There is also a generic method sortByValue which allows me to sort a HashMap according to the values as opposed to the keys and then it returns the entries in a list. Lastly, I created a Time class used by the Tweet class in order to parse out the time by year, month, day, hour, minute, and second. This allowed me to easily compare times among tweets.

2. One way in which the design of my classes is good is because the only class that would be affected by a change in the format of data or input file is the main class which parses through the files. All other classes are given more specific input and thus would not be affected by this type of change. A second way in which the design of my classes is good is that if we wanted to add more options for the user, all that we would have to do is add a method to the Menu class as the time, text, and state of every tweet is easily accessible and comparable, as the text and state are strings, and the Time class implements comparable and has a compareTo, equals, and toString method. The third way in which the design of my classes is good is that every class is needed and stands on its own, as in no two classes could intelligently be condensed into one class. For example, it is useful for me to have my Tweet and Time classes be separate as other classes, such as Menu, use the two classes for separate purposes.
3. One way in which the style of my code is good is through my naming conventions of classes, methods, fields, etc. Throughout my code I used descriptive names for each of these things so that as I was going back to review and check my code I understand what the purpose of a variable was. A second way in which the style of my code is good is by keeping things simple. I do not use complicated logic in order to solve small simple problems. Where necessary I make distinctions and separate out certain operations, but I try to avoid using unnecessary extra lines of code when a certain operation or function can be simplified to one line. A third way in which the style of my code is good is that in the Main class when the user enters an invalid input or an exception/error is come across I use meaningful print statements in order to show the user what was wrong. This was useful especially when I was testing my own program with smaller test files as well as for general debugging.
4. One way in which my code is efficient in terms of execution time and memory usage is that while I read in the tweets, I find the state in which each tweet is from and then store that in each tweet which means I don't have to do it again when asked to find all the tweets in a given state or the top hashtags from a given state. This also allows me to not have to store the latitude and longitude of each tweet because I never need to use them again. A second way in which my code is efficient in terms of execution time and memory usage is that when I perform an operation specified by the user, I print out the results directly from the Menu class as opposed to returning some object to the Main

class and then dealing with properly printing out the necessary information from that object.

5. In order to know that my code works correctly, I thoroughly tested my code using the command line and small txt and csv files. To make sure that the results of the Menu options were correct I made small files that followed the format of the tweets and states files we were given. These small files allowed me to anticipate the results of an operation, then run the operation on the file and see if the results were correct. This was also useful for testing the Main method and making sure I was parsing the files correctly. I was able to use my small test files and the debugger mode in eclipse to ensure that the correct information was being stored for each tweet and state. I also did my best to test all different edge cases of possible results from the options selected by the user in order to see where potential errors or exceptions could be thrown.