Assignment 16
(Full Score: 80 points)

(**Due in class on 5/3/Tuesday at Moodle**)

Your name:	Score:

Sets and Maps

Create a program that reads a text file, counts the occurrences of all words in the file, and outputs these words with corresponding occurrence counts in alphabetical order in another text file:

Requirements:

1. Create a Java class that has the methods for:
   a. Reading a text file.
   b. Storing all the words from this file and their occurrence frequencies.
   c. Displaying in alphabetical order the words found along with the computed occurrence counts.
2. Your code must have a *main()* method to test the calculations.
3. Use the Java Collections API class *java.util.TreeMap<K, V>* to store, access, and display the words and their occurrence counts.

Hints:

1. As the basis for your code, you may use the source file *chapter13\CheapestFlight.java* for opening and reading text files and *chapter13\MapTest.java* for *Map* operations. Keep in mind that the code in the source file *chapter13\CheapestFlight.java* has a bug – the file is never closed.
2. To split a Java *String* into words, use one of the following Java classes: *java.util.StringTokenizer* or *java.util.Scanner*.
3. Make sure that you properly select the keys and the values for the Java Collections API *java.util.Map<K, V>* interface and that you use all appropriate methods and fields from the *java.util.TreeMap<K, V>* class to simplify your code.

Bonus task (can add up to **20%** to the assignment grade):

Modify your code design to allow the word sorting based on an arbitrary sorting order defined by an instance of a class that implements the *java.util.Comparator<T>* interface. Design this code to enter and display the words in the English text in the reversed alphabetical order – from Z to A.

Submissions (one submission for one team):

1. Compressed project folder including javadoc file(s).
2. (Your code must follow Java Code/Javadoc Convention (20% of the credits.))

More Hints:

- Test your code with the attached input data file "bible1words.txt";
- Check your output (should be in a file) against the attached output file "output1.txt"

More BONUS:

1. (**30 points**) Change your code so it reads and processes the attached input file "bible1.txt" and the output should be the same (i.e., as "output1.txt").
    a. The (values) lowercase letters are "greater" than uppercases, thus in alphabetical order, uppercases are ahead of lowercases.
2. (**20 points**) Modify your code such that it will output the frequency (the number of times it occurs in the file) of each word in the input file (check against "output2.txt"):
    a. Words with larger frequencies should display first.
    b. If more than one words have the same frequency, display them in alphabetical order.