Assignment 6 Using Java Collections

Due Date: Friday April 15, before midnight
Weight: 5%
This assignment should be completed individually.

Description

This assignment is a reworking of the task you had to complete for assignments 1, 2 and 5

Essentially you have the same task: read file of English text and print out various statistics and lists on the frequency and length of words in the text. You should be able to use much of the code you wrote for assignments 1, 2 and 5.

The finished program will be run like this:

```
cat input1.txt | java A6.jar > myoutput1.txt
```

or

```
java A6.jar < input1.txt > myoutput1.txt
```

I will provide sample input and output files for the program. The output from your program must be *exactly* as shown in the examples. It can be very difficult to eyeball your code and see if it matches the required output exactly, so you need to let the computer do the comparison for you.

If you were given `input1.txt` and `output1.txt`, and you ran your program as above,

```
diff (cat output1.txt) (cat myoutput1.txt)
```

in Powershell or

```
diff output1.txt  myoutput1.ttxt
```

in Linux or MacOS will tell you if your progam is producing the correct result. No output from `diff` means all is well.

The specific requirements for this assignment are as follows.

1. For this assignment we will be using abstract data types (ADTs) from the Java Collections framework instead of writing our own.

2. Your program should operate in five general steps.

   (a) Read the words from the file and create a `HashMap` of the words. Stop words will be dealt with as in assignment 5.

   (b) Delete the stop words from the `HashMap`.

   (c) A hashtable does not have any inherent ordering of its contents. We want to be able to list the words alphabetically, by frequency and by word lenght.

   Use `TreeMap` objects (which are binary search trees, and hence have an ordering) to create lists ordered as required.

   (d) Print out the required results. See the sample output files for what is required.

   - The number of stop words is the number of stop words from the list that occured in the text.
   - Whenever you print a `Word`, print the word, its length and the number of times it occured, separated by colons.

3. You will have to research and figure out how to use `HashMap`, `TreeMap` and the supporting classes and methods.

Testing

I will supply some test input and output files for you to use. These files do not constitute extensive testing. You will need to do more testing to ensure all cases are covered.

I will be evaluating your program with test data that you have not yet seem and I will be trying to find flaws in your code!

What to Hand In

Hand in a single file, `A6.jar`.

Submit this to the Blackboard drop box provided. The jar should contain:

1. All of your `.class` and `.java` files.

2. A class called `A6`.

Do *Not* include your test input or output files or any other cruft files.

Grading

The assignment will be graded according to this rubric:

```
Total (60)

1. Documentation (15)
(a) Java doc standards           0/5
(b) Format of Code               0/5
(c) Meets other rules/guidelines 0/5

2. Testing (30)
(a) Test file1 0/10
(b) Test file2 0/10
(c) Test file3 0/10


3. Follows implementation specifications (15)
This includes;
-- Uses HashMap according to requirements
-- Uses TreeMap according to requirements
-- Code is compact and efficient.
-- BST class is generic and encapsulated.
```

I will run your program with the command line given above on three text files and compare your output to the specifications.

Outcomes

Once you have completed this assignment you will have:

- Used a hash table
- Used some parts of the Java Collections framework