Assignment 1
Avenger Favor (with ArrayLists)

Due Date: Monday Oct 5, before midnight
Weight: 7.5%
This assignment may be completed in groups of 2 to 3. Each group must submit a statement of contribution that clearly states the contribution from each team member.

Description

Jon wants to do some data analysis on the popularity of different avengers based on the frequency of appearance of the each avenger's alias (e.g., IronMan) or last name (e.g., Stark) on the internet on webpages such as Wikipedia. In this assignment you are to help him by creating a program that reads a stream of text from input, and outputs statistics on the frequency and order of appearance of avengers in the text.

For this assignment, your program will use some of the standard data structures and algorithms in Java, such as `ArrayList` and sorting and searching, to store and access the data. In the next assignments, you will use your own implementation of different data structures such as linked lists, trees, and hash tables to achieve similar goals.

This assignment will also introduce you to the world of running your program from the command line and using input/output redirection.

Your finished programs will be run like this:

```
cat input.txt | java -jar A1.jar > output1.txt
```

`input.txt` is any text file containing English text. Your program `A1` will read the standard input, one word at a time. For each word it reads, it will first clean-up the word (details provided below). Then it will check if the word is among the roster of avenger names and aliases. Your program will maintain a list of mentioned avengers and how many times their names or their aliases have occurred in the text.

The roster of avengers' aliases and last names is provided in the Implementation Details section and in the starter code. I have made sure all appearances of the avenger aliases in the input files are in one word (e.g., Iron Man will always appear as one word IronMan).

Here are the required steps with more details. For each word that your program reads from the input stream, it must do the following:

1. remove any trailing or leading blanks
2. convert the word to lower case
3. if there is an apostrophe in the word, use the substring before the apostrophe and remove the rest; e.g., "Stark's" would become "stark" "IronMan's" would become "ironman"
4. remove any punctuation or digits; e.g. "2banner" would become "banner"; "spider-man" would become "spiderman"; "rogers." would become "rogers"; "9,234" would be skipped
5. if the word is not empty at this point, then add to the total word count
6. Check to see if your list of avengers already contains this avenger, if yes, add to the previously existing object's frequency count, if it is not already on the list, add a newly created avenger object (making sure the frequency count is set to one, obviously).

Once all of the input has been read, `A1` will print:

- The total number of words in the input. This total should not include words that are all digits or punctuation.
- The total number of avengers mentioned in the input.
- The list of avengers in the order they appeared in the input stream.
- The four *most* frequently mentioned avengers ordered from most frequent to least. In case of ties, then in ascending alphabetical order of alias.
- The four *least* frequent avengers ordered from least to most. In case of ties, then in ascending order of last name length, then again in case of ties, in ascending alphabetical order of last name.
- All mentioned avengers ordered in alphabetical order of their alias (ascending order).

Important note: the output from your program must be ***exactly*** as shown in the examples, with the same spacing and formatting.

Implementation Details

There are always a number of ways any problem can be solved. For this assignment I will be very prescriptive.

- Create a class called `A1`. It will have the `main()` method, in which you should instantiate an `A1` object and then call run, and the `A1.run()` that does the bulk of the processing.
- Input is all from standard input. Create a `Scanner` object and use that for all input:

  ```
  Scanner input = new Scanner(System.in);
  ```

  All output should be to standard output. Use `System.out.print();` and `System.out.println();` for all output.
- Create an `Avenger` class to keep track of the number of times avenger last name or alias is mentioned. It must include the following private members: `heroName`, `heroAlias`, `frequency`. It should include code to maintain a count of how many times the avenger is mentioned by its last name or by its alias.
- The `Avenger` class must implement `Comparable`, and override the `equals()` and the `toString()` methods.
  - The implementation of the `compareTo()` method for the Comparable interface must order the Avenger objects in alphabetical order of alias.
  - The equals method must return true if two Avenger objects have the same alias.
  - The `toString()` method must print the following string:

    ```
    String format = heroAlias + " aka " + heroName
                    + " mentioned " + frequency + " time(s)";
    ```
- Your `A1` class should contain an `ArrayList` of `Avenger` objects.
- **Ordering and breaking ties:** To find the top four most popular or least popular avengers, I want you to use two different `Comparator`s and re-sort your `ArrayList`, using `Collections.sort()` then print the first four (if there are four) items.

  Remember that a `Comparator` must provide a *total ordering* for the objects. That includes some clearly defined behaviour in case of a tie. For example, if there are two words, 'foo' and 'bar' and they both occurred 10 times, which should come first in the list?
  - In the case of the top four most popular avengers, to break the ties the secondary ordering should be alphabetical, (a-z), by avenger's alias.

  - In the case of the four least popular avengers, to break the ties the secondary ordering should be in ascending ordering of the length of the last name (e.g., the length of "stark" is 5, and the length of "rogers" is 6). In case the length of the last names are the same, then break ties by ordering in alphabetical order of last name.

- **Testing and example files:** There are four sample input and output pairs named `input1.txt`, `output1.txt` etc. Use these to determine if your program is running correctly. You should use the `diff` command to compare your output to the samples. (Consult the instructions for Lab 1 for platform dependent details.)

  ```
  diff (cat sampleout.txt) (cat myout.txt)
  ```

  If `diff` gives no output, your program is working correctly.

  You should also devise a test plan and create a series of test files to fully exercise your program. For example, what about an empty file? or one with only some punctuation in it? Or a single word repeated 100 times? or ...?

  I will be evaluating the program against files you have not seen yet that will be meant to *test* your code.

- Here is the roster of avenger's aliases and last names:

Alias	**Last name**
captainamerica	rogers
ironman	stark
blackwidow	romanoff
hulk	banner
blackpanther	tchalla
thor	odinson
hawkeye	barton
warmachine	rhodes
spiderman	parker
wintersoldier	barnes

  I know this list is not comprehensive, and I know there may be debate in counting some of them as avengers. We will talk more about this in the next assignments.

  (for convenience, this roster is provided as an array of array of Strings in your starter code, where the first item in each inner array, i.e., avengerRoster[i][0] for $i \in \{0..9\}$ is the alias, and the second item, i.e., avengerRoster[i][1] for $i \in \{0..9\}$ is the last name.)

- This is a fairly short assignment. You really only need two classes (may be four, if you count the `Comparator` classes). Focus on writing compact, efficient code.

Documentation and Coding Standards

Your program should follow all of the coding rules and guidelines outlined in the document provided. In particular, include `Javadoc` style comments for all classes and substantial methods.

What to Hand In

Hand in a single file, `A1.jar`.

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

1. All of your `.class` and `.java` files.
   *****Make sure you include yoru java source files, otherwise I cannot mark your code for documentation and style components.***
2. A class called `A1` which has a `main()` method.
3. A class called `Avenger` which implements the Comparable interface.

Grading

A detailed grading rubric is provided at the end of this document.

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 standard Java data structures to solve a problem;
- Used polymorphism;
- Used the Java Comparable interface;
- Implemented a Comparator object;
- Used standard input and standard output re-direction.

Rubric

1. Documentation
   - (a) Java doc standards 5
   - (b) Format of Code 5
   - (c) Meets other rules/guidelines 5
2. Testing
   - (a) Test file1 10
   - (b) Test file2 10
   - (c) Test file3 10
3. Follows implementation specifications 30
   - Has two classes, A1 and Avengers with required members. (8)
   - Avenger class implements Comparable (6)
   - Avenger class overrides equals and toString. (6)
   - Implements and uses Comparator objects correctly. (6)
   - Code is compact and efficient. (4)