CS5112 and CS5052 Programming Assignment

Introduction

Gobbledegook Inc is a multinational company that offers users of its services a free *gPost* mailbox facility. However, the mailbox isn't really free. In return for the 'free' service users must agree to allow Gobbledegook Inc analyse all emails sent and received from or to the *gPost* mailbox.

Gobbledegook Inc analyses the text of the messages and builds word lists of the words used in the messages. The most frequently occurring words are then matched with marketing promotions to target users who may be interested in the products being marketed.

For the purposes of this programming assignment we are going to implement a simulation of the actual system. A sample of the messages sent to *gPost* accounts has been stored in a file called "**`gPostMessages.txt`**". Each message has the format

`gPostAddress` *`gPostBegin`* **`text of message`** *`gPostEnd`*

where

- The **`gPostAddress`** identifies the mailbox that sent the message.
- The labels *`gPostBegin`* and *`gPostEnd`* are used for the purposes of bracketing the message text (i.e. formally identifying the beginning and end of the message text).
- The **`text of the message`**. Note the text of the message may be any length and may span several lines. We will only concern ourselves with message content that we categorise as "words" (i.e. is wholly alphabetic containing only letters of the alphabet in upper, lower or mixed case). All other content will be ignored.

A typical message has an entry in the file as follows

> WillyWonka@gPost.com gPostBegin To find the golden ticket you have to buy a bar of chocolate :) Charlie's Granny and Grandad are hoping he gets a ticket but he only has enough money to buy 1 bar. I printed 5 tickets but my Oompa-Loompa workers made more than 1000000 bars :) gPostEnd

The file "gPostMessages.txt" is processed by reading the message entries and splitting them into the individual words. Noise words[1] (e.g. me, you, it, this, that, these, etc.) and other text that is not alphabetic are removed. The remaining words are stored in a list associated with the email address. Each entry in that list will contain a word and the number of times that word has occurred in messages sent or received by the user. For example, the list for *`WillyWonka@gPost.com`* would be identified by the name *`WillyWonka`* and might have a list of entries with a word and a corresponding count like the following

```
{WillyWonka -> {chocolate,1000}, {Oompa-Loompa,500}, {golden,300}, {ticket,300}}
```

When all the messages in the file have been analysed a report is produced listing all the *gPost* email addresses and, for each one, the top 10 words used and their frequencies. The **email addresses** are listed in **alphabetic order case insensitive** and for each email address the **words** are listed in **descending frequency order**. If two words have the same frequency they should be list in alphabetic order case insensitive.

You should use any structures provided by the Java Collections Framework that you think will assist the design and development of the software. You should structure the code so that it is reasonably easy to read and understand and include suitable comments where you believe they would assist a reader of the program.

[1] A full list of noise words is available in the file **NoiseWords.txt**.

Submission Requirements

You have the option of undertaking the project alone or as part of a team. If you wish to form a team please email Dermot.Shinners-Kennedy@ul.ie by **4pm Friday 25 March 2016 (GMT)** with the names and ID numbers of the team members. Teams are restricted to a maximum of four members. Management of the team is entirely a matter for the team members.

All team members are awarded the same score for the project. **The members of a team may be invited to attend a meeting to explain, justify or describe aspects of the solution.**

The end of semester closed book examination will include questions related to this programming assignment. Students who do not score at least 30 marks in the final examination may have their programming assignment score reviewed and a consequence of the review may be that they are **not** awarded the same mark as other members of the team.

An electronic copy of the solution must be emailed to Dermot.Shinners-Kennedy@ul.ie. The subject line must contain the text "JAVA Project Submission" typed EXACTLY as shown here with a single space between each word. NOTE: Your submission may be lost if you do not adhere to this requirement.

In the code submitted you should include prominent comments that contain the ID NUMBER(s) and NAME(s) of (all of) the author(s).

NOTE: ONLY .java files and any text files required should be submitted. No exe's, tar's or any other file type should be submitted. It is YOUR responsibility to ensure that the files submitted are the correct versions.

Submissions must be received on or before **4pm Friday 15 April 2016 (GMT).**