CSci 1933 Assignment 1

This assignment is due on **Friday, October 2, 2015** at **11:55 PM**

1. Introduction

Generally, this assignment is intended to provide a way for you to practice the basics of Java, including loop structures and reading data from a file.

1.1. Important notes:

- You must do this assignment in groups of 2. Only one member of the group needs to submit to Moodle site. Please submit to your own lab section Moodle site (even if the other group member is from a different lab section)
- You are not supposed to use any advanced data structure (e.g. Hashmaps), with the exception of arrays during any part of this assignment. This assignment will be a good thought exercise that will help you appreciate the value of these data structures. In case you are wondering, even Prof. Hecht did this as well when getting this assignment together.
- The output shown in the sample output throughout this assignment does not necessarily indicate what a correct solution to the assignment will produce. In general, the sample output is only intended to demonstrate the format of the output. **Please follow the output format closely, including spaces and line breaks to receive full points.**
- Assignments are structured with three levels: C, B, and A. The grading for the assignments will be done so that doing everything correctly for a particular level will result in a grade equivalent to that letter grade. The levels are also cumulative — to receive a B, you must do everything in both the C and B levels (usually, the work in each level will depend on the earlier levels).
- When an assignment description tells you to give something a particular name, you must give it **exactly that name**. Failure to do so may result in a very low or zero score due to inability to properly run your code.
- Your program must output its results in exactly the format we describe, with no additional text being written. If you insert additional output statements to aid in debugging, make sure they are removed before you submit.

2. Overview

Your task in this assignment is to write a program which analyzes and summarizes a large number of tweets taken from the public Twitter timeline. Specifically, the tweets for this assignment used the hashtag #webshop2011.

2.1. File format

The data is stored in a text file called `tweets.dat` with each tweet recorded on a separate line. Each record consists of a username (a string), followed by the text of the tweet (a string). A sample file looks like this:

```
marybethray  Of interest to #webshop2011 http://t.co/He1ujv5
wyoumans     #webshop2011 is not the same without @katypearce
```

3. Getting started

- ➢ In IntelliJ, create a new Java project and add the correct JDK to this project.
- ➢ Under this new project, right click on `src,` select New|Package. Create a package named "edu.csci1933"
- ➢ Import the following three files into your project (files are provided with this assignment):
  - `tweets.dat` (sample data file; create a folder with name "dat" in your project root folder and place this file in "dat" folder)
  - `TweetReader.java` (import this into your `edu.csci1933 package`)
  - `build.gradle` (gradle build file, place it under your project root)
- ➢ Once you have imported the files, create a new class in your `edu.csci1933` package called `TweetProcessor`, and with a `public static void main` method.

4. Overall

- ➢ Throughout this assignment, you will be operating on twitter data. Twitter users subscribe to the tweets of other users by "following" them. You will be reading Twitter data from a file, and not directly from Twitter, so we will simulate "following" data. Specifically, for this assignment, you can assume that you are following (and are thus interested in) these 10 users:
  - lorenterveen
  - pjrey
  - katypearce
  - wyoumans
  - caseyspruill
  - pstamara
  - jvitak
  - sharoda
  - MO_GY
  - barrywellman
- ➢ Portions of this assignment will refer to the people you're following, and for those portions, please refer to the usernames listed above.
- ➢ Kindly note that you are likely to find it useful to keep track of the number of users you are following in a variable. This might look something like:

```
int NUM_USERS_FOLLOWING = 15;
```

You'll obviously want to modify this to match the number of users you're actually following

Note The output shown does not necessarily indicate the correct values for your own output, but is intended to demonstrate the proper format that your output should be in. Do not assume that because your values do not match the values shown in the sample output, that your output is wrong.

5. C level

- ➢ To receive a C, you must write the code to read in the data file and print the usernames of the people you follow, the number of tweets that each of these users has made, as well as the percentage of tweets that this user produced (out of the whole collection of tweets in the dataset).
- ➢ The `TweetReader.java` file contains a class we provide, `TweetReader`, which will help you read the data from the file. Later, you will learn how to do the file I/O directly, but for now it hides some Java details that we haven't covered yet so you can complete this assignment in a straightforward fashion.
- ➢ You need to do the following in your `main` method in the `TweetProcessor` class:

1. Create a new variable to hold a `TweetReader`.
2. Initialize this variable to hold a new `TweetReader` object which can read from the `tweets.dat` file.
3. Write the code to read each tweet from the file, count the number of tweets each of the users you're following has made, and print it. It should print in the following format (see Tips section for easy formatting the print):

```
username - # of tweets made - % of total tweets (rounded
to the nearest whole percent).
```

- ➢ For this assignment, section of the assignment, you're calculating the percentage of tweets out of the whole file that the users you're following have made. Therefore, when calculating percentages, be sure to consider all of the tweets in the file, not just the ones made by the users you're following. For the purposes of the assignment, please round all percentages to the nearest whole percent. This can be done using the `Math.round()` method.

5.1. Creating new objects

- In order to do anything with most classes, you must *instantiate* them; that is, create a new object from them. In Java, you do this with the `new` keyword. For example, to create a new object of class `Color`, you could do the following:

  ```
  Color c = new Color();
  ```

  The `Color()` part is just like a method call; you can put arguments in the parentheses. It calls a method called a *constructor*, which allows initial setup of the object.
- The constructor for the `TweetReader` class requires an argument — a string — which is the filename it will read from.

5.2. Using the **`TweetReader`**

- The `TweetReader` class allows you to read from a data file, one line (or *record*) at a time. It has a notion of a "current record", and allows you to get data from the current record. When it is first created, it has no current record.
- The `advance()` method of `TweetReader` advances the current record by 1 record. The first call to `advance()` makes the first record the current record. It also returns a boolean value —`true` if it found another record, or `false` if it reached the end of the file.
- The `getTweeterID()` method gets the Twitter ID (or username), as a String, from the current record. The `getTweet()` method returns the tweet text for that particular tweet, but does not include the username. Neither of these methods can be called before `advance()` is called at least once, and neither of them can be called after `advance()` has returned `false`.

  Note If you violate the rules regarding calling `advance()` and getting record information, your program will terminate with a `RuntimeException` stating that the tweet reader is not in a valid state.
- Thus, the basic structure of code to process the records in the file looks like this (in pseudocode):

  ```
  while advance() returns true
      process a record
  ```

5.3. Outputting data

- The output in order to achieve a C grade should look like:

  ```
  katypearce - 138 - 10%
  wyoumans - 53 - 5%
  caseyspruill - 5 - 0%
  pstamara - 193 - 15%
  ```

➢ Java provides an object, `System.out`, which lets you print data to the standard output stream (which goes to your console in Eclipse, or to the terminal from which you ran the program if you're using a command line). There are a couple methods of interest here:

  `print`
    This method prints a string or other data object.

  `println`
    This method takes a string (or other data, such as an integer) and prints it, followed by an end-of-line character.

  `format`
    This method takes a string followed by several other parameters and prints the string, after inserting the other parameters in place of certain character sequences in the initial string. It is used for nicely formatting output. See Tips section of this document for more information.

➢ Look at the `PrintStream` class in the Java API reference for information on more options for outputting data.

6. B level

➢ In the C-Level solution, you provided some very brief information about the tweets of the people you follow. Now we'd like you to delve a bit deeper into understanding the behavior of the users you follow. Counting the number of tweets made can be useful, but how many of those tweets are original? On Twitter, there's an accepted practice called "retweeting", or essentially reposting what another user has posted, while providing recognition to that other user, via their username. When a user retweets another user, it will generally look like:

```
lorenterveen RT @benbendc: Summer Social #Webshop2011 http://t.co/uVjGuKC on the Computing Community Consortium Blog from Computing Research Association
```

In other words, the user lorenterveen retweeted something benbendc (another user) said.

➢ In order to begin to better understand the tweeting behavior of the users you're following, add code to your Level C solution to compute two additional pieces of information:

- How many times the people you're following retweeted someone else, and
    - For the purposes of the assignment, when counting how many times a user (eg. lorenterveen) makes a retweet, a retweet is defined as: containing the characters "RT" (without quotes) anywhere in the String
- How many times the people you're following were retweeted
    - For the purposes of the assignment, when counting how many times the people you follow are retweeted by someone else, a retweet is defined as: the text of the tweet must contain "RT @[username]" (without quotes) at the beginning of the

tweet text, where [username] is that of the person you're following (for example, "RT @lorenterveen").

➢ Your output should add both of these new computations to the end of each line, separated by a dash, as follows:

```
username - total tweets - percentage - times this user
retweeted another - times this user was retweeted
```

➢ Output for a B-Level solution will include the output from the C-Level solution, but will also print data to meet the two new requirements. Sample output will look like:

```
katypearce - 138 - 10% - 20 - 28
wyoumans - 23 - 1% - 1 - 7
caseyspruill - 5 - 0% - 2 - 3
pstamara - 193 - 15% - 16 - 4
```

7. A level

➢ Most of you probably know that various social media sites (including Twitter) make money out of the users' activity. But, do you know this happens other way round too? Click here to have a look. This segment of the assignment will concentrate on building such user-rewarding model.

➢ Suppose that the Twitter account that is following the 10 users listed above is owned by a social media analysis company, and you work for the company. The company wants to reward very active Twitter users, and in particular, those with a lot of Twitter influence (that your company happens to be following). In order to do this, your company is willing to pay influential Twitter users a small amount per tweet.

➢ For this level, you need to implement a cost calculation to account for your company's business plan. Specifically, account for the following price structure:

1. $0.05 for any time one of the followed users retweets (makes a tweet retweeting another user)
2. $0.10 for all other, non-retweet tweets by the followed users.
3. $0.25 for every time one of the followed user's tweets is retweeted.

➢ Additionally, your company would like to institute a Top Tweeter reward, where they will pay another $0.05 per tweet by the user who was the top earner according to the previous payment plan. That is, the user who has earned the most money using the previous 3 criteria will be paid another $0.05 for each of his or her tweets.

➢ However, because your company is going to be spending money, they must have a breakdown of where their money is going. In particular, they'd like to see some overall statistics about the cost to your company, in addition to some user-specific information. Please represent all monetary values in the form for `$#.##` (that is, $1.00 instead of $1).

➢ Your output for an A-Level grade must include the following overall cost data for the tweet-incentive program:

- Overall cost spent on original tweets of followed users

- Overall cost spent on retweets of others tweets by followed users
- Overall cost spent on followed users being retweeted
- Overall prize cost to the Top Tweeter

➢ You also must print the following information for each user your company is following:
- Amount earned due to original tweets
- Amount earned by retweeting others
- Amount earned by being retweeted
- Amount earned from the Top Tweeter reward, if applicable

➢ Example output for an A-Level grade will look like:

```
Overall cost: $79.00
  Overall cost of original tweets: $30.00
  Overall cost of retweeting others: $3.00
  Overall cost of being retweeted: $37.00
Cost of Top Tweeter award: $9.00

lorenterveen - 40 - 2% - 10 - 20
  Tweets: $3.00 Retweets: $0.50 Retweeted: $5.00
pjrey - 123 - 8% - 16 - 49
  Tweets: $10.70 Retweets: $0.80 Retweeted: $12.25
katypearce - 180 - 12% - 25 - 38 - Top Tweeter
  Top   Tweeter:   $9.00   Tweets:   $15.50   Retweets:   $1.25
Retweeted: $9.50

...
```

You do not need to deal with the case where 2 or more people are Top Tweeters.

8. Submitting your assignment

➢ Before you submit your assignment, **you must create a text file called group.txt in your src/ directory**. In this file, put the names and x.500 ID's of the members of your group.

➢ If you are new to Gradle, first go through Lab 2 instructions about Gradle. To check if your code works properly, you need 1) go to the project root directory in Terminal 2) run "`gradle run`" task of Gradle. Then run the "gradle tar" task of Gradle to create a compressed file. Submit this compressed file in moodle.

Tips: using `format`

➢ The `format` method of the `PrintStream` class allows you to format output by having Java substitute various values for string sequences in a template (called a "format string”). The first parameter to `format' is always a string, the format string. The data that is to be substituted into the format string is passed as the remaining parameters.

- The format string contains special codes beginning with the “%” character to indicate substitutions. Some important ones are “%s”, which tells Java to insert a string, “%d” for decimal integers, and “%f” for floating-point (`float` or `double`) numbers. If you want to include a literal “%” sign in the output, use “%%”.
- Each of the “%” sequences that inserts a value takes a parameter. The parameters are used from left to right. For example, the following call:

  ```
  System.out.format("The square root of %d is %f\n", 5, 2.236068);
  ```

  results in the following output:

  ```
  The square root of 5 is 2.236068
  ```

- `format` does **not** automatically generate a new line at the end of the string; the “\n” sequence indicates the newline character and causes the new line to be printed.

Reflections

This section will not be graded, but as a way to begin to think about future concepts that will be covered in class, please consider the following questions:

1. What if you were doing this assignment on every user in the `tweets.dat` file, instead of a subset of 10? What types of things might be different?
2. What if you wanted to print user data in a specific order (for instance, by the amount the company owed a user)? What if sometimes you wanted to print the data by the amount owed, but other times you wanted to print by the total number of tweets? Consider ways you might go about doing this. What types of advantages or disadvantages might these ways have?
3. What errors could have occurred while you were writing and running this program? What errors or problems did you encounter? What do you think caused these errors or problems to occur? How would you make sure you don't run into the same problems in the future?

---End of Assignment 1---

References:

Prof. Loren Terveen’s curriculum.