Approach

The idea for me was to find a simple and easy to understand solution that provides the required result. In my solution the users found in the (user.txt) are key to solving this task. I therefore decided to make use of a <key, value> pair approach in which the user name acts as the key. A Set data structure is used as the value in which that users' (key) followers are stored. A user (Key) can have 0 to many followers which will be stored in their respective Set. The set data structure deals with duplicates as the values are read in.

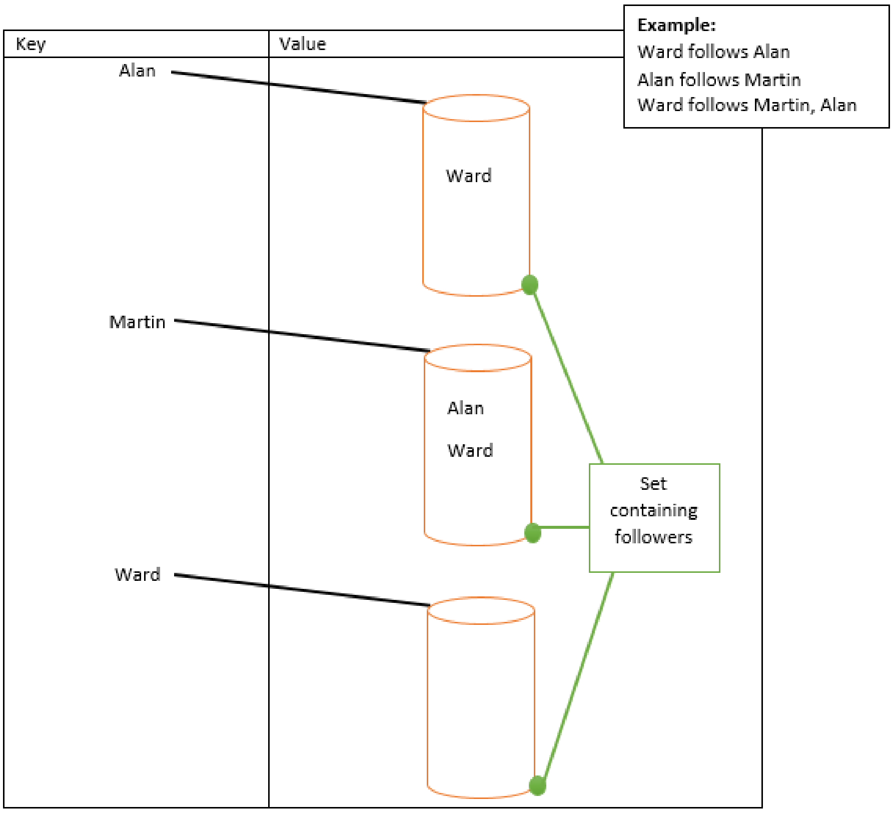
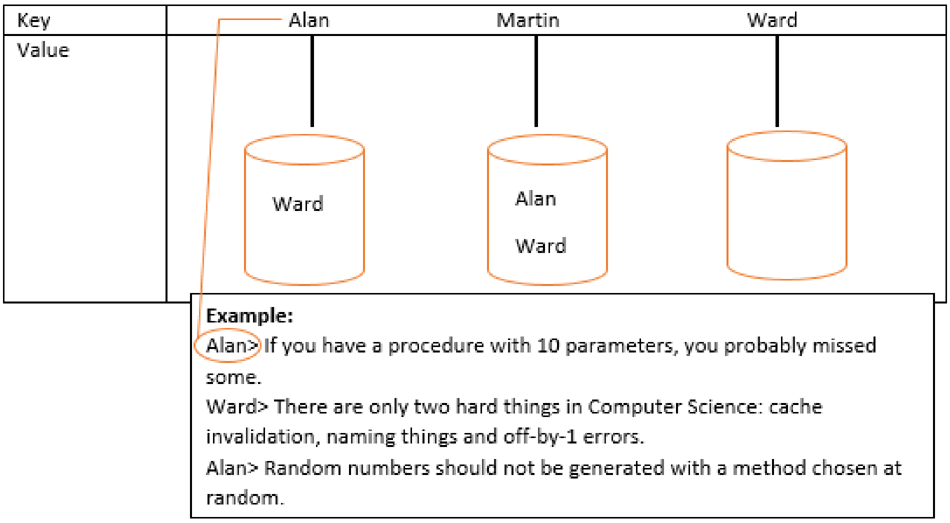


Figure 1: Sorted Map containing users as Keys and set containing followers as the value

Displaying the result

After creating a map containing all the users and their respective followers, displaying the messages was done by iterating through each key and its respective values. The twitter handle attached to the message (i.e. before the '>') was used to connect messages with the person who sent them (see figure 2)



Example:
Alan> If you have a procedure with 10 parameters, you probably missed some.
Ward> There are only two hard things in Computer Science: cache invalidation, naming things and off-by-1 errors.
Alan> Random numbers should not be generated with a method chosen at random.

Figure 2: Linking of tweets to users and followers using twitter handles

Once a handler has been attached to a key. The message is then displayed on the keys feed and on the feed of the respective followers. As seen in figure 2, the message sent out by Alan will be displayed on Alans' feed since his key matches the handle and on Wards Feed since he follows Alan.

Assumptions

1. The program must read from multiple text files. The assumption is therefore that this solution is only a test run and for that reason will be limited to a single thread (i.e. the main thread).
2. The textiles only contain 7 bit ASCII characters. Therefore will not test if characters are 7 bit ASCII
3. The Messages contained in the tweet.txt file are stored solemnly for the purpose of viewing. There is no need to store them over a longer period of time. The way in which messages are displayed is highly dependent on the mapping of the users.
4. Users are unique (i.e. no user can have the same user name as another)
5. User names in the user.txt file cannot contain a space ' ' as part of the word itself.
6. Only a comma can be used as the splitting criteria for the purpose of separating users.
7. User names are allowed to contain character values such as (" : ; _) as part of their handles. For example: The user name Spha_;) is acceptable.
8. The character '>' is used solemnly for the purpose of separating a handle from its message. (.i.e. a message does not contain this '>' character).

Motivation for chosen data structures

SortedMap

Part of the challenge was to ensure that that users are sorted alphabetically in ascending order, I therefore decided to use a sortedMap interface which implements a TreeMap from the Java collections framework. The order can also easily be altered to descending order using TreeMap.descendingKeySet(), should the need arise. This allows for the entries to be sorted in ascending order as they are read in. By using this data structure, I have the ability to create a mapping of users and followers as I read each user value in. This also make the process of attaching the tweets to the users easier as the key value is used to map messages.

Set

The set interface refers to a collection that models the set abstraction used in mathematics which denotes an environment in which duplicates cannot exist. This way even though the program can read in multiple instances of Alan for example, once Alan has been stored, a duplicate cannot exist with the same name. (See figure 1). This allowed me to address the issue of duplicate entries as the values were being read in.