Twitter Filter Builder

System Design Specification

Table Of Contents

Contents

Executive Summary

The average North-American spends ~2 hours a day on social media. Studies have found that ~3% of posts on Twitter contain abusive language. Additionally, false news stories are 70% more likely to be retweeted than true ones[1][2]. This project seeks to reduce the amount of toxic and deceptive content Twitter users are forced to interact with on their home feeds by providing them with a tool to identify and block offenders by keyword. This project uses the data curated by the ArchiveTeam Twitter Stream to examine recent years of Twitter activity and help users create a more personalized Twitter experience. The user can input keywords into the tool, and a target search area (user biography, tweets, or both) and the tool will search through all users provided in the dataset. It will then return and display a list of users associated with the input, and after confirmation from the user the members in the list are automatically blocked.

[1] https://www.amnesty.org/en/latest/research/2018/03/online-violence-against-women-methodology/
[2] https://www.statista.com/statistics/433871/daily-social-media-usage-worldwide/

Class Implementations

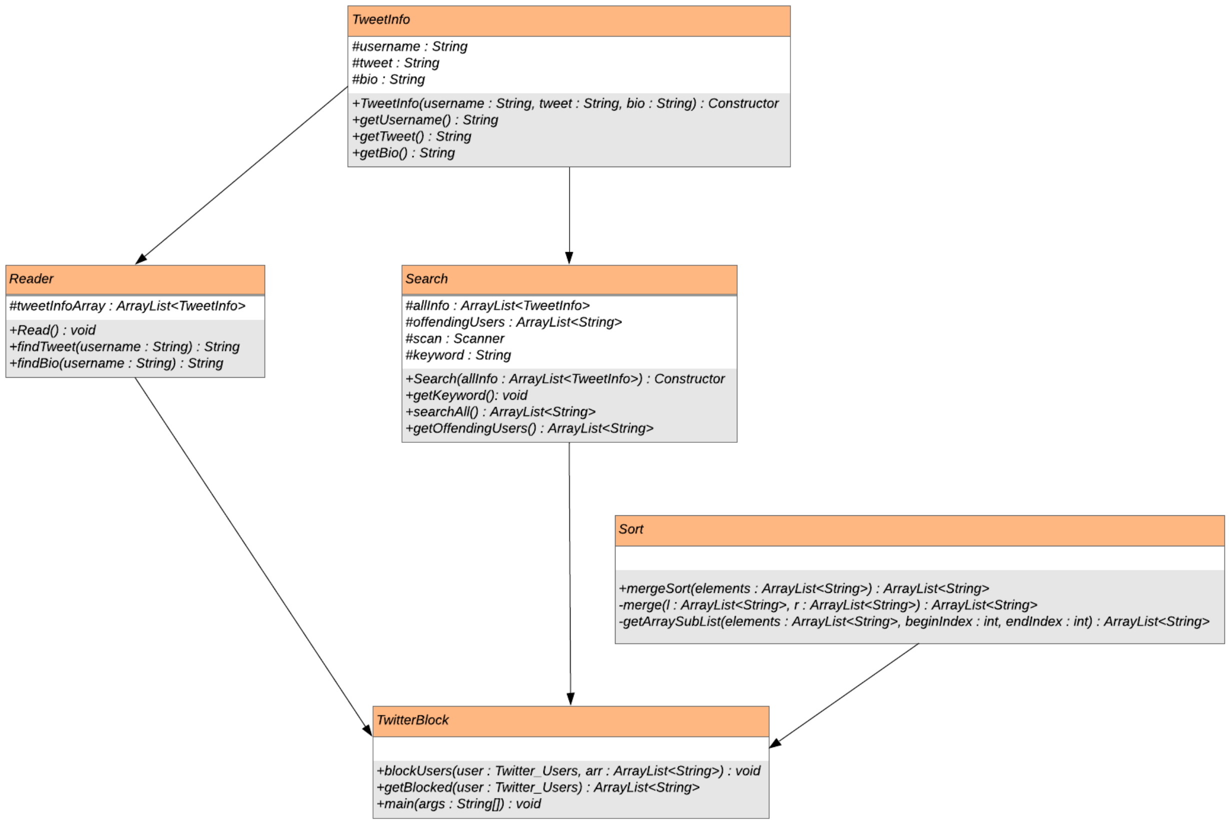
Overview:

The tool is broken down into the modules to adhere to the principle of modularity

- Reader.java
- Search.java
- Sort.java
- TweetInfo.java
- TwitterBlock.java

This program has both high cohesion and low coupling, meaning that as a whole, code that is closely related is written as one component. Combined, they are able to perform the task at hand. However, the modules are not dependent on each other, and each of these components are fully functional on their own. Independently, the reader can still be used to read values in a data set, the search and sorting algorithms can still be used, and so on.

UML:


TweetInfo
#username : String
#tweet : String
#bio : String
+TweetInfo(username : String, tweet : String, bio : String) : Constructor
+getUsername() : String
+getTweet() : String
+getBio() : String
Reader
#tweetInfoArray : ArrayList<TweetInfo>
+Read() : void
+findTweet(username : String) : String
+findBio(username : String) : String
Search
#allInfo : ArrayList<TweetInfo>
#offendingUsers : ArrayList<String>
#scan : Scanner
#keyword : String
+Search(allInfo : ArrayList<TweetInfo>) : Constructor
+getKeyword(): void
+searchAll() : ArrayList<String>
+getOffendingUsers() : ArrayList<String>
Sort
+mergeSort(elements : ArrayList<String>) : ArrayList<String>
-merge(l : ArrayList<String>, r : ArrayList<String>) : ArrayList<String>
-getArraySubList(elements : ArrayList<String>, beginIndex : int, endIndex : int) : ArrayList<String>
TwitterBlock
+blockUsers(user : Twitter_Users, arr : ArrayList<String>) : void
+getBlocked(user : Twitter_Users) : ArrayList<String>
+main(args : String[]) : void

Reader.java

Overview:

- Class that reads information from a CSV file

Requirement trace back:

- This class satisfies the requirement to read data from a database

Local Variables:

- ArrayList<TweetInfo> tweetInfoArray = **new** ArrayList<>()

Methods:

- **public void** Read() **throws** IOException
  - *Method that reads the information from a CSV file and adds it to an ArrayList of TweetInfo tweetInfoArray*
- **public** String findTweet(String username)
  - *Method to find a tweet in the generated tweetInfoArray*
- **public** String findBio(String username)
  - *Method to find a bio in the generated tweetInfoArray*

Description:

This class reads the information stored in a database, and parses the information needed storing it into an arraylist called tweetInfoArray. This generated arraylist will be the target of the search and sort algorithms used to extract users matching the block conditions.

Search.java

Overview:

- This class parses the dataset values and formats it into an arraylist

Requirement trace back:

- This class satisfies the requirement to have a searching algorithm

Local Variables:

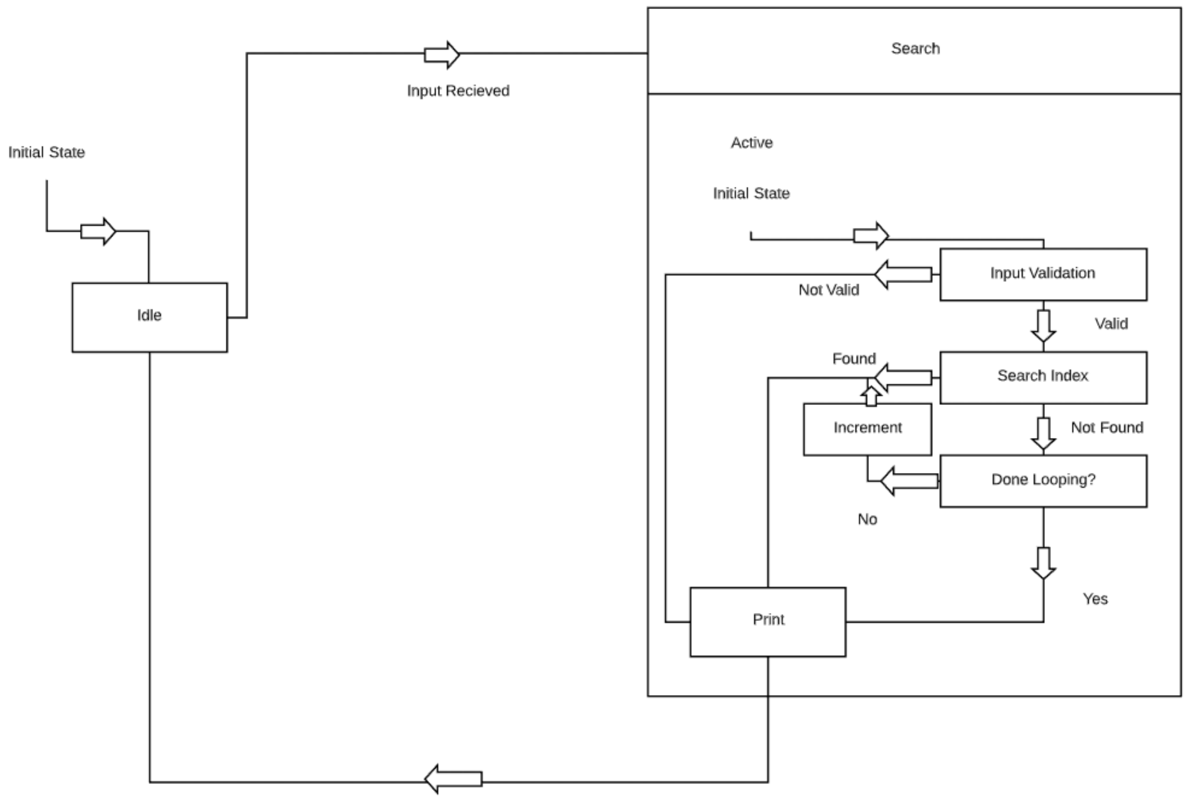
- ArrayList<TweetInfo> allTweets = **new** ArrayList<TweetInfo>()
- ArrayList<String> offendingUsers = **new** ArrayList<String>()
- Scanner scan = **new** Scanner(System.in)
- String keyword = "”

Methods:

- **public** Search(ArrayList<TweetInfo> allTweets)
  - *This method is the constructor for a Search instance*
- **public void** getKeyword()
  - *This method obtains user input for the keyword that will be searched*
- **public** ArrayList<String> searchAll()
  - *This method searches through the ArrayList and returns all searched matching the query*
- **public** *ArrayList<String> getOffendingUsers()*
  - *This method returns the list of offendingUsers*

Description:

This class reads an ArrayList of TweetInfo generated by the Reader class and searches each TweetInfo instance in this ArrayList for the keyword obtained from user input. It searches both in the biography, as well as user tweets and returns an ArrayList of users matching the search criteria.



Sort.java

Overview:

- This class *sorts elements in an ArrayList*

Requirement trace back:

- This class satisfies the requirement to have a sorting algorithm

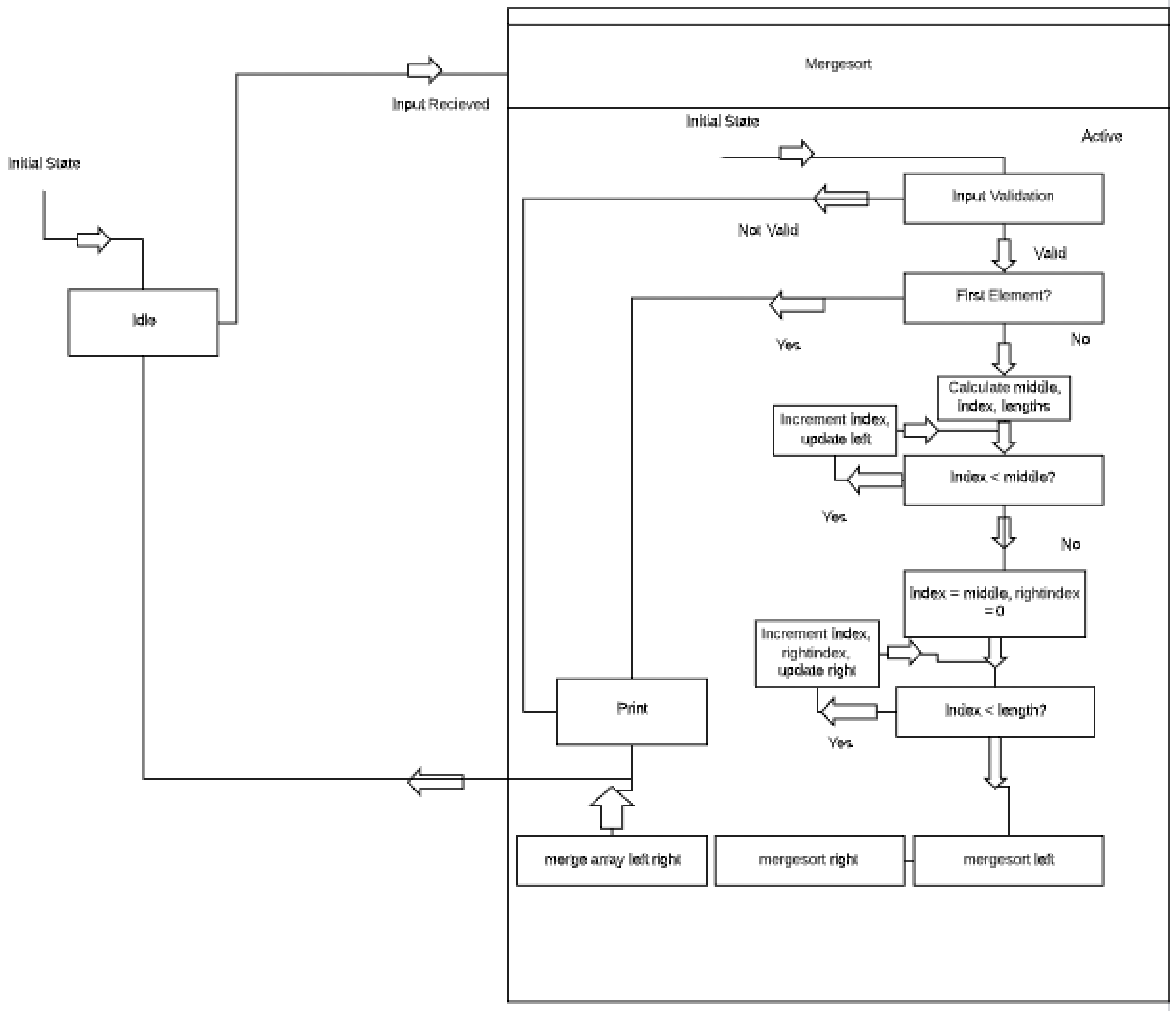
Local Variables:

Methods:

- **public** ArrayList<String> mergeSort(ArrayList<String> elements)
  - *This method is the constructor for a mergeSort instance*
- **private** ArrayList<String> merge(ArrayList<String> l, ArrayList<String> r)
  - *This method is a private method to merge two ArrayLists*
- **private** ArrayList<String> getArraySubList(ArrayList<String> elements, int beginIndex, int endIndex)
  - *This method is used to retrieve a sublist in an ArrayList*

Description:

After a list of user's is generated, or for other utility purposes this class will sort an inputted ArrayList of strings. It uses the merge sort algorithm, with the methods merge and getArraySubList being used within the merge sort itself. The returned sorted list is a new ArrayList.


Mergesort
Input Recieved
Initial State
Initial State
Active
Idle
Input Validation
Not Valid
Valid
First Element?
Yes
No
Calculate middle, index, lengths
Increment index, update left
Index < middle?
Yes
No
Index = middle, rightindex = 0
Increment index, rightindex, update right
Index < length?
Yes
Print
merge array left right
mergesort right
mergesort left

TweetInfo.java

Overview:

- This class stores relevant tweet information into a TweetInfo object

Requirement trace back:

- Although this class does not directly satisfy a requirement, it is used to facilitate management of the dataset

Local Variables:

- String username = "";
- String tweet = "";
- String bio = "";

Methods:

- **public** TweetInfo(String username, String tweet, String bio)
    - *This method is the constructor for a TweetInfo instance*
- **public** String getUsername()
    - *This method returns the username of the TweetInfo instance*
- **public** String getTweet()
    - *This method returns the tweet of the TweetInfo instance*
- **public** String getBio()
    - *This method returns the biography of the TweetInfo instance*

Description:

This class is simply an ADT for a TweetInfo object. It is used to store only needed information pertaining to a user (username, tweet, and biography) obtained from the dataset. The module also contains three getters for username, tweet and biography. Having this ADT simplifies data retrieval, and allows us to run the reader a minimal number of times.

TwitterBlock.java

Overview:

- This class authenticates users to the application and blocks users for the client

Requirement trace back:

- This class satisfies the requirement to reduce toxicity on twitter by removing undesired tweets from a user's feed

Local Variables:

Methods:

- **public static void** blockUsers(Twitter_Users user, ArrayList<String> arr)
    - *This method blocks users that use a given keyword*
- **public static** ArrayList<String> getBlocked(Twitter_Users user)
    - *This method obtains the list of users already being blocked by the client*

Description:

This class connects the client to the actual Twitter API. After searching through the list of Twitter users and obtaining the sorted list of users to be blocked, the API will iterate through the list of users and block each one individually. It can also return a list of blocked users.