POLYTECHNIC UNIVERSITY OF TURIN

Twitter Squawk

by

Montalto Monella Marco

Final project submitted in fulfillment for the
Laurea in Ingegneria Gestionale L-8

in the

Engineering faculty
DIGEP - Dipartimento di Ingegneria Gestionale e della Produzione

June 2015

Declaration of Authorship

I, Marco Montalto Monella, declare that this final project titled, 'Twitter Squawk' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a Laurea degree at this University.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: 

Date: 19 June 2015

"It's supposed to be automatic, but actually you have to push this button."

John Brunner

Acknowledgements

I take this opportunity to express gratitude to all of the Department faculty members for their help and support. I am grateful to Fulvio Corno, lecturer, in the Department of Ingegneria Gestionale. Moreover in summer 2014 I had a wonderful experience in financial markets and a special thanks goes to all the Amplify Trading mentors who taught me the fundamentals behind the traders word. I am extremely thankful and indebted to all the above mentioned, for sharing expertise and valuable guidance. Last but not least, I also thank my parents for the unceasing encouragement, support and attention.

Contents

Declaration of Authorship i

Acknowledgements iii

List of Figures v

Abbreviations vi

1 Introduction 1
- 1.1 Getting started . . . 1
  - 1.1.1 Requirements . . . 1
  - 1.1.2 Twitter Application Management . . . 2
- 1.2 Twitter in the financial industry . . . 2

2 Graphical User Interface 4
- 2.1 Menu . . . 4
- 2.2 Active filters . . . 4
- 2.3 Add filters and Remove all filters . . . 5
- 2.4 Tweets source . . . 6
- 2.5 Audio . . . 6
- 2.6 Catch window . . . 7

3 Data structures 8
- 3.1 Speaker . . . 8
- 3.2 Tweets buffer . . . 9
  - 3.2.1 Buffer refresh . . . 9
- 3.3 Twitter4j . . . 9

List of Figures

1.1 Official logo 1
1.2 Twitter Application Management page 2

2.1 Menu bar 4
2.2 Active filters 5
2.3 Add filters 5
2.4 Remove all filters 6
2.5 Tweets Source 6
2.6 Audio options 7
2.7 Catch window 7

3.1 Audio class diagram 8

Abbreviations

TS	**T**witter **S**quawk
PMI	**P**urchasing **M**anagers' **I**ndices
CSV	**C**omma **S**eparated **V**alues
TTL	**T**ime **T**o **L**ive
API	**A**pplication **P**rogramming **I**nterface

Chapter 1

Introduction

The focal point of this application is to filter tweets by string matching and optionally read them vocally.

1.1 Getting started

In order to use TS, the user has log into his own Twitter Developer account (https://dev.twitter.com) and create a new app from the Application Management page (https://apps.twitter.com).

1.1.1 Requirements

Operating System OS X Mountain Lion or above

Account A Twitter developer account

FIGURE 1.1: Official logo

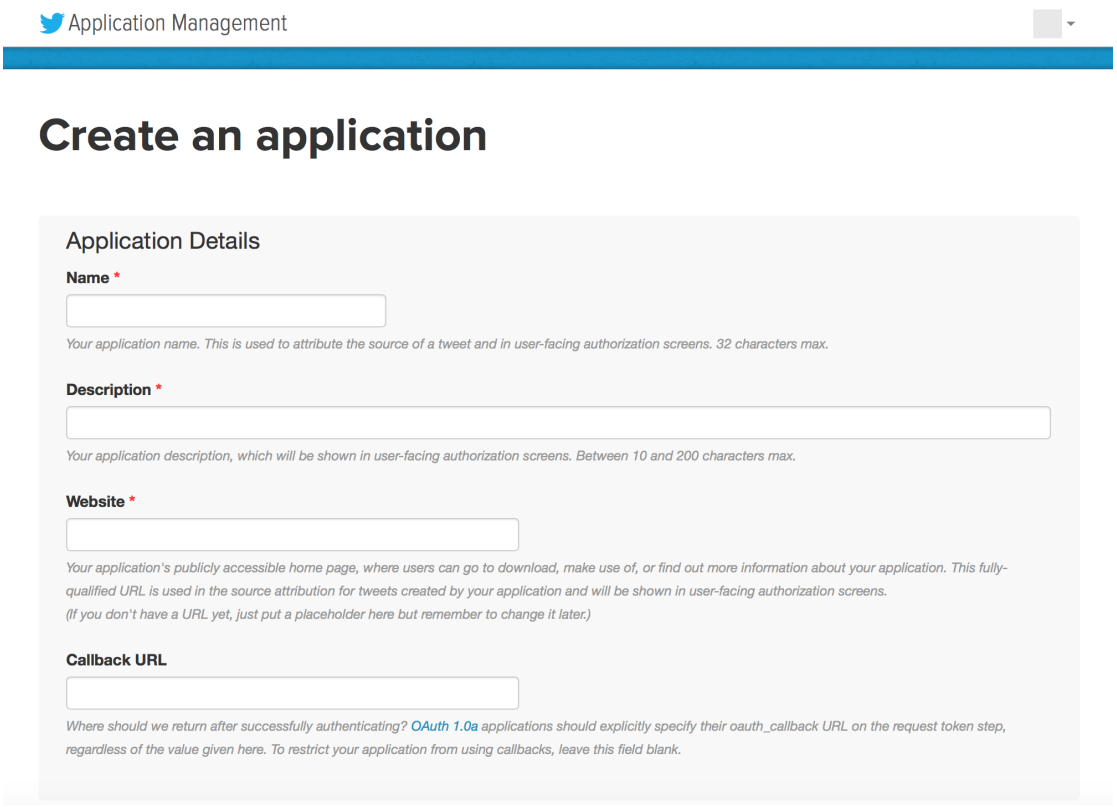

FIGURE 1.2: Twitter Application Management page

1.1.2 Twitter Application Management

Press the *Create New App* button and you will see the page in Figure 1.2. For testing use only, put a placeholder in the *Website* text field.

1.2 Twitter in the financial industry

For the first time in history, access to the observations, wisdom and emotional reactions of millions of people globally is available in real-time. Social media data represents a collective barometer of thoughts and ideas touching every aspect of the world. As social platforms increasingly become a primary means of communication for our age, asset managers, equity analysts and high frequency traders are incorporating leading indicator data from these platforms into investment decisions as a means to create alpha[1].

[1] Alpha is one of the five key measures in modern portfolio theory.

Most of the daily economic data, such as inflation rates and PMIs[2], break on Twitter immediately after the announcement, but up to 3 minutes before the official mainstream media. It is clear that social media is a real-time news-wire which can be harnessed for information to trade on. In the TS main window, an economic calendar can by found following the menu path: Help ⇒ Useful webpages ⇒ Trading Economics Calendar (http://www.tradingeconomics.com/calendar).

[2]economic indicators derived from monthly surveys of private sector companies.

Chapter 2

Graphical User Interface

2.1 Menu

TS ⇒ About TS A brief information window is shown.

TS ⇒ Quit TS Quit the application.

User ⇒ Logout Delete the stored consumer key and consumer secret used for the OAuth Protocol[1].

View ⇒ Show tips Tip labels are shown on the current window (if any).

Help ⇒ Useful webpages Open the default browser on the selected page.

Help ⇒ Send feedback Open the default mail user agent.

FIGURE 2.1: Menu bar

2.2 Active filters

This is an alphanumerical ordered list of the words inserted by the user.

To remove a single word, double click on it.

[1] See RFC 5849

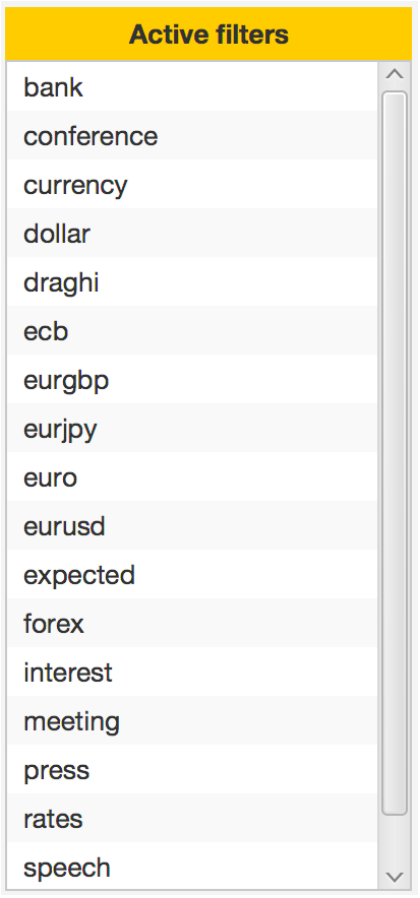

FIGURE 2.2: Active filters

2.3 Add filters and Remove all filters

Add words to perform as filter during the tweets catch. Should be inserted in a CSV format. *Remove all filters* clears the set containing the active words and refreshes the list view.

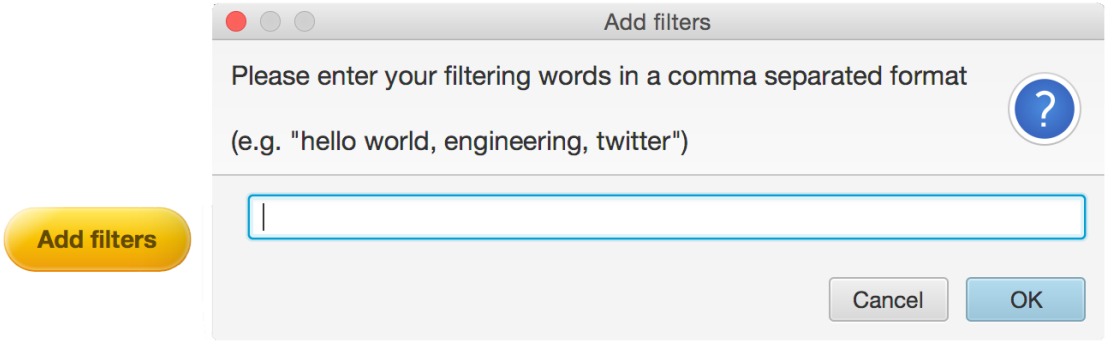

FIGURE 2.3: Add filters

FIGURE 2.4: Remove all filters

2.4 Tweets source

Tweets can be caught from the following sources:

Followings From the people that the user is currently following.

Global From a larger sample of all the global tweets.

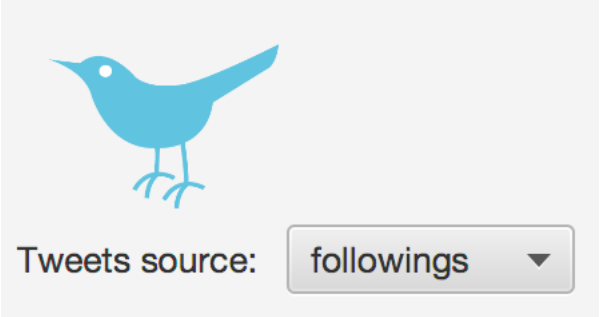

FIGURE 2.5: Tweets Source

2.5 Audio

By selecting the audio radio button, the user can choose among these four options:

Do not read links This option is useful if you do not want to listen to the spelling of URLs present in the tweet.

Do not read hashtag If activated hashtags will not be read. Otherwise the character "#" will be pronounced as "hashtag".

TTL After having read the newest one, only tweets at most T seconds/minutes old will be read. The time T can be set between 0 and 99 by the user.
Special maximum value: empty field corresponds to 24 hours.

Speech rate Choose among *Slow* (160 WPM[2]), *Regular* (200 WPM) and *Fast* (280 WPM).

[2]words per minute

FIGURE 2.6: Audio options

2.6 Catch window

While capturing, the sequence of filtered tweets can be seen on the left. The user who post the tweet which is currently being read can be identified by his/her Twitter profile picture displayed in the top-center and his/her description in the bottom-center.
To stop the catch just press the button and patiently wait for the disconnection which could take up to 5 seconds.

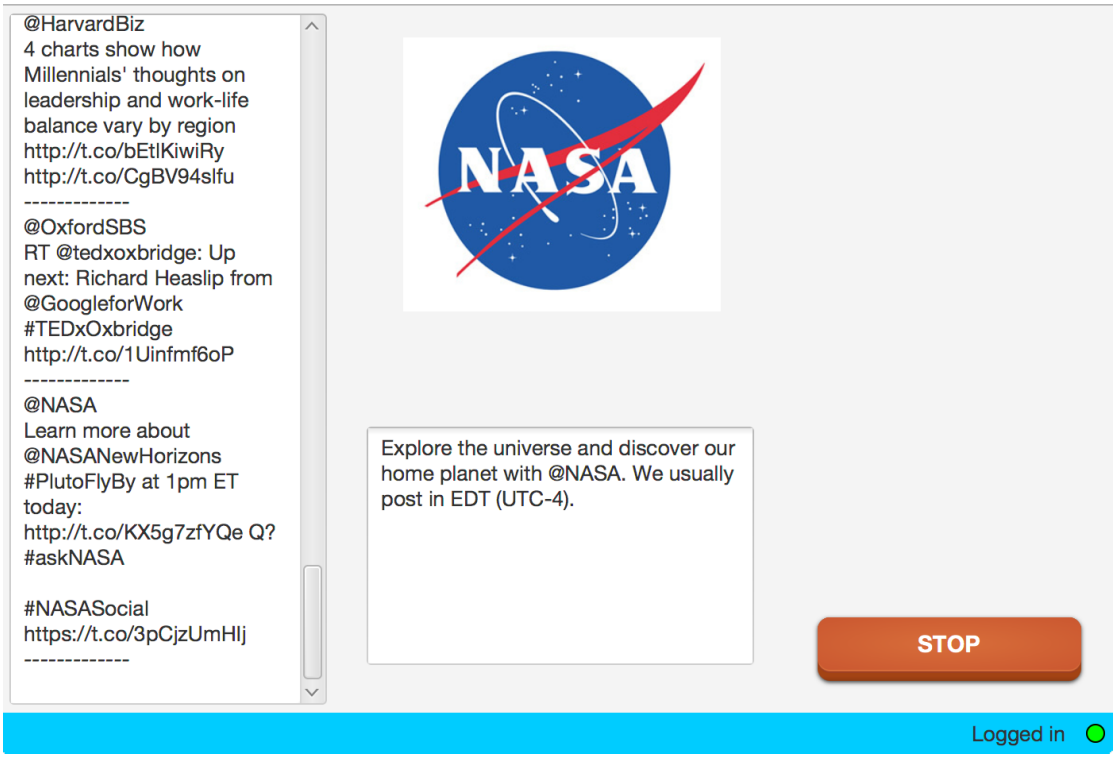

FIGURE 2.7: Catch window

Chapter 3

Data structures

3.1 Speaker

The Speaker class have two main roles: to clean up the buffer from the old tweets and vocally read the most recent one.

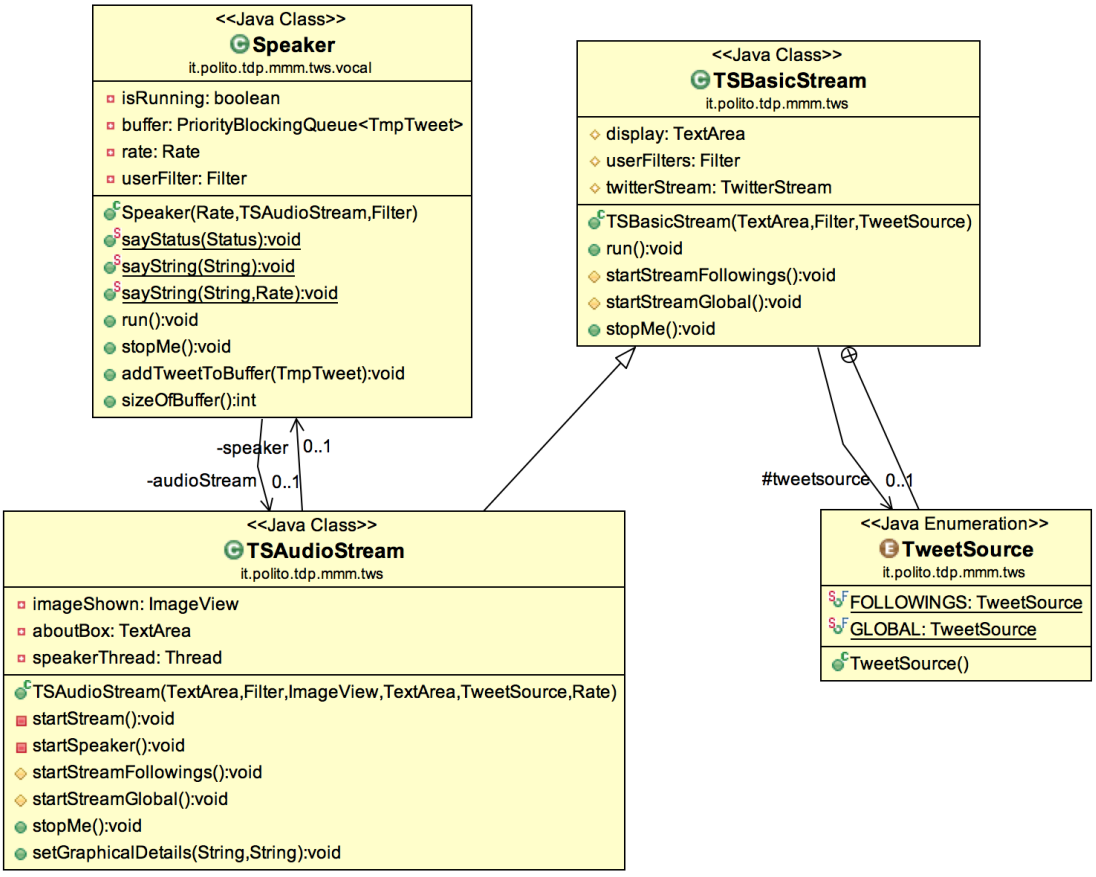

FIGURE 3.1: Audio class diagram

3.2 Tweets buffer

When the audio option is activated, the program has to keep track of the past tweets and not just display them. The Java Collections Framework offers a structure named *Priority Blocking Queue*, which differ from the simple Priority Queue by being thread-safe[1]. The queue contains a wrapper class named TmpTweet, which contains the tweet itself received from the listener and the associated TTL set by the user. Moreover, the TmpTweet class implements the Comparable interface (parametrized with TmpTweet), giving a natural order inside the queue by the posting time of the tweet.

3.2.1 Buffer refresh

Before reading a new tweet, the Speaker override the buffer using the following code:

```
this.buffer = buffer
            .parallelStream()
            .filter(s -> !s.isExpired())
            .collect(Collectors.toCollection(PriorityBlockingQueue::new));
```

Every tweet present in the buffer is independent from the others.
The Java 8 *parallelStream* method has the strength to not be serialized, thus being much more efficient on large quantities of tweets, running in less then linear time. Another approach could have been to scan through the buffer from the bottom to the top and stopping at the first not expired tweet, due to the fact that the queue is sorted by born time of the tweet. A clear disadvantage of this last mentioned algorithm is that in case of more than one tweet created at the same moment in time, only the first founded from the bottom of the queue would have been effectively removed.

3.3 Twitter4j

TS works using the underlying Twitter API *Twitter4j* by Yusuke Yamamoto which is released under Apache License 2.0.

[1] Implementation is guaranteed to be free of race conditions when accessed by multiple threads simultaneously.

Bibliography

[1] Yusuke Yamamoto. *Twitter4j - unofficial Java library for the Twitter API.* `http://twitter4j.org`

[2] Javier Fernández González. *Java 7 Concurrency Cookbook.* Packt Publishing, Birmingham-Mumbai, 2012.

[3] Rajkumar Buyya, S Thamarai Selvi, Xingchen Chu. *Object-Oriented Programming with Java - Essentials and Applications.* McGraw-Hill Education, 2009.

[4] Cave of Programming YouTube channel. *Java Multithreading.* `https://www.youtube.com/user/caveofprogramming/`

[5] RFC 5849 *OAuth protocol.* `http://tools.ietf.org/html/rfc5849`