Git Twitter Mashup App
==========

This is a simple Git Twitter Mashup App, which returns a list of projects from Github for the search term “reactive” and for each of the projects returned uses the Twitter Search API to return the most recent tweets. The projects have been restricted to 10 projects and the tweets to a maximum of 5 for better readability.

Setup, Configuration & Run
==========

1. To access the Twitter API, you need to modify the resources/app.properties to enter a valid consumer key and consumer secret.
2. The number of projects which will be queried for tweets is also configurable if so desired.
3. From the Home folder (Mashup) run “ant”. This will clean, compile, generate the jar and run the code to print out the output in command line.
4. In the Home folder, there will be an app.log file created that has the logs for the app. This is flushed on every run.
5. You can run the test suite by running “ant junit” from the command line.

3rd party libraries used
==========

commons-codec-1.11.jar - for Base64 conversion for generating the twitter bearerToken
gson-2.8.3-SNAPSHOT.jar - for converting json to java objects and vice versa

The below 2 jars are used for the logging framework:
log4j-api-2.11.0.jar
log4j-core-2.11.0.jar

These 3 jars are used for the testing framework:
hamcrest-core-1.3.jar
junit-4.12.jar
mockito-all-1.9.5.jar

Design:
======

I have used a MVC design paradigm, where the controller is the GitTwitterMashupApp.java. This accesses the GitHub and Twitter API libraries to get the data back and stores them in Java POJOs (Model). The client is a simple client that just prints out the json returned by the App.

This design is very maintainable going forward, since the client can be swapped out for a GUI client or a web client. The App can be moved to be a web app hosted on server. A rest end point can be added to it as well. Outputs in other formats can also be generated since we have all the data in models. The API libraries can be swapped out for 3rd party libraries which are more feature rich or can be enhanced to add more APIs.

A retry mechanism has been implemented to retry calling the APIs if there is a server error returned. This is because the server could be busy at the time the request comes in, and instead of returning an error straightaway to the user, the app tries to be more robust by retrying 3 times, each time with an increased time interval.

Log4j2 has been used as the logging framework and I used Mockito to write the test, so I can mock the API libraries.

Improvements:
==========

I could have made the search term enterable via the client, but chose not to to keep the execution of the app simple. This can be changed by changing the search term on the MashupClient.java. I could have also made the number of tweets and and result_type configurable. These are simple enhancements that can be made if so desired.

We can also improve the app to work around the limitation of the twitter API rate limits, by storing the requests in a queue if we hit the rate limit and try again when we would be eligible. But this would not be ideal for synchronous tasks, but would work well for async tasks.

More tests can be added to test out various other scenarios.