CMPSC 112
Introduction to Computer Science II
Spring 2015

Laboratory Assignment Four: Implementing and Evaluating a Twitter Client

Introduction

As we continue to both refamiliarize ourselves with the fundamental characteristics of the object-oriented programming paradigm and to practice using the some of the features of the Vim integrated development environment (IDE), you will take the final steps towards implementing a Twitter client. Along with learning how to use, enhance, and extend a Java class, you will also review the steps that you must take to perform console input and output. This laboratory assignment represents the second of a two-part series of assignments designed to review Java programming.

Accessing and Understanding GatorTweet

To start this laboratory assignment, you should return to the `cs112S2015-share` Git repository and type the command `git pull` in the terminal window. Now, you should have a `lab4/GatorTweet/` directory that you can explore further. Where is the Java source code in this new directory? Why is it stored in the directory where you found it? Now, you should use GVim to study the source code in the `build.xml` file. As you did for the past assignment, you can type the command "`:Ant compile`" in GVim and it will compile the three Java classes and save the bytecode in the correct subdirectories inside of the `bin/` directory. Please see the instructor if you cannot get this to work.

Connecting to the Twitter Service

At this point, you should look in the `lib/` directory and notice that there is a file called `twitter4j-core-3.0.5.jar`. This file contains the additional Java classes that we will need to communicate with the Twitter service. Before you run any program that communicates with Twitter, you must use the `export CLASSPATH` command to add this file to your `CLASSPATH` variable. In addition, you will need to ensure that the `bin/` directory is also in your `CLASSPATH`. To learn more about the library that we are using to connect to Twitter, please visit the Web site for Twitter4J, available at `http://twitter4j.org/en/index.html`. Before you try to connect to Twitter, you should also study the `twitter4j.properties` file. What does it contain? Why?

In advance of completing the next step, you should make sure that you have already created your Twitter account, followed some other Twitter users, and tweeted through the Web-based interface. Now, you can go ahead and run the `CreateProperties` program in your terminal window. You should follow this program's instructions to authorize your own Twitter client to access your Twitter account. As you complete this step, please make sure that you record the content of the `twitter4j.properties` file before and after running the `CreateProperties` program. Also, you should take a screen shot of the Web site that the program asks you to visit. Finally, you can try to run the `GetHomeTimeline` program. What output did it produce? Were you able to connect to the Twitter service? Students who could not complete this step should see the instructor.

Finishing the GatorTweet System

First, you should add the code in the `GetHomeTimeline` to your own `GatorTweet` program, thus giving your system the ability to display your own home timeline. However, we still need to add some extra features to `GatorTweet`! For instance, you should implement a feature that can post all of the valid tweets in the "Tweets.txt" file to your timeline. If the `Twitter` class in Twitter4J provides an instance method called `updateStatus` what code do you need to write to post all of the valid tweets to the Twitter system? Students who want to learn more about the `updateStatus` method can check the `http://twitter4j.org/javadoc/` Web site. Again, remember that you will need to use an iteration construct to iterate through all of the valid tweets. In order to fully implement this feature, you will also need to add a `getMessage` method to `Tweet.java`. Once you have posted each of the valid tweets to Twitter, your program should also output some debugging information indicating that each of the tweets was correctly posted.

Right now, our `GatorTweet` program has a very limited user interface. As one way to overcome this limitation, you can implement a command-line interface that will allow the user to only read the home timeline, only update the status through the "Tweets.txt" file, or to do both. In order to implement this feature, you will need to add conditional logic to your program. Finally, you should develop at least one additional feature of your choice. For instance, you could implement a random tweet generator that will create a random tweet and then post it to Twitter. Alternatively, you could implement a feature that allows you to perform direct messaging on Twitter. As you finish the `GatorTweet` system, please make sure that you add comments to all of the Java classes, including those that you did not modify or extend. You must add a comment to the header of every class and method, the declaration of every variable, and every other important line of code.

Summary of the Required Deliverables

In addition to using your Git repository to turn in all of the required items (including the screenshot), this assignment invites you to submit one printed version of the following deliverables:

1. The contents of `twitter4j.properties` before and after running `CreateProperties`.
2. A screenshot of the Web site that the `CreateProperties` program asks you to visit.
3. The final version of your `GatorTweet.java` that implements all of the required features.
4. The output from running `GetHomeTimeline` when debugging mode is and is not enabled.
5. The output from running `GatorTweet` with all of the command lines that it supports.
6. A reflective commentary on the challenges that you faced when implementing `GatorTweet`.
7. At least three suggestions for additional features that you could later add to `GatorTweet`.

Along with turning in a printed version of these deliverables, you should ensure that everything is also available in the repository that is named according to the convention `cs112S2015-<your user name>`. Please note that students in the class are responsible for completing and submitting their own version of this assignment. While it is acceptable for members of this class to have high-level conversations, you should not share source code or full command lines with your classmates. Deliverables that are nearly identical to the work of others will be taken as evidence of violating the Honor Code. Please see the instructor if you have questions about the policies for this assignment.