**CMPSC 112**
**Introduction to Computer Science II**
**Spring 2015**

**Laboratory Assignment Three: A First Step Towards Implementing a Twitter Client**

## Introduction

Twitter is a commonly used social media network. As we refamiliarize ourselves with the fundamental characteristics of the object-oriented programming paradigm and continue to learn more advanced features of the Vim integrated development environment (IDE), we will take the first steps towards implementing a complete Twitter client. Along with learning how to use, enhance, and extend a Java class, we will also review the steps that you must take to perform console input and output. In addition to helping you to review the knowledge and skills that you previously learned about programming in Java (e.g., using the Java compiler and virtual machine and implementing Java methods that contain conditional logic), this lab aims to introduce you to some "best practices" in software development (e.g., using a build system and separating source and compiled code). This laboratory assignment also represents the first of a two-part series of assignments.

## Accessing and Understanding GatorTweet

To start this laboratory assignment, you should return to the `cs112S2014-share` Git repository and type the command "`git pull`" in the terminal window. Now, you should have a `lab3/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. What do you think is the purpose of this file? For this assignment, you can type the command "`ant compile`" in your terminal window and it will compile the `Tweet.java` file and save the bytecode in the correct subdirectories inside of the `bin/` directory.

At this point, you should purposefully insert an error into the `Tweet.java` file. For instance, you can open the file in GVim and then remove one of the semi-colons at the end of a source code line. Now, type "`ant compile`" again. What output do you see on the screen? Now, you can see the that output of the build command will indicate where the error is located. After you have returned to the error location, go ahead and fix the problem by adding back the semi-colon and recompiling your program. If you would like to further experiment with automatically building your Java program, you can type the command "`:Ant compile`" directly in GVim. What action does this command perform? How is this approach different from running `ant` in the terminal?

It is also worth noting that the files and the directories for this assignment are organized in a way that is different from what you have seen in past laboratories or in-class assignments in this class or in your previous introductory computer science class. One difference that you should notice is that the bytecode files (i.e., the files ending with the "`.class` extension) are stored in a different directory that the source code files (i.e., the files that end with "`.java`"). What are the advantages and disadvantages of organizing the project in this fashion? Additionally, you should note that the source code for this project is stored in nested directories and the source code file called `Tweet.java` contains the "`package edu.allegheny.gatortweet;`" at the top of it. Why do you think that the source code is organized in this fashion? What are the benefits and drawbacks?

## Improving the GatorTweet System

You will notice that the `Tweet` class contains two instance variables and several methods. First, you should add proper JavaDoc comments to all of the variables and methods in `Tweet.java`. Once you have finished adding in all of the necessary comments, please notice that the `isValidMessage` method is not correctly implemented. You should enhance this method so that it returns `true` when a candidate message is valid and `false` otherwise. This method should classify a message as valid as long as its length is greater than zero and less than or equal to 140.

Now, you should implement a `GatorTweet` class that performs the following operations in `main`:

1. Constructs an `ArrayList` that will only hold `Tweet` objects for the valid `Tweet`s.
2. Constructs an `ArrayList` that will only hold `String`s for the invalid tweet messages.
3. Uses the `java.util.Scanner` class to read in consecutive lines of text from a file in the `tweets/` subdirectory of the `GatorTweet/` directory, called "Tweets.txt", that contains potential tweets. (It is worth noting that this file can contain both valid and invalid tweets).
4. Checks to see if the current tweet message is a valid one or not. If the message is valid, then it should construct a new `Tweet` and store it inside of the `ArrayList` for the valid instances of the `Tweet` class. If the message is invalid, then it should not construct a `Tweet` and instead store the `String` message in the `ArrayList` for invalid tweet messages.
5. Outputs all of the `Tweet`s stored in the `ArrayList` for the valid tweet messages.
6. Outputs all of the `Tweet`s stored in the `ArrayList` for the invalid tweet messages.

In the next laboratory assignment we will finish implementing a `GatorTweet` program that can interact with Twitter. For instance, we will add features that can authenticate a user with the Twitter servers, display a user's Twitter timeline, and post a `Tweet` to the Twitter site.

## Summary of the Required Deliverables

This assignment invites you to submit one printed version of the following deliverables:

1. A description of the meaning and purpose of the provided `build.xml` file.
2. Screenshots demonstrating that you could correctly run the `build.xml` file.
3. The final version of your `Tweet.java` and `GatorTweet.java` files.
4. The output associated with running `GatorTweet` with a sample "Tweets.txt" file.

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. Please see the instructor if you have questions about the policies for this laboratory assignment.