Twitter

Tian-Yo Yang

1 Introduction

This assignment will cover mostly JavaFX, anonymous inner classes, and some design concepts you have learned throughout this course.

This assignment will be a bit more open ended compared to other assignments in the past. There are a few requirements that we ask you to satisfy, but otherwise its mostly up to you. So use your creativity to surprise us! You should employ best practices and the concepts taught thus far in the course.

2 Problem Description

Twitter is an online social media networking service that enables users to send and read short 140-character messages called tweets. Registered users can post tweets and read tweets from other users.

As a Georgia Tech student, your life is complicated and you just want something simple to use for social media networking purposes. You are sick of using Twitter due to its complicated set of unnecessary features. Therefore, as a CS1331 student, you decided to make your own simple Twitter application using JavaFX.

2.1 Backend and Provided Files

We have provided you with a server that provides new Tweets randomly. You will need to use this in order to implement the "refresh" functionality. The server generates a random Tweet from choosing a random User and a random message body. Feel free to change the users and/or tweets to your liking. Also feel free to add even more users and tweets! If you do add to or change this file, make sure your tweets are appropriate. You have also been provided some images to use. You can also feel free to not use these and provide your own image files. Just make sure you submit your image files to T-Square.

In addition to the server, you have also been provided with Tweet and User. You can edit these classes to fit your version of Twitter as you see fit, but note that it is not necessary to change these

files at all.

The only class you need to write is `Twitter.java` which will be a JavaFX application. The Tweet, User, and TwitterServer classes will be helpful in completing the application.

2.2 Twitter Functionality

- "Refresh" - should randomly generate 2 to 4 Tweets and add it to the feed from the Server
- "Delete" - should remove a selected Tweet from the feed. Upon clicking the "Refresh" button, your event handler should generate random Tweets using the provided TwitterServer, add it to the list of Tweets, and update the view of the list of Tweets.
- "Like" - should mark the Tweet as liked
  - Upon clicking "Like" and "Delete", your event handler should perform the corresponding functionality on the selected Tweet and update the view of the list of Tweets.
- "Tweet" - should allow the current User to create a new Tweet and add it to the feed. Upon clicking the "Tweet" button, your event handler should get the text that has been typed in by the user in the TextArea and create a new Tweet. Your event handler should also add the newly created Tweet to the list of Tweets and update the view of the list of Tweets.

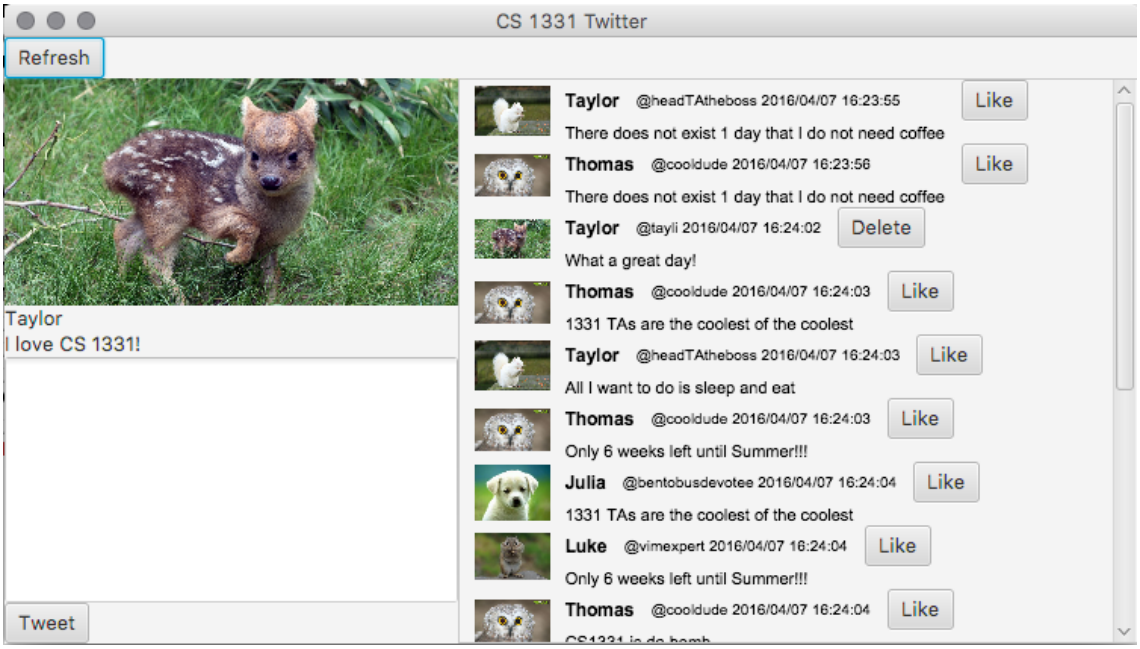
3 GUI

The UI Design is up to you for the most part, but below are some basic requirements:

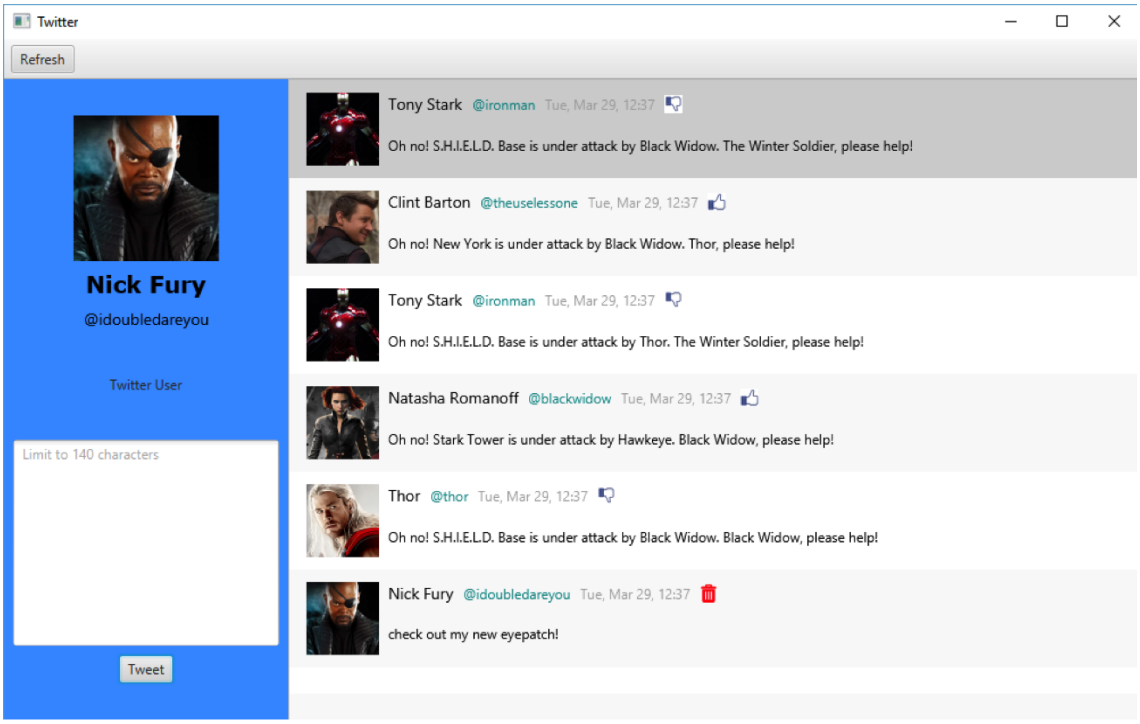
- Tweets should be listed somewhere. You can choose however you want to do this. You should be able to scroll through the tweets if they don't all fit in the view.
- There should be a profile picture displayed, the user's name, and description somewhere in the GUI.
- The "Tweet" functionality should be easy to find and use. If it isn't below the user's information it should be at the top of the twitter feed, for example.
- Like/Unlike and Delete functionality doesn't necessarily have to be next to each tweet but should be easy to find and use.

It is acceptable to change up the GUI and make it look different from the examples below. It is also acceptable to add more functionality than is required, but note there is no extra credit.

This GUI would receive full credit so long as all functionality was also present.



Additionally, here is an example of a GUI that is more than enough! Feel free to get creative with your design! There is no extra credit on this homework, but producing a finished homework you can be proud of and show your friends is your reward!



4 Tips and Tricks

One-liner hints:

- `BorderPane`
- `GridPane`
- `VBox`
- `HBox`
- `ObservableList`
- `ListView`

5 Checkstyle

You must run checkstyle on your submission. The checkstyle cap for this assignment is **100** points. Review the Style Guide and download the Checkstyle jar. Run Checkstyle on your code like so:

```
$ java -jar checkstyle-6.2.2.jar *.java
Audit done. Errors (potential points off):
0
```

The message above means there were no Checkstyle errors. If you had any errors, they would show up above this message, and the number at the end would be the points we would take off.

The Java source files we provide contain no Checkstyle errors. For this assignment, there will be a maximum of **100** points lost due to Checkstyle errors (1 point per error). In future homeworks we will be increasing this cap, so get into the habit of fixing these style errors early!

6 Javadocs

For this assignment you will be commenting your code with Javadocs. Javadocs are a clean and useful way to document your code's functionality. For more information on what Javadocs are and why they are awesome, the online documentation for them is very detailed and helpful.

You can generate the javadocs for your code using the command below, which will put all the files into a folder called javadoc:

```
$ javadoc *.java -d javadoc
```

The relevant tags that you need to have are `@author`, `@version`, `@param`, and `@return`. Here is an example of a properly Javadoc'd class:

```java
import java.util.Scanner;

/**
 * This class represents a Dog object.
 * @author George P. Burdell
 * @version 13.31
 */
public class Dog {

   /**
    * Creates an awesome dog (NOT a dawg!)
    */
   public Dog(){
      ...
   }

   /**
    * This method takes in two ints and returns their sum
    * @param a first number
    * @param b second number
    * @return sum of a and b
    */
   public int add(int a, int b){
      ...
   }
}
```

Take note of a few things:

1. Javadocs are begun with `/**` and ended with `*/`.
2. Every class you write must be Javadoc'd and the `@author` and `@version` tag included. The comments for a class start with a brief description of the role of the class in your program.
3. Every non-private method you write must be Javadoc'd and the `@param` tag included for every method parameter. The format for an `@param` tag is `@param <name of parameter as written in method header> <description of parameter>`. If the method has a non-void return type, include the `@return` tag which should have a simple description of what the method returns, semantically.

6.1 Javadoc and Checkstyle

You can use the Checkstyle jar mentioned in the following section to test your javadocs for completeness. Simply add -j to the checkstyle command, like this:

```
$ java -jar checkstyle-6.2.2.jar -j *.java
Audit done. Errors (potential points off):
0
```

There will be a javadoc cap of 20 points for this homework. Please javadoc and checkstyle as you go. Note that you can lose a total of 100 points just for style errors. Don't let this be you. Checkstyle as you go.

7 Turn-in Procedure

Non-compiling or missing submissions will receive a zero. NO exceptions

Submit all of the Java source files you modified and resources your program requires to run as well as all of the images your application uses to T-Square. Do not submit any compiled bytecode (`.class` files) or the Checkstyle jar file. When you're ready, double-check that you have submitted and not just saved a draft.

Please remember to run your code through Checkstyle!

7.1 Verify the Success of Your Submission to T-Square

Practice safe submission! Verify that your HW files were truly submitted correctly, the upload was successful, and that the files compile and run. It is solely your responsibility to turn in your homework and practice this safe submission safeguard.

1. After uploading the files to T-Square you should receive an email from T-Square listing the names of the files that were uploaded and received. If you do not get the confirmation email almost immediately, something is wrong with your HW submission and/or your email. Even receiving the email does not guarantee that you turned in exactly what you intended.
2. After submitting the files to T-Square, return to the Assignment menu option and this homework. It should show the submitted files.
3. Download copies of your submitted files from the T-Square Assignment page placing them in a new folder.
4. Recompile and test those exact files.
5. This helps guard against a few things.
   (a) It helps insure that you turn in the correct files.
   (b) It helps you realize if you omit a file or files. [1] (If you do discover that you omitted a file, submit all of your files again, not just the missing one.)
   (c) Helps find last minute causes of files not compiling and/or running.

[1] Missing files will not be given any credit, and non-compiling homework solutions will receive few to zero points. Also recall that late homework will not be accepted regardless of excuse. Treat the due date with respect. The real due date is midnight. Do not wait until the last minute!