Collaboration Lab Using Git and Javadoc

September 21, 2010

1 Overview

This lab provides you and your team with a set of programming collaboration challenges that you must solve using the git version control system. [1] Your team must complete this lab by **Thursday, September 21st** for 7 points, or pay a 1 point penalty per day late. When you have completed the assignment, **your team captain must submit the GitHub repository URL to me via Blackboard**.

Each and every team member will need to install git client software to complete this lab. Use of the EGit eclipse plugin is fine, though command-line packages work just as well. (Linux and OSX users will may want to use the command-line "git-core" package. Similar packages are available for Windows users.)

2 Instructions

1. Have your team captain create a new git *repository* on GitHub.com. Use your team name as the name of the repository.
2. The team captain should now *clone* the repository (use the HTTP URL provided by GitHub) to his/her local computer, and create a new Eclipse Java project within it.
3. With your team captain, create a simple "hello world" console application: a single driver class with a main method that prints "Hello, team!" on the first line, and "My name is *First Last*." on the second. Document the class with JavaDoc comments including the "@author" tag for the team captain. Do NOT include @author tags for the other team members. (You may, however, add other tags as you feel appropriate.)
4. The team captain must now *commit* the changes to their local repository and *push* them to GitHub.com. This will sync the Eclipse project with the centralized GitHub copy.

[1] `http://git-scm.com/`

5. Every other member of the team (in other words, everyone *except* the team captain) should now *clone* the repository–using the exact same HTTP URL provided by GitHub–to their local machines, and using the "File - Import... - General - Existing Projects into Workspace" feature to access the local copy within your programming environment.

6. All non-captain team members should now add their own new class to the project–named "FirstnameLastname"–with a single "public String toString()" method that returns a string of "Firstname Last". Document your class and method using Javadoc format. Make sure the documentation displays correctly in the "Javadoc" view within Eclipse.

7. All non-captain members should now *commit* their changes to their local *clone*, and then *push* the changes to GitHub.[2]

8. Once all non-captain members have committed and pushed their changes to GitHub, the captain should now *pull* changes into their local repository. Refresh the project view within Eclipse, and your captain should now see all the new classes!

9. With your team captain, open the driver class and create a new instance of each new class, and print it to the console, like so:

```
AliceAnderson alice = new AliceAnderson();
System.out.println("Welcome " + alice + "!");
// repeat for other team members
```

10. The team captain should now add "@author" tags to the driver class for **all** team members, save, *commit* the changes to their local repository, and *push* them to GitHub.com.

11. Non-captain members should now *pull* to retrieve the completed project.

12. Have the team captain **and only the team captain** submit the project URL via Blackboard.

[2] Depending on the activity of other team members, you may to *pull* the changes of others before you are allowed to *push* your own!

