CS 458 - Homework 3

Deadlines

Problems 3 onward are due by 11:59 pm on Friday, September 23, 2016

(Problems 1 and 2 were completed during the Week 4 Lab.)

How to submit

Submit your files for Problem 3 onward for this homework using `~st10/458submit` on **nrs-projects**, with a homework number of `3`

(Problems 1 and 2 were completed during the Week 4 lab.)

Instructions for using the tool ~st10/458submit

- If they are not already on **nrs-projects**, transfer your files to be submitted to a directory on **nrs-projects**.
- Once all of your files to be submitted are in a directory on **nrs-projects**, then use `ssh` (or Putty in a campus Windows lab) to connect to `nrs-projects.humboldt.edu` .
- use `cd` to change to the directory containing the files to be submitted -- for example,

  ```
  cd 458hw03
  ```

- type the command:

  ```
  ~st10/458submit
  ```

  ...and give the number of the homework being submitted (or whatever number you have been asked to do for lab-related files) when asked, and answer that, `y`, you do want to submit all of the files with of-interest-to-458 suffixes in the current directory. (Note that I don't mind if a few extraneous files get submitted as well -- I'd rather receive too many files that too few, and typing in all of the file names for each assignment is just too error-prone...)
- you are expected to carefully check the list of files that the tool believes have been submitted, and make sure all of the files you hoped to submit were indeed submitted! (The most common error is to try to run `~st10/458submit` while in a different directory than where your files are...)

Purpose

To make sure you know at least some `git` command-line-level basics, and to think about some of the Extreme Programming practices.

Important notes

- Note that some of your submissions for this assignment may be posted to the course Moodle site.

Problem 1

Your pair's `458lab4-1.txt` file from the Week 4 Lab, submitted during that lab using `~st10/458submit` with a number of **84**, is "counting" as Problem 1 of this homework.

(That is, you will be graded on working in your pair during that lab, and whether your pair successfully completed this file, meeting the stated specifications, during that lab.)

Problem 2

Your pair's `458lab4-2.txt` file from the Week 4 Lab, submitted during that lab using `~st10/458submit` with a number of **84**, is "counting" as Problem 2 of this homework.

(That is, you will be graded on working in your pair during that lab, and whether your pair successfully completed this file, meeting the stated specifications, during that lab.)

Problem 3

To help encourage everyone to get "up to speed" on `git`, answer each of the following questions. (Note: along with this homework handout, there is also a "GitHub Git Cheat Sheet" handout; this may be useful along with the lab projected notes, the link on the main public course web site to Scott Chacon's "Pro Git" online book, and a variety of resources from github.com.)

3 part a

Assume that, at some point in the past, you have cloned your team's private repository on GitHub into a local repository on your own computer.

Give the command you can now type to get the newest updates from the team repository on GitHub, (changing your local copy).

3 part b

Assume that you have modified a file in your local repository named `action.cpp` so that it now implements a user story **Add widget control**. Give the command you can now type to *stage* your modified `action.cpp` for a commit.

3 part c

Give the command you can now type to commit, including an attempt at a descriptive log message.

3 part d

Finally, give the command you can now type to push your committed changes to the team's private repository on GitHub.

3 part e

Give the command you can use at any time to check the current status of your repository, to see what, if

anything, has been modified.

Problem 4

Consider the following list of some Extreme Programming practices:

- Pair programming
- Release planning/the planning game
- Unit testing/test-driven development/test-first development (assume this includes the concept of writing tests BEFORE writing the code to be tested, and assume that, once written, code cannot be checked-in to the current project version unless it passes its unit tests)
- Acceptance testing/customer testing
- Frequent releases/Small releases
- Refactoring
- Simple design/YAGNI (You Aren't Going to Need It)/"do the simplest thing that could possibly work"
- Collective code ownership
- Continuous integration
- On-site customer/"extreme" customer involvement
- 40-hour week/sustainable pace
- Coding standards
- Common vocabulary/system metaphor
- User stories
- Stand-up meetings
- CRC cards - Class, Responsibilities, and Collaboration - for object-oriented programming
- Spike solutions

4 part a

You hopefully have noticed that some of these practices influence, or even depend, on some of the other practices.

Consider the practice of "Collective code ownership". List at least three other practices from the list above that make "Collective code ownership" more practical/feasible.

4 part b

CRC - Class, Responsibilities, and Collaboration - cards are used for brainstorming, for design, and for communication. They do assume an object-oriented approach. You'll find a number of templates on the web, but this is one simple version, based on the description and examples at:

http://www.agilemodeling.com/artifacts/crcModel.htm

An index card has the name of a class on the top of the card. On the left-hand side of the card, you list that class's responsibilities -- something that class knows or does. On the right-hand side of the card, you list collaborator(s) -- other classes that this class interacts with to fulfill its responsibilities.

As an example from the above web page, you might have a card with **Customer** at the top, with responsibilities listed on the left-hand-side of:

- places orders
- knows name
- knows address
- knows customer number
- knows order history

On the right-hand-side, it might have a collaborator-class listed of:

- Order

It is considered a benefit that index cards are small, and easy to move around on a desk (or perhaps a computer desktop) -- you can put cards for classes that collaborate with each other near each other, and cards that don't further apart. You can also easily throw away one or more cards and start over if one approach isn't working!

Consider a Monopoly game program. You might have a Player class in such a game.

Give an example of at least one responsibility that a Player class might have.

Give an example of at least one collaborator-class that a Player class might have to interact with to fulfill its responsibilities.

Problem 5

As you know, you have been added to organization `hsu-cs458-f16` at `github.com`. You have now also been added to a team `cs458`. Team `cs458` has access to a repository `458hw03`.

`git` is installed on nrs-projects; to provide some `git` practice that does not involve your team's eventual repository, you are going to use repository `458hw03` from nrs-projects (and it is up to you where you use Git for your team project work, of course! The beauty of Git is that you can work locally on own computer, or even multiple computers, or nrs-projects, etc.)

(Note: some of the commands given here and in the next few problems will prompt you for your password. That should be your GitHub password...)

As recommended in our Git intro, and on the Git Cheat Sheet from GitHub, and in Section 1.5, pp. 10-11 in Scott Chacon's "Pro Git", you should set your username and e-mail address for your repositories. Practice doing this on nrs-projects if you did not already do this during lab:

```
git config --global user.name "yourFirst yourLast"
git config --global user.email yourEmail@humboldt.edu
```

IF you would like to specify a default text editor that will be used when Git wants you to type in a

message, you may do so using (Pro Git, p. 11) -- it notes that, on many systems, the default editor is `vi` or `vim`.

```
git config --global core.editor desiredEditorName
```

To show that you've done this on nrs-projects -- to see for yourself, and to make a file to submit to me:

```
git config --list
```

```
git config --list > 458hw3-5-config-list.txt
```

Submit your resulting file `458hw3-5-config-list.txt`

Problem 6

Now that you have configured those settings, clone the `458hw03` repository that GitHub team `cs458` has access to onto nrs-projects. Run this command on nrs-projects (from wherever you would like this repository to be within your account on nrs-projects):

```
git clone https://github.com/hsu-cs458-f16/458hw03.git
```

...and enter your GitHub username and password when prompted.

Demonstrate that you succeeded by running the following commands (STARTING in the nrs-projects directory in which you ran the above command, which should be the one now containing your cloned repository directory) (note that >> appends to a file...)

```
echo yourName >> 458hw3-6-show-cloned.txt
```

```
pwd >> 458hw3-6-show-cloned.txt
```

```
ls 458hw03/* >> 458hw3-6-show-cloned.txt
```

Submit your resulting file `458hw3-6-show-cloned.txt`

Problem 7

Here is where we are going to live dangerously!

Now do:

```
cd 458hw03
```

...so you are in your cloned GitHub repo. You'll see there is a subdirectory `team-greetings`.

```
cd team-greetings
```

...and you'll see at least a greeting file from me in `smtuttle.txt`, and probably more, because each of you is now going to add a greeting file to this directory, also!

Notice that you haven't changed your cloned repository copy yet -- because you might be doing this problem at the same time as someone else, do:

```
git pull
```

...to make sure you have an up-to-date copy of the repository.

Then, in directory `team-greetings`, make a file *yourGitHubUsername*`.txt` that contains a

greeting and your real name.

Stage this new file using:

`git add` *yourGitHubUsername*`.txt`

...and then commit it using:

`git commit -m "adding` *yourGitHubUsername* `greeting"`

Show that you've changed and committed your local nrs-projects `458hw03` repository by running the following commands -- run the following WHILE WITHIN your repository's `team-greetings` directory!! (You should then be creating the file to submit OUTSIDE of your repository...)

```
# make sure you run the following from the team-greetings directory
#     within your nrs-projects 458hw03 repository
echo yourName >> ../../458hw3-7-show-changed.txt
pwd >> ../../458hw3-7-show-changed.txt
ls >> ../../458hw3-7-show-changed.txt
git status >> ../../458hw3-7-show-changed.txt
```

Then -- cross your fingers and try pushing your updated repository back up to GitHub:

```
git push origin master
```

Submit your resulting file `458hw3-7-show-changed.txt` (remembering to:

```
cd ../..
```

...first, to get to the directory where it is!)

(And I'll also look to see if your greeting file *yourGitHubUsername*`.txt` is successfully pushed to the GitHub version of `458hw03`.)