CMPUT 301
Team 12 Guidelines

Omar Almokdad Justin Daza Gemma Marcinkoski
Jen Switzer Andrew Whittle

Winter 2016

Contents

Chapter 1

Team Expectations

1.1 Attitude

Be positive! We can do this!

1.2 Communication

For methods of communication, Slack is the preferred way of talking to all the group. That said, email is completely viable and preferred for larger pieces of information. Please **"reply all"** to group emails.

Emails:

- Omar: omokdad@yahoo.com
- Justin: justinjo@ualberta.ca
- Gemma: marcinko@ualberta.ca
- Jen: jenwswitzer@gmail.com
- Andrew: awhittle@ualberta.ca

Contact information for the professor and TA:

- Ken Wong: kenw@cs.ualberta.ca
- Kent Rasmussen: krasmuss@ualberta.ca

Chapter 2

Using Git

2.1 Introduction

For this course we are using `https://github.com/CMPUT301W16T12` as the remote repository for our submission. The tool to add things to this repository is **git**. With git you can get the repo, add and commit changes, stay up to date, push your work and merge it into the project. It is a bit complicated to use at times, especially working with many people, but it is a powerful and useful tool.

Rule of thumb: commit often! Make your commits easy to follow. Big commits are bad because you lose a lot of work if you have to rollback, and reading the history log is difficult.

2.2 The Team

The team name is CMPUT301W16T12.

The members of the team are:

- Omar: omokdad
- Justin: jjda7a
- Gemma: gemmamarcinkoski
- Jen: waiyi
- Andrew: awhittle3

2.3 Set up

To get the exiting remote repo onto your local machine, call

```
git clone exampleurl.git
```

You are ready to go!

2.4 Branches

The master branch is the primary branch of the project, and will be used for the submission. To manage our work and keep the master branch healthy, we will be working on our own branches. Please name your branches as following:

```
awhittle/database
```

Where the first part is your user name and the second part is the name of the part of the project you are working on.

To create a branch for the first time based off of your current workspace, call

```
git checkout -b awhittle/mynewbranch
```

To switch to a branch, call

```
git checkout awhittle/myexistingbranch
```

2.5 Pulling

It is very important you pull before you start working on code, or before you push! This will give you the most up to date code for the project.

```
git pull origin -a
```

2.6 Adding and Committing

Git committing is a multi-step process. You add your changes to a staging area, then you commit your changes.

To see the status of your changes at any time, call

```
git status
```

To add all of your changes, you can call

```
git add .
```

However, if you have certain files you want to merge and not others (e.g. you don't want to commit doc changes and code changes in the same commit), you should call them by name:

```
git add src/file1.java src/file2.java
```

To commit with a short message, call

```
git commit -m "Add save method to display activity"
```

Please make your messages concise and clear. It helps communicates to the team what you have done.

2.7 Pushing

```
git push origin -a
```

This will push all your committed work on all your branches. If you want to push a specific branch call the following instead:

```
git push origin awhittle/mybranch
```

2.8 Merging

Merging can be a complex process. We recommend the rebase strategy in order to keep the master branch healthy. Please do not use merge until you have pulled and rebased your code.

2.9 Logs and Other Information

To see a log of everything on the branch, call

```
git log --oneline
```

Omit the oneline option to get a fuller picture.

To see the changes you have made to your code that you have not added or committed yet, call

```
git diff
```

This is useful for seeing what you have done and what to put in your commit message.