CISC 310
Exercise 2

GitHub Basics

At the end of today, you should be able to answer these questions confidently:

- What is a *repository*?
- Why is it (often) desirable for one project to be in multiple repositories simultaneously? What does it take to make this work?
- What is "a commit"? Relatedly, what happens when you "commit" some code?
- What do "clone," "push," and "pull" mean? When should you do these things?

Before coming to class, you should have read "Getting your Project on GitHub" and Chapters 1 and 2 of the Git book.

Part I: Overview

The git website (http://git-scm.com/) says

"Git is a free and open source distributed version control system"

Be sure you have *some* initial idea about what those 3 pieces mean!

Notice that the description of git says nothing about code! You don't have to use git with code; you can use it with any kind of project you want to keep track of. We'll mostly think about git in terms of code, just because that's what we're dealing with in this class—though of course 'code' usually includes documentation files and other kinds of "extra" stuff you'd expect to see in a software project.

A *lot* of what git offers is about collaborative development with lots of people; in this class, you're not going to be collaborating on programming projects, so you won't need many of git's features. That's a **good** thing: you'll be able to get used to the basic git operations first, so when you need to use it to collaborate later, you'll have a solid foundation. Remember, in this class, we're using GitHub for 3 essential reasons:

- I'm distributing all course information through GitHub—both the course website and, perhaps, repositories of code that we'll need during class..
- I'm distributing your project assignments, in a way that will make it a bit easier for me to grade.
- It provides you a 'backup' of your project development, and also makes it easier for us to discuss your progress if questions come up..

We won't, therefore, be using many of the most powerful ideas in git—you don't need to worry about branching and pull requests, for example. But I'm still going to ask you to go through some exercises that expose you to these ideas, so you can at least say you've seen them.

Part 2: Basic Tutorials

First, git on the command line! You won't need to use this very much, but it's important to have some fundamental understanding of what's happening when we interact with git through Eclipse:

https://try.github.io

Then, basic work with GitHub: https://guides.github.com/activities/hello-world/ — this will show you, briefly, how to create a repository on GitHub, and give you a little practice with the concepts of branches and pull requests (again, we won't need these directly in this class, but if you ever want to contribute to open-source software project, these are essential).

Don't remove any of your work from this tutorial; we'll use it in...

Part 3: GitHub and Eclipse

This "tutorial" explains how to do various git/GitHub operations inside Eclipse:
http://eclipsesource.com/blogs/tutorials/egit-tutorial/

Use this tutorial to get the information you need to carry out the following sequence of operations (which closely resembles the usual sequence of operations you'll need to carry out when you're working on programming projects). By the way, you probably won't need *every* section of this tutorial; below I've listed the table of contents with the **bold** section titles indicating the ones that are **important** for us.

Create a simple Java project in Eclipse. Create a Java source file (go to *File->New->Class*, or look for the button that looks like a small green circle with a 'C' in the middle), and write a little bit of Java code (this can be something as simple as a "Hello, world" program. Make sure this compiles and runs.

1. In Eclipse, **clone** the repository you created on GitHub in the previous tutorial.
2. Add your simple Java project to your newly cloned repository. (See the "Creating Local Repositories" section for instructions about adding a project to a repository.)
3. Be sure to **commit** the project to your repository.
4. Add a couple lines of code to your Java. Make sure the result still compiles and runs. **Commit** these changes. (Watch to see if the 'dirty' mark disappears after you commit.)
5. Look at your repository on GitHub. Is your Java project there? It shouldn't be—why not? Now **push** your commits up to GitHub. Is your Java project there? Take a few minutes on GitHub to explore the (very brief) commit history of your project, and the ways GitHub makes this information available.
6. Now, make some changes to your repository *on GitHub*. For example, add a little bit of text to the file `README.md`, or create a new file. Note how GitHub "saves" your change by asking you to commit it.
7. Look at the contents of your repository in Eclipse (Hint: one way is to open the "Git Repositories" view: *Window->Show View->Other...* and look in the *Git* menu.) You shouldn't be surprised that the change you made on GitHub doesn't appear here.
8. Now **pull** the changes from GitHub into your local repository, and make sure the changes you made *do* appear. Look at your local commit history; it should show the commits you made on GitHub too.

These are the essential operations you'll need to work on projects in this course. But before we finish, try one more thing—an example of what *not* to do.

a) Make a change on GitHub; commit it there.
b) Make a change in Eclipse, commit it there.
c) Now do a push or a pull (doesn't matter which). It should fail… why?

You should get into the habit of doing a pull from the remote repository *before* you start working on a project, just to make sure you have all the latest changes. It's possible to merge the conflicts (read the sectiosn on "Merge" and "Resolving Conflicts," if you like), but at least in this class, you shouldn't have any need to deal with that, as long as you're careful about maintaining synchronization between your GitHub and Eclipse repositories.

Tutorial Contents

More Resources

If you want to play around some more, here is a long list of tutorial and guides: https://help.github.com/articles/good-resources-for-learning-git-and-github/.