Introduction to Course Infrastructure & Java

During this course you will become familiar with several popular (and industry-relevant) software engineering tools, including IntelliJ (or Eclipse), Git, GitHub, Gradle, Travis CI, FindBugs, and Checkstyle in addition to Java. In this recitation you will make sure your environment is set up correctly. If you have problems with the course infrastructure, reach out to the course staff as soon as possible through Piazza or office hours.

To click the links in this document, open the `recitation/01/recitation1.pdf` file once you have finished the **Setting Up Your Repository** section.

Java Platform

The language used in this class is Java 8. We recommend downloading and installing the latest version of Oracle's Java Development Kit (JDK). You may choose instead to download OpenJDK 8, an open-source implementation of the JDK.

Development Environments

Integrated Development Environments (IDEs) provide a more comprehensive set of features than classical text editors for Java development. You are required to use an IDE for development in this class. There are several IDEs available, but the most common ones (and the only ones supported by the course staff) are IntelliJ IDEA and Eclipse. If you haven't already, you should install one of these.

Git and GitHub

Git is a distributed version control system popular for large software projects, and GitHub is a hosting service for Git repositories. We will be using GitHub and Git to distribute homework assignments, for you to turn in your homework, and for us to give you grades and other feedback on your work. The basic idea is that you will *clone* (make a copy of) your GitHub repository on your local computer. You can then *pull* changes from GitHub to receive our feedback and any new homework assignments, make local *commits* on your own computer to complete your homework, and *push* your completed homework back to GitHub for us to grade.

A command line interface (CLI) for Git comes pre-installed on many non-Windows systems. If you don't already have Git, you can download either the CLI or a graphical user interface (GUI) such as GitHub Desktop. Our course documents refer to CLI commands, typeset in `fixed-width` font—for example, to see the version of Git installed on your system, run:

```
$ git --version
```

Setting Up Your Repository

To set up your repository, sign up for a GitHub account; you may use your existing GitHub account if you have one. Then, fill out the web form here:

`https://garrod.isri.cmu.edu/214/registration`

After filling out that form, confirm your email. Upon confirmation, a GitHub repository will be set up for you and you will receive an email from GitHub asking you to join the 17-214 organization. If you do not get this email, go to `https://github.com/CMU-17-214/` and join the organization.

From the directory where you want to place your course materials, run the command:

```
$ git clone https://github.com/CMU-17-214/your-andrew-id
```

where *your-andrew-id* is your Andrew ID. This will create a local copy of your Git repository in a directory with your Andrew ID as its name. In this directory you will do all of your work for 17-214.

Retrieving Assignments and Grades

When changes are pushed to GitHub from another copy of your repo, you must pull those changes in turn to your local copy. You will need to do this when the course staff pushes assignments, recitations, and grades, or when you have pushed work from another clone of your repo. From your Git repository, run the command:

```
$ git pull
```

Build and Test Automation

Gradle is a build tool that automates many aspects of the development process, allowing you to build, analyze, and execute your code from the command line. It will later be useful to automatically manage dependencies. Travis CI is a web service that can execute commands on your GitHub repo each time you push, sending you an email if the commands fail. In particular, we have Travis CI run Gradle when you push to confirm that your unit tests succeed. These tools enable you to maintain the integrity of your code even as you add new features.

Checkstyle is a development tool used to identify where Java code does not adhere to style conventions. It automates the process of checking for common style flaws such as the use of magic numbers. For this course, we will be following a subset of the Sun Code Conventions. Checkstyle is set up to run automatically with Gradle and will cause your build to fail when the guidelines are not followed. We recommend installing the Checkstyle plugin in IntelliJ or Eclipse to receive immediate feedback in the IDE. Checkstyle is not an autograding tool; it is only meant to help you to avoid certain common mistakes.

Exercise: Java Practice

After you have imported the `recitation/01` project to your preferred IDE, examine the `Example` and `Main` classes. Complete the `printList` method for the `Example` class.

Turning in your work

After you are done working within a clone of your repository, you can turn in your work with

```
$ git add file1 file2 dir1
$ git commit -m "Completed recitation 01"
$ git push
```

where *file1*, *file2*, and *dir1* are the names of files and directories you have added or changed, and the commit message (after the `-m`) is an arbitrary message describing your work.

1. The command `git add` instructs Git to track changes to a set of files in your clone; this is called adding the files to your *staging area*. If you pass a directory name to `git add` then all the files added or changed in that directory and all subdirectories (recursively) will be tracked and staged.[1] You can check what files are staged with the `git status` command.

2. The command `git commit` records all the locally-tracked changes as a new version of the repository, along with a message that describes the new version. You can view recent commits with the `git log` command.

3. The command `git push` records the most recent committed version to the remote server, your repository on GitHub. GitHub will then automatically trigger a build on Travis CI.

Your work is not turned in unless you have completed all three steps. Check on GitHub to confirm that all your files are correctly pushed.

[1] Remember that `.` denotes the current directory. So, `git add .` will add all files in the current directory to the staging area.

Each commit is a *local* checkpoint of your work which you can turn in when you push your repository to GitHub. If you push your repository to GitHub but have not staged and committed your changes, those changes will not be pushed to GitHub. Commits also provide a way of rolling back to an earlier version of your work.

When you have finished a unit of work, it is a good practice to commit it to your repository by doing the above. You should commit often and write useful commit messages. It is common to commit multiple versions locally before pushing your work, although you might want to periodically push your work to GitHub before you finish your homework to back up your work.

If you attempt to push your repository to GitHub but the GitHub repository has changed since you last ran `git pull`, your push will fail. To fix this, pull the other changes from GitHub (using `git pull`) and attempt your push again.

When you are done pushing your work to GitHub, you should always check GitHub to confirm that the expected files are there. Alternatively, you can create a new clone of your repository (using `git clone`) in a new location on your computer, and test your solution in that new location. This method allows you to test exactly what the TAs will test when they clone your repository from GitHub.

Don't push changes to a homework subfolder after the deadline (unless you intend to spend part of your "late" budget). We will use the GitHub timestamp of the latest `git push` event to determine if you were on time.

Additional Resources

Here are some resources if you are interested in learning more about Git. Apart from being used heavily in this class, Git is popular in both the industrial software development and open source communities. Version control is a powerful tool in software development, and mastering it will remove many headaches that you would have encountered otherwise.

- 15-Minute Tutorial: https://try.github.io/
- Pro-Git Book (free): http://git-scm.com/book/en/v2/
- GitHub Game (free): http://pcottle.github.io/learnGitBranching/