Adam Worthington
Basic Introduction to git

The purpose of this brief guide is to set up a local repository on your machine where you can actively modify shared files with the rest of the group. It will be a step by step, comprehensive guide.

1. Initializing the repo.
    a. So normally you would have to use ‘git init’ and set up a remote to push and pull from an online repo, but fortunately for you the online repo already exists and can simply be cloned to inherit all functionality.
    b. Steps:
        i. Download and install git. “http://git-scm.com/downloads”. This will come with some crazy gui stuff that I don’t know how to use, and a very functional command line known as “git shell”.
        ii. Browse to “https://github.com/AdamWorthington/Frame”
        iii. In the bottom right of the webpage should be a url listed as “clone URL,” copy this URL in whatever communication standard you would like. Alternatively just copy this HTTPS address (do this by default if you don't know or care about SSH or other communication standards). “https://github.com/AdamWorthington/Frame.git”
        iv. At this point you need to locate your git shell. For windows simply search for “git shell”, otherwise figure out where your OS installs git to by default, and launch it from there. Once inside ‘git shell’ you will need to traverse to/create a directory that you want your files from the repo to be stored. The terminal behaves like a linux shell so use those commands (cd, mkdir, etc). Now run the following command in the shell, “git clone <Insert copied clone URL from above here>”. This will set up everything you need to access the repo.
    c. In short you now can do a pull request through “git pull” to gather all of the code the other group members have added since you last pulls. Before you commit any changes you must first add any files you have been working on through “git add …”, alternatively if I am not mistaken a “git add *.*” will only add the files that have been edited since a pull. You can use “git commit -m “message”” to commit all changes you have made to a your local repo for local source control. When you are pleased with these changes you can use “git push” to make push the files you have added to the commit to the online repo, on the master branch.
2. Setting up your IDE with these files.
    a. The following example will be assuming you are developing on Eclipse IDE. Here are the specs for my Eclipse build:
        i. Version: Kepler Service Release 2
        ii. Build id: 20140224-0627

   b. Download and install java SEDK 8 "http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html" so we all work on the same java version.
   c. You now need to import the project. Follow these steps for eclipse:
      i. File > import
      ii. In the box of options should be listed "Git".
      iii. Git > Projects from Git > Next
      iv. Existing local repository > Next
      v. Click on the "Add…" button on the upper left of the dialog box.
      vi. Browse to the directory that you ran the "git clone" command in earlier"
      vii. Click on Search
      viii. Select "Frame" from the displayed option(s) and click on Finish
      ix. Select "Frame - <Directory>" in the Select a Git Repository dialog box and click on Next
      x. Leave the "Wizard for project import" subsection as it is. (should have the "Import existing projects" selected by default). Click on Next
      xi. The "Import Projects" dialog box now opens with the following activated:
         1. "Frame - <Directory>" is checked
         2. "Search for nested projects" is checked
         3. Add project to working sets in not checked
      xii. Click on Finish
   d. You now have the project imported to Eclipse and can run the test file I have included. (Frame > src > default package > Example.java). This should just print "Success".
   e. Eclipse now recognized this project as being a git repository and you can use some nifty features/plugins to aid development. You may need to refresh the file system in Eclipse if you do a pull mid development by right clicking on the project name (Frame) and selecting the "refresh" option.
3. Resources
   a. A comprehensive reference sheet for all things git: "http://git-scm.com/docs"
   b. Branches allow us to test certain features without pushing changes to our master branch. A good resource for branches can be found here: "http://git-scm.com/book/en/v2/Git-Branching-Basic-Branching-and-Merging".
   c. Merge Conflicts occur when two people edit the same functions code in the same file and then both try to push different code to the online repo. It is not a big deal to fix but if you would like more information a good resource to reference can be found here at the link also listed above: ""http://git-scm.com/book/en/v2/Git-Branching-Basic-Branching-and-Merging". Here is a fairly good resource for a method of resolving such conlicts "http://githowto.com/resolving_conflicts".