Created by Walker Christie

What is Github?

"Github is a web-based Git repository hosting service", Github allows us to manage our code whether we're at school, home, on the moon, or anywhere. You won't be doing all the programming in school!

How are we using it?

We are going to be using Github in order to manage our robot's code. You will be using the Eclipse IDE and Github software in order to accomplish this. Certain robotics programmers who are experienced in Git will be reviewing your uploaded code to ensure that it doesn't break the project setup.

Wait.. changes have to be reviewed?

Yep! I'm sure the code you are writing is fantastic, and will most likely be added to the project. Github is complicated software that can literally destroy the entire project with one command. With a team of reviewers, we will ensure that your upload is *non-destructive* before adding it.

Great, what do I need?

In order to be programming ready, you need to have at least two programs installed on your computer. You will need Github (either for Windows or Mac) and Eclipse. Github's interactive interface is very new, and requires you to have at least Windows 7 or Mac OSX 10.9. This means you cannot *commit* changes with the Windows XP computers unless you have the Git command line installed. Feel free to ask those who know about Github on how to do that.

Lets get started

Start the process by installing Github for Windows, I've included the full URL and a shorter bit.ly link below

https://windows.github.com/
http://bit.ly/1DpR5NJ

Once you have Github installed, it's time to install Eclipse, the links are below

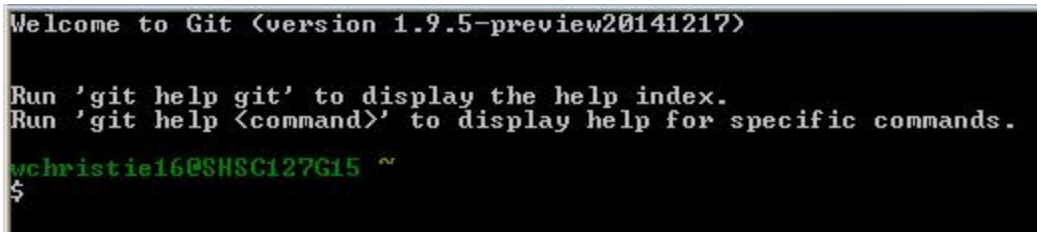
https://eclipse.org/downloads/
http://bit.ly/1xVvXgC

Eclipse is an interesting installation as it downloads a .zip file to your computer. Simply take the (extracted) zip file and drop it where you want Eclipse to be. I normally put Eclipse in my Documents folder

Now it's time to open Eclipse, upon opening eclipse.exe you will be prompted where you would like to store your projects, choose a location and remember where set it to.

Let's set up github.

Getting the project setup on your computer
Start by opening the Git Shell. If you are on Windows XP and have installed an older version, this is called Git bash. Your computer prompt should look something like this



First you need to cd into your project directory, you do this by typing
cd /c/wherever/my/workspace/is
For example, I would type: *cd c/Documents\ and\ Settings/wchristie16/workspace*

Now why did I put a \ before every space? The reason for this is because the Git shell only recognizes space characters if you put a \ before them. When you put a space *without* a \, you are telling the git shell to go into a different folder. A little confusing, I know. Just remember to put a \ before every space!

Anyways, go ahead and type that into the git shell, press enter.

Once you've set your cd, it's time to clone the git repository. You do this by typing..
git clone https://github.com/SHSRobotics4311/2015-Season.git
(This URL is subject to change, if the URL is not found, ask someone where the current repository is located)

This will download (or clone) the robotics repository from Github onto the local computer. Your command prompt should now look something like this

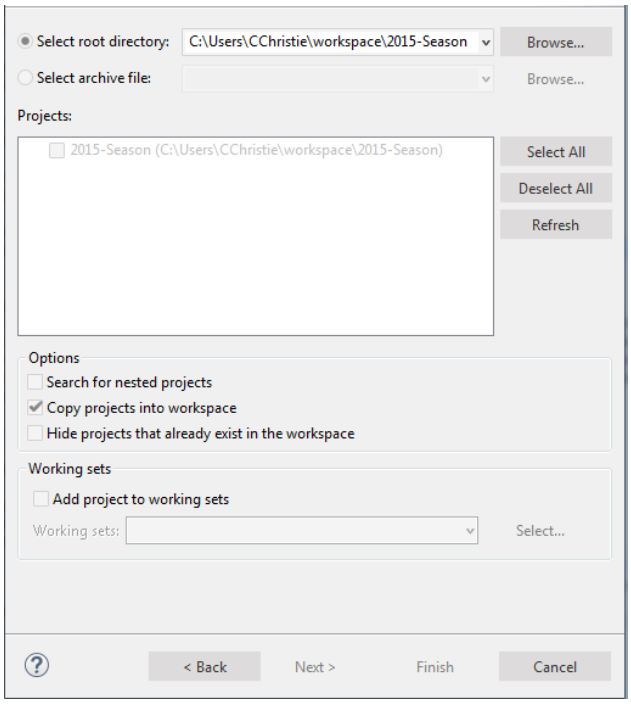
```
wchristie16@SHSC127G15 ~
$ cd "C:\Documents and Settings\wchristie16\workspace"

wchristie16@SHSC127G15 /c/Documents and Settings/wchristie16/workspace
$ git clone https://github.com/SHSRobotics4311/2015-Season.git
Cloning into '2015-Season'...
remote: Counting objects: 46, done.
remote: Compressing objects: 100% (17/17), done.
remote: Total 46 (delta 7), reused 0 (delta 0)
Unpacking objects: 100% (46/46), done.
Checking connectivity... done.
```

Do NOT write git clone ever again (unless you received an error the first time). Git clone downloads the repository from Github. If you ever want to pull changes from the repository, you should use the *pull* github command. Using the *clone* command more than once could cause you to overwrite your project and all the hard work you've done.

Now that Github is setup, let's start setting up our project in Eclipse!

Open Eclipse and select File > Import
When the Import menu appears, expand the "General" folder and select "Existing projects into workspace"



Press the Browse button and search for the "2015-Season" folder that was cloned from github. This *should* be in your workspace folder. Once you've located the folder, check the project that appears underneath the "Projects:" label

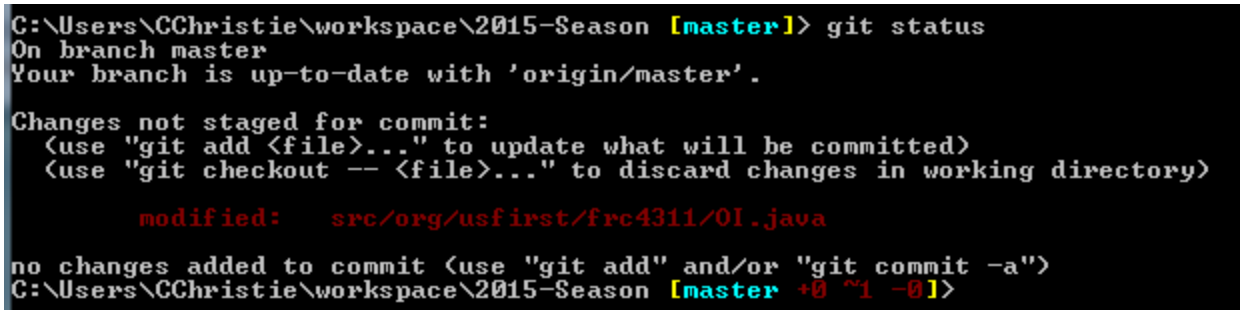
The 2015 project on my screen is faded out because I have already imported the project, yours should be selectable, however. Once you've selected the project, just press the Finish button.

Alright, now you should have Github setup with Eclipse! Now you must learn how to send your changes to Github in order to be reviewed

Committing Changes
Alright, so you've just solved an impossible bug and are super excited to share the fix with your team! How do you do this? It all starts by committing your changes. Many would suggest working with the Github for Windows software, but I am going to show you how to do this through the shell, as it works on all operating systems.

To start, open the git shell and type cd to your Eclipse project, we did this a bit back, so just jump back a few pages if you forget how to cd (change your current directory)

Let's view all of the changes that we have created inside our github project. To test this, I am going to add a comment to the OI.java file. I am now going to check what changes have been made using the *git status* command. Git status shows you all of the changes that have been made while you've been writing in your Eclipse project.



You'll notice my modified OI.java file appears in the changes not staged for commit. To add these to my next commit, all I need to type is *git add .*
The add command allows me to add modified files to my next commit. By typing add . I am telling git that I want to add **all** of my modified files. I can add individual files by typing *git add filename*

Once you've typed your add command, you are ready to *commit* your changes. But what does commit actually mean? A git commit is basically telling git "okay, I am completely sure that I want to update these files to our project". Committing doesn't actually send anything to the server. Yet.

Let's commit our modifications by using the commit command, I will also add a small message to our commit (which is **extremely** important, though optional) using the -m operator. Let's commit our changes below.

git commit -m "Updated OI file with new comment"

Let's break down what this command is doing. First we are telling git to commit our changes (*git commit*). Then we are adding a message to our commit about what we are doing (*-m "Message"*). Why is it so important to add a commit message? Commit messages tell the reviewers why your changes are so important. Without a commit message we don't know what you did to the file or why it was necessary.

```
C:\Users\CChristie\workspace\2015-Season [master +0 ~1 -0]> git add .
C:\Users\CChristie\workspace\2015-Season [master +0 ~1 -0]> git commit -m "Updat
ed OI file with new comment"
[master f315e7f] Updated OI file with new comment
 1 file changed, 1 insertion(+), 1 deletion(-)
C:\Users\CChristie\workspace\2015-Season [master]>
```

Finally, we are to push our commit to the server! This is covered in the next section.

Pushing

Finally! This is where we actually communicate with the Github website, the final step. This step is very easy, and you shouldn't have too much trouble with it. To begin, start by typing the push command

git push origin master

What this does is tell git "Hi! I am **push**ing my changes to the repository which **origin**ates in the **master**". Simple, right? Well, if you did everything correctly your screen *should* look like this

```
C:\Users\CChristie\workspace\2015-Season [master]> git push origin master
Counting objects: 13, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (7/7), 509 bytes | 0 bytes/s, done.
Total 7 (delta 3), reused 0 (delta 0)
To https://github.com/SHSRobotics4311/2015-Season.git
   64b1709..f315e7f  master -> master
C:\Users\CChristie\workspace\2015-Season [master]>
```

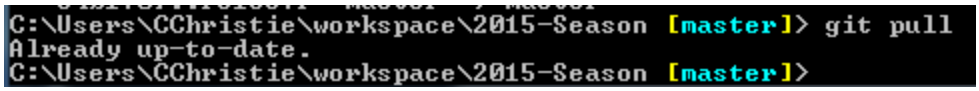
If you are asked to log in to Github, do not be afraid. Simply put in your email and password, then the changes should be committed!

That's it! Wasn't too hard, huh.

Pulling
But what are you supposed to do when you want to download other team member's changes? This is where pulling comes in. Pulling allows you to merge other's changes with your own. Generally, you should always pull changes before starting to write code. So remember to do this before changing anything to the project! Eventually this will become so quick that you can do it in around 5 seconds, so don't feel as though it is a hastle.

To start navigate to your project using the cd command (don't remember? Find out in the commit section)

Once you've done that, simply type *git pull* into the prompt. You might get a message like this



If you get this message, your project currently matches the project on Github.com. This is fine, continue on with writing your code. If github is telling you that it has found changes, it will automatically start downloading said changes, and then you should be able to write your code.

Github terms for newbies
Commit: This takes all of the changes you have created and prepares them to be sent to the Github website. Remember to add a commit message! (-m "message")

Add: This command allows you to add individual files that have been changed. You can add all files, individual ones, or even files that haven't been modified! (Generally not a good idea though)

Push: This takes your most recent *commit* and sends it to the server to be reviewed

Pull: This pulls down changes that you don't yet have from the website. For example, say there is an updated file that you don't have. This will pull **only** that file from the server so that you don't have to redownload the entire project

cd: CD or current directory changes the directory where you are currently "looking in". Generally you should be inside the git project's directory when working with the Git shell or Bash. cd can be wonky sometimes, so don't be afraid to ask for help if something goes wrong with it

Eclipse: The IDE (Integrated Development Environment) where we write the robotics code in

Git Shell: Similar to command prompt, allows you to perform tons of git commands. The git shell is normally colorful, however, making it easier to see what your doing.

Git Bash: Similar to command prompt and git shell, the only difference is that git bash is older and doesn't have the organization or colors that the git shell has. Use this if the computer you are working on is old (such as the school computers)
Github for Windows / Mac: This is new software developed by the creators of Github which allows you to do easy and quick commits, clones, pulls, pushes, etc instead of typing commands into the shell. This is only available on newer operating systems however, so get used to writing in the shell!

Development process cheat sheet
I understand that there are like 1 million things to do here, and that you are probably saying to yourself "where do I start", because of this I have created a little cheat sheet.

If you have never installed github before: Start by cloning the project into your workspace, then import your project into Eclipse. Write your code, add changes, commit, push

If you have a git project already running: First pull changes from Github, then write your code, lastly add, commit, and push

The end
So I understand that this is a bit long. This is supposed to act as a reference on how to manage the monstrosity that is Github. I can guarantee that you will run into some issue while using github, as the program is massive. Don't be afraid to ask for help as this is complicated. Sometimes you just need a second pair of eyes to understand a problem. Good luck and have fun programming!

- Walker