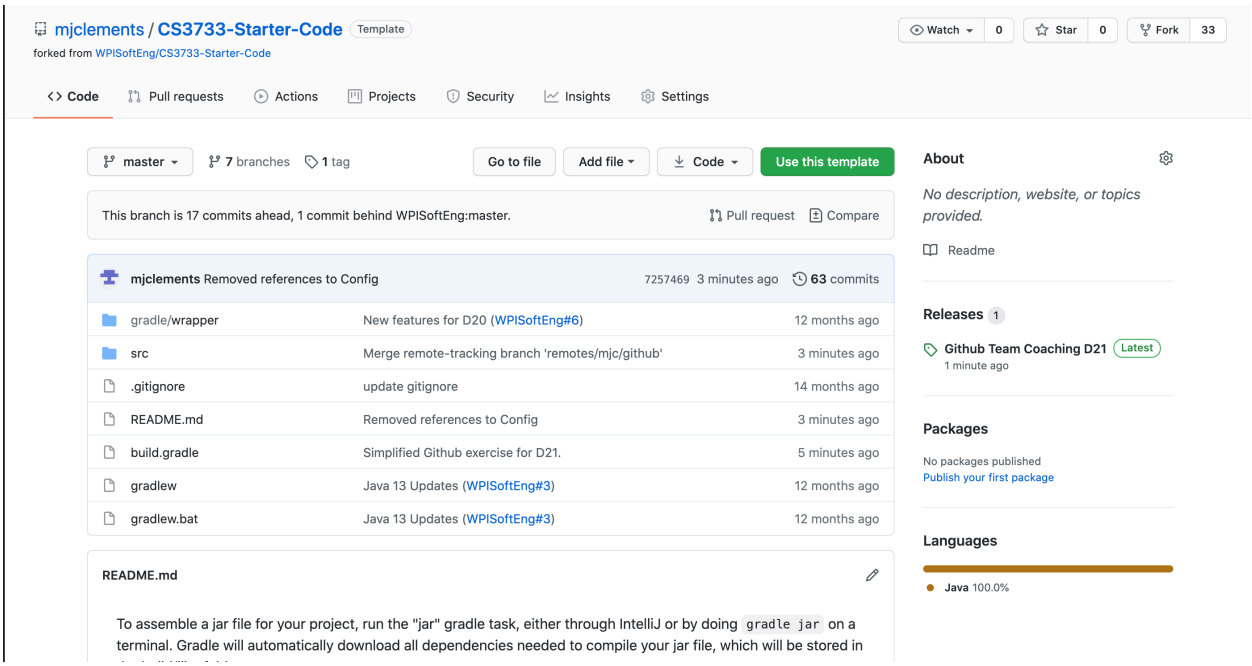
IntelliJ, GitHub, and Travis CI

Step 1. Starting a New Project (One Member Per Team)

This section describes how to start a new Gradle project. Exactly one member of the team should complete this section. If you are not this member, you can ignore this for now. It may be useful later though!

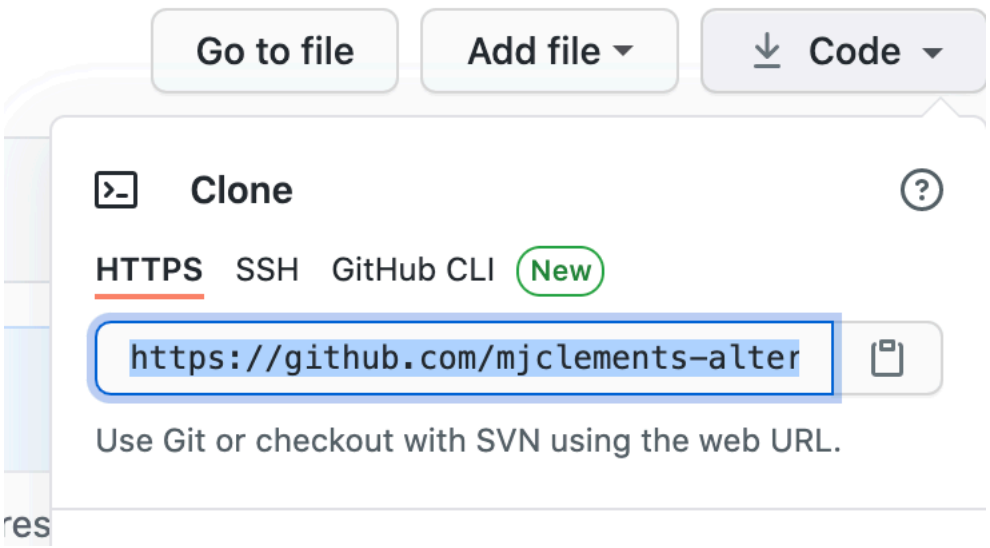
1. Install IntelliJ 2020.3.1 Ultimate using the Educational License.

2. Have **one member** log into Github and navigate to https://github.com/mjclements/CS3733-Starter-Code
3. Click “Fork” in the upper right-hand corner.

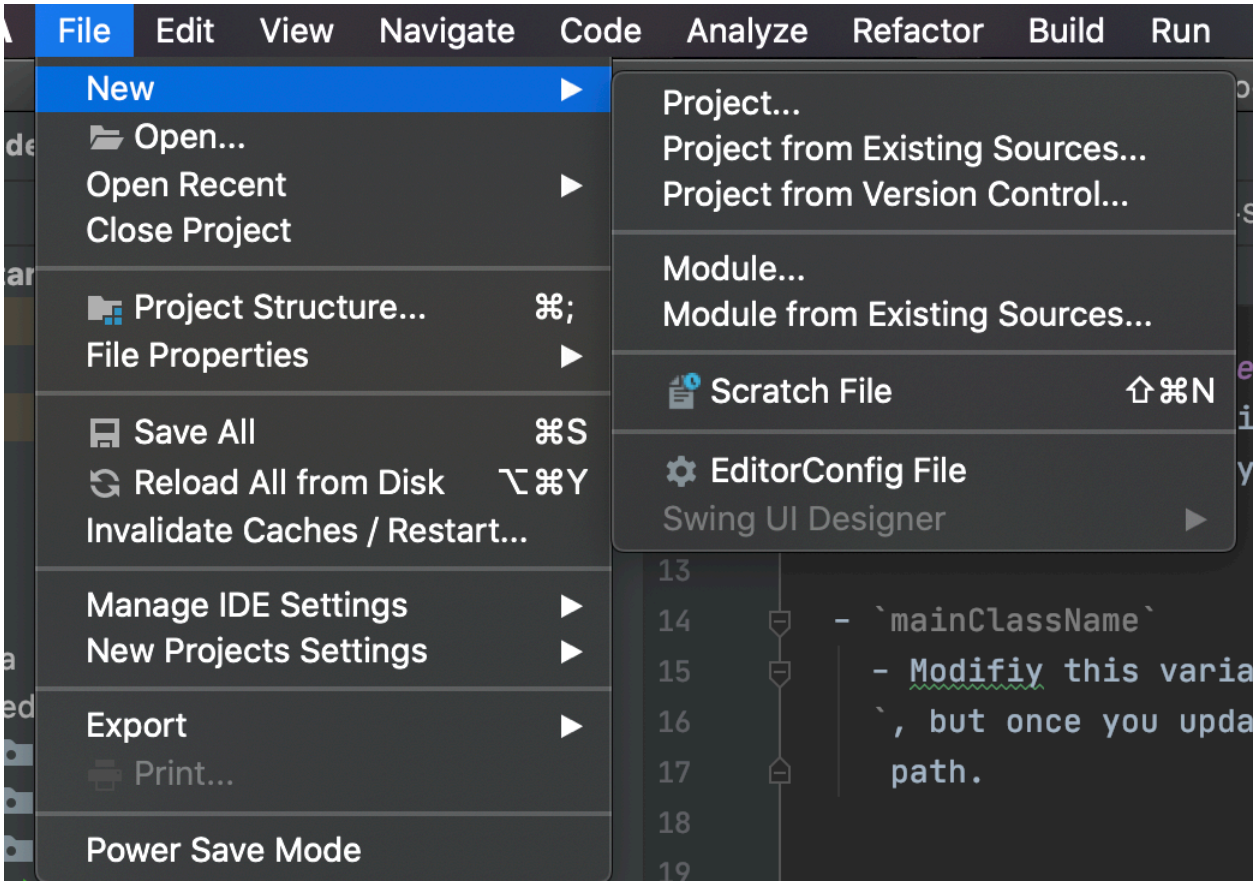


4. It will take you to a page called [your_github_username]/CS3733-Starter-Code

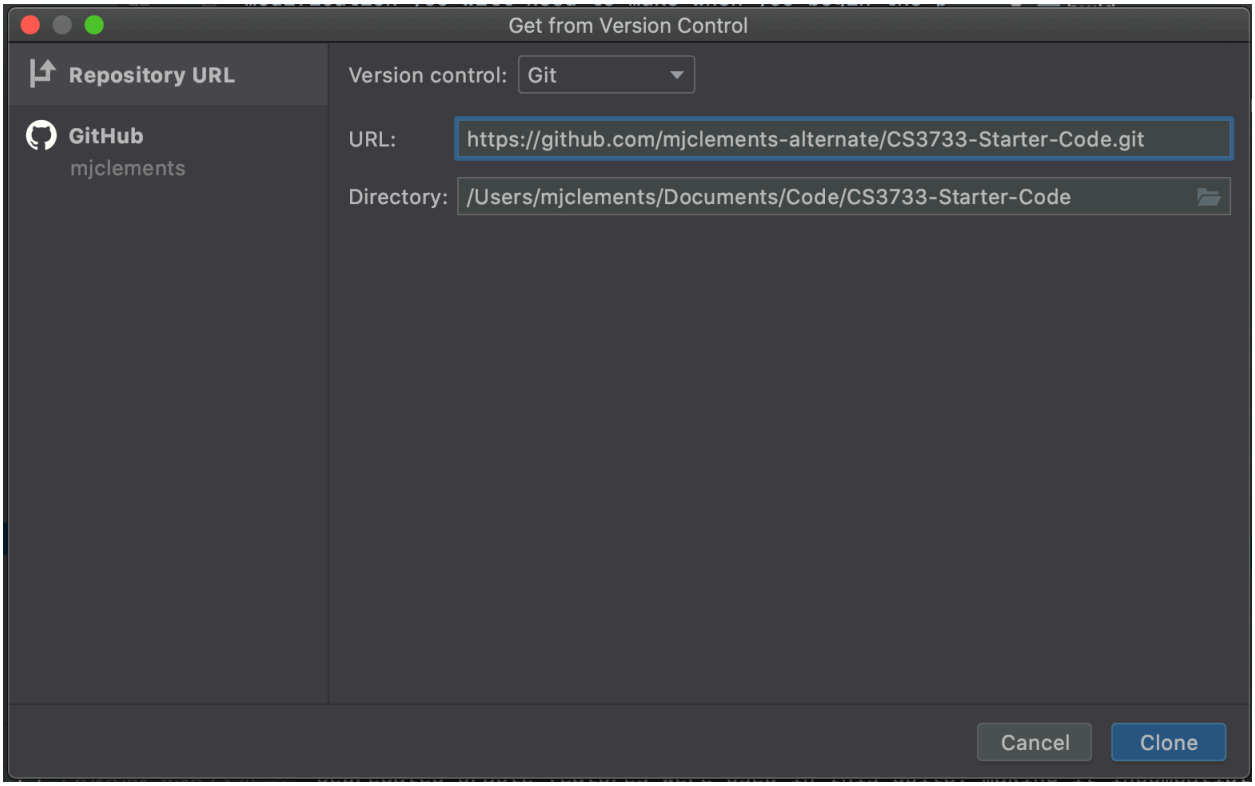
5. Here, click “Code” and then copy the URL it says under HTTPS.



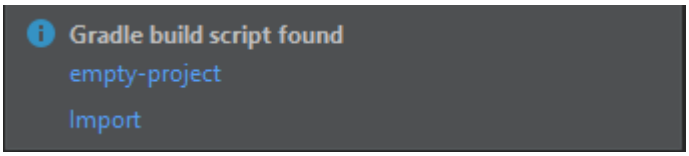
6. Open IntelliJ and click **File->New->Project from Version Control.**



7. Enter the URL you copied earlier.

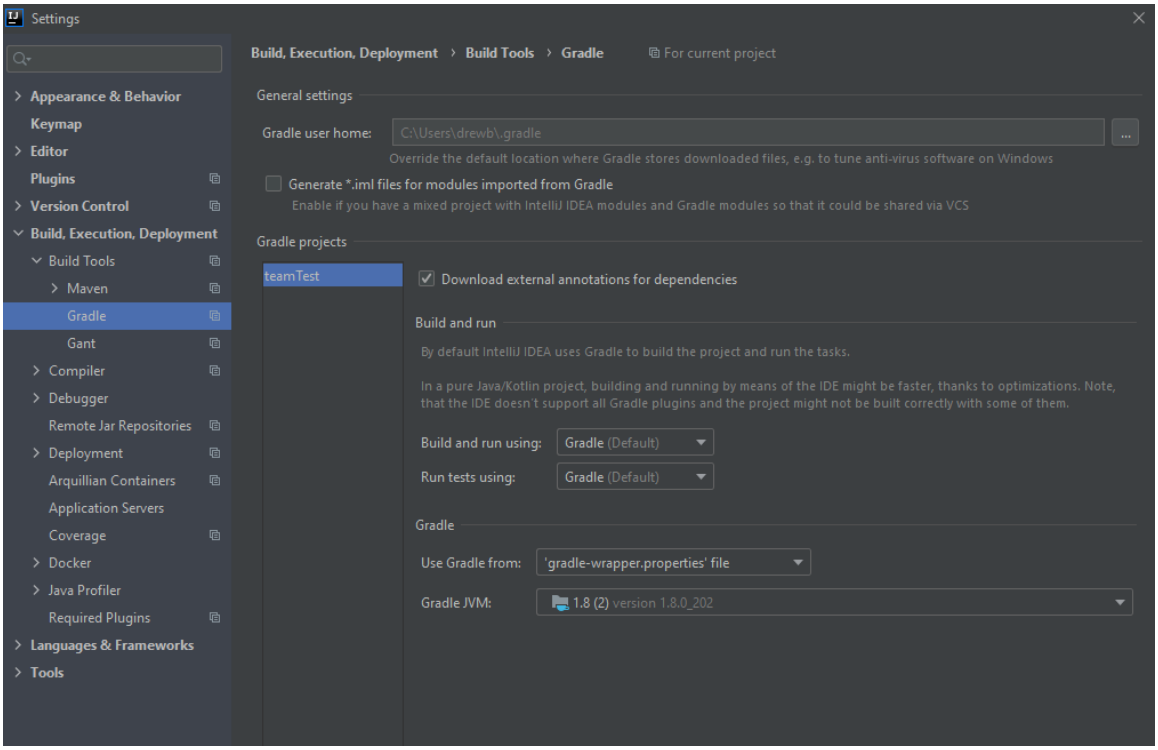


8. There should be a pop-up that appears in the bottom-right that looks like the below:



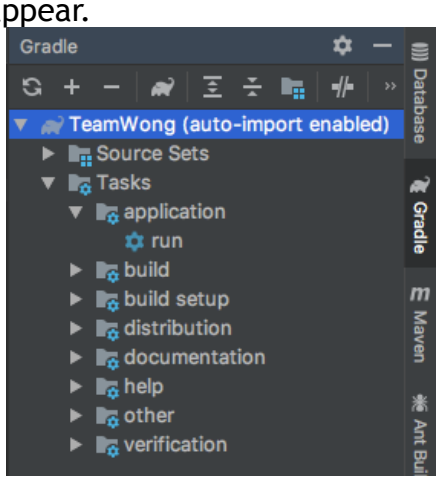
Click import on this to configure the project as a Gradle project

Note: Make sure the JDK that Gradle is using for its Gradle JVM is Java 8 (Java version “1.8.0_2XX”), the JVM may default to the version that your computer uses in its environment variables. If this is the case, you may see “package does not exist errors”. This can be fixed by setting the Gradle JVM in the Intellij settings



9. In the src/main/java folder, navigate to the class called Main and observe its main method. E.g.

10. On the right-hand side of IntelliJ, you should see a Gradle menu option. Open it and the window should appear.



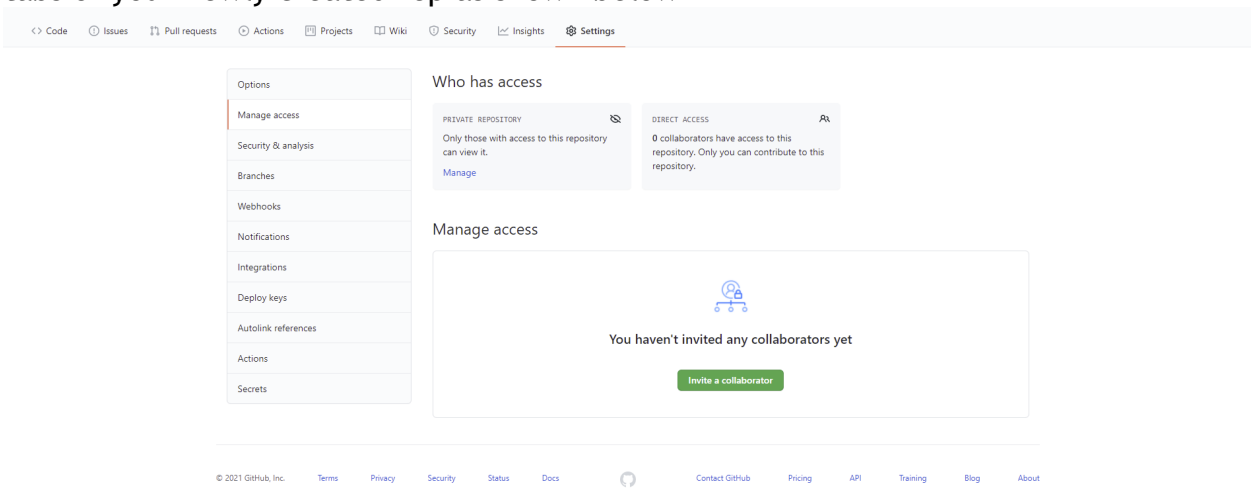
11. Open application then click run. If you had a System.out.println() call in your main method, then you should see the text appear in the console.

Step 2. Beginning Collaboration on Github

Once the above steps have been finished, it is time to make sure your team can contribute to the GitHub repo.

IntelliJ, GitHub, and Travis CI

1. **Make sure to also invite all of your teammates as collaborators on GitHub for the repo you create:** this can be done through the setting and manage access tabs of your newly created rep as shown below

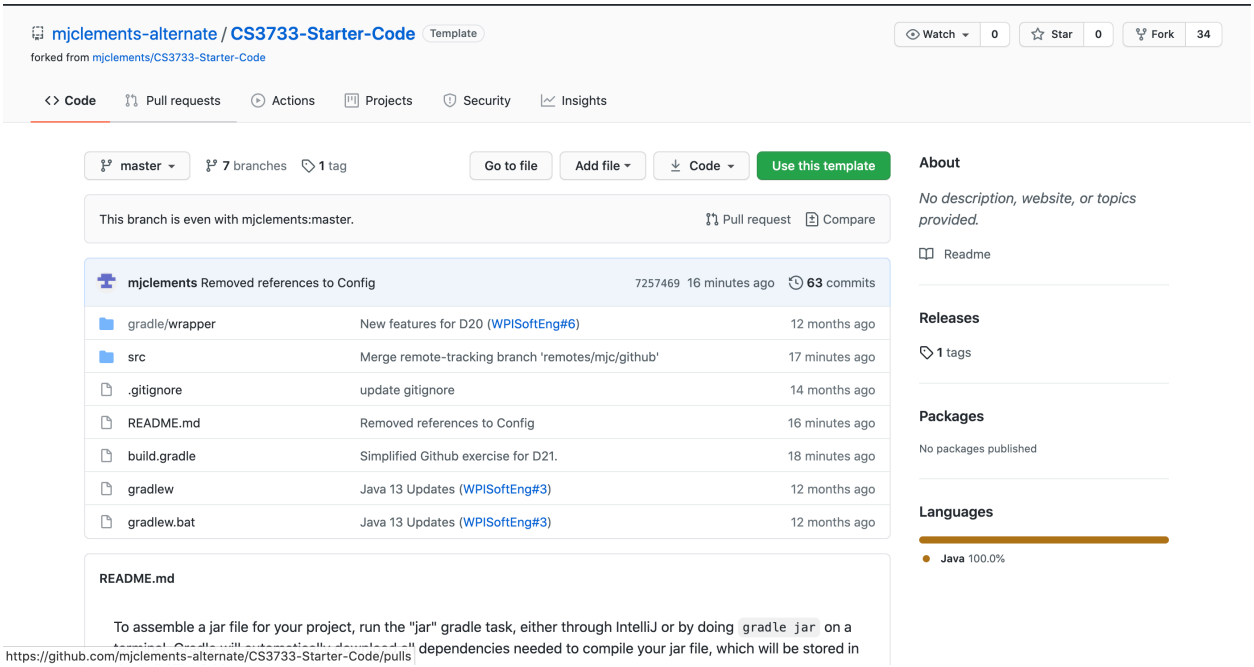


2.

Step 3. Importing a project from GitHub (all other members)

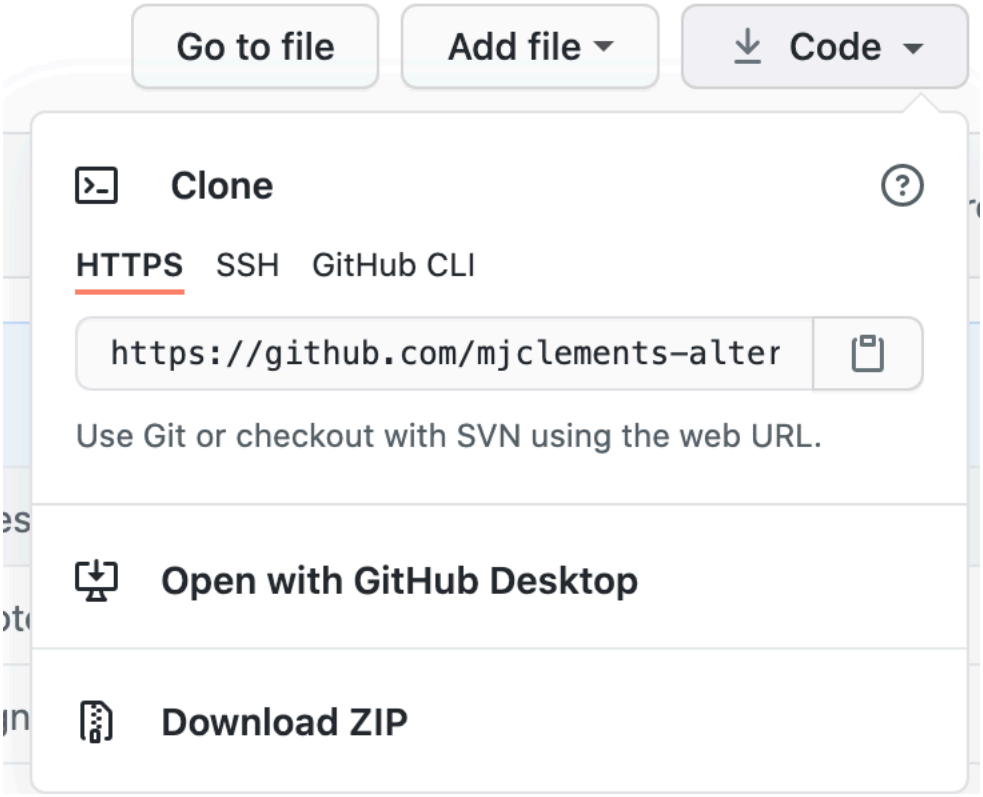
After the project has been shared on GitHub and you have been added as a collaborator, do the following.

1. Navigate to the Github page your teammate created.

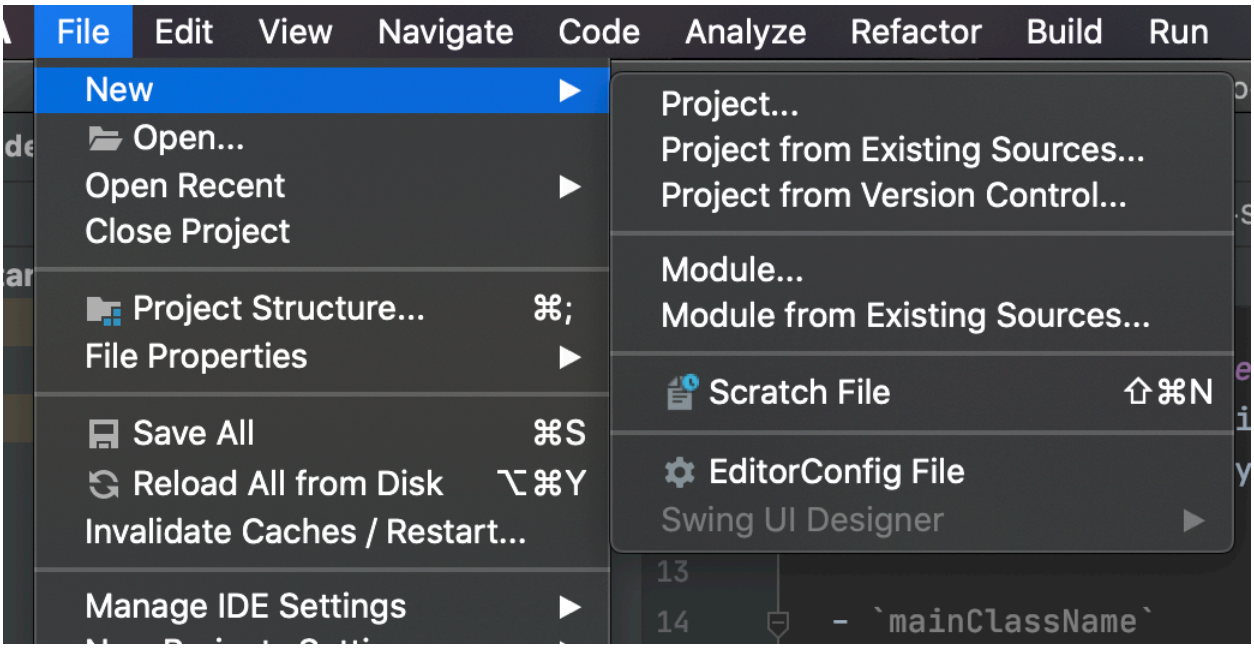


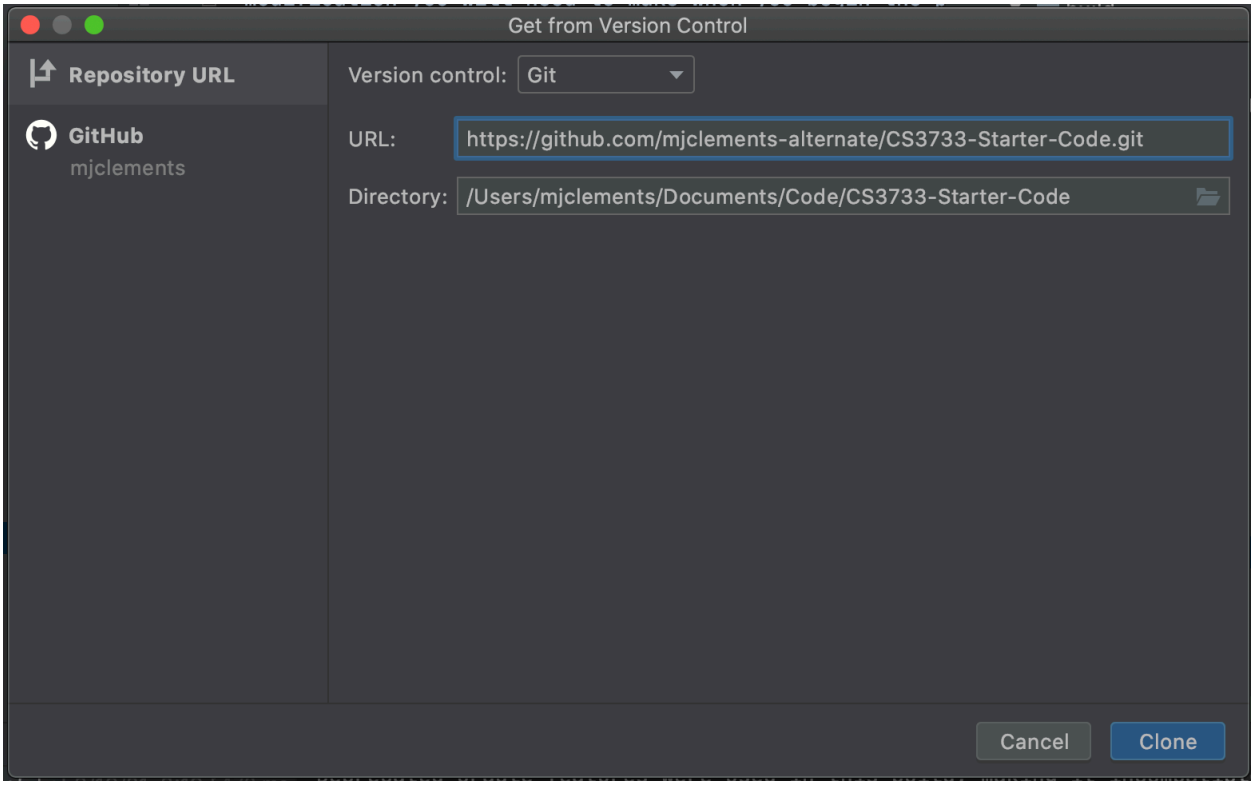
https://github.com/mjclements-alternate/CS3733-Starter-Code/pulls

2. Copy the link under Code -> HTTPS:

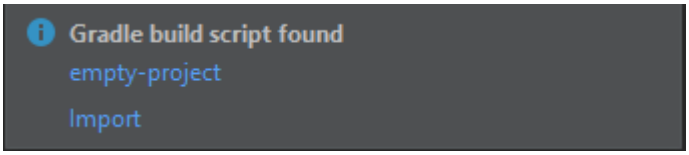


3. Clone the repository using IntelliJ
    a. File-New Project from Version Control
    b. Paste the link you copied earlier





4. There should be a pop-up that appears in the bottom-right that looks like the below:



Click import on this to configure the project as a Gradle project

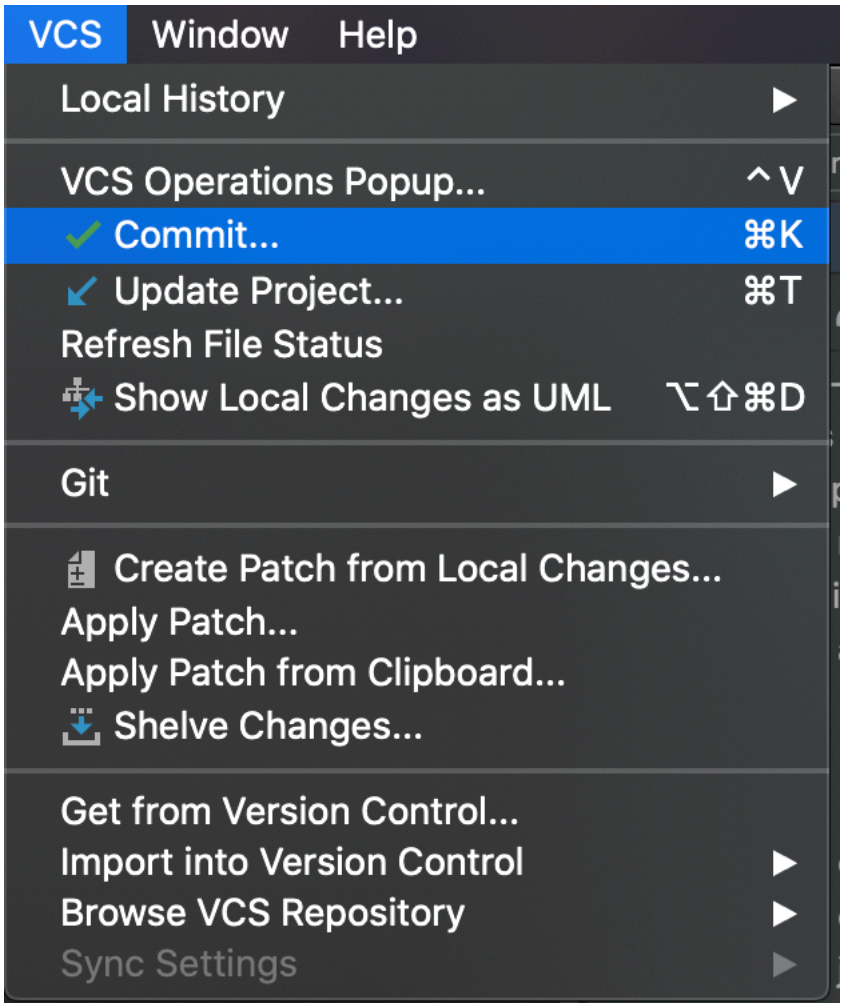
Note: Make sure the JDK that Gradle is using for its Gradle JVM is Java 8 (Java version "1.8.0_2XX"), the JVM may default to the version that your computer uses in its environment variables. If this is the case, you may see "package does not exist errors". This can be fixed by setting the Gradle JVM in the Intellij settings

Step 4. Committing changes (all members)

1. Go the src/main/java folder and add a System.out.println() call with your name. E.g.

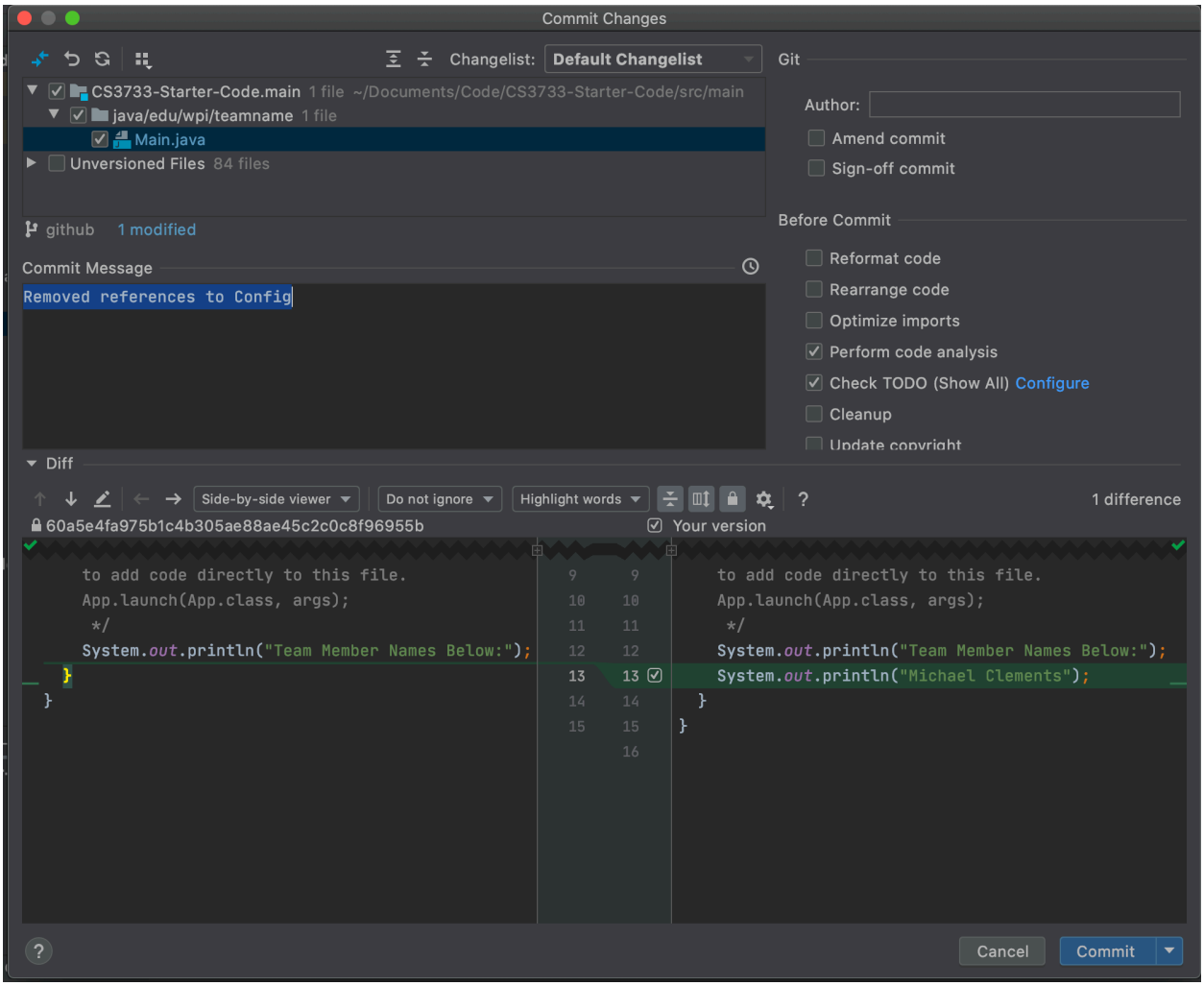
```
System.out.println("Kit");
```

2. In IntelliJ add and commit your files
    a. **VCS -> Commit**

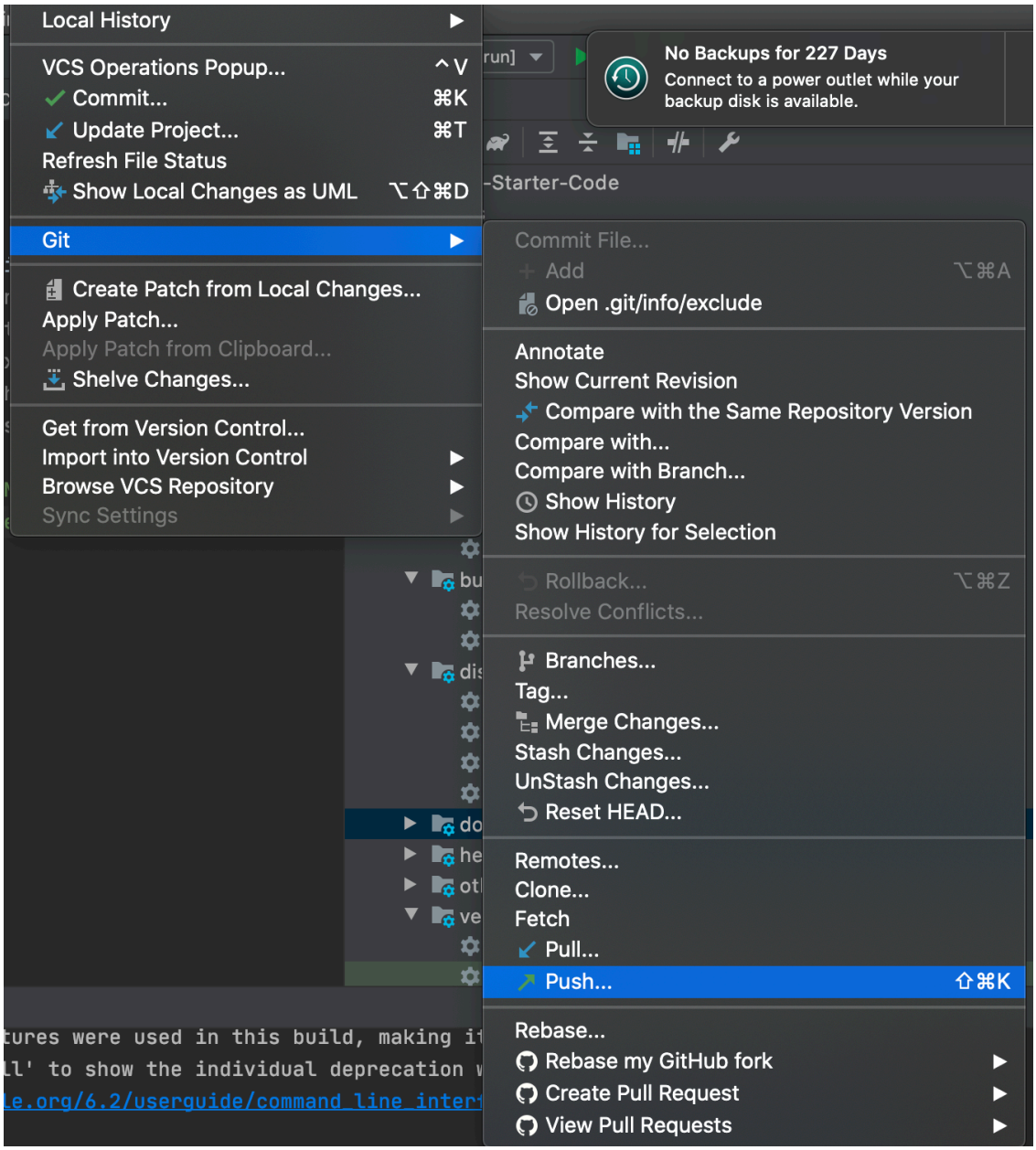


    b. Select **java/edu/wpi/teamname/Main.java**
    c. Add a commit message
    d. Press commit.

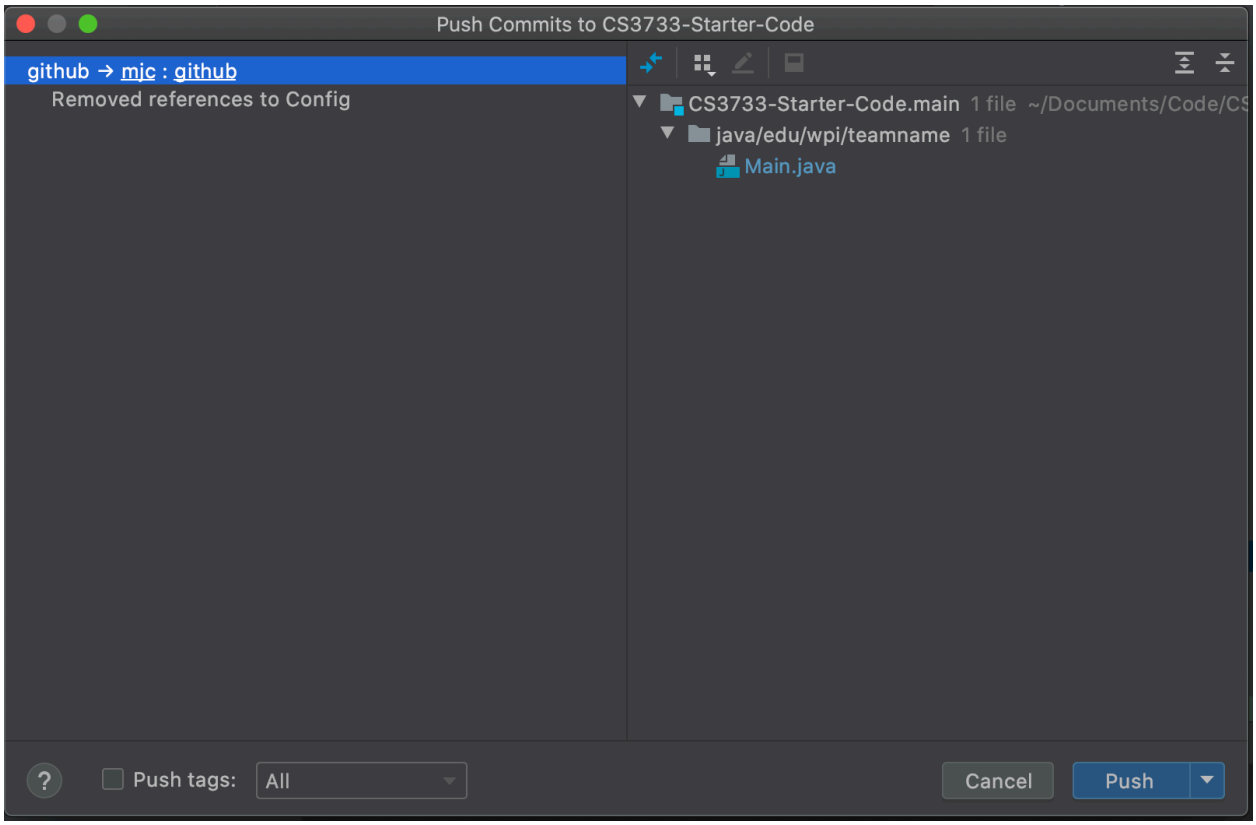
IntelliJ, GitHub, and Travis CI



3. At this point, your files are committed locally, but not pushed to the remote. You now need to push them
    a. Go to VCS->Git->Push.

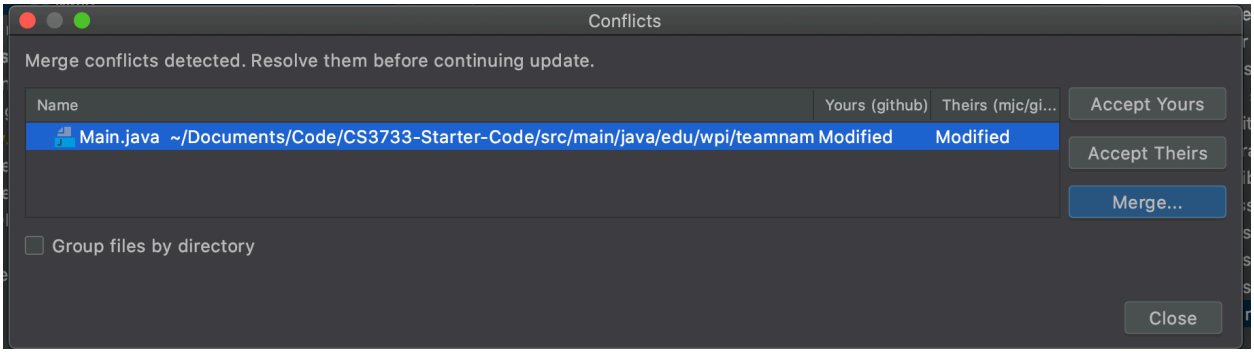


    b. Press push!



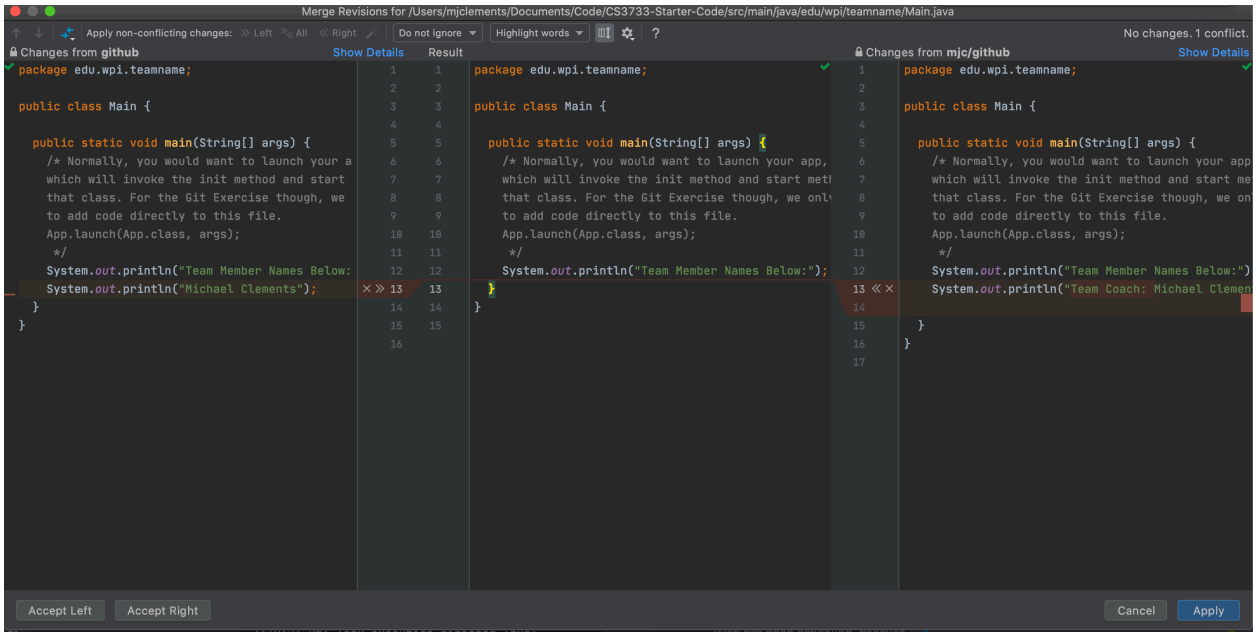
If the push is rejected, that means that there are changes on the remote that you do not have. Pull these changes by clicking “Merge.” (This causes a git pull).

When you pull changes, it is likely you will have a merge conflict, because each team member is modifying the same line of the same file. IntelliJ will show you this with its merge conflict tool To resolve this conflict, select a file and press Merge…



IntelliJ's merge tool is quite powerful. On the left, you will see your code, on the right your teammates, and on the middle your merge resolution.

Click the arrows to 'accept' a highlighted block of code and the x to 'reject' that block. In this case, you will likely want to accept the code from both sides.

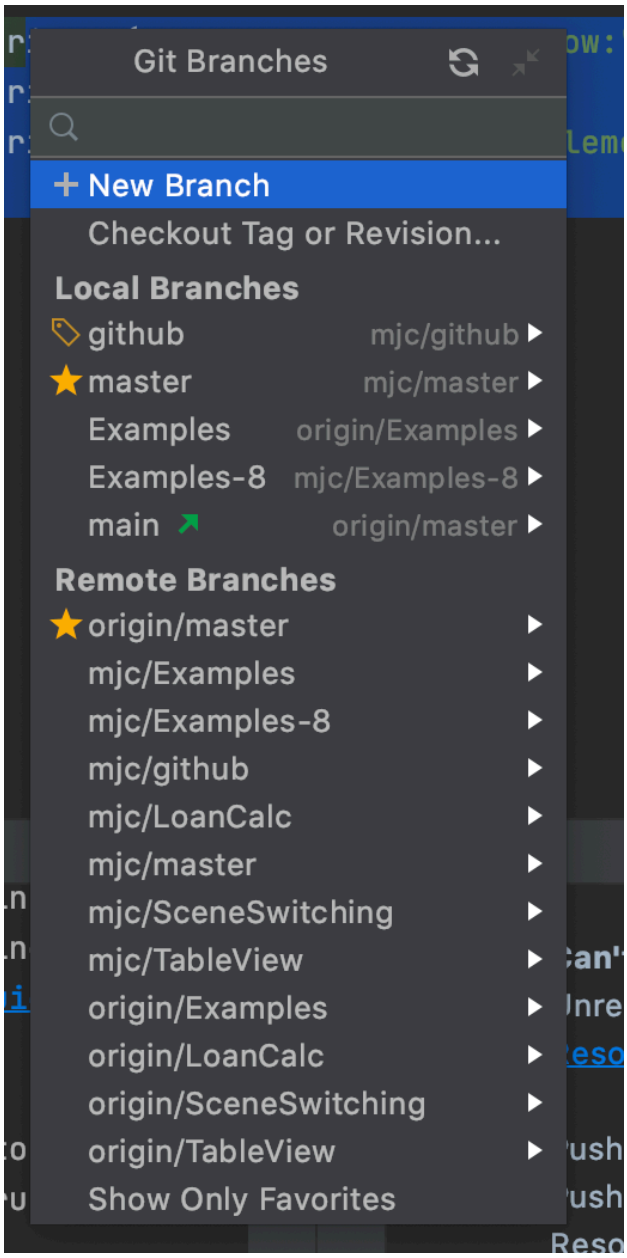


Once all highlighted blocks of code are taken care of, press apply, and attempt to Push again. Unless there are new conflicts, it will work!

Step 5. Branches

If you have time, each team member should create a feature branch. Create a function that simply uses System.println.out() to display the text "Branch – " followed by your name. When done, merge your branch with the master branch.

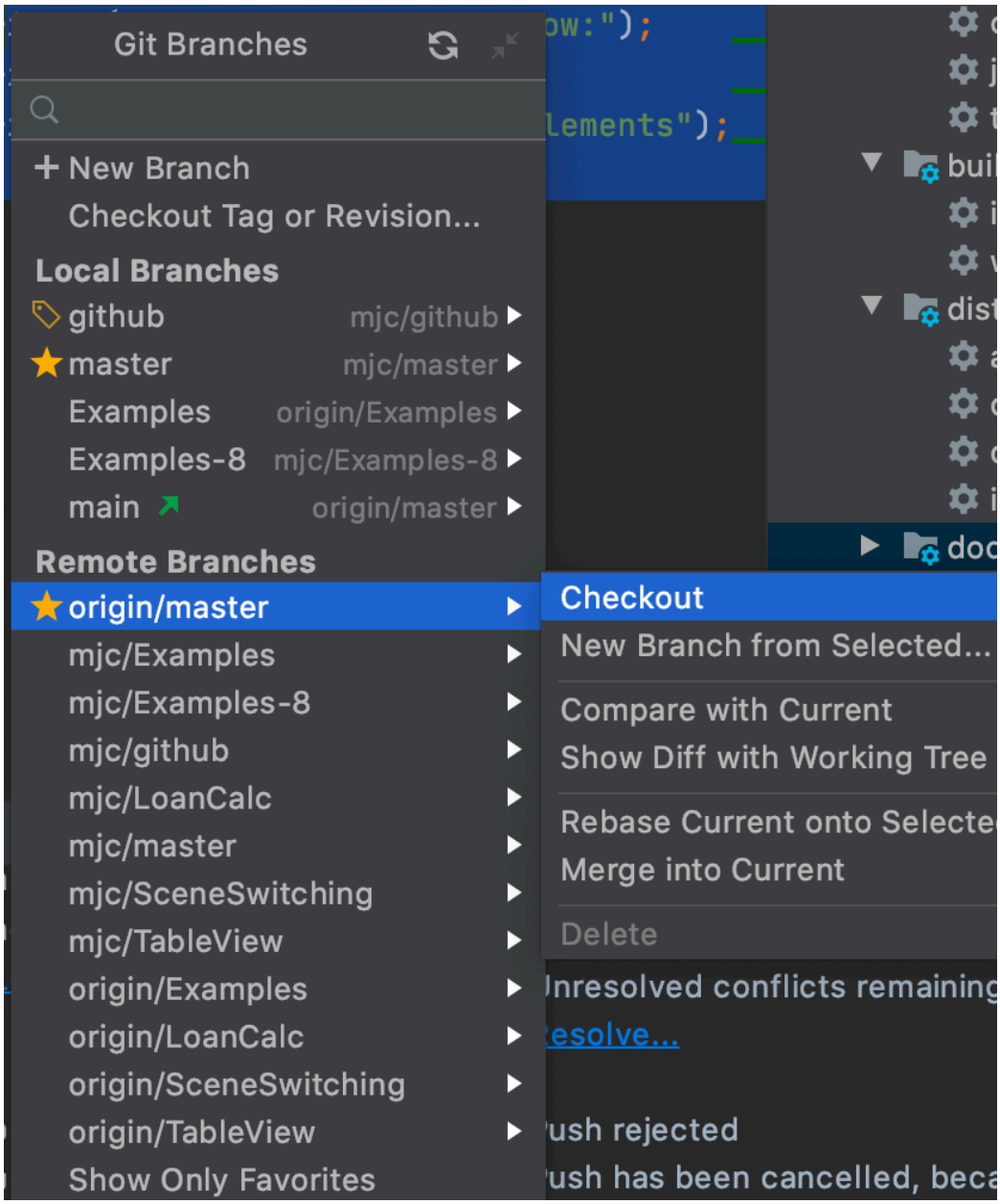
IntelliJ, GitHub, and Travis CI



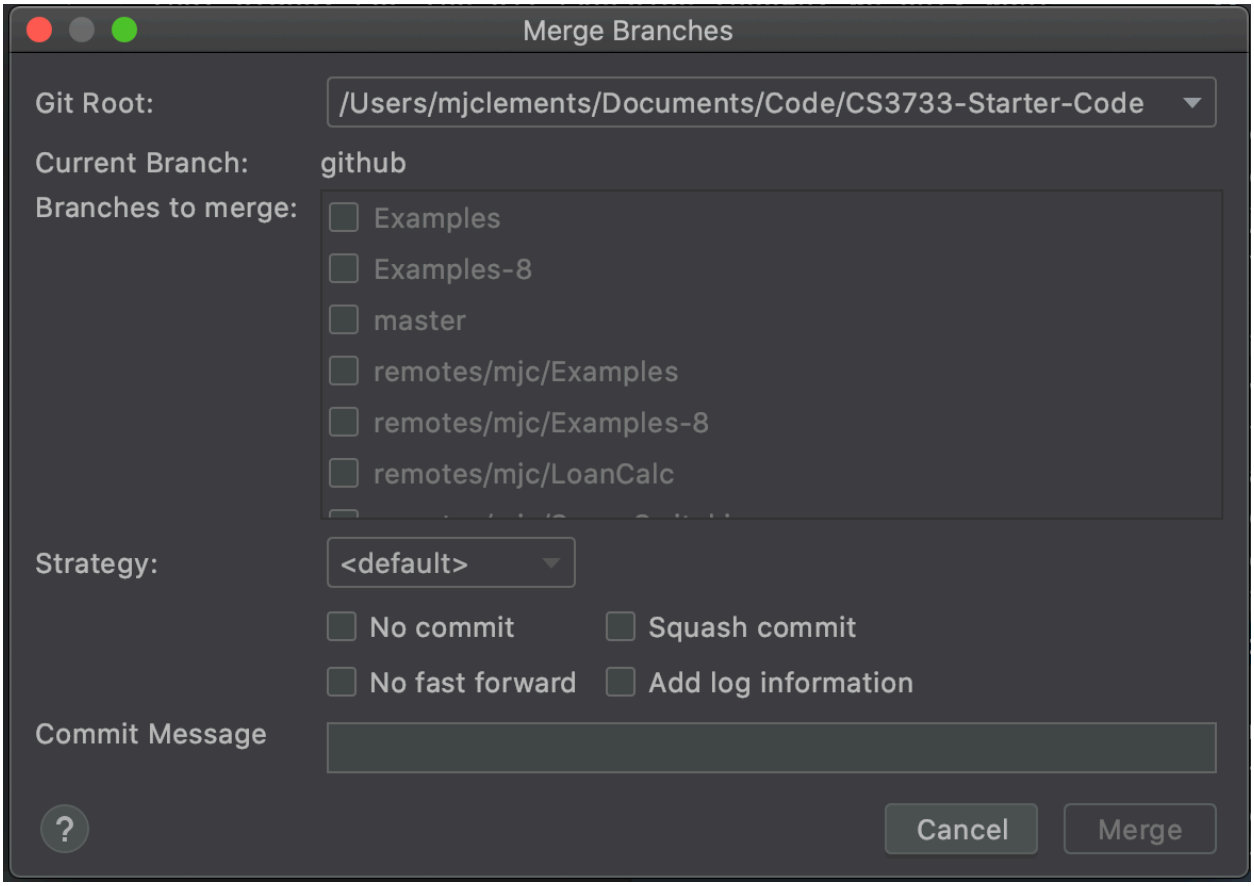
1. Create a new branch using the IntelliJ
   a. VCS-> Git-> Branches
   b. [Popup] -> + New Branch

2. Add your System.out.println() of code and commit/push it

3. Checkout the master branch again
    a. VCS -> Git -> Branches
    b. [Popup] -> origin/Master -> Checkout



4. Merge in your feature branch
    a. **VCS -> Git -> Merge Changes**
    b. **[Popup]** Select your branch. Select remotes/origin/master
    c. Press Merge.
    d. Merge, as described above.
    e. Push!



IntelliJ, GitHub, and Travis CI