

CS193HW2: IDE Instructions

Step 1: Clone this repository

Without getting too deep into the commands of git, we want to "clone" our repository and edit it locally like any other file (if you are interested, [check out this guide](#))

IntelliJ has a great deal of support for exactly this! We can store a local version of our repository, and push any changes to the remote repo easily!

How to clone remote repo to local repo and manage it with IntelliJ

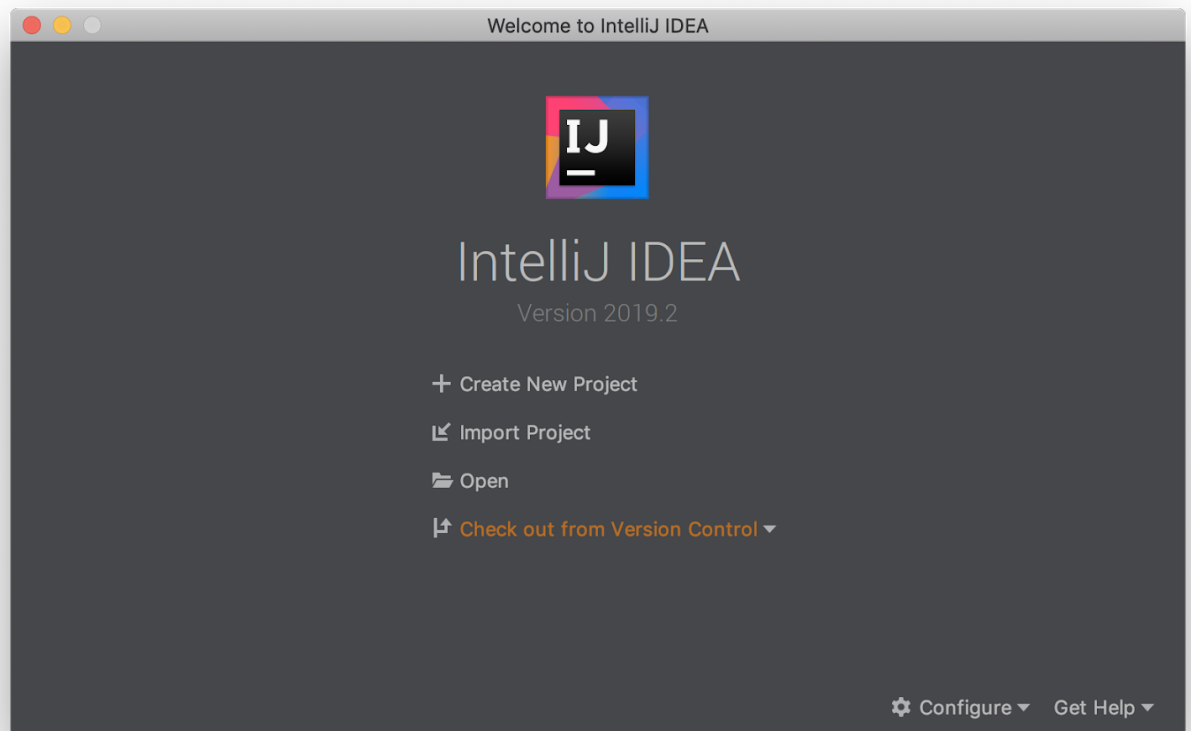
1. Close any projects you currently have open

File -> Close Project

You should now see the main IntelliJ screen (It looks like the picture below).

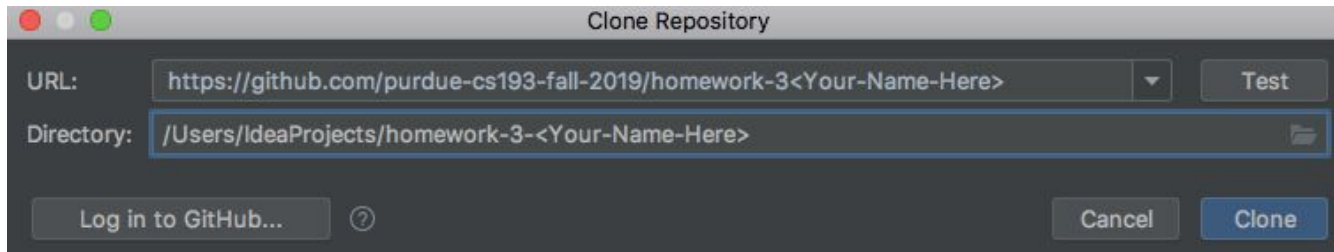
NOTE: If you just clone without letting IntelliJ configure everything for you, it's a pain in the butt to fix. Please let IntelliJ create everything for you!

2. Checkout a project from Version Control. This is done from the welcome screen in IntelliJ!

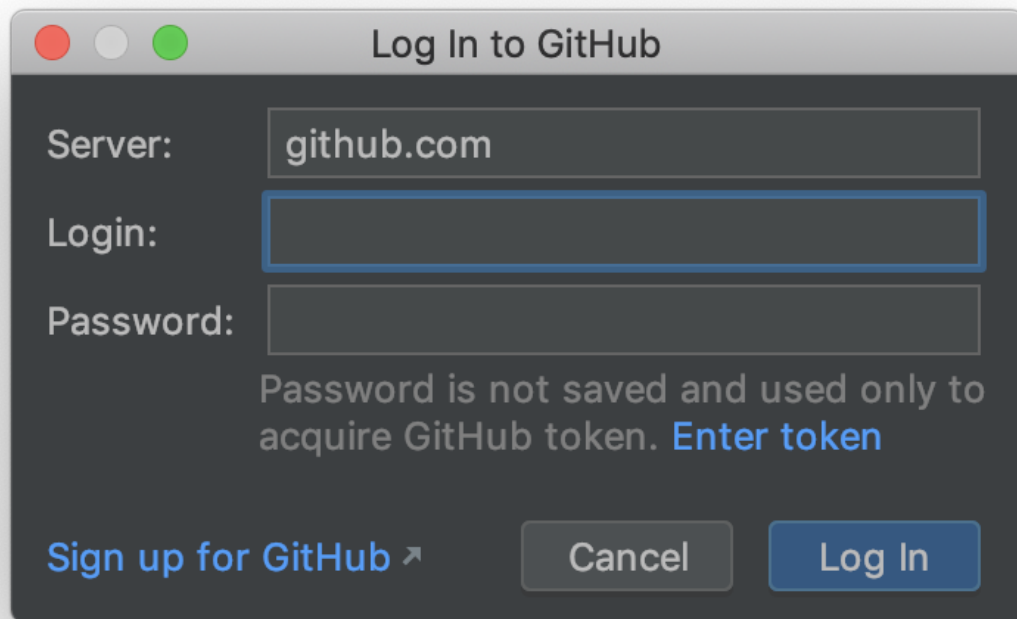


Click Check out from Version Control -> Git -> Log in to GitHub...

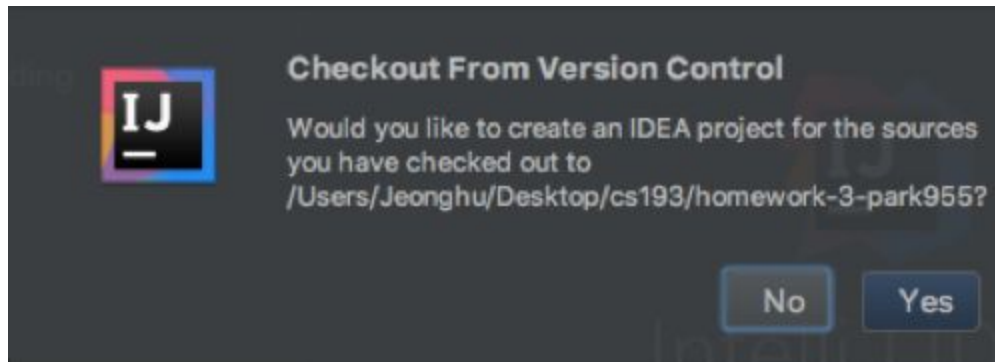
3. Click 'Clone'



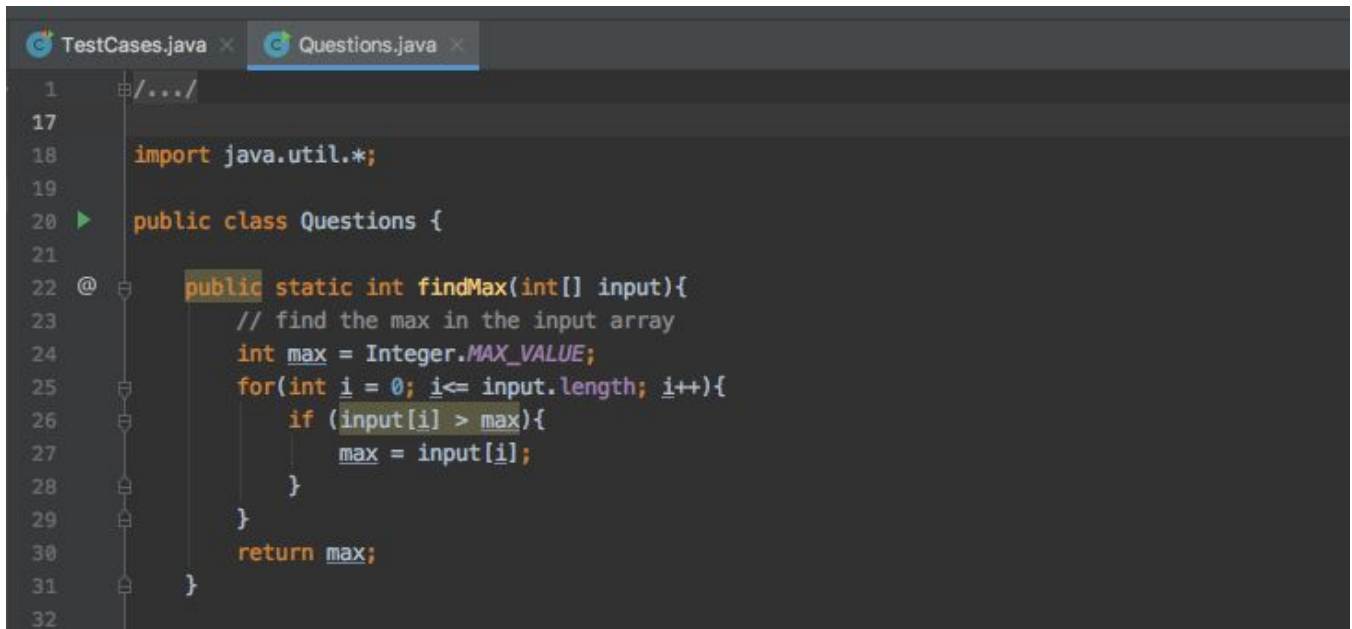
4. Enter your GitHub credentials if asked for. (Note: The order of steps 3 and 4 might be flipped)



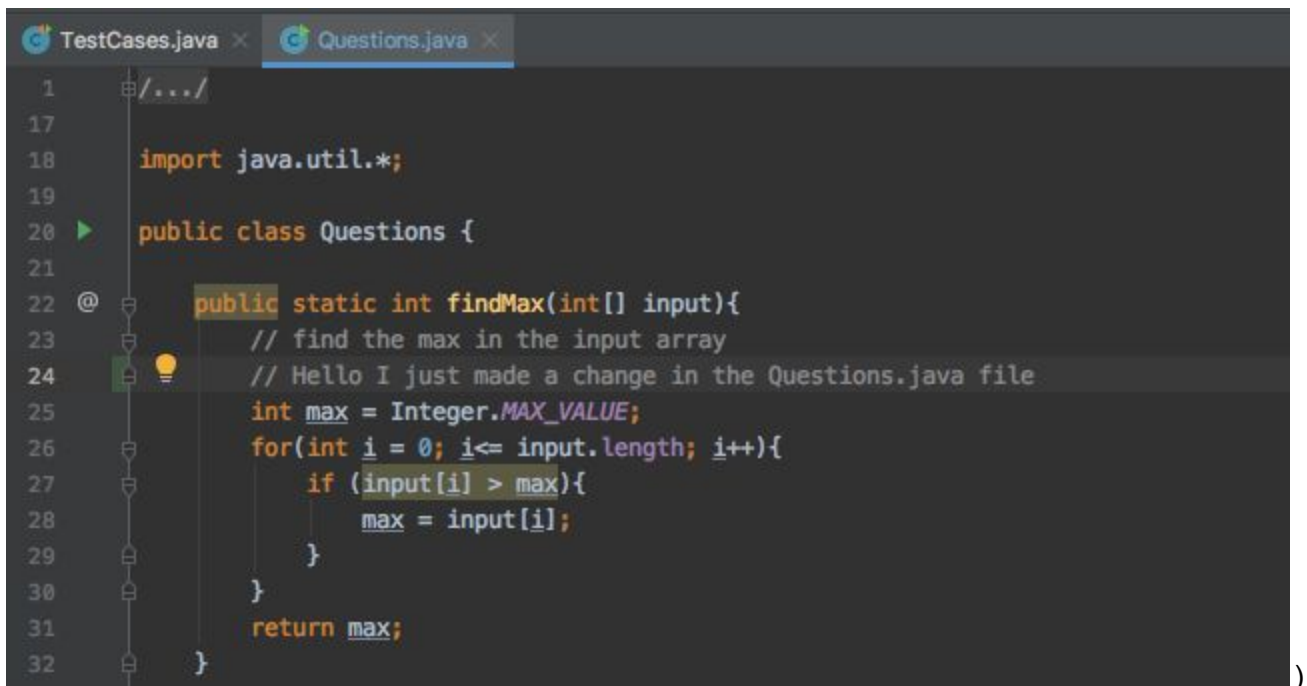
5. Confirm checkout from version control. Click 'yes'



6. IntelliJ will now ask you if you want to customize setup or rely on the default. You can just click 'Yes' or 'Confirm' on the screens that follow
7. Make some changes to the documents



```
1  .../
17
18  import java.util.*;
19
20  public class Questions {
21
22  @ public static int findMax(int[] input){
23      // find the max in the input array
24      int max = Integer.MAX_VALUE;
25      for(int i = 0; i<= input.length; i++){
26          if (input[i] > max){
27              max = input[i];
28          }
29      }
30      return max;
31  }
32
```



```
1  .../
17
18  import java.util.*;
19
20  public class Questions {
21
22  @ public static int findMax(int[] input){
23      // find the max in the input array
24      // Hello I just made a change in the Questions.java file
25      int max = Integer.MAX_VALUE;
26      for(int i = 0; i<= input.length; i++){
27          if (input[i] > max){
28              max = input[i];
29          }
30      }
31      return max;
32  }
```

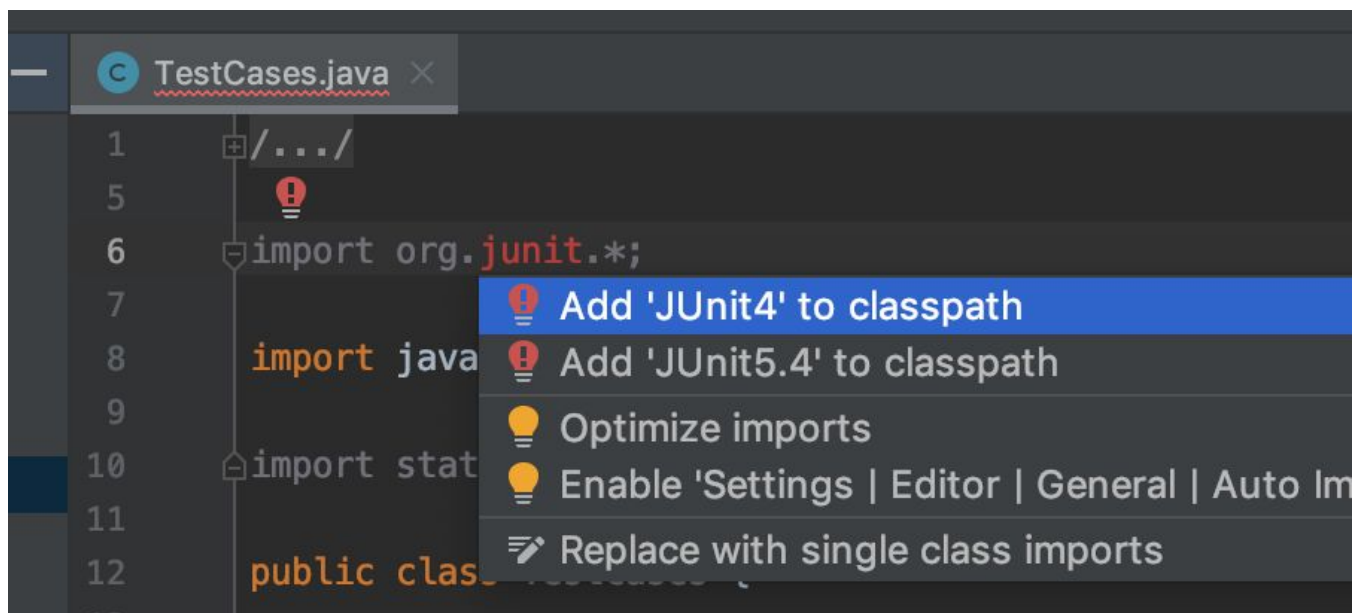
Step 2: Running and debugging JUnit testcases

1. Configure IntelliJ for JUnit



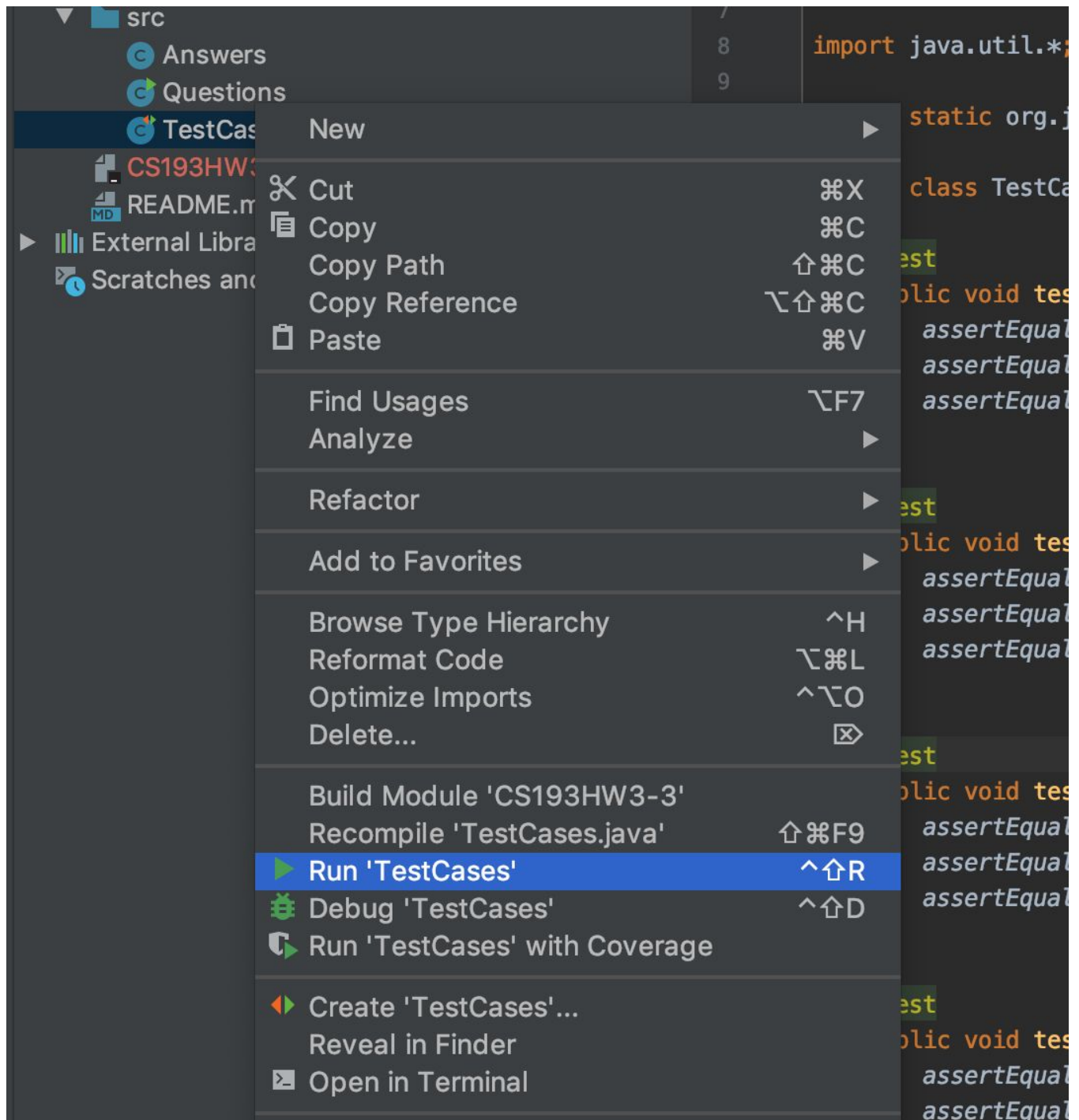
```
1  + /.../  
5  
6  import org.junit.*;  
7  
8  import java.util.*;  
9  
10 import static org.junit.Assert.assertEquals;  
11  
12 public class TestCases {  
13
```

You just need to hit ALT + ENTER and choose "Add JUnit4 to classpath". ALT + ENTER is your friend in IntelliJ :)






2. Run all the test cases

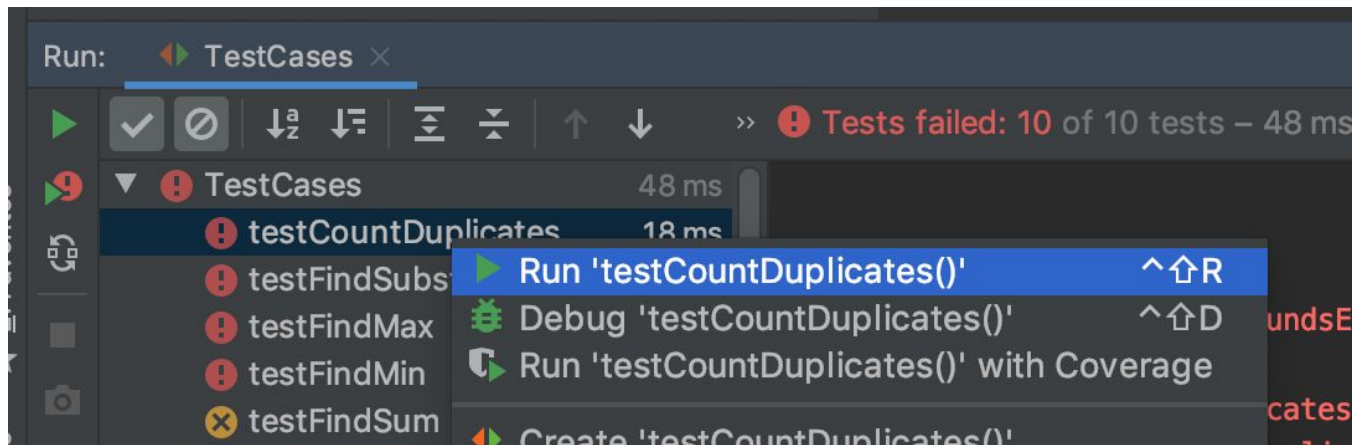
You can run all test cases by right clicking the test case file.



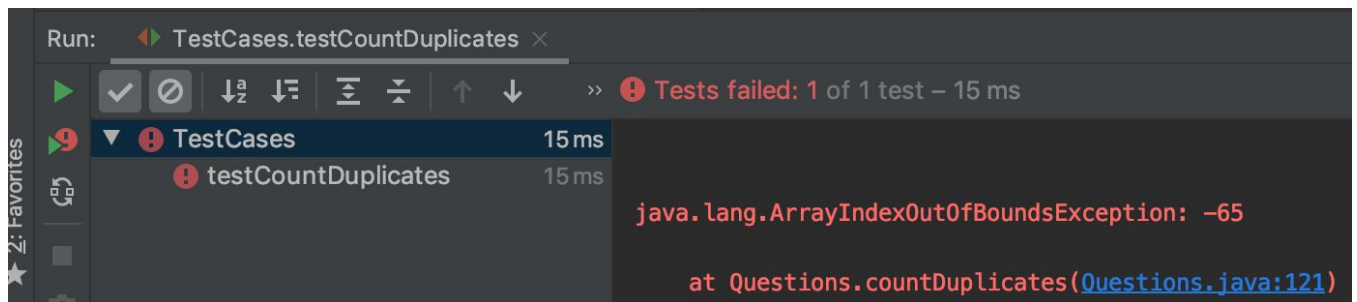
Note on test case results:

-  is good, because you passed all checks
 -  means you got a wrong result in a test case
 -  means an exception was encountered when running the case
3. Run a particular test case

You can also isolate a single test case and run only that one by right-clicking it.

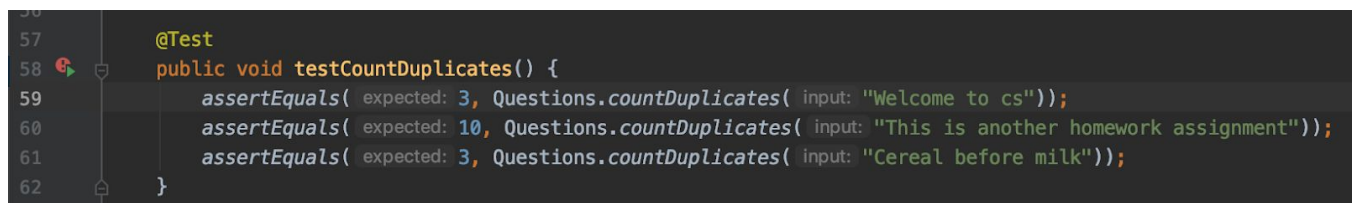


This results in:



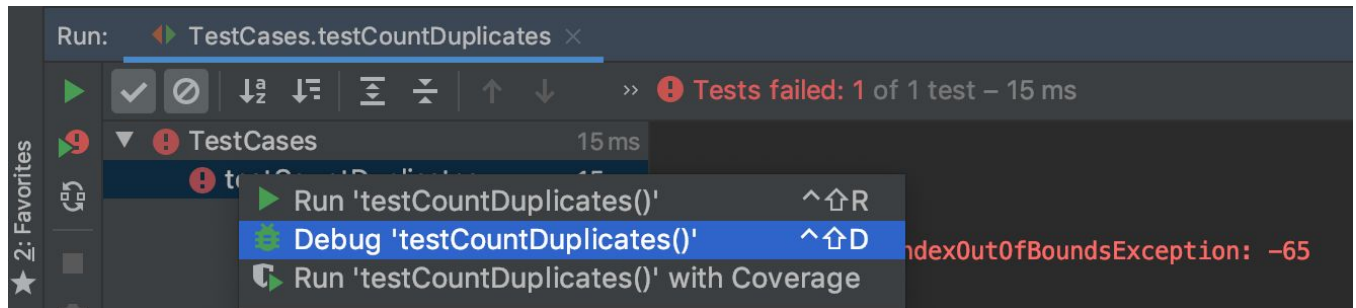
4. Find the line where it goes wrong

Once the test case fails, you can pinpoint the exact issue by double-clicking the failed case. This will show you either the assert statement you got incorrect, or the line where an exception was encountered.



5. Set method and line breakpoints and start test case in Debug mode

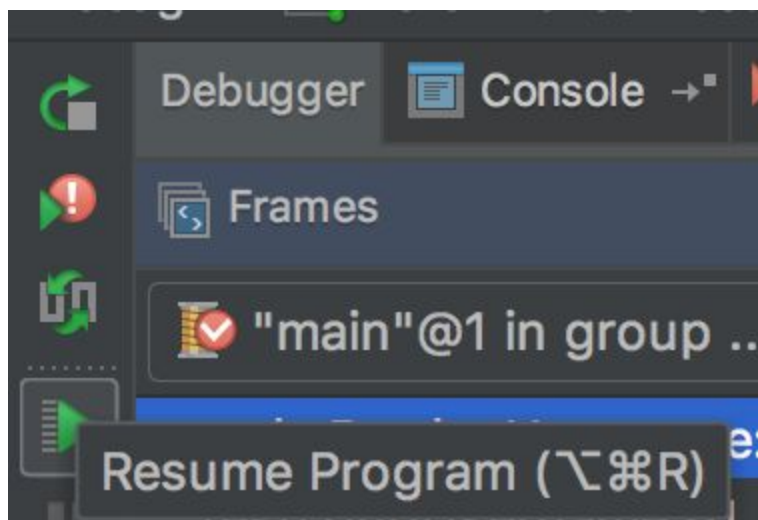
Set some breakpoints in the methods you suspect this test case has touched, and then run the test case in Debug mode.



6. Continue stepping through the code and find the issue

You can see the step-by-step execution of your code, as well as the values of variables and how they change.

To move onto the next breakpoint sector, you have to resume the program.



7. Take advantage of the information you find, and just keep stepping!

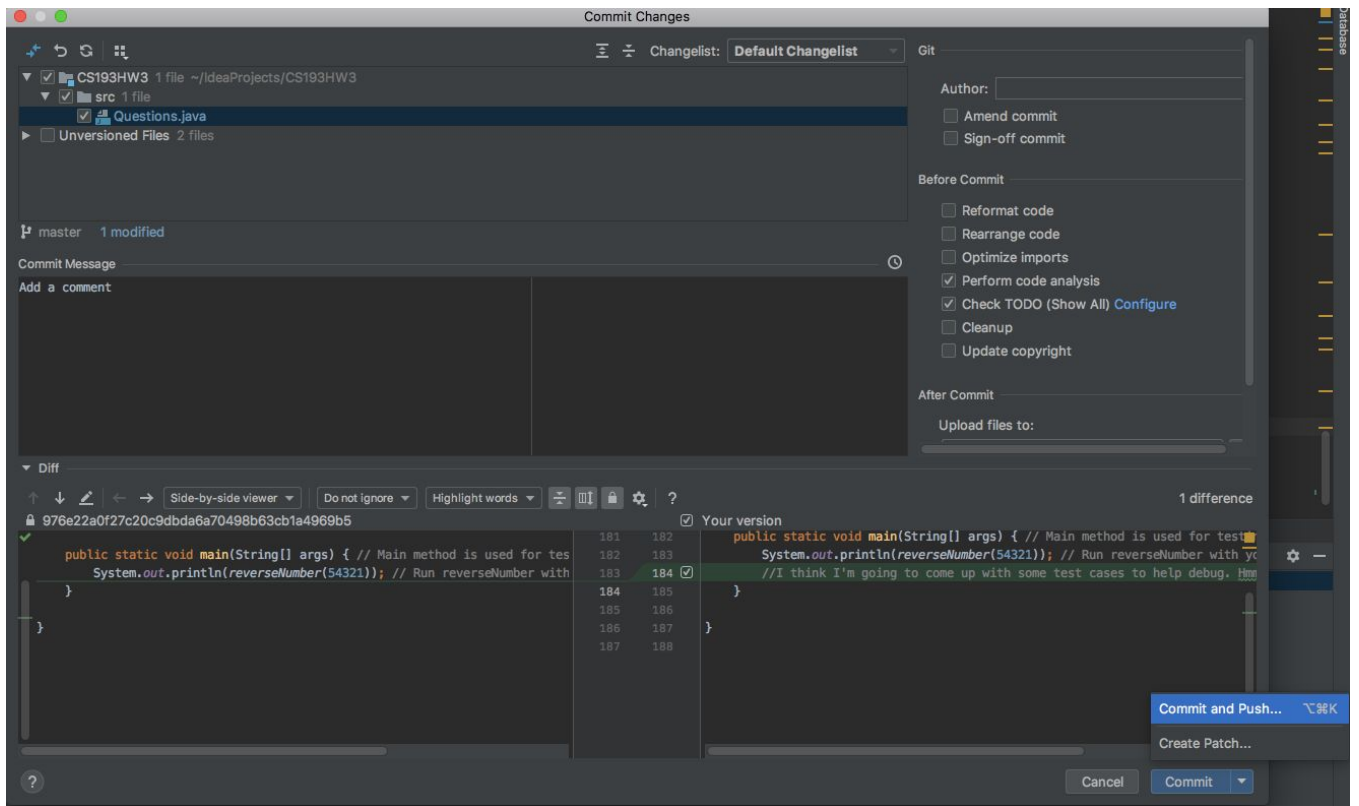
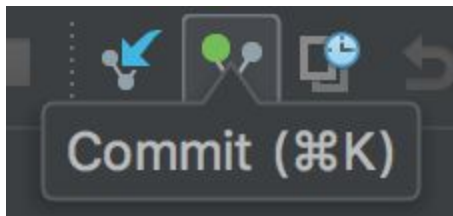
Step 3: Fix those bugs

Use print statements, the debugger, or any strategy you'd like to fix the bugs in the methods and pass the test cases. You might also want to try to understand the algorithms, such as binary search, as they commonly show up in technical interviews (although you're not required to do this). **You are NOT allowed to make any changes to the test case file. If you feel there is a need to make changes, please email your TA or post on Piazza. It will be an automatic 0 if changes are made!**

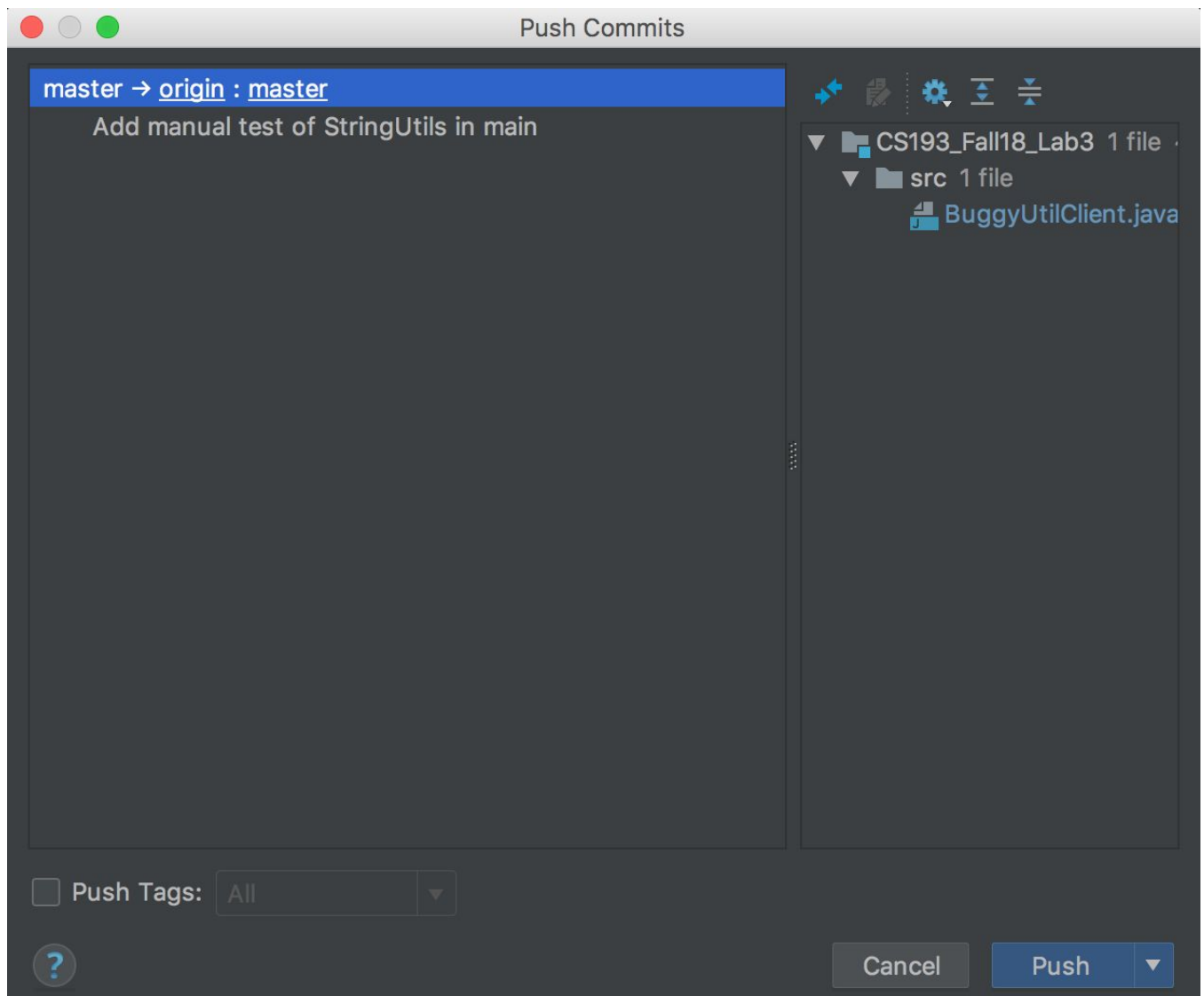
[Click here for IntelliJ's debugger documentation](#)

Step 4: Push your changes to GitHub!

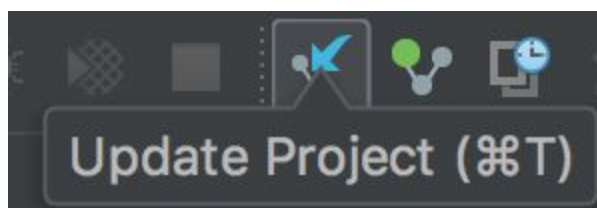
1. Commit and push your changes. **Note: Depending on version, the pull and commit may look like blue and green arrows instead.**



2. Confirm you'd like to push changes



NOTE: Each time you open this project, you should **PULL** to make sure your local repository is up to date with the remote repository



As with every other homework, if your code isn't on GitHub we won't be able to grade it! Once you're happy with the fixes you've made to the code (likely when all the test cases pass), save your work to GitHub.