# 

**SE-314: Software Construction**

**Class: BESE 12B**

# Lab 03: Unit Testing

|  |  |
| --- | --- |
| Name | Saddam Hussain |
| CMS | 370126 |
| Class/Section | BESE-12-B |

**Date: 06th Oct 2023**

**Time: 10:00 AM** **- 12:50 PM   
 02:30 PM – 04:50 PM**

# Instructor: Ms. Naema Asif

# Lab Engineer: Mr. Aftab Farooq

### 

**Introduction:**

# Lab 03: Turtle Graphics

Students will have hands-on experience of JUnit and Git.

Material:

https://ocw.mit.edu/ans7870/6/6.005/s16/psets/ps0/

**Lab Tasks**

Solve problem 0 -3 of part 1 listed on the link.

**Problem 0: Install and Setup Problem 1: Clone and Import Problem 2: Unit Testing**

**Look at the source code contained in RulesOf6005.java in package rules . Your warm-up task is to implement**

A white background with black text

Description automatically generated

**Once you’ve implemented this method, run the main method in RulesOf6005.java .**

public static void main(String[] args) is the entry point for Java programs. In this case, the main method calls the mayUseCodeInAssignment method with input parameters. To run main in RulesOf6005 , right click on the file RulesOf6005.java in either your Package Explorer, Project View, or Navigator View, go to the *Run As* option, and click on *Java Application* .

Run the unit tests.

To run the tests in RulesOf6005Test , right click on the RulesOf6005Test.java file in either your Package Explorer, Project View, or Navigator View, and go to the *Run As* option. Click on *JUnit Test* , and you should see the JUnit view appear.

If your implementation of mayUseCodeInAssignment is correct, you should see a green bar, indicating that all the tests (there’s only 1 test, containing 2 assertions) passed.

Try *breaking* your implementation and running RulesOf6005Test again.

You should see a red bar in the JUnit view, and if you click on testMayUseCodeIn- Assignment , you will see a *stack trace* in the bottom box, which provides a brief explanation of what went wrong. Double-clicking on a line in the failure stack trace will bring up the code for that frame in the trace. This is most useful for lines that correspond to your code; this stack trace will also contain lines for Java libraries or JUnit itself.

fix your implementation so it’s correct again. Make sure the tests pass.

Passing the JUnit tests we provide does **not** necessarily mean that your code is perfect. You need to review the function specifications carefully, and **always write your own JUnit tests** to verify your code.

**Tasks Steps:**  
  
The lab 3 tasks are as follows.

1. Setup your eclipse and install git. Show proof of git install by adding a screenshot of git in cmd.

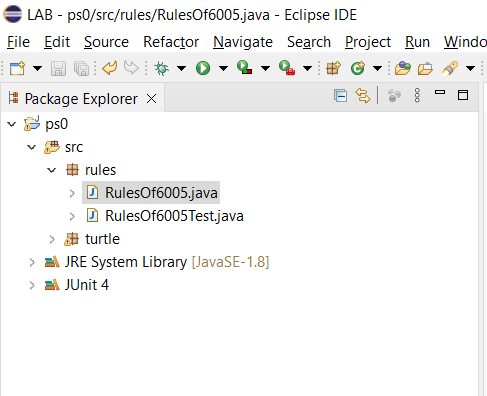
A black background with orange lights

Description automatically generated

2. Download the code from the link given in lab manual: <https://ocw.mit.edu/ans7870/6/6.005/s16/psets/ps0/>

3. Clone the project and show screenshot.

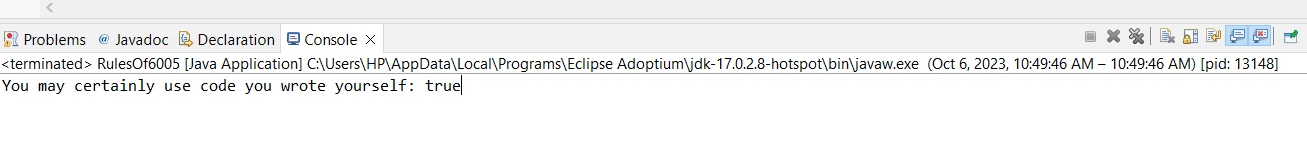
4. Open project on eclipse.



5. Implement the method mayUseCodeInAssignment according to the rules given in: <https://ocw.mit.edu/ans7870/6/6.005/s16/general/collaboration.html>

|  |
| --- |
| /\*\*  \* Judge whether a given piece of code may be used in an assignment (problem  \* set or team project) or not, according to the 6.005 collaboration policy.  \*  \* **@param** writtenByYourself true if the code in question was written by  \* yourself or, in the case of a team project, your teammates,  \* otherwise false.  \* **@param** availableToOthers if not writtenByYourself, whether or not the  \* code in question is available to all other students in the class.  \* Otherwise ignored.  \* **@param** writtenAsCourseWork if not writtenByYourself, whether or not the  \* code in question was written specifically as part of a solution to  \* a 6.005 assignment, in the current or past semesters. Otherwise  \* ignored.  \* **@param** citingYourSource if not writtenByYourself, whether or not you  \* properly cite your source. Otherwise ignored.  \* **@param** implementationRequired whether the assignment specifically asks  \* you to implement the feature in question.  \* **@return** Whether or not, based on the information provided in the  \* arguments, you are likely to be allowed to use the code in  \* question in your assignment, according to the 6.005 collaboration  \* policy for the current semester.  \*/  **public** **static** **boolean** mayUseCodeInAssignment(**boolean** writtenByYourself,  **boolean** availableToOthers, **boolean** writtenAsCourseWork,  **boolean** citingYourSource, **boolean** implementationRequired) {  // If you wrote the code yourself, you can use it.  **if** (writtenByYourself) {  **return** **true**;  }  // If the code is not available to others, you can use it.  **if** (!availableToOthers) {  **return** **true**;  }  // If the code was written as part of a 6.005 assignment in the current or past semesters, you can use it.  **if** (writtenAsCourseWork) {  **return** **true**;  }  // If you properly cite your source, you can use it.  **if** (citingYourSource) {  **return** **true**;  }  // If the assignment specifically requires you to implement the feature, you can use it.  **if** (implementationRequired) {  **return** **true**;  }  // If none of the above conditions are met, you should not use the code.  **return** **false**;  } |

6. Run the method to see output. Screenshot output.



7. Run the unit tests given in the RulesOf6005Test. Screenshot output.

A screenshot of a computer error message

Description automatically generated

8. Break the implementation and run the test to fail them. Screenshot output.

I have made the implementation wrong by changing this block of code:

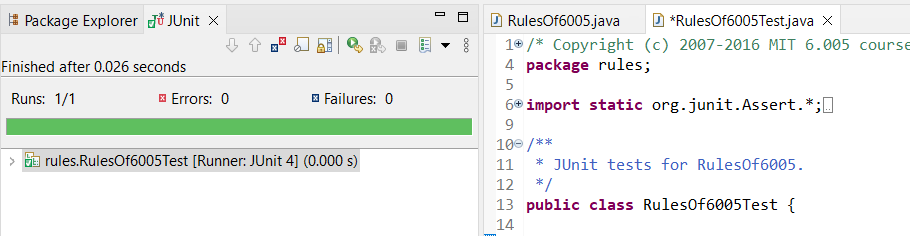
|  |
| --- |
| // If you wrote the code yourself, you can use it.  **if** (writtenByYourself) {  **return** **false**; |

A screenshot of a computer

Description automatically generated

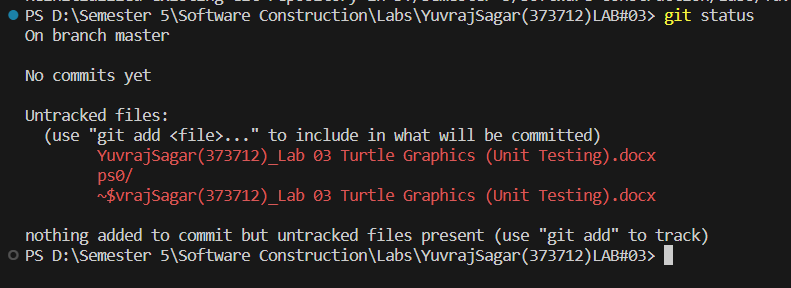
9. Make a min of two more unit tests in RulesOf6005Test file.

|  |
| --- |
| // Test case: You wrote the code yourself and it's not available to others.  *assertTrue*("Expected true: self-written code not available to others",  RulesOf6005.*mayUseCodeInAssignment*(**true**, **false**, **false**, **false**, **false**));    // Test case: The code is publicly available, not written as coursework, but you properly cite your source.  *assertTrue*("Expected true: un-cited publicly-available code with proper citation",  RulesOf6005.*mayUseCodeInAssignment*(**false**, **true**, **false**, **true**, **false**));  } |



10. Also at the end add commits and push the changes into the repository. Screen shot.

**1. Git Status:**

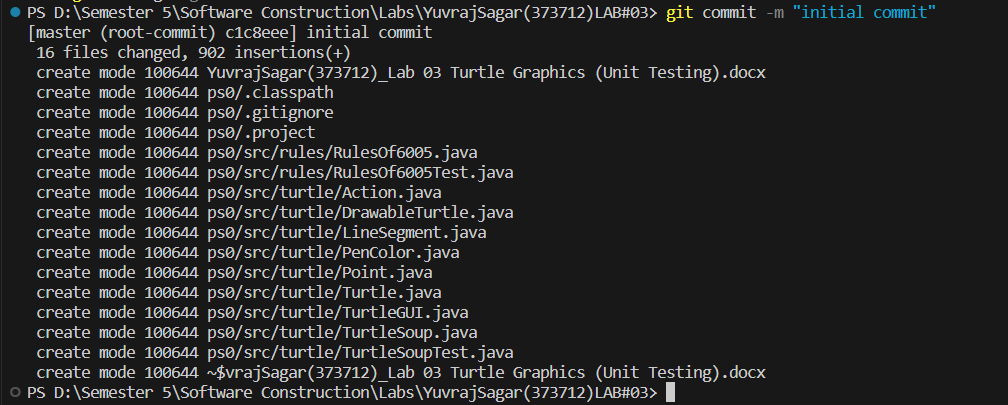


**1. Git Add :**

A screen shot of a computer screen

Description automatically generated

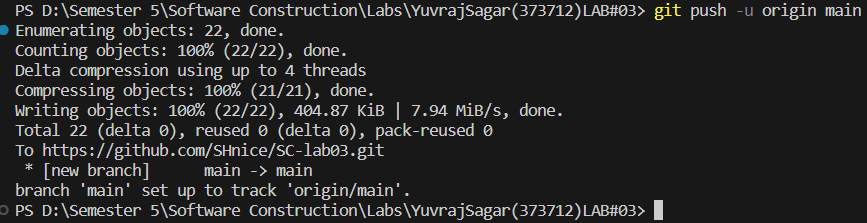
**1. Git Commit:**



**1. Git Add Remote Repository:**

****

**1. Git Push:**

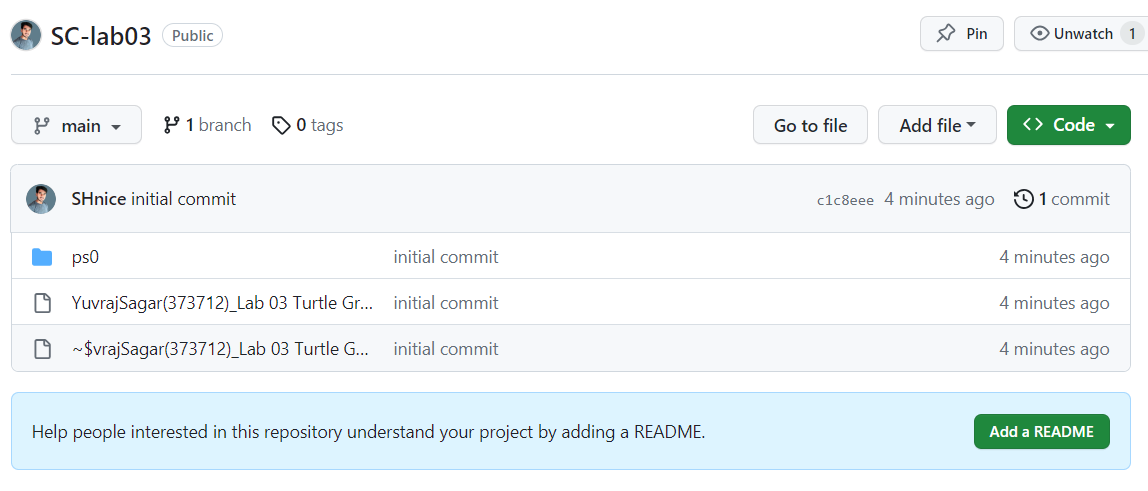


**1. Git Log:**

A computer screen with white and yellow text

Description automatically generated

11. Zip code and upload both report and code.



## Answer:

**Source Code: Zip your source code and upload it as well.**

**Solution**

**Deliverables**

Compile a single word document by filling in the solution part and submit this Word file on LMS. This lab grading policy is as follows: The lab is graded between 0 to 10 marks. Insert the solution/answer in this document. You must show the implementation of the tasks in the designing tool, along with your completed Word document to get your work graded. You must also submit this Word document on the LMS. In case of any problems with submissions on LMS, submit your Lab assignments by emailing it to [aftab.farooq@seecs.edu.pk.](mailto:aftab.farooq@seecs.edu.pk.)