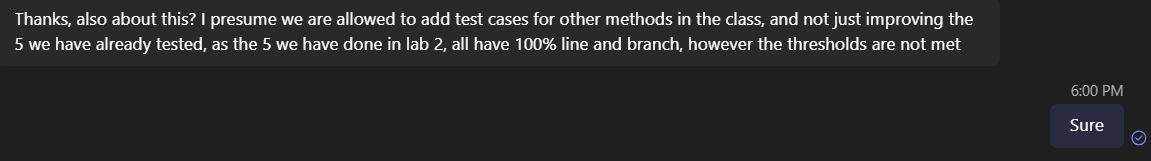
Notes:

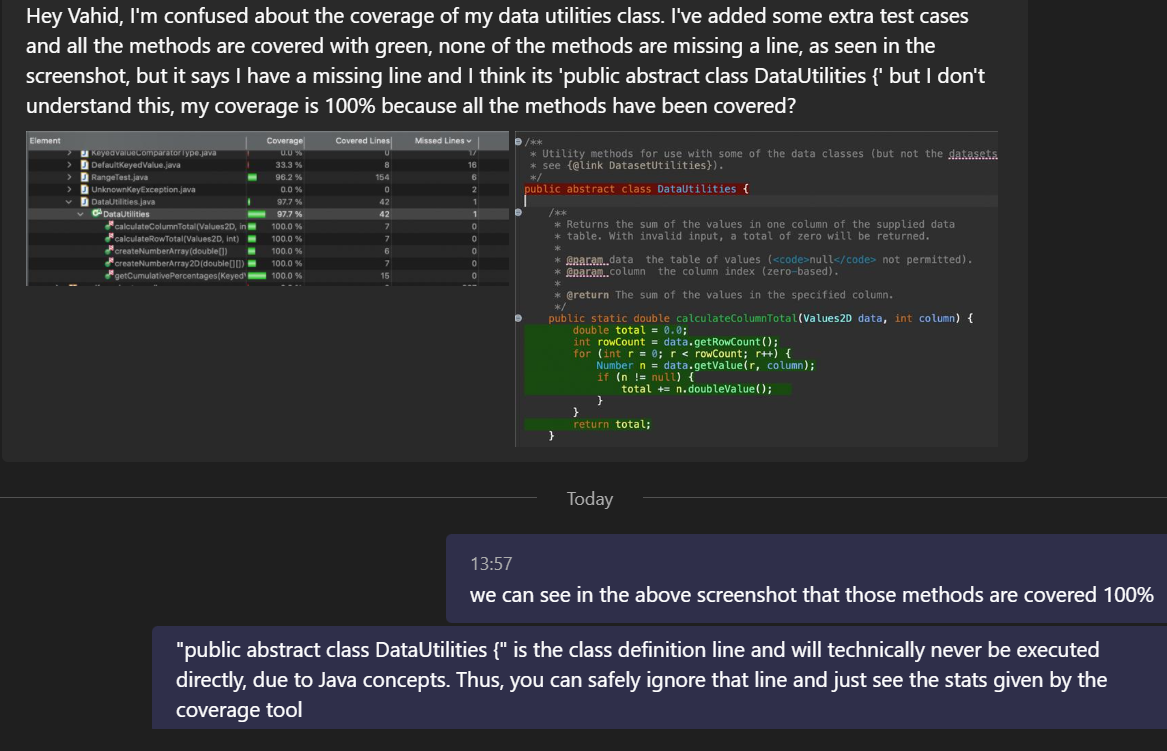
* This is a “LIVE“ document, i.e., more examples and items will be added to this document as we progress in the labs.
* Getting issues/error messages in the project set-up are often due to not following the lab document step-by-step carefully and/or not installing Java/JDL and the Eclipse IDE properly
* Each given issue/error message in the project set-up could be due to MANY possible root causes. We discuss the solution ideas that we are aware of. Students are encouraged to review online resources (such as StackOverflow) also which offer many other solution ideas as well.

# Technical FAQ (about testing)

## Which methods to add test cases for

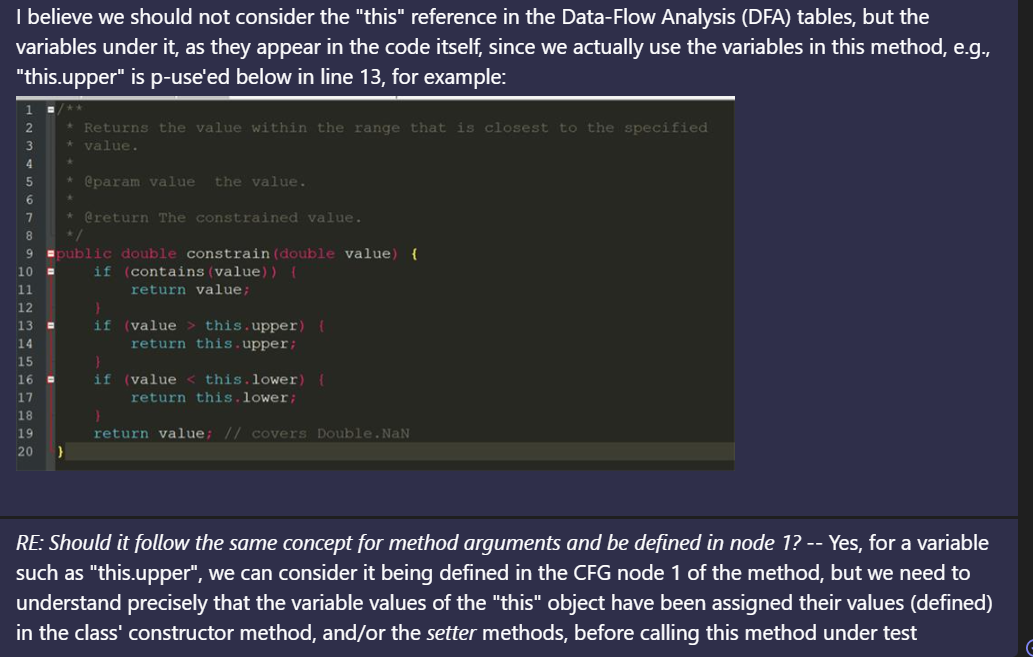


## Coverage of the class definition line in the code



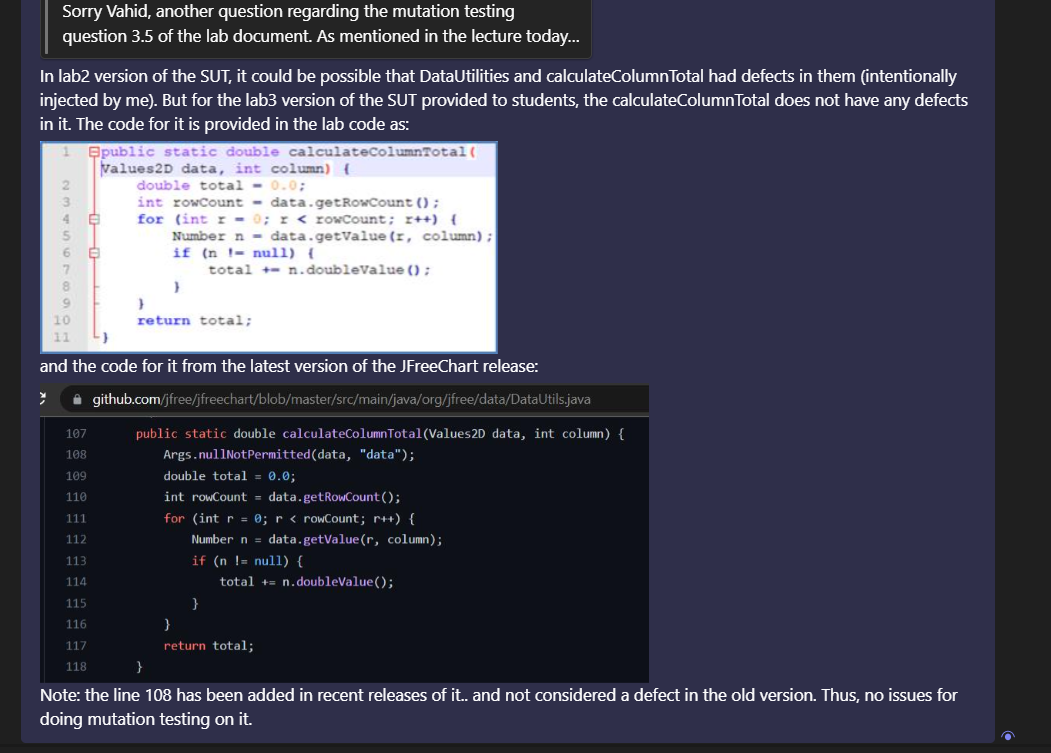
## Dealing with the OO this reference in Data-Flow Analysis (DFA)

*“I am completing the DFA question (3.4) of lab3 and just have a question regarding the self reference(this) of the range class. Do we need to define it in the definition and uses table? Should it follow the same concept for method arguments and be defined in node 1?”*

**

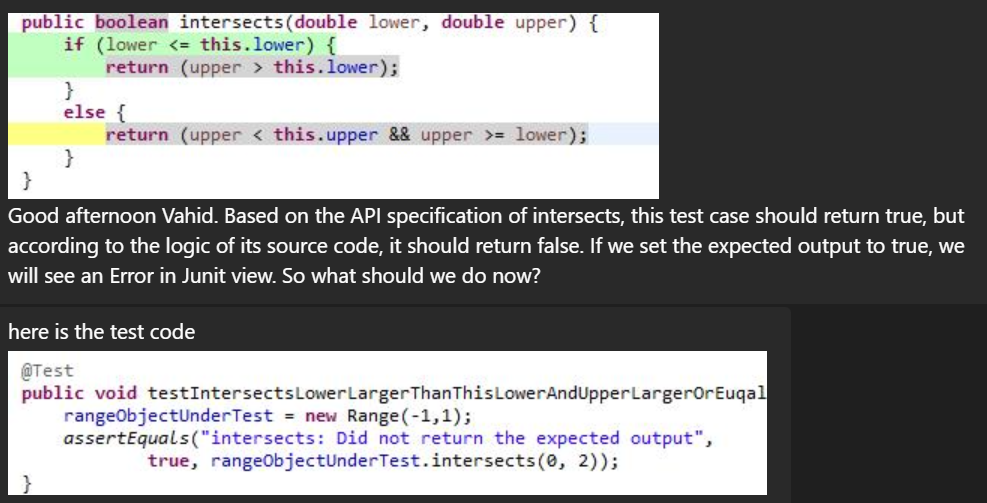
## Are there any defects in the calculateColumnTotal method for the lab3 SUT version? And will it impact the work on mutation testing?

*“.. a question regarding the mutation testing question 3.5 of the lab document. As mentioned in the lecture today, the initial test suite needs to be failure free before Mutation testing can be carried out. However the test suite my group has developed has test failures already and we believe there already exist defects in the calculateColumnTotal method. How should we go about completing this part of the lab to get our initial test suite all passing/ failure free?”*



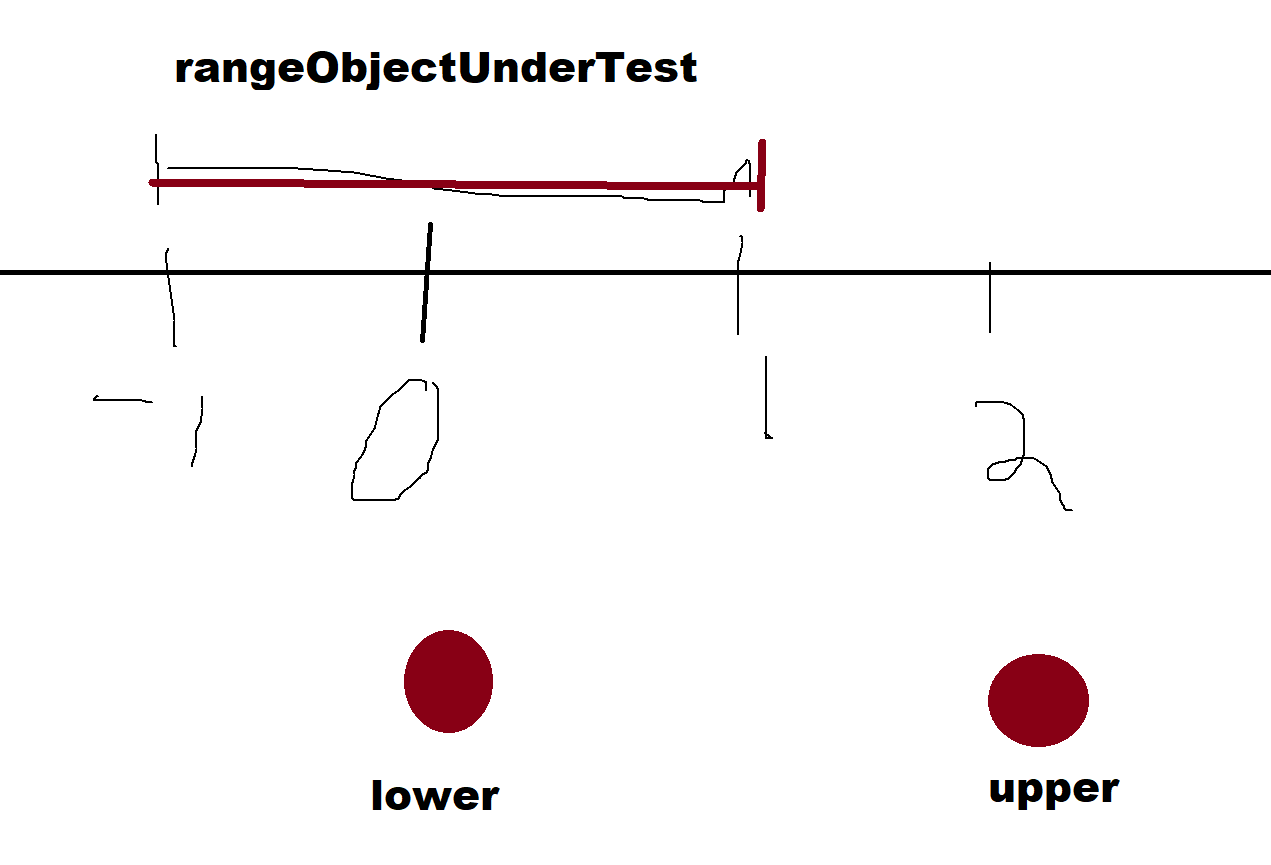
<https://github.com/jfree/jfreechart/blob/master/src/main/java/org/jfree/data/DataUtils.java>

## Is there an intentional defect in one of the methods of the Range class?



**Answer:**

Here is the visualization of your test-case above:



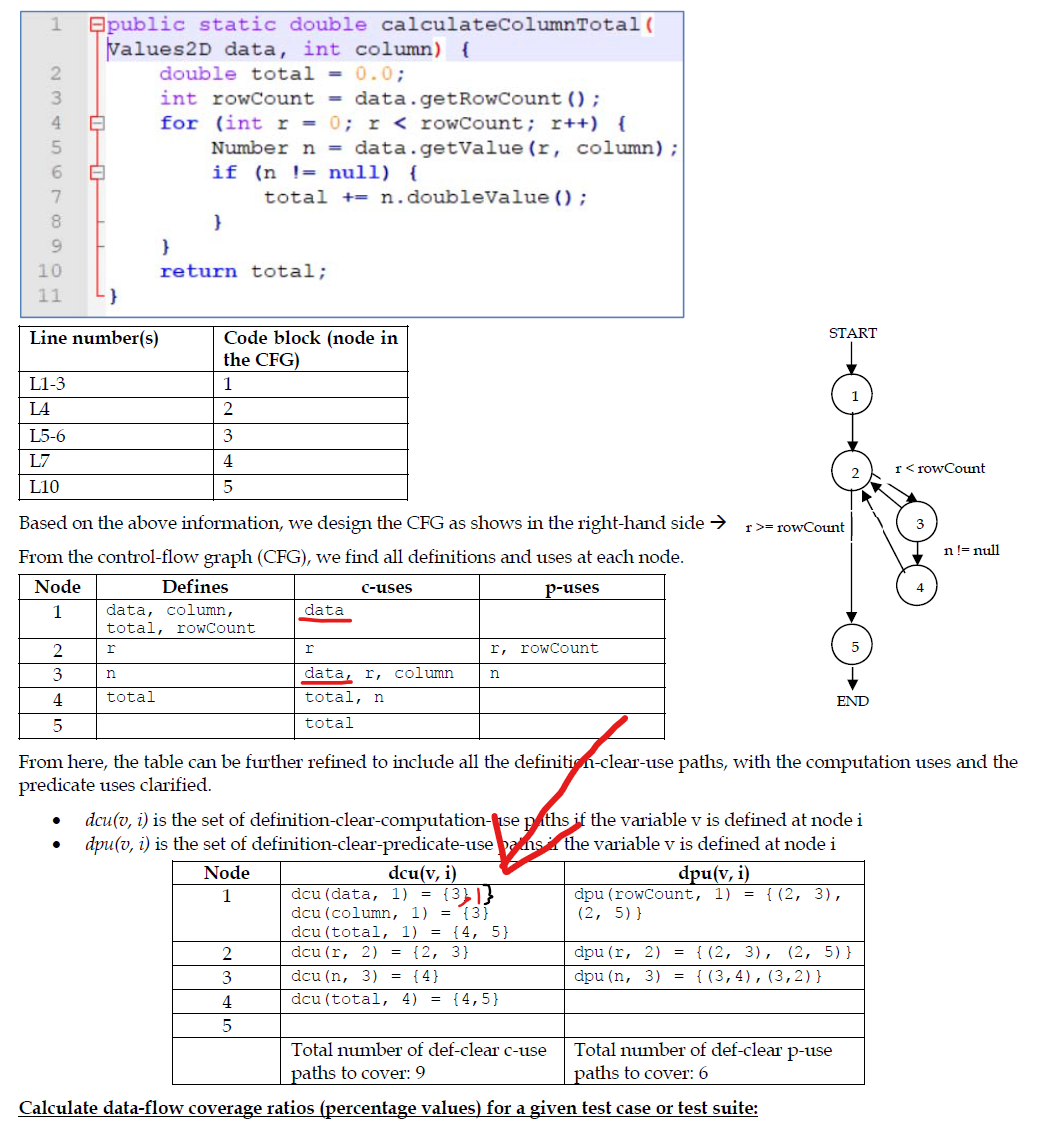
Thus, we should get True, as our visualization above also confirms the API JavaDoc.

In fact, there is an intentionally-injected defect in the method under test above. By inspecting the code manually, you can determine what that defect is 🙂.

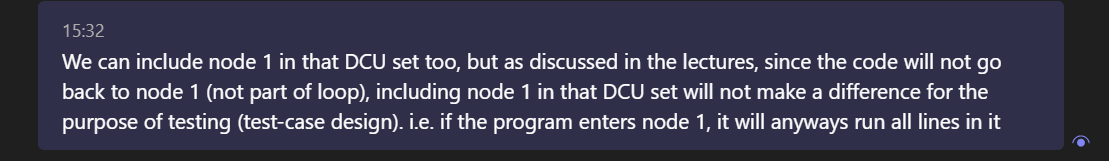
Thus, as per the rules of our module, we should use the JavaDocs as the requirements to determine the expected output/behavior, and let your test-case(s) for that method fail.

## Is there a typo in a DCU set in the Appendix 1?

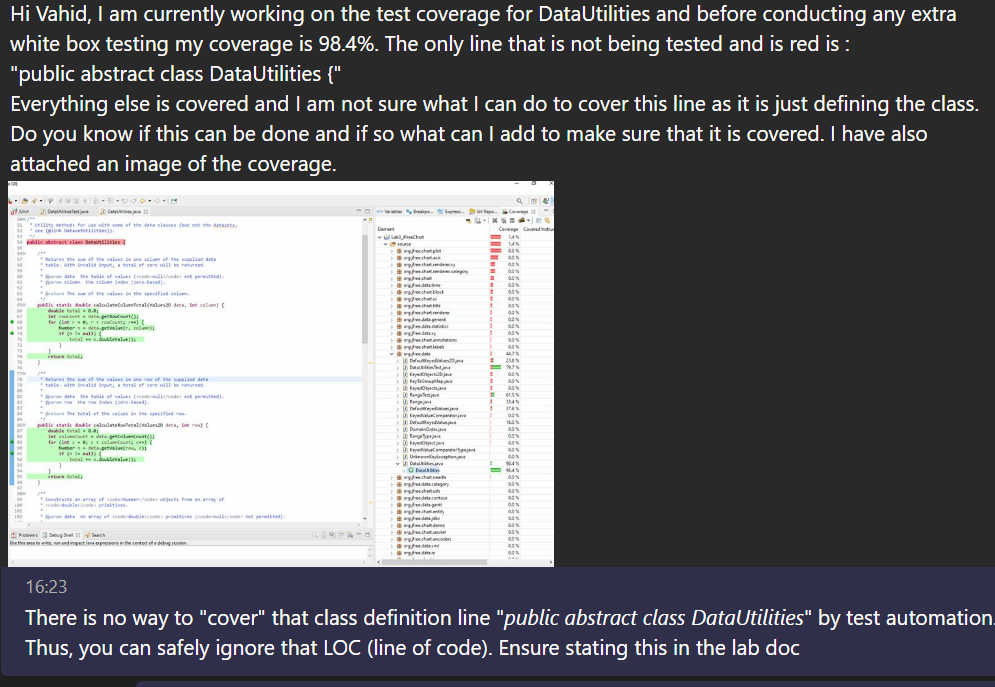
*I came across what seems like a typo in Appendix 1 in the lab docs 3 pdf. It looks like dcu(data,1) = {3} should actually be dcu(data,1) = {3, 1}.*



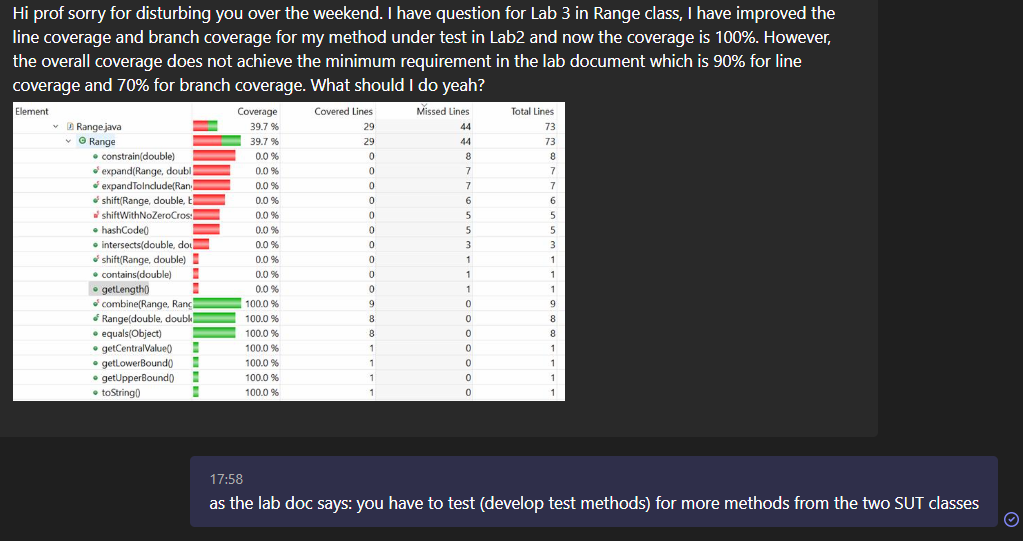
**Answer:**



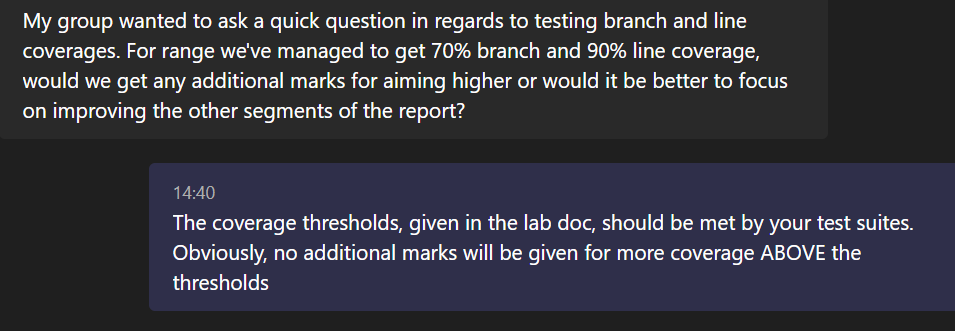
## Covering class definition lines



## If you need to get more coverage value in the SUT class level, you need to develop more test methods for more methods from the two SUT classes

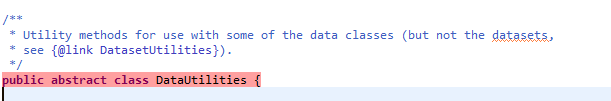


## Will there be “additional” marks for more coverage ABOVE the thresholds?



## No way to cover/execute the class “header” lines

*Hi, having a bit of an issue when trying to cover the following for the line coverage for DataUtilities not sure how to cover this line shown in the image:*



**Answer:**

There is no way to cover / execute the class “header” lines. Thus, you can exclude that in your coverage assessment.

# Non-technical FAQ (Project set-up, IDE, etc.)

## ***I was wondering if we could use IntelliJ for the lab assignment?”***

The labs have been designed using the Eclipse IDE. If a student wants to use another IDE, it is up to them to do the project set up, and port the settings and steps, etc. Unfortunately, the teaching team will be unable to provide support for other IDEs.

## Getting the error message "Could not find or load main class” in Java / Eclipse

Your project set up in the IDE has issues, or you need to use Eclipse in the right way. ... See:

<https://www.google.com/search?q=eclipse+java+could+not+find+or+load+main+class>

## If the code-coverage plugin cannot be found in your Eclipse (is not installed)

Answer:

**All the lab machines are configured properly to run the labs, and you can use them.**

The plugin can be easily installed from its update site at <http://update.eclemma.org> on any Eclipse installation of version 3.5 or above. You need to [follow the steps for installing new plugin’s in Eclipse](https://docs.oracle.com/javame/dev-tools/jme-sdk-3.3/ecl/html/setup_eclipseenv.htm#:~:text=2.1.&text=In%20Eclipse%2C%20go%20to%20Help%20%3E%20Install%20New%20Software).

It is also available from the [Eclipse Marketplace](https://marketplace.eclipse.org/content/eclemma-java-code-coverage).

Further details: <https://www.eclipse.org/community/eclipse_newsletter/2015/august/article1.php#:~:text=The%20plugin%20can%20be%20easily,new%20Coverage%20As%20launch%20configuration>.

## If you are getting Java compile errors in other classes of the JFreeChart code-base (outside the org.jfree.data package)

Answer:

**All the lab machines are configured properly to run the labs, and you can use them.**

Like the following red error icons in the package icon in the project explorer view:

Those Java compile errors should not actually occur. But apparently, the setup of some groups are somewhat different than the lab’s suggested setup.

You can safely ignore those compile errors since apparently those are in some other classes / packages of the JFreeChart code-base, outside the org.jfree.data package.

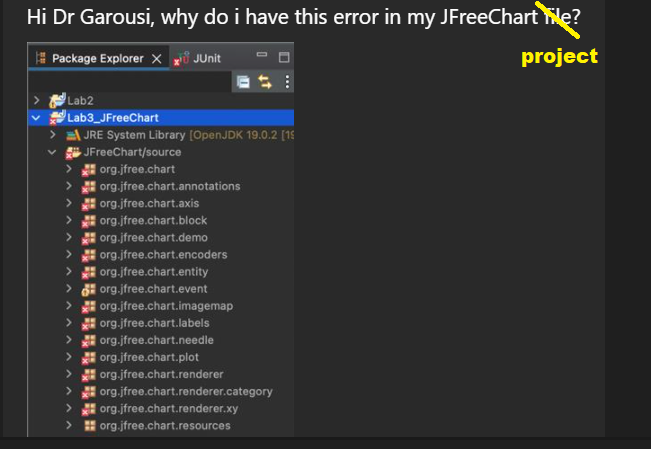
You can safely command the IDE to ignore those compile errors and run the JUnit with coverage feature, as instructed in the lab doc.

By ignore, we mean just develop your JUnit test-code for the org.jfree.data package, and run the test-code in coverage mode.

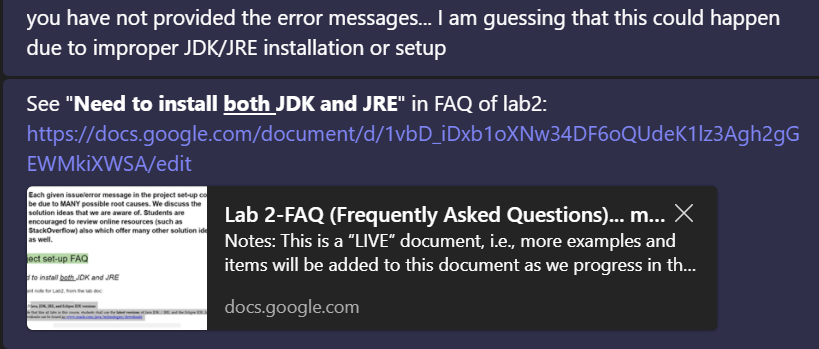
## If you are getting Java compile errors in most of your project packages / Java files, like the following:

Answer:

**All the lab machines are configured properly to run the labs, and you can use them.**



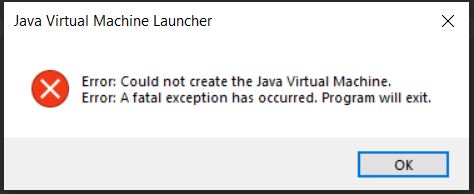
**RE:**



[**https://docs.google.com/document/d/1vbD\_iDxb1oXNw34DF6oQUdeK1lz3Agh2gGEWMkiXWSA/edit?usp=sharing**](https://docs.google.com/document/d/1vbD_iDxb1oXNw34DF6oQUdeK1lz3Agh2gGEWMkiXWSA/edit?usp=sharing)

## JVM "fatal error"

*We are getting JVM "fatal error". A screenshot is attached below.*

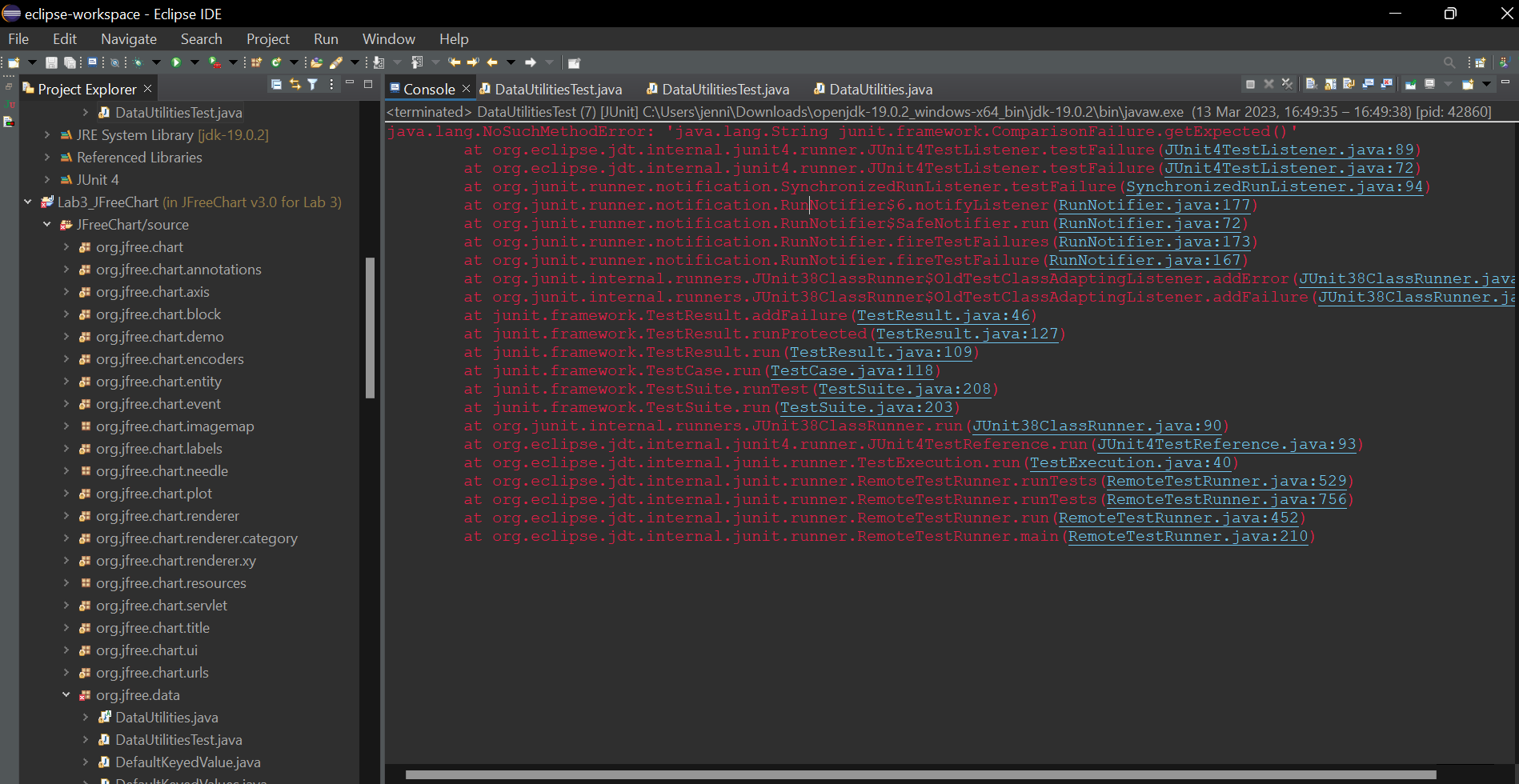


**Answer:**

That issue seems to be related to the settings of your PC/laptop and the JVM / JRE / JDK on it. All the lab machines are configured properly to run the labs, and you can use them. If you want to fix the issue on your own machine, you should get help from the online forums, as many people have offered solutions online:<https://www.google.com/search?q=could+not+create+the+java+virtual+machine>

## Getting JUnit java.lang.NoSuchMethodError on junit.framework.ComparisonFailure

I seem to be getting the following error when running my code



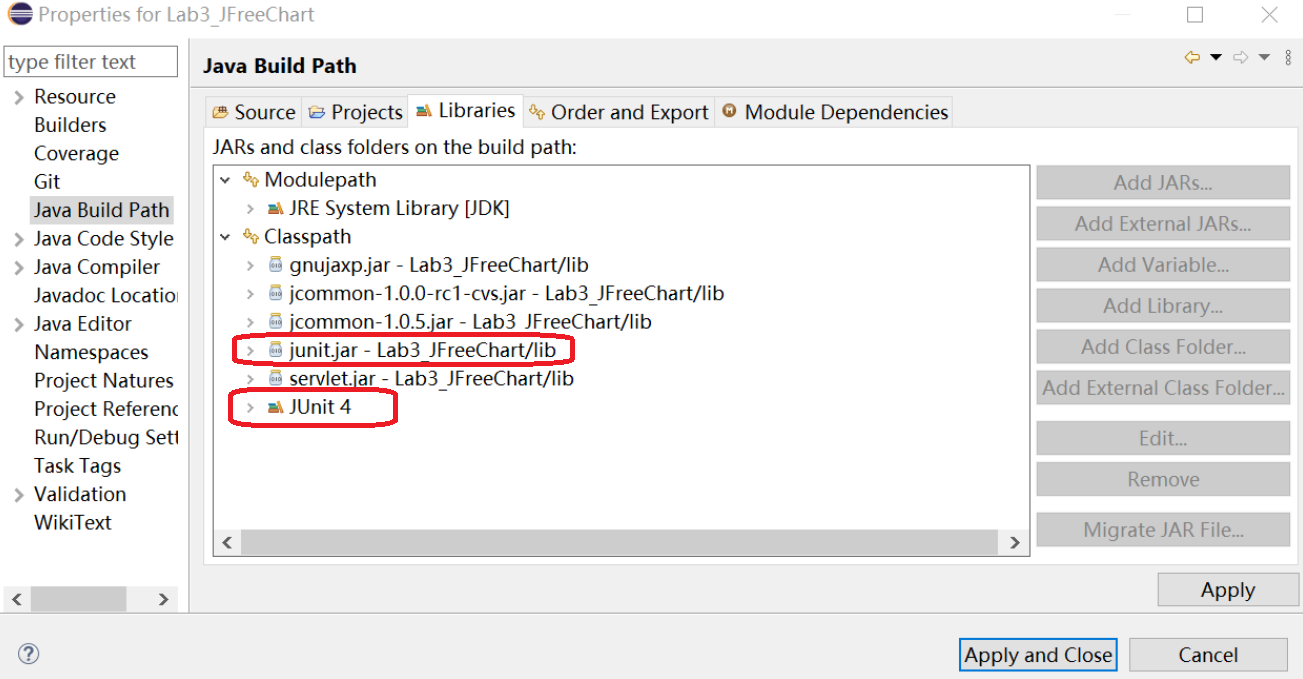
**Answer:**

This is most probably you are, mistakenly, using JUnit 3 and not version 4 as the labs require. See the lab2 docs to use JUnit4. Also check your import statements in top of Java files to ensure you are importing the JUnit 4 files (you need to distinguish between JUnit 3 and 4 imports). These are discussed in:

* <https://stackoverflow.com/questions/4019016/junit-java-lang-nosuchmethoderror-junit-framework-comparisonfailure-getexpected>
* <https://www.google.com/search?q=nosuchmethod+junit+comparisonfailure>

## My test-code is mixing up JUnit versions

My test-code is mixing up JUnit versions. Here is my project settings:



**Answer:**

There is a clear mistake above, as **both** JUnit4 and also junit.jar (that is for JUnit 3) have been included. You cannot include both JUnit versions!! And thus you should NOT include the junit.jar file, but only include JUnit4 in your project. You should just delete junit.jar from the list above.