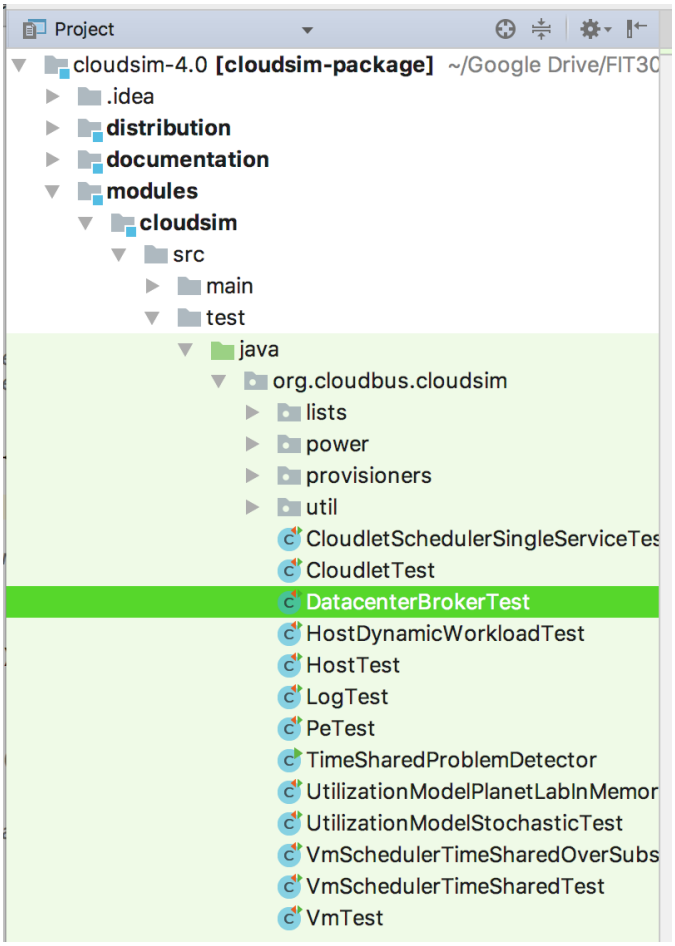
1.0 Introduction

1.1 Overall approaches to test the project

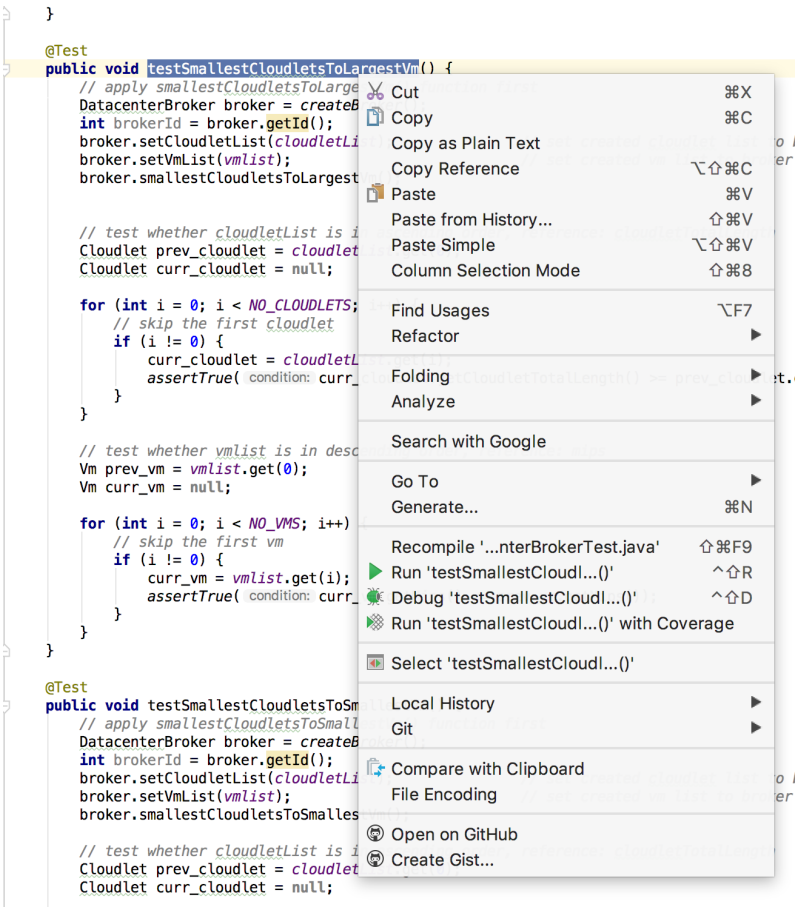
Step 1

Locate the source code test class “DatacenterBrokerTest” as shown below.



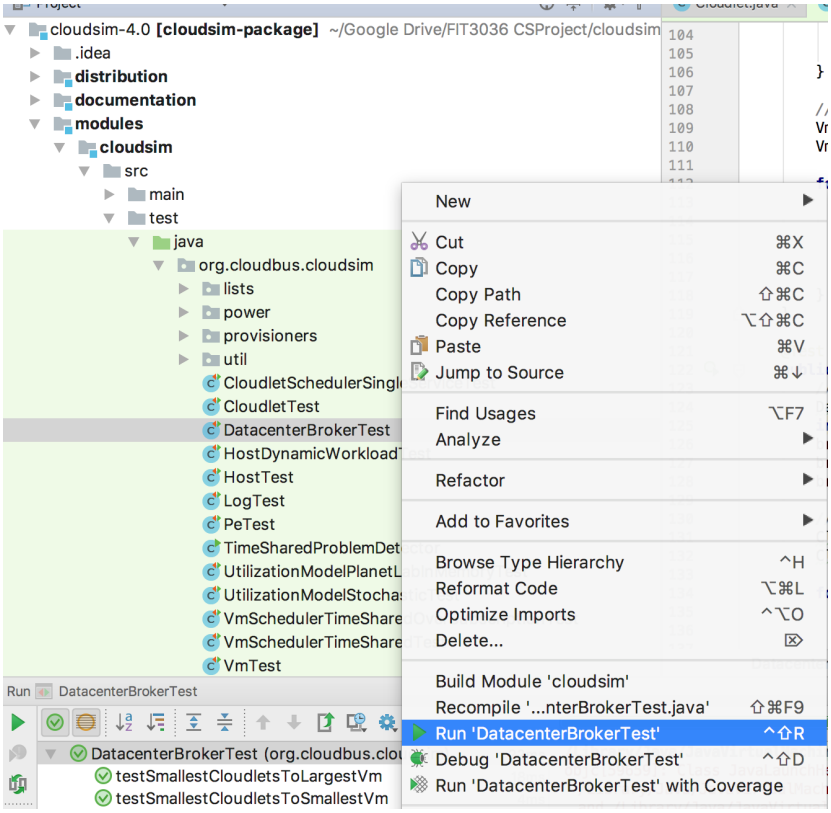
Step 2

To run unit test, go to function “testSmallestCloudletsToLargestVm()”, right click on the function name and select “Run testSmallestCloudletsToLargestVm()” as depicted below. This step applied to all individual functions to be tested. And be note that there are a total of two unit tests.



Step 3

To test the entire test suite, click the test suite file name “Run DatacenterBrokerTest”. A successful execution will be colored green, and a failure to execute will be colored red.



1.2 Test targets

- Percentage of successful tests
  Function “testSmallestCloudletsToLargestVm()” and function “testSmallestCloudletsToSmallestVm()”. A 100% test success rate is required for class “DatacenterBroker” to be considered passing test cases.

- Code coverage
  The entire test suite does not have an explicit aim for code coverage. Or in other words, the test suite does not need to have code coverage. Generally speaking, code coverage represents the percentage of source code of a program being executed when a test suite runs successfully. In my test suite, the unit tests are designed for two sorting functions. As long as the sorting does its job correctly in one place - sort the list in ascending or descending order, it would be able to execute successfully elsewhere. Because what sorting does is to change the order of the elements in the list. No matter what order the elements in the list, the list is still of original data type. It has no effect on the program logic.

  Also, it is important to note that code coverage is easy to achieve with low-quality testing. What matters in testing is thoughtfulness of designing the test cases.

- Testing phase completion criteria

As long as the sorting functions pass the basic test suite, the testing phase is completed. In previous project specification, it is stated that execution time needs to be tested thoroughly in order to make sure there is indeed a performance improvement of the proposed algorithm in comparison to the random resource allocation algorithm. This performance measurement is represented in the Result and Discussion & Analysis sections of final report. We observed that there is an average 90% execution time improvement.

2.0 Test Report

Unit testing and performance testing

Test method

- The test method is straightforward. For both testing functions, we only need to check whether the current inspecting element is greater than or equal to the previous inspected element in the case of sorting in ascending order, and that whether the current inspecting element is smaller than or equal to the previous inspected element in the case of sorting in descending order.
- The performance testing has been written in the final report Result and Discussion & Analysis sections. We observed an average 90% execution time improvement.

Test output

- For function tests, the expected output is every currently inspecting element be greater than or equal to the previous inspected element in the case of sorting in ascending order, and every currently inspecting element be smaller than or equal to the previous inspected element in the case of sorting in descending order. The actual test output is all test cases passed and testing process finished with exit code 0. The expected output matches the actual output.
- For performance test, the expected output is overall execution time improvement. The actual output is an average 90% execution time improvement. The expected output matches the actual output.

Test evaluation

- In test suite, two sorting functions complete the job of sorting tasks, resources in their corresponding sizes. This provides the basis of experimenting on tasks and resources allocation of different sizes.
- In performance test, two proposed algorithms 1) assign the smallest cloudlet to the largest vm, and 2) assign the smallest cloudlet to the smallest vm improves the cloudlets execution time successfully. This answers my problem statement, “how to schedule loT health care request with higher efficiency”.

3.0 Conclusion

In this test report, I discussed how to unit test the program. For performance testing the program, refer to final report Result and Discussion & Analysis sections. I also discussed the test targets - what is considered successful experiment tests, test method - the rationales behind the testing and why code coverage is not applicable in my testing cases, test output - how actual output matches expected output, and test evaluation - whether the test and test results echo my project objectives.