CS209A Tutorial Week 12

In this tutorial we implement test cases for the following reference code provided for you to use: `ClientStudent.java` and `ServerStudent.java`. The class ServerStudent will load data which is stored in `StudentsGrade.csv`, it then, as a server, waits to recieve a query command (which is generated by the client class ClientStudent).

There are five methods (apart from main) in the reference code `ServerStudent.java`:

(1) `public void readFile()`
(2) `public String handleNameCommand(String s)`
(3) `public String handleGradeCommand(String s)`
(4) `public String handleTopCommand(String s)`
(5) `public String handleCommand(String s)`

To complete this tutorial, you will create a testing class for the server named: `ServerStudentTest.java`. Then write test cases that are to test the server class. You should include at least the following tests that are for the functionality of each method.

(1) `public void testReadFile()`
a. check the size of `gradeMap`
b. check the size of `orderedList`
c. check the content of `gradeMap`
d. check the content of `orderedList`
e. check if the `orderedList` is correctly sorted.

(2) `public void testHandleNameCommand()`
a. input a name in the `StudentsGrade.csv`
b. input a name not in the `StudentsGrade.csv`
c. check each name in the `StudentsGrade.csv`

(3) `public void testHandleGradeCommand()`
a. input a string can present two integers (a = b, a < b, a > b, b>100 or a < 0 etc.), and check the result
b. input a string cannot present two integers, and check the result

(4) `public void testHandleTopCommand()`
a. input a string can present an int($n \leqslant 0, n > 100$), and check the result
b. input a string cannot present an int, and check the result

(5) `public void testHandleCommand()`
a. check the command “NAME”, no argument, one argument, or more arguments.
b. check the command “GRADE”, no argument, one argument, or more arguments.
c. check the command “TOP”, no argument, one argument, or more arguments.
d. input some undefined commands.

(6) (**Optional**)Integration test (“end to end”). This test is to check whether the different parts of the server are able to work together properly. It should test receiving a message from the

client and then generate a response that is in the correct format. In the test you should use the idea of “mocking” so that it is not necessary to actually have a real connection between the client and server, instead you will make a fake client message and call the various classes in the server.

You can also create more test cases according to the week 9 tutorial.

You can use `@FixMethodOrder` to fix the method running order from method (1) to method (6).