This assignment is to perform full unit test of the Model classes (classes that represent the table attributes in the database) AND the corresponding error message classes. **Example Reservation.java and ReservationErrMsgs.java**

1. **JUnit**
   1. Use Eclipse to perform the JUnit testing of these classes. Create a JUnit test case file in Eclipse for EACH class under test
   2. Develop your test cases in Excel. Create a separate test case file in Excel for EACH class.
   3. Use **parameterized tests** from files (M06 - JUnitParamsRunner) to read the csv version of the test case tables.
   4. You do NOT need to deliver CFGs or use Basis Path to develop test cases BUT you MUST
      1. have full JaCoCo coverage.
      2. use MCDC tests for logical expressions.
      3. test all BVs as we did in 5321
      4. have extreme range values
   5. How to test these units (classes).
      1. **DAOs** - you have two choices:
         1. You may call these with the actual methods (more of an integration level test) and NOT mock out the calls with EasyMock **(this is the preferred approach but see next**).
         2. You may mock out the calls with EasyMock if you so desire. If you are testing things that you cannot put in the database you **MUST** use this approach. Example - only one kind of System User can exist at a time.
      2. **ErrorMsgs** - Test these as a single JUnit test class with its twin - in otherwords test Reservation.java and ReservationErrMsgs**.**java together.
2. Run **JaCoCo** - see **Steps for creating JaCoCo report.docx**
3. Run **PitClipse** (search in Eclipse Market place for this plugin) against EACH JUnit test. Change mutators to all: Window -> Preferences -> Pitest -> Mutators -> All Mutators.
   1. Run PitClipse against each JUnit test case. Resolve any issues.
   2. Once you get all issues resolved or explained then run PIT against all tests - you will turn in the directory containing all the PIT results. See **Steps for submitting PIT reports.docx**

Submission requirements (submit this as a SINGLE zip file named TeamX\_A03\_Deliverables.zip)

1. Include a folder containing your project (call this MyApp.zip) - **this must be all the files in your application.**
2. Include a folder containing JaCoCo results
3. Include a folder containing PIT Results
4. Include a folder containing Excel test files
5. Include a single Word document titled TeamX\_TestReport.docx (X is your team number)
   1. Create a Test Links section that provides the names of the following zip files
      1. Java source code
      2. Test Case Tables (xls) format
      3. JaCoCo results
      4. PIT Results
      5. Excel test files
   2. For Each JUnit test case list the following:
      1. Section X - *Class Name* (X is an integer value incremented with each class and Class Name is the actual class name)
      2. Section X.1 Test Case Table - reference the test case table file name (xls version)
      3. Section X.2 JaCoCo Coverage Summary (summarize the results here). For any non-coverage items you must explain why you don't have all green source lines or fix it.
      4. Section X.3 PIT test results (summarize the PIT issues here). For any non-coverage items you must explain why you don't have all green source lines or fix it.
6. Include the **Usage check validation mapping.xlsx** file.
7. Include A03 Test Results Summary Table (Excel spreadsheet)
8. Include two Powerpoint charts
   1. Team Number and team members
   2. Task assigned (who did what)

Presentation requirements

1. Each team will have 10 minutes to present the details of their test. The discussion will focus on giving the details of the test report. Each team member must present.