Assignment-2: Due date is **Monday**, **May 1st** at 11:59pm (**No late submissions will be accepted**)

The goals of this assignment are: (a) to learn how to **manually** create a test suite; and (b) to learn how to use a **Cobertura** code coverage tool in order to assess the comprehensiveness of a test suite.

ASSIGNMENT-2 DETAILS:

Part 1) Preliminaries and Warm up

Get the latest files from the class repository by running

```
git remote add upstream "https://github.com/aburasa/CS362-001-SP17"
git checkout master            # make sure we're on the master branch
git fetch upstream             # pull any information about changes in upstream
git merge upstream/master -m "Sync" # merge new files
```

You should now have new files in your projects/aburasa/assignment-2/

Copy assignment-2 folder into your projects/youonid/assignment-2/

The directory "projects/assignment-2/Calendar/src/main/java/edu/osu/cs362/" contains 4 different classes for which you are to develop unit tests. We will be using a tool called JUnit which is a unit test framework for Java. The file projects/assignment-2/Calendar/src/test/java/edu/osu/cs362/ApptTest.java is provided to you with an example unit test that tests the Appt.java class.

First cd into the projects/youronid/assignment-2/Calendar directory and make sure that you can compile the program using mvn compile.

Part 2) The Calendar Application

You just joined a team that is working on a Calendar application. Your first job is to become a "subject expert" in the Calendar application. Note that the primary source of information about the Calendar implementation itself is the **Appt.java**, **CalDay.java** and **TimeTable.java** classes provided in the class repository. You can use **CalendarMain.java**, which contains the main program, to run some data inputs and see how the Calendar application works. This is a typical testing experience, where you are not given a complete specification, but must discover one for yourself. To run and execute the CalindarMain file use java -cp ./target/classes/ edu.osu.cs362.CalendarMain

Unfortunately, it looks like the current code base has very small tests. Instinctively, you know that these tests are probably not sufficent, but you decide to write one or more test cases and see how the code should behave.

Your task is simply to test the **Appt.java**, **CalDay.java** and **TimeTable.java**. Write unit tests for the three classes excluding **CalendarMain** (i.e., do not use CalendarMain at all) following your own intuition on what a good test suite might look like. Check these tests in as **ApptTest.java**, **CalDayTest.java**, and **TimeTableTest.java.**

Part 3) Code Coverage using *Cobertura tool*

Use *Cobertura* tool to measure code coverage for all of the test cases using `mvn cobertura:cobertura`. This should generate a set of files in the **/target/site/cobertura** folder including an index.html file. Open this file with a web browser. Now measure the coverage of your code. If you have less than **80%** line coverage and Branch Coverage for each class, create more unit tests (i.e., test cases) until you achieve it (if you can!).

Take a screenshot of your coverage and submit it to Canvas ScreenShotCoverage.pdf (**10 points**).

Part 4) Write up your report that conatins the following sections: (60 points)

- **Program Description** (**15 points**)
  - Provide details of the Calendar application?, what does each class do (provide the number of public methods, private methods, source lines of code(SLOC)) ? (**15 points**)
  - **Optional**: you can use JavaNCSS tool for counting source lines of the code, methods, or classes.
- **The Utility of the code coverage tool** (**35 points**)
  - Commentary on how well the test cases cover the source code of the Calendar application. (**10 points**)
  - How many test cases did you generate?. Point out some code, **if any,** that your test cases do not cover. I want you to look at the Calendar code coverage and find out what parts of your code are not covered so that in future you can improve your test suite. (**15 points**)
  - From a testing perspective, what is the advantage of using a code coverage tool like *Cobertura*?. how it helped/didn't help, etc. (**10 points**)
- **Unit Testing Efforts (10 points)**
  - Discuss your unit testing efforts and your experince testing the Calendar application. (**5 points**)
  - Document the status and your view of the reliability of the Calendar application. (**5 points**)

SUBMISSION INSTRUCTIONS:

- **Canvas** – **Assignment-2.pdf** that contains the sections and the answers to the above questions, and **ScreenShotCoverage.pdf** (**70 points**)
- **The class github repository (**under **projects/your-onid/assignment-2/Calendar/) (30 points)**
  - Submit your complete maven project of assignment-2 to your onid-folder on your class github. The common maven project should contain:
    - src/main/java package (Calnedar java programs)
    - src/test/java package (The test cases that you generated)
    - pom.xml

  Note: dont submit the target folder.

  - The code must run and compiled.
  - Create a new branch of your repository called "youronid-assignment-2" contains your final submission. This branch must be created before the due date to receive credit.

** Add a comment in Canvas and give the **URL** for your fork (under Assignment-2).
10 points off for not submitting the URL