**Assignment 2**

**Project Description**

**Graduation Timeline for Students**

* The graduation requirements for a new degree in Computer Science are the following:
  + Group-1: Long Programming and Design: 2 courses from A-D
  + Group-2: Data Structures and Algorithms: 2 courses from E-H
  + Group-3: Hardware Sequence: 2 courses from I-L
  + Group-4: Data Analytics: 2 courses from M-P
  + Group-5: Electives: 2 courses from Q-Z
* Each student can be in any one of the following states, one for each group listed above.
  + StateOne - Student has taken most courses in Group-1
  + StateTwo - Student has taken most courses in Group-2.
  + StateThree - Student has taken most courses in Group-3.
  + StateFour - Student has taken most courses in Group-4.
  + StateFive - Student has taken most courses in Group-5.
* Each student starts in StateOne.
* A change of state occurs only when a student's number of courses in another group is more than what (s)he has in the current group.
* A student should take 3 courses per semester.
* Pre-requisites in each group: In any group, except Group-5, a student can only take courses in the Alphabetic order. In Group-5, the order does not matter. For example,
  + a student **cannot** take course F before taking course E as they are in the same group.
  + a student **can** take course X before course W as course order in Group-5 does not matter.
* Use State pattern to capture the degree requirements and implement your software as discussed in class.
* Use the below rules and guidelines while working on the assignment.
  + If a course cannot be allowed due to pre-requisites, then save it in a wait-list for the student. You can design your own scheme/algorithm for when/how courses are processed from the wait-list, and describe your scheme/algorithm in your README.md.
  + A course and its pre-requisite/s cannot be assigned to a student in the same semester. So, a course, if put into the wait-list, cannot be removed until at least the current semester is over.
  + To be eligible for graduation,
    1. the student needs to complete 10 courses.
    2. the student needs to take 2 courses per group.
  + A student may want to take several courses from a Group, even though just 2 are required for graduation.
  + Once a student is eligible for graduation, stop processing more courses for the student.
  + It is OK if the student takes less than 3 courses in the final semester. Note that a student may take more than 4 semesters to graduate.
  + Depending on the input, it is possible that a student does not graduate.
  + Courses are processed one at a time in the given order. Note, however, that the courses in the wait-list (if any) can be processed in any order by using a scheme/algorithm of your choice (as mentioned above).
* From the command line accept the following args: ***input.txt output.txt***
* Your driver code should do the following:
  + Read command line arguments.
  + Create an instance of the Context Class and call a method in it to determine the student's outcome, along with the input file handle.
  + Call a method in Results, via the method in FileDisplayInterface, to write the data stored in Results to output.

**INPUT FORMAT**

The program should accept a single input file. The input file will contain just one line corresponding to the course sequence for a student. Below is the format of the input file.

<studentID>: <course> <course> <course> ... <course>

* The input will be well formed. The courses will be white space delimited.
* The input for each student is their preferred sequence of courses.
* It is possible that the input is an empty file or not available. If the input does not exist, or is empty, then just print a meaningful error message to stderr, print the stack trace, and exit.

An example input is shown below.

1234: A B C D E F G H I J K L M N O P Q R S T U V W

**OUTPUT FORMAT**

The program should accept the name of an output file to store the results. The format in which results need to be persisted to the output file is shown below.

<studentID> <course completed> <course completed> ... <course completed> -- <# semesters> <# state changes>

An example of a possible output for the input given above is shown below.

1234: A E I B F J C G K D H L M Q R N -- 6 0

* Use the Results class to store and print (when the appropriate methods are called) the sequence of courses completed for the given input for a student.
* If a student does not graduate, write a line at the end of the output file indicating this. In this case, you should set the *# semesters* to 0.

**Code Organization**

* Your directory structure should be the following ( note the change in package names ):
* -firstName\_lastName\_assign2
* ---studentCoursePlanner
* ----- README.md
* ----- src
* ----- build.xml
* ---courseplanner
* ----------driver
* -----------------Driver.java
* ----------util
* -----------------Results.java
* -----------------FileProcessor.java
* -----------------FileDisplayInterface.java
* -----------------StdoutDisplayInterface.java
* ----------state
* -----------------CoursePlannerStateI.java
* ----------other packages and classes that you need