KINGDOM OF SAUDI ARABIA
Ministry of Education
UNIVERSITY OF JEDDAH
Main Campus- Girl's
COLLEGE OF COMPUTER SCIENCE & ENGINEERING
COMPUTER SCIENCE & AI DEPARTMENT

جامعة جدة
University of Jeddah

المملكة العربية السعودية
وزارة التعليم
جــامـعة جدة
شطر الطالبات
كلية علوم وهندسة الحاسب
قسم علوم الحاسب والذكاء الاصطناعي

CCCS 121, 1st Term 2020
Assigned: Monday, Oct 07, 2019
Phase1: Monday, Oct 21, 2019
Phase2: Thursday, Nov 28, 2019

Purpose:

The purpose of this project is to:

1. Introduce Object Oriented Style of programming.
2. Practice your knowledge of design classes in Java.
3. Practice the implementation of various kinds of classes in Java.
4. Organize the data and the functionality into required classes using object oriented concepts.
5. Learn to use and implement Polymorphism, Object Explicit casting, ArrayList class, File I/O (Reading/Writing from/to files).

Read Carefully:

- This Project is worth 10% of your final grade.
- The project consists of two phases (Phase 1 is worth 4% and Phase 2 is worth 6%)

- **WARNING**: This is an individual project; you must solve it by yourself. Any form of cheating will result in receiving **– 4%** (less than zero) in the program.

    - The deadline for Phase 1 is by **11:00 PM on Moday, Oct 21, 2019.**
    - The deadline for Phase 2 is by **11:00 PM on Thursday, Nov 28, 2019.**
    - **Note:** once the clock becomes 10:59PM, the submission will be closed! Therefore, in reality, you must submit by 10:58 and 59 seconds.

- LATE SUBMISSION: you are allowed to make a late submission, but there is a penalty. If you submit **within 24 hours** of the due date (so on Friday by 10:59PM), you will receive a 25% deduction. You will NOT be able to submit after this date/time.

Deliverable and Blackboard Submission:

- This project must be submitted online via blackboard
- If your file is empty or you upload the wrong file, it will be solely your responsibility, and your grade will be zero.
- **Phase 1 (UML part)**
  you need to submit a **UML class diagram** (in pdf, gif or jpeg format) with a brief discussion of such design and should be named as:
  CCCS121Project_StudentId_SectionName.pdf
  Example: CCCS121Project_1110348_LA.pdf

- **Phase 2 (the coding part)**
  You must include only the source files in your submission (do not submit any *.class files!) (Your submission must be one zip file).
- The zip file should be on the format CCCS121Project_StudentId_SectionName, if it's not in this format you will lose points.

  NOTE: your name, ID, section number AND EMAIL should be included as comments in all files!

Project Description:

Task: College of Computer Science & Engineering Academic Affairs Electronic (CCSEGeAA).

This customization will help Academic Affairs in fulfilling new Faculty requirements and keeping track of student academic activities. First, the system must continue with providing the basic learning management systems functionalities. For example, students may able to registering courses, printing schedule or adding dropping, etc… Moreover, faculty can choose courses to teach, print course student lists.

In addition, to the basic functionality the updated **CCSEGeAA** system must generate validation reports on registration (adding courses) activities to help in addressing problems quickly.

The validity reports generate reports on students' cases that need attention. These two cases are: students registering courses under 12 hours and graduate students that must register Senior project 1 but they did not. The requirements of every needed command will be described below.

More Details are as follows:

Your program will use File I/O to read input data from the given file with name [inCCSEGeaa.txt]. Make sure that the file exists or display a message that it does not exist. Once the file is open all CCSEG-eAA information must be read into the application. This means the Course, Instructor, Student and Sections data must be read and stored in the appropriate data structure in the CCSEG-eAA application. All data entry commands will start with the word "input". This is followed by an integer indicating the number of records that need to be read. The description of every data entry command is given in the coming sections.

The commands you will have to implement are as follows:

- **InputDepartmentPlan**– following this command the file consists of an integer indicating the total number of courses (52 courses) in the CS department plan. Thus, students are required to complete all these courses in order to graduate with a bachelor's degree in Computer Science.

The Course information stored in the input file is organized as follows:

1) If the line begins with the word "Level" then it will be followed by the level number. This is used to group the courses according to their levels in the plan. When you read this line you must read the integer that follows and store it in all courses of this level.
2) If the line begins with a Course code then you must read all remaining course information Course Number, Course Name, Course Credit, Theory and Lab. Moreover, the Course Name is stored with "?" marks instead of blanks to ensure that the complete course name is read. This must be replaced by blanks after it is read. For example:

In the input file the course name is: "Writing?Skills"

In the Application CCSEGeAA it must be replaced with a blank space

Furthermore, the words theory and lab if they are read from the input file then the Boolean variables in the Course class are set to true otherwise they are set to false. The level number is read from the beginning and should be repeated with every course in the same level. [see input file]

In the output file that must be created by the application with name [outCCSEGeaa.txt]. must include the following message after all Course data is inserted. Also, see sample output file.

The Message: "Department Plan(Courses) Data has been Inserted"

- **InputInstructorData** – Again this command is followed by an integer indicating the number of instructors teaching in the Faculty. Their information includes Instructor's name, Instructor's ID, Beginning of work year, Qualification (PhD, MSc, etc..) and Instructor's Schedule (List of Sections). The instructors Schedule will be chosen in a separate command (coming later). Similar to Course the name of the Instructor's name and her qualifications will include "?" mark. Thus, the "?" mark must be with a blank once read into the application. [see input file].

In the output file that must be created by the application with name [outCCSEGeaa.txt] must include the following message after all Instructors data is inserted. Also, see sample output file.

The Message: "Department Instructor's Data has been Inserted"

- **InputStudentData** – Here the Faculty Students' information is read into the application. The information includes Students name (String) , student id (Integer), student starting (Registration) date at faculty (Integer) , GPA (double), Total Number of hours studied and the students current schedule. Note that the student schedule will be registered in a separate command (coming later). [see input file]

In the output file that must be created by the application with name [outCCSEGeaa.txt] must include the following message after all Students data is inserted. Also, see sample output file.

The Message: "Department Student's Data has been Inserted"

- **InputSectionData**- Finally, the Department Schedule information is read from the input file. The Schedule is built out of Sections information (current available courses). These sections are the ones registered by students and added into their schedules. Same thing goes with the instructor's; their schedules include the sections they teach. The Section information includes ID, Name, Course Information (determined by the pair course code and course num in the input file), Size(No. of students in section), Slot for Theory part (Day & Time), Slot for Lab part (Day & Time), Room and Lab numbers, Instructor name and finally, list of students (Student class) Registered. Note that if a Theory or Lab part of the course is not given then the term "N/A" is added to indicate that it's not taught in the course. Also, the Course information is stored in the Sections array as a reference to the class course described above.

In the output file that must be created by the application with name [outCCSEGeaa.txt] must include the following message after all Sections data is inserted. Also, see sample output file.

The Message: "Department Schedule (Sections) has been Inserted"

- **RegisterCourse** – This command allows the students to register the courses they want. Then it checks the validity of their choices. Firstly by verifying that the given Student ID is a valid ID. Secondly by checking for the **two cases listed** below.
  The cases that the system should check for are as follows:
  1. If a student is a graduate [TotalHourse > 100] and did not register CCCS411 then a warning message must be added to the registration report, **[see input file]**
  2. If a student has registered for less than 12 hours then a warning message must be added to the registration report, **[see input file]**

The result of RegisterCourse command will be the student's registered schedule. This will be followed by a report stating the problems with the requested courses, if any. In the input file the requests start with the student number then it is followed by a list of requested section IDs, which means all input data are integers. The request line ends with a "-1" indicating no more courses are requested, [**see input file].** Keep in mind that once a student registers a course and it is added to his Schedule then AT THE SAME TIME it must be added to the Section Student list found in the Department Schedule.

The output will be the registered schedule of the student followed by a report on the validity of the requests.**[see output file]**

- **InstructorLoadRequest**- Up to this point no instructor has requested a course to teach. Thus, under this command the total number of requests is given. This is followed by a list of integers the first one will indicate the Instructors ID then the remaining integers are the list of Section IDs that the instructor wants to teach. It's important to keep track of class aggregations and similar to Student Registration, once a section is added to Instructor load his name (instructor) must also be added in the Sections array.

- **Print All InstructorsLoads**- it will only printout all the instructors' schedules.

- **Print Section Student List**- This command reads the Sections ID and prints all its registered students.

Further Details are as follows:

You have to create **six classes** in this program.

- Course.
- CCSEGMember is a super class of Instructor and Student classes.
- Instructor.
- Student.
- Section.
- CCSEG-eAA is a tester class to create objects and invoke appropriate methods for the program to execute successfully. And you have to create **three** ArrayLists in the void main method:
    1. private static Course[] deprtCourses;
    2. private static ArrayList< CCSEGMember > deprtMember = new ArrayList< CCSEGMember >();
    3. private static ArrayList<Section> deprtSchedule = new ArrayList<Section>();

Important Notes:

- Use class & object, ArrayList of Object, passing object to method, dynamic binding and Inheritance is mandatory.
- Use and implement Polymorphism and Object Explicit casting.
- Use the instanceof operator
- Use Files, Reading/Writing from/on files.
- Your program output must be exactly same as given sample output files.
- Your display should be in a readable form.
- Organize your code in separated methods.
- Document your code with comments.
- Use meaningful variables
- Use dash lines between each method.

Input and Output Format:

Your program must generate output in a similar format to the sample run provided.

Sample input: See sample input files.

Sample output : See sample output files.

Suggestions:

- Read AND fully understand this document BEFORE starting the program!
- Once the solution is 100% clear to you, then begin making your code.

Final suggestion: START EARLY