Course Scheduler Application

Overview:

Your task is to create a course registration system. Students can register for a certain number of courses. Each course is taught by one faculty member. Each faculty member can teach a certain number of courses.

The system stores the first, middle, and last name of everyone who attends the university (students and faculty members), along with the email, telephone number, street address, city, state, and zip code. There is also a unique id number that is assigned by the system when the student or faculty member is first entered into the system.

For each student, the system also tracks the date of birth, current GPA, and the date the student started attending the university.

For each faculty, the system tracks the hire date and whether the faculty member is tenured.

Courses belong to a department (e.g. CS or BIO), have a code (e.g. 1A, 4A, or 4B), and a description (e.g. Introduction to Computer Science). The course id uniquely defines a course and is simply a combination of the department and code (e.g. CS1A or BIO1B). Each course has minimum and maximum number of students per session. If there are not enough students to fill a session, that session is cancelled. If no sessions for a course are scheduled, the entire course is cancelled.

For each course you can have multiple sessions. A session has a unique id (e.g. c123486). Each session id is system generated and must be unique across all sessions.

Each course has a minimum and maximum number of students who can enroll. If a course does not meet the minimum number of students, it will be cancelled.

Your solution should be built to be as flexible and dynamic as possible. You should create a way to swap scheduling algorithms during runtime. For example, maybe one of your scheduling algorithm runs through all sessions and students and fills sessions sequentially. You do not have to define multiple scheduling algorithms, but your code must be written to allow someone else to use your code, define a new algorithm at run time, and use that algorithm at run time. In your report, you must document how you were able to achieve this.

You must also create a flexible way to change the algorithm with how unique ids are determined (i.e. for student/faculty ids and for session ids). You should create a separate algorithm for student/faculty ids and for session ids.

Guidelines:

1. Make sure you follow the Java coding style guidelines we discussed in class

2. Appropriately comment your classes so that Javadoc documentation can be automatically generated. Since we did not get to cover this in class, you may want to refer to this tutorial that I think is pretty good:

   https://www.tutorialspoint.com/java/java_documentation.htm

3. Use packages to organize your code effectively. Separate library classes from application level classes

4. Make sure all of your methods are robust and cannot lead to broken code if they receive invalid input

5. Use exceptions to report errors

6. Hard code as little data as possible for maximum run time flexibility

7. Use constants where necessary

Input:

I would suggest multiple input files for your data. You will require student data, faculty data, and course data. You will need the following types of input:

1. All student information
2. All faculty member information
3. All course information
4. The number of sessions to schedule for each course
5. All of the courses each student wishes to take
6. Control parameters, such as the number of courses a student can take and the number of sessions a faculty member can teach

As the program creator, it is your job to create the input format and document it.

Output:

The format for your output will be graded on how intuitive it is to follow.

Given the input your task is to generate output that has the following:

ScheduledCourseSessions.txt:

- Each course that was scheduled (full details)
- Each session for that course (i.e. session id)
- The full name and id number of the instructor for the session
- The number of students in the session
- For each student in the session, the full name and id number.

UnscheduledCourseSessions.txt:

- Each course that was not scheduled (i.e. not even one session scheduled) along with the minimum number of students that needed to be in the course.

Faculty.txt:

- Print all details for each faculty member and list the full course details of each course and session he/she has scheduled along with the session id and number of students in that session

ScheduledStudents.txt:

- For each student show all details and list the session id, course id, and course description for the classes taken.

UnscheduledStudents.txt:

- List full details for each student that has no classes to take.

Program Interface:

When your application loads, it should read all the import files and generate output files and show the following statistics (as an example):

Total Students	100
Total Faculty	20
Total Courses	10
Total Sessions Scheduled	50
Total Courses (not sessions) Unscheduled	2
Total Students With No Classes	3

You must run your application to demonstrate a variety of test cases and identify each test case in your report.

Your sample runs should show different properties of the system (some unscheduled students, some unscheduled courses, etc.). The size of each test case should be between 20 students (minimum) to 100 students.

Report structure:

1. Table of Contents
2. Printed source code (well organized)
3. Detailed javadoc documentation (print without the frames).
4. Essay describing: a) your high-level approach, b) your algorithms, c) key design decisions you made and why, d) what you learned by doing this exercise, e) rate the exercise from 1 to 10 and explain why (I will not give you 0 points if you rate this as a 0, but give honest feedback and do not wait until the last day to do the exercise and say it was too much work because you procrastinated ☺). In your essay use headings for each section.
5. Input and output documentation – provide a section identifying your input format and output format.
6. Test Cases – print the input and output for each test case and explain what each test case is supposed to demonstrate. Read through your test case input and output and mark it up. In your markup explain how the output of the test case gave you confidence that your solution actually works. Also, explain how you generated the test data for your test cases.

Scoring:

You will be graded on multiple criteria. It is not just about how well your code works. Please make sure what you submit is easy to grade. Keep a clean report structure. Highlight key areas. If I cannot make it through your report in about 3-5 minutes and understand the key ideas, you cannot expect a good score.