CourseManager

Description

In this assignment, you are tasked with implementing a class named CourseManager. This system works in conjunction with predefined Student and Course classes to implement a Credit-based Course Selection System. This system needs to include functionalities for students to select courses based on credit points or cancel the selection, as well as generate results for credit-based course selections.

In the credit-based course selection system, there exist two states:

when ifOpen is set to true, it means that the credit-based course selection phase is currently underway;

else, when ifOpen is false, it denotes that the course selection period has ended, In this state, the system determines the final list of successful enrollments based on the course capacity and the credit bids placed.

For Student

Within the course selection system, each student has an initial credit value credits, a list of enrolled courses enrollCourses, and a list of successfully selected courses successCourses.

During the credit-based course selection phase, a student can:

1. Bid credits to enroll courses. A student can not enroll one course twice.
2. Modify the credit values allocated to a enrolled course.
3. Cancel course enrollment, with the bid credits being refunded.
4. Update enrollCourse(upon course selection or cancel).
5. Query their list of courses they have bid credits for and the corresponding credits.

After the credit-based course selection phase ends, a student can:

1. Update successCourse (upon course withdraw).
2. View their list of successfully enrolled courses.

For Course

Each course has a set capacity, along with a list of students who have bid credits enrollStudent

and their corresponding credit lists credits, as well as a list of students who have successfully enrolled in the course successStudents.

During the credit-based course selection phase, a course can:

1. Obtain the list of credit points bid by students credits.
2. Get a list of students who have bid credits enrollStudents.
3. Update enrollStudent and credits (when course enrollments, withdrawals, or changes to credit bids occur).

After the credit-based course selection phase ends, a course can:

1. Update successStudents.
2. Query the list of students who have successfully enrolled.

For the CourseManager

The course selection system maintains a status ifOpen and holds a list of all students students and a list of all courses courses.

During the credit-based course selection phase, the course selection system can:

1. Update student and course information based on course selection applications.
2. Update student and course information based on course withdrawal applications.
3. Return to a student their list of courses for which they have bid credits and the corresponding credits upon querying their course status.

At the end of the credit-based course selection phase, the course selection system can:

1. Generate the enrollment results for each course based on the students' credit bids and the course capacities, subsequently updating relevant student and course information accordingly.

Problems

Student and Course classes are in CourseManagerTest.java

1. Review and understand the provided Student class. [No Points]
2. Review and understand the provided Course class. [No Points]

3. Implement and submit the CourseManager class. [100 Points]

Notes for this assignment

1. Do not use package statements in your source code.

2. The output of methods should follow the specification exactly. Do not print extraneous information within any methods.

3. The CourseManager class must interface correctly with the provided Student and Course classes without modifying them.

We will explain the classes mentioned in this assignment, in which the Student and Course classes will explain the fields involved, please refer to the specific methods in codes provided on your own , and the CourseManager class will explain the fields as well as the use of the methods that need to be implemented and the points of note

Class1: Student (already implemented, not to be submitted)

Review the provided Student.java file. The Student class represents a student with specific attributes and behaviors. Use this information to determine how the CourseManager should interact with students.

Fields:

```
// Basic information about one student and the studentID is unique.
// Same student ID or same student objects can both be considered as
// representing the same student.
private String studentID;
private String email;
private String name;
```

```
// The manager class instance, in which all the methods related to the
interaction between the student and the course are implemented.
// Each student only enrolls in one cousermanager.
private CourseManager courseManager;
```

```
// Credit points available for students, will be assigned in the constructor
// or set method.
private int credits;
```

```
// Record of courses for which credits have been bid.
private ArrayList<Course> enrollCourses;
```

```
// Successfully selected courses after the end of the credit-based course
// selection phase.
private ArrayList<Course> successCourses;
```

```
public Student(String studentID, String email, String name, int credits) {
        this.studentID = studentID;
        this.email = email;
        this.name = name;
        this.courseManager = null;
        this.credits = credits;
        this.enrollCourses = new ArrayList<>();
        this.successCourses = new ArrayList<>();
    }
```

Class2: Course (already implemented, not to be submitted)

Review the provided Course.java file. The Course class represents a course with attributes such as capacity, students enrolled, and enrollment credits. Utilize this information when implementing the CourseManager.

Fields:

```
// Basic information about one course and the curseID is unique.
// Same course ID or same course objects can both be considered as representing
// the same course.
private String courseID;
private String courseName;
```

```
// Maxmium capacity for this course
private int maxCapacity;
```

```
// The manager class instance, in which all the methods related to the
interaction between the student and the course are implemented.
// Each course only enrolls in one cousermanager.
private CourseManager courseManager;
```

```
// Students who bid credits for the course and the corresponding credits bid
private ArrayList<Student> enrollStudent;
private ArrayList<Integer> credits;
```

```
public Course(String courseID, String courseName, int maxCapacity) {
        this.courseID = courseID;
        this.courseName = courseName;
        this.maxCapacity = maxCapacity;
        this.courseManager = null;
        this.enrollStudent = new ArrayList<>();
        this.credits = new ArrayList<>();
        this.successStudents = new ArrayList<>();
    }
```

Class3: CourseManager (need implemented)

Implement a class named CourseManager in CourseManager.java. This class is responsible for

managing all course-related and student-related functionality in the system. In the class

CourseManager you need to define:

Fields:

Constructor:

Methods:

```
// Successfully selected students for this course after the end of the credit-
based course selection phase.
private ArrayList<Student> successStudents;
```

```
private ArrayList<Course> courses
// Maintains a record of all courses successfully registered.
// It is guaranteed that students enrolled in a course must exist in students.
```

```
private ArrayList<Student> students
// Maintains a record of all students successfully registered.
// It is guaranteed that courses student enrolled in must exist in courses.
```

```
private boolean ifOpen
// Represent system open(true) or not(false).
```

```
public CourseManager()
// Constructor, initializes the course and student lists, and set the system
// default status open(true).
```

```
public ArrayList<Student> getStudents()
// getter for students

public ArrayList<Course> getCourses()
// getter for courses

public void setIfOpen(Boolean ifOpen)
// setter for ifOpen

public boolean getIfOpen()
// getter for ifOpen

public void addCourse(Course course)
// Register a course. Add a course object to courses and set the courseManager
of the course object to this manager. It is guaranteed that all courseIDs are
unique.

public void addStudent(Student student)
```

```
// Register a course. Add a student object to students and set the courseManager
of the student object to this manager. It is guaranteed that all studentIDs are
unique.

/**
* Attempts to enroll a Student in a Course.
* Enrollment will only be successful if the course exists, the student has not
already enrolled, the credits is greater than 0 , and they have enough credits to
bid.
* If successful, the student's credits will be reduced by the amount bid on the
course. The corresponding list in Student and Course should be updated.
* Only available when ifOpen is true. Return false if ifOpen is false.
* @return boolean Returns true if student enroll this course successfully;
otherwise, it returns false.
```

```
*/
public boolean enrollStudentInCourse(Student student, String courseId, int
credits)
```

```
/**
* Modifies the number of credits a student has bid on an already enrolled
course.
* The modification will only be successful if the course exists, the student is
currently enrolled in the course,and the new bid is within the student's
available credits. This can be used to increase or decrease the bid.
* On a successful bid modification, the student's credits will be updated to
reflect the new bid amount. The corresponding list in Student and Course should
be updated.
* Only available when ifOpen is true. Return false if ifOpen is false.
* @return boolean Returns true if the bid modification was successful;
otherwise, it returns false.
*/
public boolean modifyStudentEnrollmentCredits(Student student, String courseId,
int credits)
```

```
/**
* Drops a student's enrollment from a specific course.
* The drop will succeed only if the course exists, and the student is currently
enrolled in it.
* If ifOpen is true, upon a successful drop, any credits the student had bid
for this course will be refunded in full. The corresponding list in Student and
Course should be updated.
* Only available when ifOpen is true. Return false if ifOpen is false.
* @return boolean Returns true if the student successfully drops the course;
otherwise, it returns false.
*/

public boolean dropStudentEnrollmentCourse(Student student, String courseId)
```

It should be noted that, as in the case of our normal selection, if the number of electors exceeds the maximum number of courses, the principle of " same credit, same drop " will be followed.

Eg. If the maximum number of students is 10, and 15 students choose this course, and after sorted by enrolled points, if the 8th, 9th, 10th, and 11th students are all of the same credit, then the four of them will not be able to choose the course successfully, and the number of successful students will be seven.

```
/**
* Completes the course registration process. Change ifOpen to false.
```

```
* This method resolves which students get into each course based on their bids
and the course capacities. Students with higher bids are prioritized. The
corresponding list in Student and Course should be updated.
* Only successStudents in class Course and successCourses in class Student need
to be updated.
*/
public void finalizeEnrollments()
```

```
/**
* Retrieves a list of courses with associated credits for a given student.
* Each String in the list includes the course ID and the points bid by the
student in enrollCourses, follow the format: "courseID: enrollmentCredits"
(without quotes).
* Only available when ifOpen is true. Return null if ifOpen is false.
* @return A list of Strings, each representing the courses and the respective
credits the student enrolled.
*/
public ArrayList<String> getEnrolledCoursesWithCredits(Student student)
```