Assignment 3

Description

In this Assignment, you need to implement four classes: **Student**, **Course**, **Grade**, and **GradeSystem**.

- The class **Student** represents a student and contains attributes such as student name and student number.
- The class **Course** represents a course and contains attributes such as course name and course number.
- The class **Grade** represents a grade, and a grade belongs to a student who chooses a course, so the grade object needs to include the courses and student objects it is associated with.
- The class **GradeSystem** represents a grade system. There are student list, course list, and grade list in the grade system. Through these lists, various operations such as querying student information, querying course information, and querying grade information can be realized.

Problems

- Problem 1: Design a class named Student [Easy, 15 marks]
- Problem 2: Design a class named Course [Easy, 15 marks]
- Problem 3: Design a class named Grade [Easy, 20 marks]
- Problem 4: Design a class named GradeSystem [Medium, 50 marks]

Notes for this assignment

1. You should submit all source code directly into your OJ system, do not compress them into one folder.
2. No Chinese characters are allowed to appear in your code.
3. No package included.
4. The output must strictly follow the description and the sample in this document. Do not print anything in all methods.
5. You may need to use String.format().

Problem 1 Student

Design a class named **Student** in **Student.java**

In this problem you need to submit: **Student.java**

Student describe each student. In class **Student** you need to define:

Fields:

- ```
  private int sID
  //Student ID. Each student's student ID is unique and is automatically generated by class Student. Starting from 1, each new student's ID will increase by 1.
  ```
- ```
  private String name
  //Student name
  ```
- ```
  private static int student_Cnt
  //Initialized to 1, plus 1 while a Student object is created
  ```

Constructor:

- ```
  public Student(String name)
  //Constructor, automatically generate student ID and generate student object according to the name provided by the user
  ```

Methods:

- ```
  public int getsID()
  ```
- ```
  public String getName()
  ```
- ```
  public void setName(String name)
  //Modify student name.
  ```
- ```
  public void setsID(int sID)
  //Do not allow to set sID, do nothing in this method.
  ```
- ```
  public static int getStudent_Cnt()
  //Get the value of cnt
  ```
- ```
  public String toString()
  //When print an object of this class, follow the format: "Student: %s, sID: %d"
  ```

Hints:

- You can add other methods or attributes that you think are necessary.

Problem 2 Course

Design a class named **Course** in **Course.java**

In this problem you need to submit: **Course.java**

Course describe each Course. In class **Course** you need to define:

Fields:

- ```
  private int cID
  //Course ID. Each course's course ID is unique and is automatically generated by class Course. Starting from 1, each new course's ID will increase by 1.
  ```
- ```
  private String name
  //Course name.
  ```
- ```
  private static int course_Cnt
  //Initialized to 1, plus 1 while a Course object is created
  ```

Constructor:

- ```
  public Course(String name)
  //Constructor, automatically generate course ID and generate Course object according to the name provided by the user
  ```

Methods:

- ```
  public int getcID()
  ```
- ```
  public String getName()
  ```
- ```
  public void setName(String name)
  //Modify course name.
  ```
- ```
  public void setcID(int cID)
  //Do not allow to set cID, do nothing in this method.
  ```
- ```
  public static int getCourse_Cnt()
  //Get the value of cnt
  ```
- ```
  public String toString()
  //When print an object of this class, follow the format: "Course: %s, cID: %d"
  ```

Hints:

- You can add other methods or attributes that you think are necessary.

Problem 3 Grade

Design a class named **Grade** in **Grade.java**

In this problem you need to submit: **Student.java**, **Course.java**, **Grade.java**

Grade describe each Grade. In class **Grade** you need to define:

Fields:

- ```
  private Course course
  //Course object corresponding to this grade
  ```
- ```
  private Student student
  //Student object corresponding to this grade
  ```
- ```
  private float grade
  //The grade(>=0);
  ```
- ```
  private float GPA
  //The GPA calculated by grade
  ```

Constructor:

- ```
  public Grade(Course course,Student student,float grade)
  //Constructor, innitialize student, course, grade and GPA.
  ```

Methods:

- ```
  public static float CalGPA(float grade)
  //A static method to calculate GPA by grade.
  ```
- ```
  public Course getCourse()
  ```
- ```
  public float getGrade()
  ```
- ```
  public Student getStudent()
  ```
- ```
  public float getGPA()
  ```
- ```
  public void setCourse(Course course)
  //Do not allow to set Course, do nothing in this method.
  ```
- ```
  public void setGPA(float GPA)
  //Do not allow to set GPA, do nothing in this method.
  ```
- ```
  public void setGrade(float grade)
  //After set the grade, update the GPA at the meantime.
  ```
- ```
  public void setStudent(Student student)
  //Do not allow to set Student, do nothing in this method.
  ```
- ```
  public String toString()
  //When print an object of this class, follow the format: "sID: %d, cID: %d,
  Grade: %.1f, GPA: %.2f"
  ```

Hints:
  ```

- You can add other methods or attributes that you think are necessary.
- GPA conversion table:

Grade	GPA
0~59	0
60~62	1.15
63~66	1.63
67~69	2.08
70~72	2.42
73~76	2.78
77~79	3.09
80~82	3.32
83~86	3.55
87~89	3.73
90~92	3.85
93~96	3.94
97~100	4.00

Problem 4 GradeSystem

Design a class named **GradeSystem** in **GradeSystem.java**

In this problem you need to submit: **Student.java**, **Course.java**, **Grade.java**, **GradeSystem.java**

GradeSystem describe each Grade System. In class **GradeSystem** you need to define:

Fields:

- ```
  ArrayList<Student> studentList
  //The list of students
  ```
- ```
  ArrayList<Course> courseList
  //The list of courses
  ```
- ```
  ArrayList<Grade> gradeList
  //The list of grades
  ```

Constructor:
  ```

- ```
  public GradeSystem()
  //Constructor, innitialize studentList, courseList, gradeList. For a list with no elements in it, its size should be 0 and its reference should not be null.
  ```

Methods:

- ```
  public ArrayList<Course> getCourseList()
  ```
- ```
  public ArrayList<Grade> getGradeList()
  ```
- ```
  public ArrayList<Student> getStudentList()
  ```
- ```
  public boolean checkStudent(int sID)
  //If a student is not in the student list, return false.
  //Else return true.
  ```
- ```
  public boolean checkCourse(int cID)
  //If a course is not in the course list, return false.
  //Else return true.
  ```
- ```
  public boolean addStudent(Student student)
  //If a student is in the student list(that is his/her sID is in the student list), return false.
  //Else add the student into the student list, return true.
  ```
- ```
  public boolean addCourse(Course course)
  //If a course is in the course list(that is its cID is in the course list), return false.
  //Else add the course into the course list, return true.
  ```
- ```
  public boolean addGrade(Grade grade)
  //If a grade satisfied:
  //   - is in the grade list(that is its corresponding student's sID && its corresponding course's cID already exists a grade in the grade list)
  //   - its grade in the grade list is less than 60
  //   - the new grade to be add is not less than 60
  //Update the grade in the grade list by the new grade and return true.
  //Else if the grade is not int the grade list && its corresponding student's sID already exists in studentList && its corresponding course's cID already exists in courseList, add the grade into the grade list, return true.
  //Else return false.
  ```
- ```
  public float GPA(int sID)
  //Return the GPA of student with sID. The GPA is the average GPA of all the courses the student enrolled in.
  //If the student has not taken any courses, return 0.
  ```

- ```
  public float Average(int cID)
  //Return the average score of course with cID.
  //If the course has no students, return 0.
  ```

- ```
  public ArrayList<Grade> listStuGrade(int sID)
  //Return an ArrayList<Grade> of the student with sID, the order of the
  grades is in accordance with the course order in the CourseList. Skip
  courses the student didn't take.
  ```

- ```
  public ArrayList<Grade> listCouGrade(int cID)
  //Return an ArrayList<Grade> of the course with cID, the order of the grades
  is in accordance with the student order in the StudentList. Skip students
  who didn't take the course.
  ```

Hints:

- You can add other methods or attributes that you think are necessary.
- Do not sort the lists.
  ```