COL 106 Assignment 1

Updates

1. CGPA calculation updated in **INFO student** query section.
2. Note added in **SHARE** query section.
3. Sample query and result text file updated.
4. Java version to be used added in Important notes section.
5. Result of **SHARE 2015cs1325 COL106** query is updated.
6. Signature of **gradepoint** function in GradeInfo_ interface is updated
7. Course file updated. Title of course COL113 updated to Distributed Systems
8. Implementation of **gradepoint** function in GradeInfo_ interface is added.
9. "You must not use any integer constant in your code." is removed from the description.
10. CGPA formula added for reference in **INFO student** query section. < 6:00 P.M , 3 August 2019>
11. README file should be added in **src** folder. < 2:29 P.M, 4 August 2019>
12. Deadline extended to 9th Aug(Friday), 11:55 P.M
13. Interface GradeInfo_ updated with grade() function <<<12:47 P.M, 7 August 2019>>>
14. Test cases uploaded for reference Download from here .

Assignment

In this assignment you have to use both linked lists and arrays to store data about students, hostels, courses, etc.

You may only use external packages java.util.Iterator and java.io.* for this assignment.

· First, implement the following LinkedList_ interface, which uses a Position_ interface:

```
public interface Position_<T> {// Supports any class T
   public T value();          // Return value at position
   public Position_<T> after();// Returns the position after this position in its list
}

public interface LinkedList_<T> {// Supports any class T
   public Position_<T> add(T e);              // Add element e to this list, returns it position in the list
   public Iterator<Position_<T>>  positions();// Returns an iterator for all positions in the list
   public int  count();                      // Returns the number of elements in the list
}
```

A student has three entities: department, hostel and courses. In this assignment, you are required to maintain at least these three linked lists to manage these entities: **allHostels** includes all hostels, **allDepartments** includes all departments and **allCourses** includes all courses. In addition, each element in the lists allHostels, allDepartments, allCourses itself should maintain lists that includes all students in that hostel, department, or course, respectively.
· In order to organize these entities, use interface Entity_:

```
public interface Entity_ { // Entities Classes Hostel, Dept, and Course all have this functionality.
   public String name();                // Returns this  entity's name
   public Iterator<Student_> studentList();        // Returns all students of this entity
}
```

· You may implement classes Hostel, Department, and Course, which are derived from Entity_. You must also implement interface Student_:

```
public interface GradeInfo_ {
   enum LetterGrade {
        A, Aminus, B, Bminus, C, Cminus, D, E, F, I;
   } // I is the place-holder grade for the current semester, where grade has not been earned yet
   public static int gradepoint(GradeInfo_.LetterGrade grade)// Returns the points earned for each letter grade
```

```
        {
                if (grade == GradeInfo_.LetterGrade.A) return 10;

                else if (grade == GradeInfo_.LetterGrade.Aminus) return 9;

                else if (grade == GradeInfo_.LetterGrade.B) return 8;

                else if (grade == GradeInfo_.LetterGrade.Bminus) return 7;

                else if (grade == GradeInfo_.LetterGrade.C) return 6;

                else if (grade == GradeInfo_.LetterGrade.Cminus) return 5;

                else if (grade == GradeInfo_.LetterGrade.D) return 4;

                else return 0;

        }

   // I grade for every course. Override the function in implementation as needed.
    public default LetterGrade grade()
        { return LetterGrade.I;
        }

}


public interface CourseGrade_ {         // Tuple of course and grade
   public String coursetitle();          // Returns course title
   public String coursenum();            // Returns course code, e.g., COL106
   public GradeInfo_ grade();            // Returns student's letter grade
}


public interface Student_ {
   public String name();                // Returns student's name
   public String entryNo();             // Returns student's entry number
   public String hostel();              // Returns student's hostel name
   public String department();          // Returns student's department name
   public String completedCredits();    // Returns student's credits earned
   public String cgpa();                // Returns student's cgpa until the previous semester
   public Iterator<CourseGrade_> courseList();// Returns an iterator for all courses for this student
                                              // including those in the current semester
}
```

· Finally, implement a `public class Assignment1`, with a static main that has two parts: private static functions `getData` and `answerQueries`. The main method takes as command line argument three strings; each string is a file name. The first file (student record file) contains personal data about the students, the second (course file) contains the list of courses taken by students, and finally the third (student query file) contains some queries to process. The first two files are passed to the method getData. The third file is processed by the method answerQueris. i

getData reads information per student from the specified student record file and course file, inserting the information in the respective linked lists (e.g., allHostels, allDepartments, allCourses), updating all data structures.

Student record file will contain record details of student in this order: entry number, name, department, and hostel. These names are single word strings, separated by space (See examples later). Every line contains one record of one student. The student record file will hence act as your source of student information.

The course file lists on each line the grade of one student in one course. It has four strings: the first indicates the student's unique entry number, the second indicates the course number with no spaces (e.g., COL106), the letter grade, and the course title. The title may have multiple words (space separated).

answerQueries will read the student query file, which contains one query per line. For each query, your program must output the correct answer to System.out, one answer per line in the specified format -- but in reverse order of the queries, \ie, the first query's answer comes at the end. Three types of queries have to be supported. Their formats and their explanation are given below:

INFO student

> INFO is the keyword. List information about student with the given entry number or the given name: Entry number, Name, Department, Hostel, CGPA, and the course-numbers of all courses taken with the obtained grades in each, sorted lexicographically by the course number (each separated by space), in that order.

Note: Each course is of 3 credits. **While computing CGPA, please ignore the courses which have grade I. CGPA should be rounded off to 2 places of decimal.** Use below formula for computation of CGPA. Here n is the number of courses taken by student.

$$CGPA = \frac{\sum_{\substack{i=1 \\ Grade_i \neq I}}^{n} CourseCredits_i * GradePoint_i}{\sum_{\substack{i=1 \\ Grade_i \neq I}}^{n} CourseCredits_i}$$

SHARE studententrynumber entityname

SHARE is the keyword. Provide space separated list of the entry numbers of all students who share the given entity with him or her.

Note:The result of query SHARE will be sorted in lexicographical order

COURSETITLE coursenumber

COURSETITLE is the keyword. Provide the full title of the given course number.

Example

Example student record file, course file, and student query file have been provided. The required solution for the included query file is given in the third file. Your output must match this format to be counted correct.

Submission instructions

Submission will be at moodle. Link will be provided in due course.

Please ensure that you follow the following set of instructions meticulously while submitting the assignment on Moodle. Also, please note that in case of deviation from this procedure will render your submission faulty, and your assignment might not be evaluated at all by the script. Please follow these steps:

1. On eclipse, by default, all your source files (.java files) are saved into a directory called "src" inside your project. You have to compress this directory (to zip format) and rename the zip file in this format:

   `your-entry-number_assignment1.zip`

   Example: If your entry number is 2012CSZ8019, the zip file should be named 2012CSZ8019_assignment1.zip. It should expand to a directory called `src`, which contains all your .java files. Please make sure that you follow exactly this naming format.
2. Next, convert the zip file to base64 format. On any linux system, you can do this easily by issuing the following command on the terminal :

   ```
   base64 entrynumber_assignment1.zip > entrynumber_assignment1.zip.b64
   ```

   This will create a file with the same name with a b64 extension appended at the end.
3. Upload this b64 file on moodle and submit it (in the "Submission" section). After Submission, there is an option of "Evaluate" in the "Edit" section. Click on "Evaluate". On the right side of the window, you will get information about the submission status. On clicking the "Evaluate" link, you should see a script running and producing an output of your codes' performance against the standard set of test cases that we supply internally for the assessment. A dialog box will show you the messages on how your codes performed against the benchmarks that we provided.

Important notes

- Please note that the immediate feedback is based on simple checks and to inform you whether your java code produces some right output for a basic test case. It is not the real grading. Hence, the grade that you see at submission is only to indicate if submission succeeded. During the actual evaluation, we will run many other tests and provide the final grade based on those. That grade will be uploaded on moodle after grading.
- Please do not submit any executable or other types of files along with the source code. The "src" directory should only contain your code (.java) files and README file. Information about README file is present in Submission Requirements of course home page.
- As you approach the deadline for assignment submission, multiple students will submit their files together. As a result, the server will be slow to respond to clicks on "Evaluate" link. In fact, we may even turn off the immediate feedback (Evaluate) option up to an hour before the submission in case of excessive load. You should submit well in advance to check for basic gradability. You may always modify and re-submit your assignment until the deadline. In any case, when

the response to "Evaluate" request is slow, please be patient and re-try after 30 seconds. Incessant clicking will only make it worse. Please also note that no emails regarding slow response to "Evaluate" will be entertained on the day of submission.

- Following is the full specification of the JDK that will run on the server for evaluation purpose. So please make sure that your own system also satisfies these requirements and that your java codes run in this setup. It will be extremely difficult for us to entertain requests that say the code ran on some other version of JDK, but does not comply with the server's version of JDK:

```
openjdk version "1.8.0_212"
OpenJDK Runtime Environment (build 1.8.0_212-8u212-b03-0ubuntu1.16.04.1-b03)
OpenJDK 64-Bit Server VM (build 25.212-b03, mixed mode)
```

- Note that you must click on "Evaluate" after you submit the b64 file. Submitting the file without clicking on "Evaluate" may mean that your code does not meet the desired specifications. In that case, your submission will not be evaluated for the assignment.
- The system will only be able to evaluate submissions in base64 format. Direct uploading of zip/java/txt/other formats would not let the system evaluate your submission.

Dept. of Computer Sc. & Engg.
IIT Delhi
Bharti Building 422
New Delhi 110016
Ph. (+91) 11 2659-6032