CSCI 3901 Assignment 1

Due date: 11:59pm Thursday, September 21, 2023 in Brightspace

Problem 1

Goal

Get practice in decomposing a problem, creating a design for a program, and implementing and testing a program. Practice basic Java programming.

Background

The people who assess applicants to university graduate programs have not memorized all variants of transcripts. Instead, staff translate information like transcripts into an equivalent of a canonical university so that the assessors can compare applications on the same basis. The translations depend on the original university of the applicant and on some standard translation scales[1].

Typical items that require translation are

- Course grades
- The measure of duration for a course
- Required course lists

In some other instances, such as an English language proficiency test score, what's important is knowing if the score meets the admission minima, should a score be required.

In still other cases, we compute specific information, like a grade point average.

For Dalhousie admission, we consider course grades as letter grades on a 4.3 scale and measure course durations in credit hours. Letter grades translate to grade points to then calculate a GPA:

Letter grade	Grade point
A+	4.3
A	4.0
A-	3.7
B+	3.3
B	3.0
B-	2.7
C+	2.3
C	2.0
C-	1.7

[1] For example: https://www.scholaro.com/

D	1.0
F	0.0
W	No GPA value
Pass	No GPA value
NC	No GPA value
CR	No GPA value
ILL	No GPA value

Grades like “Pass” are not counted in grade point averages.

Problem

Write a Java class, called “AdmissionInfo” that holds and converts admission information for candidates. (Aside: Computer Science at Dalhousie has not automated the assessment of applications. Each application is assessed by people.)

The class can have information that applies to all applicants

- Grade scale translation systems
- Core course names

Each applicant in the system will have their own personal information

- Their transcript information
- Potentially an English language proficiency result

Your class will help answer questions on admissibility for applicants:

- The applicant’s GPA, in Dal 4.3 standard
- What the transcript looks like, were it at Dal
- The latest academic term when the candidate passed each requisite core course with at least the minimum grade

You will also submit a small “main” method in a class called “A1” that will demonstrate a minimal use of the Java class.

AdmissionInfo

The AdmissionInfo class must have the following methods:

Constructor AdmissionInfo( )

Initializes an instance of the object

Boolean gradeScale(String scaleName, BufferedReader scaleInfo)

Record a grading scale that university transcripts can use. scaleName identifies the scale and will appear in transcripts that use this scale. scaleInfo provides the scale itself as a sequence of lines, each line representing a grade equivalence; this stream is designed to accept content from a file.

The first line in the file will denote the nature of the grades at the originating university. That first line will either be the word “alphabetic” or “numeric”.

Every other line in the file is a grade equivalence line. A grade equivalence line in the scaleInfo file consists of two strings separated by a tab character. The first string is the grade on the originating transcript and the second string is the corresponding letter grade at Dalhousie.

The first string in the grade scale is either a letter grade of its own or is a pair of integers separated by a hyphen sign that indicates a range of integers that all correspond to the one letter grade. A scaleInfo stream will not mix numeric and letter grades.

If the original grade scale is numeric then a full scale should include all grades from 0 to the highest grade mentioned in the scale. There should not be any numeric gaps in the scale.

As an example, a percentage conversion scale may use the following lines to begin the stream content:

```
numeric
0-49	F
50-54	D
55-59	C-
60-64	C
```

An example of a letter conversion scale may use the following lines to being the stream content:

```
alphabetic
AA	A+
AB	A
BB	A-
BC	B+
```

The method should return true if the grading scale is ready to be used under the given name and is complete. The method should return false otherwise.

Boolean coreAdmissionCourse(String courseStem)

Admissions will have a set of courses of interest. This method records a string whose inclusion in a course name marks the course as a course of interest to report on later. The courseStem is the text that should appear somewhere within a course title. The courseStem is strictly a string to match; it is not something complex like a regular expression.

For example, a courseStem of “algorithm analysis” will match course names of “advanced algorithm analysis”, “topics in algorithm analysis”, “algorithm analysis and design” but will not match the course name “algorithm design and analysis”.

The method returns true if the courseStem can now be used to identify a course of interest in the future. Otherwise, the method returns false.

Boolean applicantTranscript(String applicantId, BufferedReader transcriptStream)

Accept and store the transcript of an applicant for later analysis. The applicant is identified by the applicantID field. The transcript appears as a sequence of lines in the transcriptStream parameter. The transcriptStream structure follows the format of a text file’s content.

The transcriptStream contains a fixed number and order of header lines followed by course grade lines.

A header line consists of an information identifier, a tab character, and then the information itself. The header line identifiers, in the order in which they must appear in the input stream, are:

- Institution
- Grade scale
- Student name
- Student identifier
- Program
- Major

The subsequent course grade lines have one course per line and have six tab-separated fields on a line. The fields, in order, are

- Term (in YYYY-MM format)
- Subject code
- Course number within the subject code
- Course title
- The course credit hours
- Student grade in the course

A sample file for fictitious courses starting in September 1950 would then look like:

```
Institution         Dalhousie University
Grade scale         Dalhousie standard
Student name        Mike McAllister
Student identifier  B00123456
Program             Bachelor of Computer Science
Major               Computer Science
1950-09        CSCI    1001   Introduction to computing on the abacus     3          B
```

1950-09	ENGL	1000	Writing without clay tablets	2	B-
1951-01	CSCI	1002	Advanced abacus techniques	3	A-
1951-01	MATH	1100	Arithmetic without using your fingers	4	B

The method returns true if the transcriptStream is well-formed, has valid grades, and will now be accessed using the given applicantId.

Boolean translateTranscript(String applicantId, PrintWriter converedTranscript)

Translate the transcript of the given applicant into one with Dalhousie letter grades. "Print" the translated transcript to the PrintWriter parameter, using a format that matches the input format for the applicantTranscript() method, i.e. I should be able to take the PrintWriter output and re-enter it as an applicantTranscript input stream and have it be accepted. Call the output grade scale "Dalhousie standard".

The method should return true if the PrintWriter parameter contains a fully translated transcript. If the PrintWriter output shouldn't be used as a transcript then return false.

Double applicantGPA(String applicantId, Double maxHours, Set<String> coursesToExclude)

Return the GPA, to two decimal points, for the applicant identified by applicantId. Only use the most recent "maxHours" credit hours of courses in the GPA calculation and exclude any course whose title contains any of the words in the coursesToExclude set. For example, we might exclude all course titles that include the string "lab" in the title to focus only on the in-class grades.

If the recent credit hours do not meet maxHours exactly then include as many courses as needed to first exceed the maxHours limit. If there are several courses to choose from to cross the maxHours threshold then first include the course with the highest course number and, if there is a course number tie, then include the one with the alphabetically smaller subject code.

If a course was taken more than once then each instance of the course can be included in the GPA calculation, subject to the constraint of including the recent courses.

Grades are converted to a numeric value according to the grade conversion scale in the background section of this introduction. The contribution of any course is weighted by the credit hours of the course; the GPA is a weighted average of the course grades. If a grade does not have a numeric equivalent in the Dal grading scale then that course is excluded from GPA calculations.

Return the computed GPA as the method value. Return a negative number in the case of any error in the calculation process.

Map<String, Integer> coursesTaken(String applicantId)

Calls to the coreAdmissionCourse method provides words in course titles that will be of interest to us. coursesTaken returns a Map where the Map key is the word from coreAdmissionCourse and the Map value is the number of months, following the first course registration in the transcript, when the course was last taken. If the word from coreAdmissionCourse is not in any course title in the applicant's transcript then that word is not a key in the returned Map.

Courses taken in the applicant's first study term will be listed as in month 0 of the program.

For example, suppose that an applicant has the following course names and terms taken as their complete transcript

1950-09	Introduction to computing on the abacus
1950-09	Writing without clay tablets
1951-01	Advanced abacus techniques
1951-01	Arithmetic without using your fingers
1951-05	Beyond the abacus

Suppose also that coreAdmissionCourse has been given three words of interest: "tablet", "abacus", and "algorithm"

coursesTaken on this combination of information would return the following map

abacus	8
tablet	0

No course contains the word "algorithm" so it isn't in the map, the oldest course mentioning an abacus happened 8 months after the earliest named term (1950-09), and the course mentioning a tablet happened in the first academic term.

Return a null value if there is any error condition in the method.

A1

The A1 class has the minimal code to show that you can make the translations for an applicant from content that you have stored in files, getting at least the applicantId from keyboard input.

Assumptions

You may assume that

- The grade scale for Dalhousie is called "Dalhousie standard".

- The first line of the grade scale stream will always be either “numeric” or “alphabetic”, as long as there is some further grade scale information. You must be prepared for an empty stream, though.

Constraints

- If in doubt for testing, I will be running your program on timberlea.cs.dal.ca. Correct operation of your program shouldn’t rely on any packages that aren’t available on that system.
- Methods cannot throw exceptions in the case of error conditions.
- All comparisons should be case invariant.

Notes

- Make a plan for your solution before starting to code. Write that plan and the parts as part of your external documentation.
- You can create more than one class for your solution. Group information as it makes the most sense.
- Pick a few aspects of the solution to implement at a time rather than try to write code that solves everything in one pass of coding. Use the marking scheme to guide what parts could look like and/or where to prioritize your efforts.
- Simplify the operations of a method the first time that you implement it. Rather than try to think of all possible scenarios at one, start with a typical and simple set-up, get the implementation working on that, and then add in complications incrementally. For example, have the method to compute a GPA first work on a Dalhousie transcript before trying to include the variations to handle transcripts under other grading schemes.

Marking scheme

- Documentation (internal and external) – 3 marks
- Program organization, clarity, modularity, style – 4 marks
- Provide a translated copy of a transcript – 6 marks
- Provide correct Dal-based GPAs – 4 marks
- Report when each of the core courses was taken – 6 marks
- Demonstration that your code can do something using your main() method – 2 marks

The majority of the functional testing will be done with an automated script or JUnit test cases.

Test cases for AdmissionInfo class

gradeScale

- scaleName is null
- scaleName is the empty string
- scaleName hasn’t been used before
- scaleName has been used before

- scaleInfo is null
- scaleInfo is an empty stream
- scaleInfo contains blank lines
- scaleInfo line doesn’t have 2 tab-separated fields in it
- scaleInfo line has 2 tab-separated fields in it
- scaleInfo has one line in it
- scaleInfo has multiple lines in it
- scaleInfo has one alphabetic grade that translates to a Dal grade
- scaleInfo has one alphabetic grade that translates to a grade not on the Dal scale
- scaleInfo has one alphabetic grade that contains a – sign (like B-)
- scaleInfo has a numeric grade range that is properly formatted
- scaleInfo has a numeric grade with no ending grade
- scaleInfo has a numeric grade with no starting grade
- scaleInfo has a numeric grade where the starting grade is greater than the ending grade
- scaleInfo has a hole in the numeric grade range
- scaleInfo is a good numeric grade conversion that is in numeric order
- scaleInfo has a numeric grade conversion that is in random order

coreAdmissionCourse
- courseStem is null
- courseStem is an empty string
- courseStem hasn’t been sent in before
- courseStem has been sent in before
- courseStem is a single word
- courseStem is a multi-word string
- courseStem is all lower case
- courseStem is all upper case

applicationTranscript
- applicantId is null
- applicantId is an empty string
- applicantId hasn’t been seen before
- applicantId has been seen before
- transcriptStream is null
- transcriptStream has no contents
- transcriptStream has a good transcript in it
- transcriptStream is missing one of the header lines
- transcriptStream has a header line without 2 tab-separated fields
- transcriptStream has a header but no courses
- transcriptStream has a header and just one course
- transcriptStream has a header and multiple course lines
- transcriptStream has the courses in chronological order
- transcriptStream has the courses out of chronological order
- a course line doesn’t have the six tab-separated columns

- a course line doesn’t have a properly-formatted date entry
- a course line has a grade that doesn’t correspond to the original grade scale scheme
- a course line has credit hours that are negative
- a course line has a course title that is empty
- a course line has a course subject that is empty
- a course line has a course number that is empty
- a course line has a credit hour that is empty
- a course line has a grade that is empty
- a course has a negative numeric grade
- a course has an alphabetic grade
- call applicationTranscript before gradeScale for the transcript’s scale
- call applicationTranscrpt after gradeScale for the transcript’s scale
- call applicationTranscript for a Dalhousie transcript

translateTranscript

- applicantId is null
- applicantId is an empty string
- applicantId is an application loaded into the class
- applicantId is not an application loaded into the class
- convertedTranscript is null
- transcript has one course
- transcript has no courses
- transcript has many courses
- transcript converts from Dal to itself
- transcript converts from a known conversion scale to Dal
- transcript converts from an unknown conversion scale
- using a letter system for the original transcript
  - translate using the first letter in the grade conversion
  - translate using the last letter in the grade conversion
  - translate using a letter grade not in the conversion file
- using a numeric system for the original transcript
  - translate using the first number range
  - translate using the last number range
  - translate using a grade range where the top and bottom of the range are the same
  - translate a grade of 0
  - translate a grade that is the maximum in the number scale
  - translate with a grade above the maximum in the number scale
  - translate a grade on each edge of a boundary of the numeric grade scale

applicantGPA

- applicantId is null
- applicantId is empty
- applicantId is an application loaded into the class

- applicantid is not an application loaded into the class
- maxHours is null
- maxHours is 0
- maxHours is some positive value
- coursesToExclude is null
- coursesToExclude is an empty set
- get a GPA of a single course that matches each grade value of the Dal grading scale
- get a GPA of more than one course
- get a GPA of a transcript with no courses
- get a GPA of a transcript where the transcript has more hours than maxHours
- get a GPA of a transcript where the transcript has fewer hours than maxHours
- get a GPA of a transcript where there is a choice of which is the last course to add to exceed maxHours
    - break the tie with just the course number
    - break the tie with the subject code as the course numbers match
- get a GPA of a transcript where the course hour tally hits exactly maxHours
- get a GPA of a transcript where the course hour tally will exceed maxHours
- get a GPA that must ignore some grades that have no grade point value
- get a GPA of a transcript where the courses weren't added in chronological order
- get a GPA that excludes courses that match the coursesToExclude list
- get a GPA that excludes courses in coursesToExclude but where the letter case doesn't match
- get a GPA that would include the same course taken twice, but at different times and with different grades
- ask for a GPA before the grading scale is read-in
- ask for a GPA from courses that have different numbers of credit hours

coursesTaken
- applicantId is null
- applicantId is empty
- applicantId is an application loaded into the class
- applicantId is not an application loaded into the class
- call when no core course stems are loaded into the class
- call when only one core course stem is in the class
- call when several core course stems are in the class
- have a course stem match some course name with the same upper/lower case
- have a course stem match a course name when the cases differ
- have a course stem match the start of a course name
- have a course stem match the end of a course name
- have a course stem match in the middle of a course name
- have a course stem match a course in the first term
- have a course stem match a course in the last term
- have a course stem match a course in some intermediate term
- have some course stems match no courses in the transcript

- have the course stem match the whole course name
- have the course name include all parts of the course stem, but with extra text in the middle to break the match
- have a course stem match a course taken more than once and the latest was the first instance in the transcript lines
- have a course stem match a course taken more than once and the latest was the second instance in the transcript lines