COMP167 - Major Programming Assignment 1 – Spring 2025

Introduction

In this assignment, you will create a Java application that will track your grades for this course. It will allow you to input and save grades, show how you are doing in each grade category (exams, labs, quizzes, etc.) and show you how you are doing overall in the course according to your point total.

Enumeration:

This project utilizes enumerated types which helps your program's readability. Here is an example:

```
public enum Category {
      Quiz, Exam, Lab, MajorProgram, FinalExam, Attendance, Assignment;
}
```

The code above would appear in a separate file named Category.java and would serve as the data type for variables that could only have the values: `Quiz, Exam, Lab, MajorProgram, FinalExam, Attendance, and Assignment`. Notice, these are not Strings! Internally, Quiz=0, Exam=1, Lab=2, etc.; however, the enumerated type is a more elegant way of representing these values. The following examples should also prove helpful to you:

String to Enumeration type:

```
Category cat = Category.valueOf( stringValue );
```

The above can be used to convert a string to an enumerated value. For example, if *stringValue* contained the value "Lab", then the variable *cat* would be assigned the enumerated value of Lab.

Enumeration to String:

```
String value = cat.name();
```

The above converts the enumerated tag to a string. So, if *cat* was equal to the enumerated value Assignment, then the String value "Assignment" would be assigned to *value*.

UML for the Category enumeration type:

<<enumeration>> Category
Quiz Exam Lab MajorProgram FinalExam Attendance Assignment

Classes

You are required to implement the following classes at a minimum. You may add other classes (and methods) if you need them.

GradeLevel Class

This class will define each grade level that can be obtained in the course (A to F).

GradeLevel	
-letterGrade:String -minPoints:int -percentage:double	Letter grade (e.g. A, A-, B+, .., , D) Minimum points needed to earn this grade in the class Equals the minPoints / (maximum achievable points in the course)

+GradeLevel()
+GradeLevel(//All params)
+//Accessor and Mutators
+toString():String
	Use the input file format to determine which properties should be included and how they should be formatted.

GradingScale Class

This class will maintain the list of all GradeLevels.

GradingScale	
-gradeLevels:ArrayList<GradeLevel> -maxPoints : int	All GradeLevel objects from A down to D Maximum points achievable in the course.

+GradingScale()
+GradingScale(maxPoints:int)
+//Accessor and Mutators
+toString():String	

	Use the input file format to determine which properties should be included and how they should be formatted.

Assignment Class

This class will track the points earned/received and the maximum points for each class assignment. For example, if a student makes a 79 on a 100 point exam, then *points*=79 and *maxPoints*=100.

Assignment	
-points:int -maxPoints:int	Points earned on the assignment Maximum points (e.g. Most labs have maxPoints equal to 20)

+Assignment()
+Assignment(//All params)
+//Accessor and Mutators
+toString():String
	
	Use the input file format to determine which properties should be included and how they should be formatted.

AssignmentCategory Class

This class will maintain the list of all assignments within a particular Category for a student.

AssignmentCategory	
-assignments:ArrayList<Assignment> -category:Category -pointsEarned:int -pointsAssigned:int	Holds all the Assignment objects for this category The grading category (Quiz, Lab, etc…) Points earned in this assignment category. Rolling total of all assignment maximum points.

+AssignmentCategory()
+AssignmentCategory(category:Category)
+//Accessor and Mutators
+toString():String
	
	Use the input file format to determine which properties should be included and how they should be formatted.

StudentAssignments Class

This class will maintain the list of all assignment categories for a student.

StudentAssignments	
-assignCategories:ArrayList<AssignmentCategory> -gradingScale:GradingScale	All grades arranged in AssignmentCategory objects. Course grading scale
+StudentAssignments()	No-arg constructor

+addAssignment(cat:Category, points:int, maxPoints:int):void +readGradeFile(File inputFile):void +saveGradeFile(File outputFile):void +getGradeReport():String +//Accessor and Mutators +toString():String	Add a new assignment in the appropriate GradeCategory object. See input file description in Section 4 See output file description in Section 5 See sample report format below Should contain all the data in the same format of the input file.

Handling ArrayLists

Each ArrayList should only have five methods in the enclosing class contract: getNum, add, remove, get and set. So if you have an ArrayList named *widgets* that stores items of type Widget, then the associated UML behaviors would be:

+getWidget**s**Size() : int //Return the number of items in the ArrayList widgets.
+getWidget(index:int) : Widget //get the Widget at location index in ArrayList widgets
+setWidget(index:int, item:Widget):Widget //store item at location index in the ArrayList widgets and return the previous item.
+addWidget(item:Widget):void //Append the Widget to the ArrayList.
+removeWidget(index:int) : Widget //remove and return the Widget at location index

```
Grade Report

Quiz 32/45 Points
10/15
8/15
14/15

Exam 0/0 Points

Lab 35/40 Points
15/20
20/20

MajorProgram 76/100 Points
76/100

Final Exam 0/0 Points

Total: 143/185 = 77.30%

Grading Scale
A 1300   83.33%
A- 1220   78.21%
B+ 1135   72.76%
B 1050   67.31%
B- 970   62.18%
C+ 885   56.73%
C 800   51.28%
C- 690   44.23%
D+ 570   36.54%
D 450   28.85%
```

Figure 1 - Sample Grade Report Output

Input File

The name of the input file will be supplied using command-line arguments. If no command-line argument is supplied, then your program should prompt the user for the input file using the JFileChooser class. Figure 2 shows the format of the input file. The grade categories along with the associated grades can appear in any order in the input file. For example, the Quiz grades could be listed last in the file.

Maximum Points for course
A min_points_for_A
B+ min_points_for_B+
…
D min_points_for_D
*
Quiz num_quiz_grades
Grade_0 MaxGrade_0
…
Grade_{n-1} MaxGrade_{n-1}
…
FinalExam
num_Final_Exam_grades
Grade_0 MaxGrade_0
…

Figure 2: Input File Format

Output File

The format for the output file should be identical to that of the input file. In other words, after writing your output file, you should be able to read it back in as an input file. The toString() methods of your classes are designed to make file output simple.

Graphical User Interface

If you would like to add a GUI to your application, look for the GUI addendum. It will be posted later in the same folder as this assignment. You should not attempt this portion of the assignment if you have not completed the other classes.

Grading

If your project does not compile, it receives a grade of zero. If you do not document your program according to the documentation guidelines, the graders have been instructed to deduct **up to 25%.**

Level 1 (40%): Implement the Category enumeration type, GradeLevel and GradingScale classes. Create a main method that calls another method (readInputFile) that opens a file named “grades.txt” and reads and populates GradeLevel objects and stores them in a GradingScale object. The method readInputFile will eventually be a method in the StudentAssignments class. Call and print the output of the GradingScale toString() method inside of a JOptionPane. **Due 2/16/2025**

Level 2 (55%): Implement the Assignment and AssignmentCategory classes. Instantiate and populate Assignment objects using the mutator methods. Demonstrate that your classes are correct by calling the toString() method and sending the output to a JOptionPane. **Due 2/22/2025**

Level 3 (85%): Modify your main so that it uses command-line arguments to provide the inputfile name. Add the logic to obtain the input file name from a JFilechooser if no command-line argument is provided. Implement the StudentAssignments class. Call the addAssignment method and add a new assignment for each assignment Category. Save the updated StudentAssignments object using the saveOuputFile method. – **Due 02/25/2025**

Level 4 (100%): Implement the GUI. See the GUI addendum for details **- Due 02/28/2025**

Extra Credit: Talk to me after you finish Level 4 for extra credit ideas.

Submission

You will submit your assignments through GitHub. You will create a branch to work on a level. When you feel that the level is complete, you will make a pull request for one of the teaching assistants. They will give you feedback or give you a grade on the level. After receiving a grade, you can move on to the next level. More information will be given in lab or in class about submitting your code.