Project #2 – 140 points

Due Date

Monday, February 27, 11:59pm.

Submission Instructions

1. Zip your project folder and submit the file to Canvas. The zipped file MUST include the following items.
   - **Source folder src,** containing the source code *.java files [100 points]
     - (a) Create a package to organize the source files; use all lowercase letters for the package name; you will **lose 2 points** if this is not done properly.
     - (b) MUST include the **@author** tag in the comment block on top of EVERY Java class and put the name(s) who implemented the Java class, or you will **lose 2 points**.
   - **Class Diagram** [10 points]
   - **JUnit** test classes
     - (a) Date class, test the isValid() method. [10 points]
     - (b) International class, test the tuitionDue() method. [5 points]
     - (c) Roster class, test the add and remove methods. [10 points]
   - **Javadoc** folder in which you must include all the files generated by the Javadoc. [5 points]
2. The submission button on Canvas will disappear after **February 27, 11:59pm**. It is your responsibility to ensure your Internet connection is good for the submission. **You get 0 points** if you do not have a submission on Canvas. **Projects sent through the emails will not be accepted.**

Project Description

Your team will extend the software developed in Project 1, to add additional functionalities for processing the tuitions based on the enrollment of the semester. For tuition purpose, students are classified as New Jersey resident and non-New Jersey resident. International students are considered as non-resident and pay health insurance fee. The table below shows the tuition and fees for a single semester.

Tuition/Fees	Full Time Students			Parttime Students	
	Resident	Non-Resident	International	Resident	Non-Resident
Tuition	$12,536	$29,737		$404 per credit hour	$966 per credit hour
University Fee	$3,268			80% of full-time rate	
Health Insurance Fee	not applicable		$2,650	not applicable	

Students enrolled more than 16 credit hours will pay additional for the credit hours exceed 16, using the same per credit hour rate as the parttime students. Students enrolled at least 12 credits are considered as full time, otherwise are classified as parttime. For all students, the maximum number of credits enrolled is 24, minimum is 3. International students must enroll at least 12 credits, except for the international students who are participating in the study abroad program. The maximum number of credits enrolled for the international students in the study abroad program is 12.

Students may have different tuition remissions as listed below. However, parttime students DO NOT qualify.

- Resident students are eligible for scholarship with a maximum amount of $10,000.
- Non-resident students from the tristate area get the tuition discount: $4,000 for New York state residents, and $5,000 for Connecticut residents.
- International students participating in the study abroad program do not pay the tuition. They pay the fees only.

- Here are some tuition calculation examples.
  - Resident students with 18 credit hours: 12,536 + 3,268 + 404 * (18 – 16) = $16,612
  - Resident students with 18 credit hours and $10,000 scholarship: 12,536 + 3,268 + 404 * (18 – 16) – 10,000 = $6,612
  - Non-resident students with 12 credit hours: 29,737 + 3,268 = $33,005
  - Tristate student with 9 credits: 966 * 9 + 3,268 * 80% = $11,308.40
  - International student with 12 credits: 29,737 + 3,268 + 2,650 = $35,655
  - International student in study abroad program with 12 credits: 3,268 + 2,650 = $5,918

Project Requirement

1. You MUST follow the Coding Standard posted on Canvas under Week #1 in the "Modules". **You will lose points** if you are not following the rules.
2. You are responsible for following the Academic Integrity Policy. See the **Additional Note #14** in the syllabus.
3. Test cases for grading are included in the file **Project2TestCases.txt** and the associated output is in the file **Project2ExpectedOutput.txt**. Your project should read the test cases from console in the same order without getting any exceptions and without terminating abnormally. Your program should be able to ignore the empty lines. The graders will run your project with the test cases in **Project2TestCases.txt** and compare your output with the expected output in **Project2ExpectedOutput.txt**. You will **lose 2 points** for each output not matching the expected output, OR for each exception not caught.
4. Each source file (.java file) can only include one public Java class, and the file name is the same with the Java class name, or you will lose **-2 points**.
5. Your program MUST handle bad commands; **-2 points** for each bad command not handled, with a **maximum of losing 6 points**.
6. You cannot import any Java library classes, EXCEPT the **Scanner**, **StringTokenizer, Calendar**, **DecimalForamt** and the necessary **Exception classes**. **You will lose 5 points** FOR EACH additional Java library class imported, with a **maximum of losing 10 points**.
7. You are not allowed to use the Java library class **ArrayList** anywhere in the project or use any Java library classes from the Java Collections, or **you will get 0 points for this project**.
8. When you import Java library classes, be specific and DO NOT import unnecessary classes or import the whole package. For example, `import java.util.*;`, this will import all classes in the `java.util` package. You will **lose 2 points** for using the asterisk "*" to include all the Java classes in the `java.util` package, or other java packages, with a **maximum of losing 4 points.**
9. You **CANNOT perform read or write with Scanner or System.out** in all classes, EXCEPT the user interface class TuitionManager.java, and the print methods in Roster class. You will lose **2 points** for each violation, with a **maximum of 10 points off.**
10. Change the **Student class** to abstract and add the method and abstract methods listed below. The instance variables and methods remain the same as in Project 1. You cannot change or add the instance variables or methods, or change the method signatures below. **-2 points for each violation**. You can add overloading methods.

```
public abstract class Student implements Comparable<Student> {
        ...
    public boolean isValid(int creditEnrolled){} //polymorphism
    public abstract double tuitionDue(int creditsEnrolled); //polymorphism
    public abstract boolean isResident(); //polymorphism
}
```

11. **Polymorphism** is required. You must create the Java classes below to extend the Student class and override the methods in Student class if necessary. All instance variables must be "private". You cannot add additional instance variables. **-2 points** for each violation. No restrictions in adding constants, constructors, and methods.

- **Resident** class extends Student class; contains only one instance variable: `private int scholarship;`
- **NonResident** class extends Student class; no additional instance variables.
- **TriState** class extends NonResident class; contains only one instance variable: `private String state;`
- **International** class extends NonResident class; contains only one instance variable: `private boolean isStudyAbroad;`

12. You MUST add or change the Java classes listed below. **-5 points** for each class missing.
    (a) **Enrollment class** to hold the list of students who are enrolled in classes this semester. You cannot change or add the instance variables, or **-2 points for each violation**. You can add additional methods.

```
public class Enrollment {
   private EnrollStudent[] enrollStudents;
   private int size;
   public void add(EnrollStudent enrollStudent){} //add to the end of array
   //move the last one in the array to replace the deleting index position
   public void remove(EnrollStudent enrollStudent){}
   public boolean contains(EnrollStudent enrollStudent){}
   public void print() {} //print the array as is without sorting
}
```

    (b) **EnrollStudent class** to keep track of the credits enrolled by a student. You cannot change or add the instance variables, or **-2 points for each violation**. You should override the equals() and toString() methods.

```
public class EnrollStudent {
    private Profile profile;
    private int creditsEnrolled;
}
```

    (c) **TuitionManager class** – refactor RosterManger to process additional line commands. You must keep the **`public void`** `run()` method **under 40 lines** for readability, or you will **lose 3 points**. When your project starts running, it shall display `"Tuition Manager running..."`. It should continuously process the line commands until the "Q" command is entered. Before the software stops running, display `" Tuition Manager terminated."`. Below is a list of new commands and changes needed to make.

    - **LS – load the student roster from an external file**. The file "**studentList.txt**" is provided. Each line of the file is an instance of Student. The data tokens are delimited by commas. The first data token on each line specifies the student status: R is Resident, N is Non-Resident, T is Tristate, and I is International. You can assume the line commands in the text file contains only valid data tokens.
    - **Adding students** with different status to the roster. The add command must check the invalid conditions stated in Project 1. In addition, you must catch the exceptions due to not enough data tokens or invalid data tokens.
        - **AR** – add a Resident student, for example, **`AR John Doe 4/3/2003 CS 29`**
        - **AN** – add a NonResident student, for example, **`AN Leo Jones 4/21/2006 ITI 20`**
        - **AT** – add a Tristate student, for example, **`AT Emma Miller 2/28/2003 CS 15 NY`**
        - **AI** – add an International student, for example,

```
AI Oliver Chang 11/30/2000 BAIT 78        //is not study abroad
AI Oliver Chang 11/30/2000 BAIT 78 false  //is not study abroad
AI Oliver Chang 11/30/2000 BAIT 78 true   //is study abroad
```

    - **E** – enroll a student with the number of credits. For simplicity, we ignore the details of the courses enrolled and only process a total number of credits enrolled. For example, **`E Carl Brown 10/7/2004 24.`** If the student is entered again, the newly entered credits will replace the number of credits previously entered.
    - **D** – drop a student from the enrollment list, for example, **`D Carl Brown 10/7/2004`**

- **S** – **award the scholarship** to a resident student, for example, `S Roy Brooks 9/9/1999 10000`
- **PE** – display the current enrollment list, based on their order in the array.
- **PT** – display the tuition due based on the credits enrolled, with the order in the enrollment array. Only the students who are enrolled will be displayed. The tuition due amount should be displayed with 2 decimal positions, or you **will lose 2 points.** Use the format() method in DecimalFormat class or String class to format the double.
- **SE** – semester end to add the enrolled credits to the credit completed in the roster and print out the students who have already completed 120 credits or more. This command will only be performed once before the Q command.

(d) **RunProject2 class** is a driver class to run your Project 2. The main method will call the **run()** method in the TuitionManager class.

```
public class RunProject2 {
    public static void main(String[] args) {
        new TuitionManager().run();
    }
}
```

13. **JUnit test classes.** Write code to test the following methods.
    - **isValid()** method in the Date class; reuse the 7 test cases (5 invalid, 2 valid) and code from the testbed main() in Project 1.
    - **tuitionDue()** method in the International class; 2 test cases: is or is not study abroad.
    - **add()** methods, in the Roster class. 4 test cases: true or false for adding International and true or false for adding Tristate.
14. **Class Diagram.** Create a class diagram to document your software structure for Project 2. Hand-drawing the diagram is not acceptable. You can follow the links below to create a class diagram online and save it to your device as a picture, which can be included in your submission. You can also use other UML tools if you like. For each rectangle (class), include only the instance variables and public methods.
    - http://draw.io/ → Create New Diagram → UML → Class Diagram
    - https://online.visual-paradigm.com/app/diagrams/ → Create Something → Class Diagram
15. You are required to **generate the Javadoc** after you properly commented your code. Your Javadoc must include the documentations for the constructors, private methods, and public methods of all Java classes. Generate the Javadoc in a single folder and include it in the zip file to be submitted to Canvas. **Please double check your Javadoc** after you generated it by **opening the index.html file** in the folder, and ensure the descriptions are NOT EMPTY. You will lose points if any description in the Javadoc is empty. You will **lose 5 points** for not including the Javadoc.