Due Date

October 11, by 11:59pm

Submission

1. One of the team members is responsible to submit **the zipped project folder** to Canvas. The zipped file MUST include the following subfolders.
   - Source folder **src**, including *.java files [50 points]
     - MUST include all team members' names using the **@author** tag in the comment block on top of EVERY Java class.
   - JUnit Test classes for Date class, Roster class and International class. [20 points]
   - Javadoc folder **doc**, including all the files generated. [5 points]
   - Class diagram, one copy per team. [5 points]
2. The submission button on Canvas will disappear after **October 11, 11:59pm.** You get 0 points if you do not have a submission on Canvas. DO NOT wait until the last minute. DO NOT send the projects to me or the graders through the emails.

Project Description

You will be developing a simple software that maintains a student roster for tuition purpose and processes the tuition. The software shall be able to add and remove students to and from the roster, update student information, calculate tuition dues and process payments. For tuition purpose, students are classified as New Jersey resident and non-New Jersey resident. The international students pay the non-resident tuition and fees, as well as additional fees. Tristate students are students from Connecticut and New York, and they get special discount on the tuition. The table below shows the tuition and fees for a single semester. In addition to the fee schedule below, for the students who enrolled more than 16 credit hours will pay additional tuition for the credit hours exceed 16, using the same per credit hour rate as the parttime students. A student who enrolled at least 12 credit hours is considered as a full time, otherwise will be classified as a parttime student. However, International students must enroll as a full time. For all students, the maximum number of credit hours is 24, minimum is 3.

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

Students may have different tuition remissions as listed below. However, parttime students do not qualify.

- A resident student could get a one-time financial aid 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 who are participating in the study abroad program do not pay the tuition. However, they need to pay the fees. The maximum number of credit hours for the international students in the study abroad program is 12 credit hours.
- Here are some tuition calculation examples.
  - Resident students with 18 credit hours: 12,536 + 3,268 + 404 * (18 – 16) = $16,612
  - 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. This is a group assignment. You MUST work in pair to get the credit for this project.
2. You MUST follow the software development ground rules, or you will lose points for not having a good programming style.
3. Sample input and output are posted as "project2TestCases.txt" and "project2SampleOutput.txt" on Canvas for your reference. The graders will be using the test cases in "project2TestCases.txt" to run your project. Your project should take the sample input in sequence without getting any exception and without terminating abnormally. Your output should match the output in "project2SampleOutput.txt". You will **lose 2 points** for each incorrect output or for each exception.
4. Each Java class must go in a separate file. **-2 points** if you put more than one Java class into a file.
5. Your program MUST handle bad commands. **-2 points** for each bad command not handled.
6. You are NOT allowed to use any Java library classes, except **Scanner**, **StringTokenizer**, **Calendar**, **DecimalFormat** and the necessary **Exception classes**. You will **lose 5 points** for each additional Java library class imported.
7. When you import Java library classes, do not use the * to import all classes in that Java package. **-2 points** for each violation.
8. You MUST create a Class Diagram for this project to document the software structure. The diagram **is worth 5 points.** You will **lose the 5 points** if you submit a hand-drawing diagram.
9. You must create Java classes with the following inheritance relationships. **-5 points** for each class missing, or incorrect inheritance structure. All instance variables must be "private" or **lose 2 points** for each non-private modifier. All static constants must be private, except for the ones used by more than one classes.
   - **Student** class defines the common data items and operations for all student instances; each student must include a profile, including name and major, to uniquely identifies the student in the roster. There are other common data items may need to be defined in this class.
   - **Resident** class extends Student class and includes specific data and operations to resident students.
   - **NonResident** class extends Student class and includes specific data and operations to non-resident students.
   - **TriState** class extends NonResident class and includes specific data and operations to students live in Connecticut or New York.
   - **International** class extends NonResident class and includes specific data and operations to students from overseas.
10. You are not allowed to have redundant code in this project; **-2 points** for each violation below.
    - Unused code: you write code that was never called or used.
    - Duplicate code segments for the same purpose in more than one places/classes.
    - Define common instance variables in each of the subclasses listed in #9, while they should be defined in the superclass.
    - Define specific instance variables in the superclasses listed in #9, while they are not used by all the subclasses.
    - Define some variables as instance variables, but those variables should really be local in the methods within the class.
11. **Polymorphism is required**. An instance of student Roster holds a list of students with various status, including resident, nonresident, international and tristate students. You cannot use the getClass() method to check the class type **listed in #9** above, or you will **lose 10 points**. Tuition calculation will be based on the actual type of the instance in the roster.

12. You must include an overriding **toString()** method method **in ALL classes listed in #9** above. **-2 points** for each toString() method missing. The toString() methods in all subclasses MUST reuse the code in the superclass by calling the toString() method defined in the superclass. **-2 point** for each violation. Override the equals() method where necessary. Always add the tag **@Override** on top of the overriding methods, **-2 point** for each violation.
13. **Student class** must include a do-nothing method listed below. You cannot change the signature of this method, or you will lose **5 points**. All the subclasses MUST override this method and add the **@Override** tag.

```
public class Student {
    public void tuitionDue() {
    }
}
```

14. In addition to the classes listed in #9, you must include the following classes. **-5 points** for each class missing. You CANNOT perform I/O in all classes, EXCEPT the **TuitionManager** class, and the print methods in the **Roster** class. You will **lose 2 points** for each violation. The floating-point numbers must be **displayed with 2 decimal places**, **or -1 point** for each violation. You can use **DecimalFormat.format()** or **String.format()** methods for this purpose.

- **Date** class. Import this class from your Project 1. The class implements the Java Interface Comparable. However, in this project, any years before 2021 are considered as invalid.
- **Profile** class. Define the profile of a student, including name and major. A student profile uniquely identifies a student in the student roster. You cannot add or change the instance variables. You must override the **equals()** and **toString()** method with the **@Override** tag. **-2 points** for each violation.

```
public class Profile {
    private String name;
    private Major major; //5 majors and 2-charater each: CS, IT, BA, EE, ME
        ...
}
```

- **Roster** class. This a growable array list data structure with an initial capacity of 4, and automatically grows (increases) the capacity by 4 whenever it is full. The array list does not decrease in capacity. You must define the two instance variables below. You are not allowed to add additional instance variables**, -2 points** for each violation. You must implement the methods listed below or **lose 2 points each**. The remove() method maintains the relative sequence of objects in the array after the deletion, **-3 points** if this is not done. You can define necessary private and public methods to be used by the client TuitionManager. However, all the public methods defined in this class take either no parameter, or only a single parameter (Student student), **-2 points** for each violation. You cannot use **Arrays.sort()** or **System.arraycopy()** or you will **lose 5 points**.

```
public class Roster {
    private Student[] roster;
    private int size; //keep track of the number of students in the roster
    …
    private int find(Student student) { }
    private void grow() { }
    public boolean add(Student student) { }
    public boolean remove(Student student) { }
    …
}
```

- **TuitionManager** class.
  This is the user interface class that performs Input/Output (read/write.) This class uses an instance of Roster class to process the tuitions. You are not allowed to use Java library class **ArrayLis**t or other Java collection classes, or you will **get 0 points for this project**. This class handles all exceptions and invalid data before it calls the methods in Roster class to complete the associated commands. For example, **InputMismatchException**, **NumberFormatException**, **NoSuchElementException**, invalid credit hours, invalid major code, or invalid state

codes, etc. Whenever there is an exception or invalid data, display a message on the console. Please see the sample output for the messages. You will **lose 2 points** for each exception not caught or invalid data not checked or messages not displayed.

A command line always begins with a command in uppercase letters followed by some data tokens delimited by a comma. The commands are case-sensitive, i.e., the commands with lowercase letters are invalid. In addition, you are required to handle the bad commands not supported. **-2 points** for each invalid command not handled.

You must implement a run() method to handle all commands listed below. This is the only public method. You should try to keep the run() method under 50 lines. You can define additional helper (private) methods and try to keep all the private methods under 40 lines so it's easier for you to manage your code. Each major and the state is a 2-character string; they are not case sensitive, i.e., CS, cS, Cs, or cs are considered as valid for Computer Science major; and ny, Ny, nY, or NY are valid state codes for New York state.

Valid command lines are listed as follows.

- **Adding a student** to the roster; for example,

```
AR,John Doe,CS,20        //adding a resident student with 20 credit hours.
AN,John Doe,CS,20         //adding a nonresident student with 20 credit hours.
AT,John Doe,CS,20,NY     //adding a tristate student from NY.
AI,John Doe,CS,20,false  //adding an international student; false is not study abroad, true is.
```

You could get a command line that is missing some of the data items; in this case, discard the command line and do not process the incomplete command line.

- **Removing a student** from the roster; it is possible that a given student is not in the roster.

```
R,John Doe,it   //removing a resident student from the roster
```

- **Calculate tuition dues** for all students in the roster.

```
C      //calculate tuition due for all students in the roster
```

- **Pay tuition**; for example, the command line below submits a payment of $1,000 on 7/13/2021. Students can submit payments multiple times as long as the tuition due is greater than the payment amount. The system will only save the last payment date. Any payment date is valid only if the year is 2021 and before today's date.

```
T,John Doe,CS,1000,7/13/2021
```

- **Set study abroad status to true** for an international student. If the number of credit hours is greater than 12, set it to 12; set the payment to 0, clear the payment date and recalculate the tuition due.

```
S,John Doe,CS,true.
```

- **Set the financial aid amount** for a resident student. You must validate the amount, which shouldn't be negative or exceeding $10,000.

```
F,John Doe,CS,1000
```

- **Print the roster**, for example,

```
P     //print the roster as is.
PN    //print the roster sorted by student names.
PT    //print only the students who have made payments, ordered by the payment date.
```

- **RunProject2** class. This is the driver class to invoked the **run() method** in TuitionManager and run Project 2.

15. You must follow the instructions in the Software Development Ground Rules and comment your code. 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 in this project.

Generate the Javadoc in a single folder and include it in your project folder to be submitted to Canvas. You will **lose 5 points** for not including the Javadoc.

16. You must create a JUnit test class for the following classes. No need to create the test design document. **Each test method** in the JUnit classes **is worth 5 points**.
    - **Roster** class – write test cases for the **add()**, and **remove()** methods.
    - **International** class – write test cases for the **tuitionDue()** method.
    - **Date** class – write test cases for the **isValid()** method.

Sample Input (test cases)

See project2TestCases.txt

Sample Output

See project2SampleOutput.txt