Lab 7&8 – 30 points x 2

For this two-part lab, we are going to apply the five-step process we learned in class to help us design and develop programs:

1. Gather requirements.
2. Use CRC cards to find classes, responsibilities, and collaborators.
3. Use UML diagrams to record class relationships.
4. Use `javadoc` to document method behavior.
5. Implement your program.

We are going to step through this process for a very small and simple use case. In order to guide you through the process, you can follow through and emulate Section 12.3 in the textbook which does the exact same thing, but for printing an Invoice.

FOLLOW THE STEPS IN ORDER!

Part 1 – Requirements (Lab 7 - 5 Points)

(You only need to design and implement what is necessary to achieve the following set of requirements)

We are going to build a very simple class registration system. A Class has assigned to it 1 Instructor and 1 LectureHall. A Class can have many registered Students. A LectureHall has a capacity, meaning how many students it can hold. Students, Instructors, and AdminStaff are all People belonging to the University and all People have an ID# and an email address. In addition, Students have a major, a list of Classes they are taking, and a single final cumulative semester grade. Instructors and AdminStaff have a salary. An AdminStaff must be able to 1) enroll students in classes, 2) assign a Class a LectureHall, and 3) assign a Class an Instructor. Instructors must have the ability to 1) assign/change each student's single final cumulative semester grade and 2) also have a list of classes they are teaching (that they have been assigned).

In the interest of simplicity, we do not provide a user interface (except for bonus below). Just provide a test program that tests each of the functions/operations applied above.

For the 5 points, you must list

1. All the nouns in these requirements, representing candidate classes.
2. All the verbs.

Save this information as PDF file called Part1.pdf

Part 2 – CRC Cards (Lab 7 - 10 Points)

As we did in class, discover and identify classes and create CRC cards including classes, responsibilities, and collaborators. You can/should also follow Section 12.3.2 from the textbook. Make sure to include all appropriate responsibilities and their collaborators. Also remember that as you discover responsibilities, you may need to add other responsibilities to other classes.

You can create your CRC Cards as tables in word or any other electronic file format you wish. Save it as a PDF called Part2.pdf

Part 3 - UML Class Diagrams (Lab 7 - 15 Points)

After you have discovered classes and their relationships with CRC cards, you should record your findings in a UML diagram. The dependency relationships come from the collaboration column on the CRC cards. Each class depends on the classes with which it collaborates. Ensure you use all appropriate UML relationships that we have taught in class from our selection of 4 relationships. Follow Section 12.3.3 from the textbook as an example.

You can submit your UML Class diagrams in any electronic format you wish. There are many online UML tools you can download for free including ArgoUML, which is fairly popular and easy to use, and https://www.draw.io/ tool. You could also just draw it by hand and scan it in, ensuring it is legible.

Submit your UML diagram as a PDF called Part3.pdf

Part 4 – Javadoc (Lab 8 – 10 Points)

The final step of the design phase is to write the documentation of the discovered classes and methods. Simply write a Java source file for each class, write the method comments for those methods that you have discovered, and leave the bodies of the methods blank. REMEMBER TO USE PACKAGES as we discussed in class! Then, use Javadoc comments to record the behavior of the classes.

Look at Section 12.3.4 for an example. You can also search online for tons of resources like https://www.youtube.com/watch?v=Hx-8BD_Osdw .

Javadoc is in the form of

```
/**
        Overall method comments
        @param param1  Description of parameter 1
        @param paramn  Description of parameter 2
        @return  Description of what is returned
*/
    Public int myMethod (int param1, String param2){
```

…

Proper Javadoc in your code is worth 10 points. We will be looking very closely at it.

Part 5 – Implementation (Lab 8 – 20 Points)

After you have completed the object-oriented design, you are ready to implement the classes. You already have the method parameter variables and comments from the previous step. Now look at the UML diagram to add instance variables. Aggregated classes yield instance variables.

Follow all the implementation rules and standards we have been following thus far. For testing, make an additional tester class program that runs through and tests (Actual versus expected, 4 step process) for the functions/operations designed and implemented above. This includes checking that things were added to the appropriate lists.

Part 6 – Bonus (Lab 7+8 – 10 points)

Create a Graphical User Interface (GUI) for your course registration system. If you need help or review on GUIs, read Chapter 10 in the text book. There are no real requirements here except that the GUI utilizes the classes you developed. Points here will be given based on how many features and how well laid out your GUI is. It is bonus because you will have

to read ahead and learn some skills on your own. So, do not feel pressured to do it, but it will help you with future material to come.

Submission

For Lab 7 (Part 1, 2, and 3), submit the three correctly named PDFs to CANVAS by 1159pm on Monday March 28th .

For Lab 8 (Part 4 and 5 and Bonus), Turn in a .zip file or all your .java files to CANVAS by 1159pm on Monday March 28th.

Grading will be based on conforming to the standards we reviewed in class as well as following the requirements of these two Labs.