Class, Responsibility, Collaboration (CRC) Cards Lab – Ticket Vendor Specifiation

0. Purpose of this activity:

The CSC207 project will be a simulation of a common interaction between client and software developer. The client tries to express in words what they want a program to do. This is called the *specification*. The software developer has to read the specification and decide how to translate them into a design for a program.

Next week, you will be given a specification for a program which you will design, write, and present as your CSC207 project. Today, you will practice this skill by designing a program that you will *not* write.

For the following specification, discuss possible designs with your group and draw a UML Class Diagram for the design you think is best. Why is it best? Does it include all of the features discussed in the specification?

NOTE: Be sure to bring this handout AND your class diagram to your next lecture. We will be using both of them.

1. Specification:

Create a program that keeps track of ticket sales for an auditorium that has 32 rows of seats with a varied number of seats in each row. Each ticket is associated with a seat (row letter and seat number such as A12), a price (low ~ \$10, medium ~ \$30, or high ~ \$50), whether or not the seat is for sale or complementary (for example, sold to the public or given to friends of a performer), name of occupant (who bought the ticket), their e-mail address, and the date, time, and name of the performance so that tickets for each seat can be sold to different people at different times.

It should be possible to get a list of names of all people who bought tickets for a particular date. For a given performance, it should also be possible to print a grid of x's and o's where an x represents a seat that is occupied (that is, it's associated ticket has been sold) and an o represents seats for which no ticket has yet been sold.

Anyone who buys a high-priced ticket automatically joins the Gold Members Club which is an e-mail list. It should be possible to generate an e-mail list of all people in the Gold Members Club, and to delete someone from that e-mail list upon request.

A ticket holder should be able to create and access an account that stores the seats, dates, and names of performances attached to all of the tickets which they have purchased.

2. The Activity

Read the specification again, looking for potential classes (usually nouns) and their responsibilities (usually verbs). In groups discuss possible class structures. You can draw inheritance diagrams on this paper (or scrap paper) for each idea you generate.

Instructions for drawing a class diagram: If class `B` extends class `A`, then show an arrow pointing from `B` to `A`. If a class in the program explicitly inherits from a Java API class (e.g., `ArrayList`), then show this relationship as well in your diagram. To distinguish the Java API classes from the ones in your program, draw a rectangle around each Java API classname and an oval around each class name from your program.

Once you have created your diagram, reread the specification as a group, and discuss whether a program based on your design would be able to handle all of the required details.

Imagine you are a programmer, and you've been handed the design to implement. Would you be able to write a working program according to that design? Where would you need more information? If you also had the specification, would that be enough? Would you need to add more classes, methods, or variables?