COSC 1P03 Lab 8

Change of plans: there's really no need for an independent 'debugging lab', so instead we'll take the time to cover this right! That means this topic will spread across this week *and* next. But, we only have 1 week of labs left. That is true, so lets pair this back a bit and aim to finish up to the CRC cards. That will give you a good start on how to break up a problem, and create an object based solution.

It is recommended to break up into groups, but feel free to simply operate as pairs if it's easier.

Ye olde floral shoppe

Consider the following initial Problem Statement:

> *A flower shop needs an order management system for storing and prioritizing orders for flower arrangements. The shop maintains a catalog of possible arrangements; each of which has one or more types of flowers, housed in some design of vase. Customers enjoy browsing the catalog. New arrangements are occasionally added, but never completely retired. Each order is some number of any combination of said arrangements. The shop does not maintain separate customer records; customer names are attached only to orders, which also have each order's price. Cancellations are common, and to prioritize profits priority is always given to more expensive orders before less expensive ones. The only exception is that 'rush' orders are given ultimate priority. At any time, the shop could need to consolidate a listing of all flower types needed for outstanding orders for inventory management reasons. Such listings include the number of orders requiring any such types. Of course, it's also necessary to dispense the next order to be fulfilled, removing it from the backlog.*

Is this complete? Does it leave any questions unanswered?

Exercise 1 — Refining the problem statement

Your lab demonstrator will give you all 5 or so minutes to decide if there are any improvements or clarifications that could be made to this problem statement. You shouldn't need to *remove* anything, but you might wish to add or clarify.

Before continuing, your lab demonstrator will work with you as a class to decide on the final phrasing.

Exercise 2 — Analysis: major classes and model

As we did in lecture, start by considering nouns. What major classes should we probably have?

Additionally consider *how many* instances of each class you might see. Note: I **really** hope you didn't skip lecture, and understand the significance of this question.

Map out a basic sketch of the direct connections between your classes, including the **multiplicities** you've determined.

Your lab leader will give you a decent amount of time to work this part out.

Your lab leader will now have you discuss this as a group, and will likely draw an analysis model on the board. You don't necessarily have to go with this, but if you disagree it'd be best to say why. Collectively, you can work this out!

Exercise 3 — Identifying Use Cases

Before getting any further, try to work out every *use case*: every basic mode of operation, or fundamental program task that would need to be implemented.

Have a chat amongst yourselves before the lab leader covers it together.

Exercise 4 — Design: Class, Responsibilities, and Collaborators

You'll spend this time working out CRC cards.
Fill them out for *each* of the decided-upon classes.

Remember:

- Fill out responsibilities for knowing first (instance variables)
  - Start with the leftover nouns from the refined problem statement for initial ideas
- Then responsibilities for doing (methods)
- Then collaborators based on the methods (recall that a collaborator is *another* class it depends on for doing *its own* work; not some other class that depends on *it*)

This should take you a decent while, if you're doing it correctly.

Class:	**Collaborators**
Responsibilities	
Knowing:	
Doing:	

Class:	**Collaborators**
Responsibilities	
Knowing:	
Doing:	

Class:	**Collaborators**
Responsibilities	
Knowing:	
Doing:	

Class:	**Collaborators**
Responsibilities	
Knowing:	
Doing:	

Class:	**Collaborators**
Responsibilities	
Knowing:	
Doing:	

Class:	**Collaborators**
Responsibilities	
Knowing:	
Doing:	

Class:	**Collaborators**
Responsibilities	
Knowing:	
Doing:	

Class:	**Collaborators**
Responsibilities	
Knowing:	
Doing:	

What's next on the agenda? Well, let's take stock:

- We should probably sort out our inputs/outputs
    - What goes in?
    - What's produced?
    - What's the formatting/representation?
- We need to work out detailed designs (interfaces)
- We could get at least a start on the implementation
    - If you want to get fancy: since you're (hopefully) paired-up or more, once you have the interfaces worked out, a different person *could* work on each concrete class!

Wherever the 'breaking point' is between the two weeks, it's suggested to *try* working with the same people, if possible.