Assignment #4

Practice with class design and GRASP

CS 4354 Summer II 2016
Instructor: Jill Seaman

Due: at the beginning of class **Monday, 8/1/2016**
Submit a "hard copy" (probably hand-written, optionally computer-generated) only.
Do this assignment with your partner and submit one copy with both names on it.

---

1 A preliminary (incomplete) version of a Library Management System is available on TRACS (lms.zip). In that code, find ONE example of each of the problems listed below AND provide new code to fix the problem.

1. Sharing Mutable References (unintentionally).
2. Not Separating Accessors and Mutators.
3. Side Effects.
4. Violating the Law of Demeter (aka "Sharing Mutable References intentionally").

2 Below is a (very mild) rant from an experienced C++ programmer who is trying to use the DecimalFormat class in a Java program. Use it to make a slightly more formal critique of the DecimalFormat interface, by using the statements below as evidence that the interface is lacking in **three** of these desired qualities: Cohesion, Completeness, Convenience, Clarity, or Consistency. Be convincing.

The mild rant:

1. Ok, I can construct a DecimalFormat object by giving it a pattern string. Then I call format(x) on my object and it returns the value by formatting x according to the object's pattern. I get that, but how do I make the pattern? In the table in the API, I see that if I use a 0 in the pattern, it will be replaced by a digit, but if I use a #, it will be replaced by a "Digit, zero shows as absent". What does that mean? If the corresponding digit is 0 will it be a space? Or just not appear in the output string? Is "0#0" a valid pattern, if so what does it mean?
2. How do I get my numbers to line up in columns in a table? I want to tell the DecimalFormatter to always generate a string of width 10, but I can't find a setWidth method. Why can they just have something like setw(10) in C++?
3. So there are a bunch of setters and getters that affect how the number gets formatted, like setMinimumFractionDigits(int). But if you want to change the pattern, you use a method called applyPattern(String). And if you want to access the current pattern as a string you use a method called toPattern(). Why not just setPattern(Sting) and getPattern()?

3A Consider the checkout resource use case for the **Library Management System:**

Some requirements related to checking out resources:

- Students may check out books for 4 weeks, and faculty for 3 months.
- The library also has other resources that can be checked out, including music CDs, software and videos. These resources may only be checked out for one week at a time.

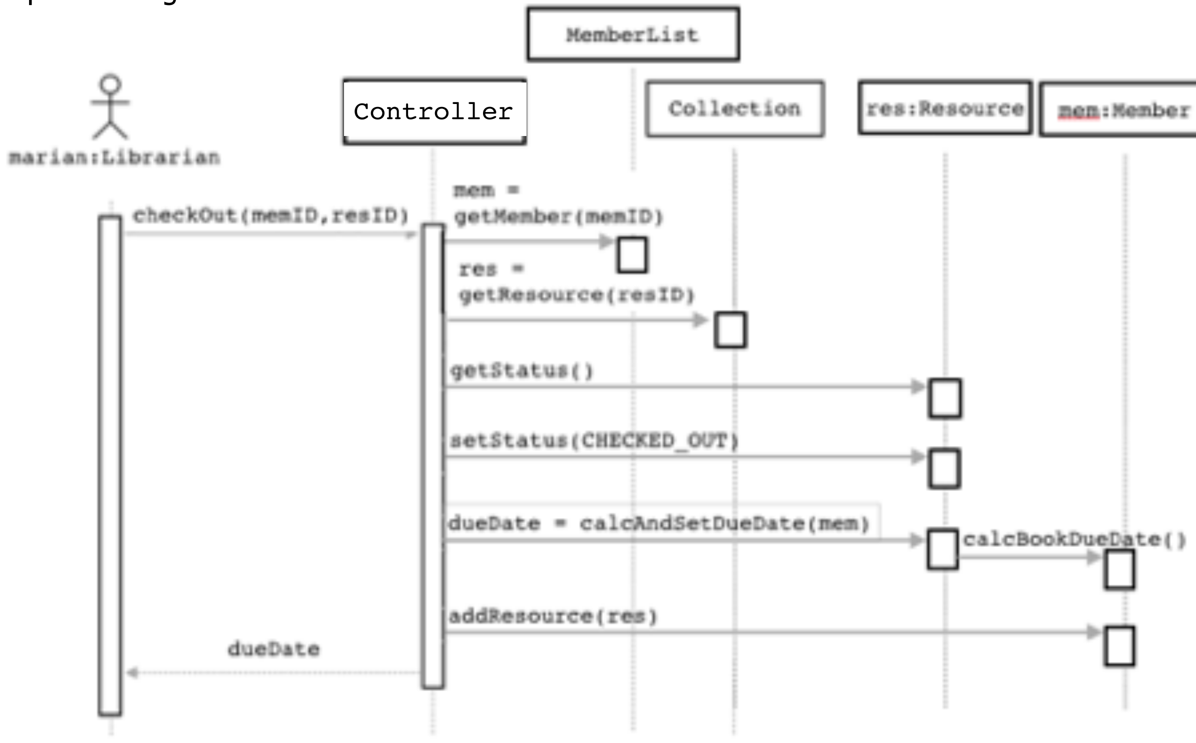
The checkout resource use case:

1. A member provides the librarian with their library ID card and a resource.
2. The librarian enters the member ID# and the resource ID#.
3. The system displays the due date for the resource.

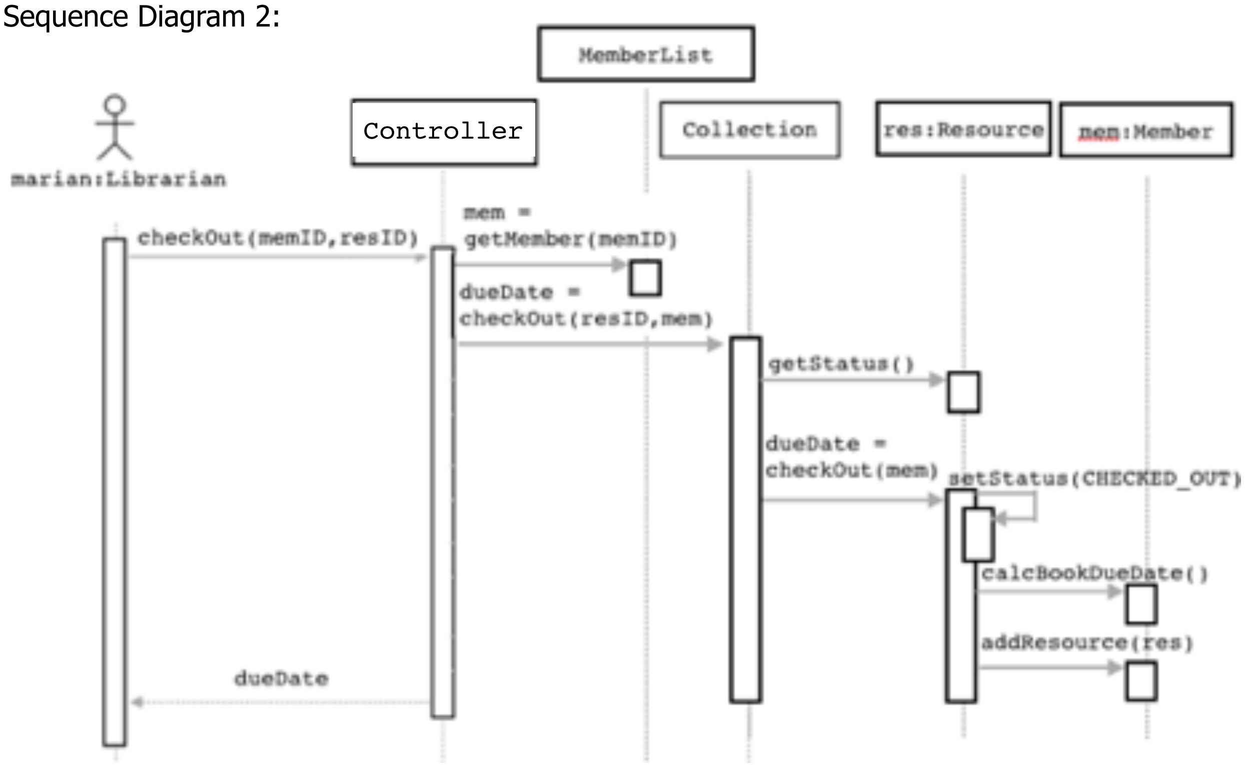
Note that the due date depends on the type of resource and (for a book) the type of member. So the resource and member must collaborate for the book's due date.

Evaluate the two sequence diagrams below that each describe the same interaction: checkout resource. **Which one is better and why?** Mention at least 2 GRASPatterns in your explanation.

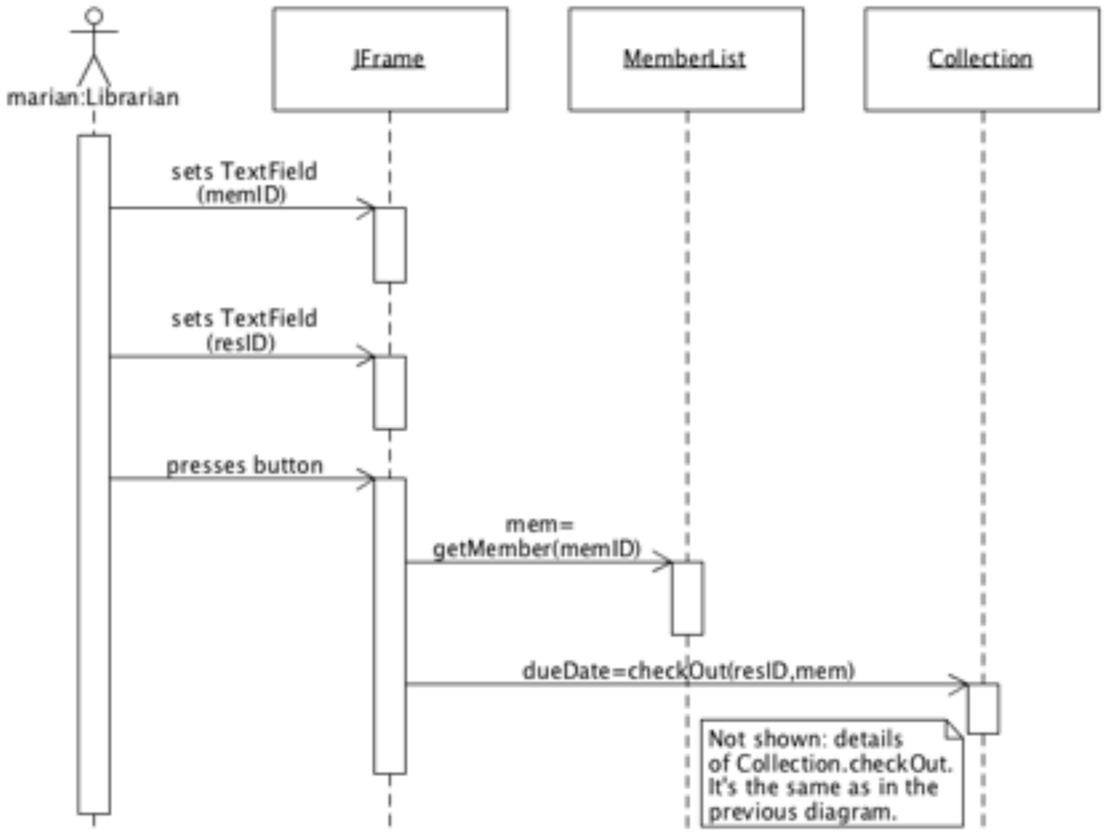
Sequence Diagram 1:



Sequence Diagram 2:



3B Given the following sequence diagram, which models a UI component (JFrame) accessing the Library Management System, apply one of the GRASP patterns to make it better. Indicate which pattern you are using, and include the modified diagram.


marian:Librarian
JFrame
MemberList
Collection
sets TextField (memID)
sets TextField (resID)
presses button
mem= getMember(memID)
dueDate=checkOut(resID,mem)
Not shown: details of Collection.checkOut. It's the same as in the previous diagram.