CPS122 - OBJECT-ORIENTED SOFTWARE DEVELOPMENT

Team Project

Due Dates: See syllabus for due dates for each milestone and quizzes

This project spans much of the semester, to be completed as a series of milestones, each building on its predecessor. You will work on it in teams of two students each, using "pair-programming" principles. (One team of 3 will be allowed if necessary) Each milestone will be due on the date shown in the syllabus.

Since each milestone builds on the previous one(s), we may use some time in class to discuss solutions to a milestone when it is turned in, and student submissions for each milestone may be posted or made available on the course web site so that you can get the benefit of your colleagues' insights as well as your own when doing the next milestone. (Though the work you turn in for each milestone must be produced by your team, you may make use of ideas from any source from previous milestones as a foundation for working on each new milestone.) You will also find it very helpful to refer to the "Addressbook Example" and "ATM Example" systems accessible from the course web page.

Your mission is to design and implement a system to satisfy the requirements listed below. You have some discretion in terms of the details of some aspects of the user interface, etc., so long as your project fulfills the requirements as stated. While you will only implement a subset of the complete set of requirements, your design should be such that implementing the full functionality would be straight-forward.

The final project grade for each individual will be based on team grades for each milestone, plus individual project quizzes to be given at two points in the semester. Also, at the end of the project, within 24 hours of the final due date, *each team member* must fill in and email to the professor an assessment of how the team worked together, and the extent to which the other team member contributed to the team's work. This assessment may be reflected in the assignment of a project grade for each individual team member, which may be adjusted up or down from the grade computed from the milestones and quizzes, based on peer assessment.

Library Checkout System Requirements

The software to be developed is to be used by the manager and desk clerks of a library that checks out various items. These items belong to two categories: books and DVDs. For items checked out frequently, the library will have several copies; in other cases, there may be only one copy of a given item.

As noted above, each item is either a book or a DVD. Each item has a title and a unique call number. Both kinds of items have a brief description (the title of a book; the title of a DVD followed by "(DVD")). A book also has a principal author and a DVD also has a lead actor and a rating.

Each copy is associated with the item it is a copy of, and has a unique copy number within that item (e.g. the first copy of any item is copy 1 of that item, the second is copy 2 etc, Given the call number and copy number for a copy, it is possible to learn whether it is currently checked out and - if so - to whom and when it is due.

The library maintains a record for each patron that shows the patron's name, address and phone number. A new patron must provide this information before checking anything out for the first time, but thereafter the patron need only supply his or her phone number to enable the system to access the stored information. (The system assumes that the phone number uniquely identifies the patron.)

To check out one or more copies to a patron, the clerk enters the patron's phone number (or complete information for a first-time patron), and the call number and copy number for the copy(s) being rented. For each copy, the system will display the description and the date the item will be due (calculated from the current date and the checkout period for the item it is a copy of).

Copies (of any category) being returned can either be handed to a clerk, or they can be placed in a returns slot in the wall of the library if the library is closed. In either case, the clerk must enter the call number and copy number for each copy that has been returned into the system. Of course, the clerk does <u>not</u> need to enter a patron phone number for returns, since the system keeps a record of to whom the copy is checked out - in fact, the patron may not even be present it the copies are left in the return slot at night. If a copy is returned late, a late fine is assessed, to be collected the next time the patron tries to check out something.

If all copies of a given item are checked out, the system will allow a patron to make a reservation for that item, to be filled by the first copy of that item returned. (If more than one patron reserves an item, reservations will be filled on a "first-come, first-served' basis.)

The system must ultimately provide the following functions. Some functions (those marked M) can only be performed by the manager; others (marked C) are normally performed by a clerk. (The manager may also perform clerk functions if he/she chooses to do so.).

1. Core functions to be implemented initially. (C)
   a. Check out one or more copies to a patron.
   b. Record the return of one or more copies.
   c. Renew a checked-out copy of a book, displaying the new due date.
   d. Report the status of a specific copy (description of the item it is a copy of, checkout status [on shelf; checked out - if so, to whom and when the copy is due; on hold - if so for whom].)

2. Manage list of patrons (C and M as noted)
   a. Add a new patron (C)
   b. Modify information stored about a patron (C)
   c. Delete a patron. (M)

3. Manage list of items. (M and C as noted)
   a. Add a new item, recording appropriate information for either a book or a DVD. (M)
   b. Delete an item (M)
   c. Respond to inquiries about a particular item - kind and description, plus whether a copy is available for checkout now. (C)

4. Manage collection of individual copies available for checkout. (M)
    a. Add one or more newly acquired copies of an item.
    b. Delete a lost, damaged, or no longer needed copy.
5. Manage records of outstanding late fines owed by a patron. (C and M as noted)
    a. The system will automatically add a fine (during 1b above) if a patron returns a copy of an item late. If the patron is present it should be possible to collect the charge on the spot - if the copy being returned is late, the clerk should be asked if the patron is present and wishes to pay the charge now. (C)
    b. Indicate that the patron has unpaid fines when the patron attempts to check out or renew something . This is done automatically when a patron's phone number is entered during 1a or 1c above, and the clerk is told to ask the patron whether he/she wishes to pay the fines now. (A patron who chooses not to pay can still check out the copy) (C)
    c. Record the payment of one or more fines owed by a particular patron. The patron has the option of paying all outstanding fines, or just specific one(s). This is available as part of 5a or 5b above, and is also directly available if a patron comes in and asks to pay fines. (C)
    d. Respond to patron inquiries about fines (the item checked-out, when it was due, and when it was returned.) This option is available whenever the clerk attempts to collect outstanding fines (5c above), and is also directly available if a patron comes in and asks about outstanding fines. (C)
    e. Cancel a specific fine. (M)
6. Accept a patron reservation for an item for which all copies are currently checked out, to be filled later on a "first come, first served" basis. Reservations are made for an <u>item</u> (not a specific copy), and will be satisfied by the next copy of that item that is being returned or received. (C)
    a. Enter a reservation for a specific item.
    b. Place a newly-returned or received copy "on hold" for the first patron who has a reservation for it. (Done automatically during 1b above when a copy is returned and there are one or more outstanding reservations for the item. The clerk is told the name and phone number of the patron for which the copy is on hold so as to be able to phone the patron to let him/her know that a copy of the reserved item is in.)
    c. Cancel a reservation. This may be initiated by the patron at any time, or may need to be done if the patron for whom an copy is being placed on hold cannot be contacted or doesn't want the item (in which case it is put on hold for the next patron on the list, or returned to the shelves if there is none.)
7. Produce a patron report for management upon request, showing the following information, with the following reported for each patron: (M)
    a. Name and other basic information (e.g. address, phone)

b. The total number of copies the patron currently has out (whether or not overdue) and the number of these which are overdue.

c. Information about currently overdue copies. There should be one line of information for each copy, including its description and when it was supposed to be returned.

d. Information about fines currently owed. There should be one line of information for each fine, including the description of the item that was returned late, the date on which it was due, the date on which it was actually returned, and the amount of the fine. In addition, if the patron owes one or more fines, the total amount of all fines should be shown.

It should be possible to produce this report for all patrons, or only for those with overdue items or only for those with fines.

8. Produce an item report for management upon request, showing all items, with the following information for each item: (M)

a. Complete description (title and author for book; title, leading actor, and rating for DVD).

b. Total number of copies currently owned (should equal sum of next three items, each of which should also be separately reported)

- Number of copies currently on the shelf
- Number of copies currently rented out
- Number of copies on hold for some patron

c. Number of reservations pending for the item.

The purpose of this report is to help management decide whether to buy more copies of a given item or to sell off some copies when interest in checking out an item declines.
It should be possible to produce this report just for books, just for DVDs, or for both kinds of item

9. Keep information on disk between runs of the program, using some sort of persistence mechanism.

a. When the system is shut down, it would first automatically save all information to disk.

b. It should be possible to manually save all information to disk at any time via a menu option (C).

c. When the system starts up, it should automatically read the saved file of information, if there is one.

10. Provide for system startup (by launching the program) and shutdown (via a quit menu option) (M)

Rules for Checkout and Renewal

Items have a checkout period for their copies, which will be different for different kinds of items.

- Books can be checked out for 28 days.
- DVDs can be checked out for 7 days

Books can be renewed for one additional checkout period (added to the oiginal due date, not the date on which the book is renewed), but DVDs cannot be renewed. A book that is overdue, or that has already been renewed for this patron, cannot be renewed.

Incremental Development

Because developing the whole system as described above could be daunting, the project requirements have been divided into three iterations that complete part of the project, and additional iterations could be defined to finish everything.

1. Iteration #1 - requirements 1, 9, and 10 (all parts of each). (Assume, for this iteration, that a copy is being returned is on time - checking for late returns comes later)

2. Iteration #2 - requirements 2a, 3a, 4a.

3. Iteration #3 - requirements 7a-c [you can ignore requirements dealing with fines since that's not implemented yet] , 8a-b [copies on hold will always be 0, of course].

   (The set of requirements for iteration 1 is quite a bit smaller than those for iterations 2 and 3. The requirements for iteration 1 are the central requirements for the entire system. Spend the time needed to get the overall architecture / design right, and think about how you will implement the requirements for iterations 2 and 3 even though they are not due yet; this will pay off later!)

You can see that quite a number of the requirements will not actually be implemented during the course of the semester. The goal of the project is to give you experience developing software - not to provide a salable system to fund a trip for the professor during the summer :-)

To simplify your life, the professor will furnish you with code for a partially-complete system including most of the GUI code needed for iteration 1. (But you will have to add to the GUI to fulfill all the requirements for this iteration and the next iteration.) The professor will also furnish a class that will make managing date information much easier plus a facility for diddling with the date to facilitate testing. All of this will be made available for you to copy.

Project Milestones

Each iteration will consist of a series of milestones, requiring you to turn in portions of the work for that iteration. Each milestone has a two-part number - e.g. milestone 1-2 is the second milestone for the first iteration. In each case, unless otherwise noted, the milestone pertains only to the requirements for that iteration - but some milestones include requirements that pertain to the whole system, not just the one iteration it is part of. (Be careful to note which is which). The artifacts to be turned in for each milestone are as follows:

(Preliminary - not graded) A list of the names of the team members. Also, read the article “All I Really Need to Know about Pair Programming I Learned in Kindergarten” (`http://collaboration.csc.ncsu.edu/laurie/Papers/Kindergarten.PDF` - also linked from Blackboard site) and discuss it with your partner. Turn in a brief written summary of your discussion (approximately one page).

1-1 • Use-case diagram covering <u>all</u> requirements (not just iteration 1)
- Sequence (flow) of events for each use case for requirements for <u>this</u> iteration. Use the format illustrated in Figures 3.6 and 3.7 on pages 46 and 47 of the text. (You will be given a flow of events for the checkout book use case. as an example)
- Initial functional test cases for the requirements for <u>this</u> iteration.

1-2 • Class diagram showing the entity objects needed for all requirements (not just iteration 1) Since the class responsible for a use case will likely not be an entity class, it therefore does not need to appear in this diagram!
• CRC cards for the entity classes involved in the use cases done for this iteration. (The professor will give you CRC cards for the boundary and controller classes.)

1-3 • JUnit test cases for the unimplemented model classes in the starter code furnished by the professor: `Book, DVD, Copy, Patron,`. (Both class `Item` and a tester for it have been fully implemented for you. Obviously you should not make changes to it!)
• Implemented code for these classes plus `LibraryDatabase` that passes the JUnit tests. (The JUnit tests for `LibraryDatabase` are included in the starter code. [Note: your code will be tested against the professor's JUnit tests, so make sure you test well! Be sure an up-to-date version of the code is on the server. You do not need to turn in any printed code or email anything to the professor for this milestone.]

1-4 # • Tested code for this iteration. The professor will give you sequence diagrams for the use cases for this iteration to help you write your code. You do not need to turn in any printed code. But you must turn in a list of patrons, items, and copies you have added to the database as described below - (on paper or via email). [Be sure an up-to-date version of the code is on the server. **Very important:** Use"Clean and Build" as the very last thing you do after making all your changes to be sure the executable is up to date.]

.(In order to test your code, the class `LibraryDatabase` in the starter code "hardwires" the creation of a few patrons, items, and copies at system startup. You will need to add some more to this, as directed in class. The "status" report for a copy should correctly reflect checkout/return/ renew operations, and invalid operations should be handled correctly. Information should be saved correctly on disk between runs.

2-1 To help you get going on this iteration, you will be given the following:
• Flows of events for each use case for requirements for this iteration.
• A sequence diagram for one of the use cases for this iteration..
You will be responsible for developing the following for this milestone:
• Initial functional test cases for all use cases for this iteration.
• Sequence diagrams for the remaining use cases for which a sequence diagram has not been provided.
(For this milestone you will need to turn in paper copies of your work as well as emailing an electronic copy to the professor as you have done for previous milestones.)

2-2 # • Add code for this iteration to the code you wrote previously. Of course, the code you wrote for iteration 1 should work correctly with this new code! You do not need to turn in any printed code or email anything to the professor for this milestone. [Be sure an up-to-date version of the code is on the server. **Very important:** Use"Clean and Build" as the very last thing you do after making all your changes to be sure the executable is up to date.]

3-1 # For this iteration, you will be working with the code developed by another team. To help you get going on this iteration, you will be given the following:

- Flows of events for each use case for requirements for this iteration.
- A sequence diagram for the one of the use cases for this iteration..

You will be responsible for developing the following for this milestone:

- A sequence diagram for the remaining use case for which a sequence diagram has not been provided.
- Tested code for the use cases for this iteration. You do not need to turn in any printed code for this milestone. [Be sure an up-to-date version of the code is on the server. **Very important:** Use"Clean and Build" as the very last thing you do after making all your changes to be sure the executable is up to date.]
- You will also be required to email to the professor a brief written report on your experience modifying the other team's work. Requirements for this will be given out via Blackboard or in class.

3-2

- Test plan for thoroughly testing the requirements for all three iterations.
- Test report based on executing this plan to test another team's version of the project (as assigned by the professor - will not be your own or the one you worked on for milestone 3-1), You will need to turn in a printed copy of this, and must also email an electronic copy to the professor as you have done for earlier milestones.

(Your grade for this milestone will be based on both the thoroughness of your plan and testing and on the extent to which you discover errors existing in the other team's code but don't "find" spurious errors. The professor may deliberately add one or more intentional errors to the code you test to be sure you have something to find!)

# A lab session will be set aside as a work session for this milestone. However, you should start the work well ahead of time so you can use the lab session as a time to get help with troublesome spots. **Do not wait until the lab session to start work on the milestone!**

Turn In for Each Milestone

Except for the "code' milestones - as noted above - work for each milestone should be submitted in paper form plus electronic form. Documents should be produced in a format that can be read using a freely-available reader (e.g. pdf, .png, or ,html) and diagrams should be produced using appropriate modeling or drawing tools, to the extent possible. Documents produced with a tool whose developer does not provide a freely-available cross-platform reader (e.g. Microsoft Word) are not acceptable, though documents produced with such a tool may be converted to an acceptable form (e.g. .pdf) and submitted that way if you wish. In the case of a diagram produced with astah, print the diagram as a .png file and turn this in and email it to the professor. (Don't email a .astah file!).

Your Java code should be built as a NetBeans project, and up-to-date code for "code" milestones should be left on the server., (Each team will be given a team account on the server for the purposes of this project.)

Grading of Milestones

The artifacts turned in for each milestone will be worth 10% of the total project grade. Each milestone will be graded and returned to you shortly after it is turned in.

Optionally, you may redo portions of most artifacts to improve your final grade. If you choose to do so, your final version will be graded, and your grade for the milestone will be the average of the initial and final grades. (This provision allows you to redo portions of artifacts - not to do them for the first time! If an artifact is totally missing or does not cover all the requirements when this is explicitly stated above, you may not do the missing parts later!) Redone artifacts are due one week after the graded original is returned to you. The option of redoing artifacts will not be available for Milestones 3-1 or 3-2 because they are due so close to the end of the semester.

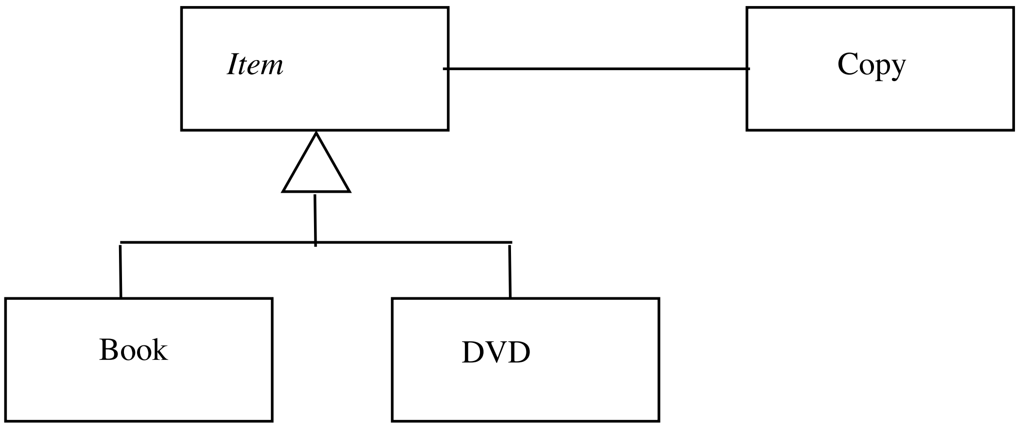
Quizzes

There will be two individual quizzes (each work 10% of the total project grade for each individual), given as shown in the course schedule. The first will focus on understanding code or diagrams given you by the professor; in the second, you will be asked to discuss what changes you would make to the system to actually implement one or more of the future requirements and/or some other added functionality.

Implementation Notes

IMPORTANT: REFER BACK TO THESE AS YOU ARE DOING THE PROJECT. THEY ARE MEANT TO SAVE YOU SOME SIGNIFICANT GRIEF!

1, For reasons that will be discussed more fully in class, you will need separate classes to represent items and copies. (Do not try to use the same class for both purposes). Moreover, since there are two type of item which store distinct information and handle certain operations (e.g. calculating a due date) differently, you will need a class hierarchy for items - an abstract base class and separate concrete subclasses for the two types of item. In the case of copies, though, you don't need a hierarchy - a single class can be used provided each copy object "knows" the item it is a copy of and delegates operations that behave differently for the two types (e.g. calculating a due date) to the item. Thus, this portion of your class structure will look something like this:



2. The professor will furnish a NetBeans starter project. You should study what is there **carefully** before beginning to write your sequence diagrams and your own code. The project folder will contain a README file that you should refer to frequently while doing coding! Knowing and understanding what is furnished for you wlll save you a lot of grief when you are doing your own code. Understanding this code will be the focus of the first of the individual quizzes.

A README file in the starter project will contain further implementation notes for the coding of the three iterations. Be sure to study these **carefully** when you get to the coding milestones.