CS211 - OBJECT-ORIENTED SOFTWARE DEVELOPMENT

Semester Project

Due Dates: See syllabus for due dates for each milestone

This project is a semester-long project 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 may be necessary depending on the total enrollment in the class.) Each milestone will be due on the date shown in the syllabus.

Since each milestone builds on the previous one(s), we mauy 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 "ATM Example" systtem accessible from the course web page.

Your mission - whether or not you decide to accept it[1] - is to design and implement a system to satisfy the requirements listed below. You have considerable discretion in terms of the details of the user interface, etc., so long as your project fulfills the requirements as stated. Note that you are only required to implement a subset of the complete set of requirements, but your design should be such that implementing the full functionality would be straight-forward. Note, too, that you may be given some additions/changes to the requirements late in the semester. (Changes like this are typical of real software development projects.) A well-designed system will make incorporating these changes relatively easy.

The final project grade will be based on grades for each individual milestone. At the end of the project, within 24 hours of the final due date, *each team member* must email the professor with an assessment of how pair programming worked (or did not work) for the team. 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 quiz(zes), based on peer assessments here and in connection with iteration 3.

Video Rental Store System Requirements

The software to be developed is to be used by the manager and clerks of a store that rents both movies and video games. (Some functions will be available only to the manager. The system must provide some suitable mechanism to ensure that only the manager can access such functions.).

To be able to rent from the store, a customer must provide his/her name, address, and phone number. A clerk enters this information into the system, and then gives the customer a card which contains a unique customer ID. The customer must present this card to be able to rent anything.

Movie DVDs are rented for a specified rental period for a specified rental charge, and game disks are rented for a specified rental period for a specified rental charge. If a movie DVD is returned late, it is charged an extra rental charge for each rental period (or fraction thereof) it is late; likewise, if a game disk is returned late it is charged an extra rental charge for each rental period (or fraction thereof) it is late. (The rental periods and prices charged are established by management, and can be different for movies than for games.)

[1] If you don't get this touch of humor, go to your favorite (real) video store and rent "Mission Impossible" :-) .

DVDs and game disks available for rental are displayed in the store in their boxes. Each has a unique ID number. To rent one or more items, the customer brings it/them to a clerk, who enters the customer's ID number from the customer's card, and the DVD/disk ID number(s) from the box(es). When all have been entered, the system will calculate the total charge owed and the clerk will collect it from the customer. (A future improvement may use a bar code scanner to scan customer cards and DVD/disk ID numbers, but for now the system depends on the clerk typing the appropriate ID numbers.)

DVDs and disks being returned can either be handed to a clerk, or they can be placed in a returns box in the store, or they can be inserted into a returns slot in the wall of the store if the store is closed. In any case, a clerk must enter the ID number of each DVD/disk that has been returned into the system. Of course, the clerk does not need to enter a customer id for returns - in fact, the customer may not even be present it the items are left in the return slot at night.

The system must load data from disk at startup, and automatically save data to disk at shutdown.

The system must ultimately provide the following functions. The ones marked M are performed by the manager; those marked C are normally performed by the clerks. (Note that the manager may also perform clerk functions if he/she chooses to do so.).

1. Manage rental and return of movie DVDs and game disks. (C)
   a. Rent one or more DVDs and/or disks to a customer.
   b. Record the return of one or more DVDs and/or disks.
   c. Report the status of a specific DVD or disk (title, checkout status [on shelf; rented - if so, to whom and when due; on hold - if so for whom].)

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

3. Manage list of titles of movies and titles of games available for rental (M and C as noted)
   a. Add a new title (M)
   b. Delete a title (M)
   c. Respond to inquiries about a particular title - general information about it, plus whether a copy is available for rental now. (C)

4. Manage inventory of individual DVDs / disks available for rental. (M)
   a. Add one or more newly acquired DVDs or disks.
   b. Delete a lost, damaged, or no longer needed DVD or disk.

5. Manage records of outstanding late charges owed by a customer. (C and M as noted)
   a. Add a late charge if a customer returns a DVD or disk late. (The customer may drop off returns without interacting with a clerk, so late charges incurred may have to be recorded now and collected the next time the customer comes in for a rental. However, if the customer is present it should be possible to collect the charge on the spot.) The charge is computed and added automatically when the DVD or disk is returned (during 1b above) and the clerk is asked if the customer is present and wishes to pay the charge now. (C)

b. Indicate that the customer has unpaid late charges when the customer attempts to rent an item. This is done automatically when a customer's id is entered during 1a above, and the clerk is told to ask the customer whether he/she wishes to pay the late charges now. (A customer who chooses not to pay can still rent the item) (C)

c. Record the payment of one or more late charges owed by a particular customer. The customer has the option of paying all outstanding late charges, or just specific one(s). This is available as part of 5a and 5b above, and is also directly available if a customer comes in and asks to pay late charges. (C)

d. Respond to customer inquiries about late charges (the title rented, when it was due, and when it was returned.) This option is available whenever the clerk attempts to collect outstanding late charges (5c above), and is also directly available if a customer comes in and asks about outstanding late charges. (C)

e. Cancel a specific late charge. (M)

6. Accept a customer reservation for a title for which all copies are currently rented, to be filled later on a "first come, first served" basis. (C)

   a. Enter a reservation for a specific title.

   b. Place a newly-returned item "on hold" for the first customer who has a reservation for it. (Done automatically during 1b above when a DVD or disk is returned and there are one or more outstanding reservations for the title. The clerk is told the name and phone number of the customer for which the item is on hold so as to be able to phone the customer to let him/her know that the item is in.)

   c. Cancel a reservation. This may be initiated by the customer at any time, or may need to be done if the customer for whom an item 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 customer on the list, or returned to general stock if there is none.)

7. Produce a customer report for management upon request, showing the following information, with the following reported for each customer: (M)

   a. Name and other basic information (e.g. address, phone)

   b. Total number of DVDs/disks the customer currently has out

   c. Information about currently overdue DVDs or disks. There should be one line of information for each item, including its title and when it was supposed to be due.

   d. Information about fines currently owed. There should be one line of information for each fine, including the title 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 customer owes one or more fines, the total amount of all fines should be shown.

   The manager must be able to choose whether to produce such a report for

   - All customers.
   - Only customers that have one or more overdue items
   - Only for customers that owe one or more late fees.

8. Produce a title report for management upon request, showing all titles, with the following information for each title: (M)

   a. Name and other basic information.

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 rented out
- Number of copies on hold for some customer
- Number currently in stock

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 title or to sell off some copies when interest in renting a title declines.

9. Miscellaneous. These changes, once made, apply to subsequent rentals, but not to ones that are currently outstanding. (M)
   a. Set the rental rate for a particular class of item (movie or video game.)
   b. Set the rental period for a particular class of item (movie or video game)

10. Manually save all information to disk at any time. (This is in addition to the automatic save to disk which occurs at shutdown) (C)

Of course, information about customers, inventory, rentals, late charges, and reservations will need to be preserved between program runs through some sort of persistence mechanism.

Incremental Development

Because developing the whole system as described above could be daunting, the project requirements have been divided into several iterations:

1. Iteration #1 - requirement 1 (all parts); ability to save and re-load data between program runs (automatic at program startup/shutdown plus manual via "Save" menu option - requirement 10.) (Assume, for this iteration, that an item being returned is on time - checking for late items comes later)

2. Iteration #2 - requirements 2a, 3a, 4a, 7a-c, 8a-b.

3. Iteration #3 - Surprise! You will design and implement some subset of the remaining requirements, and/or new or changed requirements, to be specified later in the semester. You will be assigned to work with the software produced by some other team for this iteration, and will turn in both a modified system and a brief written discussion of how the other team's system design did/did not facilitate making the required changes.

Note that the set of requirements for iteration 1 is quite a bit smaller than that for iteration 2. The requirements for this iteration are the central requirements for the entire system. Spend the time needed to get the overall architecture / design right ; this will pay off on subsequent iterations!

Note 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 to Bermuda during the break!

To simplify your life, the professor will furnish you with code for a partially-complete GUI. The code you will be given will cover most of the requirements for Iteration 1 of the project; but you will have to implement additional portions for this and subsequent iterations. 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 available for you to copy from the server.

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 all requirements (not just iteration 1)
- Flow of events for each use case for requirements for this iteration.
- Test cases for the requirements for this iteration.

1-2 *
- Class diagram showing the entity objects needed for all requirements (not just iteration 1) Note that the class responsible for a use case will likely not be an entity class, and therefore does not need to appear in this diagram!
- CRC cards for the classes involved in the use cases done for this iteration. (This will likely involve some classes that did not appear as entity classes in your class diagram)

1-3 *
- Sequence diagrams for the use cases you did for this iteration.
- Overall design for the GUI (expressed as a state diagram) covering all requirements - (partially reverse engineered from GUI code furnished by the professor, with additions for the remaining requirements)

1-4
- Tested code for this iteration. Note that, in order to test your code, it will be necessary to "hardwire" the creation of a few customers, titles, and DVDs at system startup. The "status" report for an item should correctly reflect rental / return operations, and invalid operations should be handled correctly. Information should be saved correctly on disk between runs. [Drop the complete project in the drop box. Turn in a printed of classes (or portions of classes) that you wrote or changed significantly.]

2-1 *
- Flows of events for each use case for requirements for this iteration.
- Test cases for the requirements for this iteration.
- Sequence diagrams for the use cases you did for this iteration.

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!

2-3
- Test plan for thoroughly testing the requirements for both iterations 1 and 2.
- Test report based on executing this plan to test another team's version of the project (as assigned by the professor)

(Your grade for this milestone will be based on both the thoroughness of your plan and testing and on the correctness of your code as revealed by another team's testing of it.)

3-1 (To be announced)

3-2
- Add tested code for this iteration to the code written by the team whose code you are modifying. Of course, the modified code should still work correctly!
- Test report showing the results of testing your modifications.

* You may work together with other team(s) on this milestone if you wish, though each team must produce and turn in its own artifacts.

Turn In for Each Milestone

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 or html,) and diagrams should be produced using appropriate modeling or drawing tools, to the extent possible. Note that documents produced with a tool that whose developer does not provide a freely-available reader (e.g. Microsoft Word) are not acceptable, though documents produced with such a tool may be converted to an acceptable form and submitted that way if you wish.

Your Java code should be built as a NetBeans project, and code for "code" milestones should be placed in the Drop Box on the server.

In addition, you should keep an electronic archive of the source form of all documents. This will be given to another team for Iteration 3.

Grading of Milestones

The artifacts turned in for each milestone will be worth 10% of the total project grade. The milestone will be graded and returned to you when 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. (Note: 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 2-3, 3-1, or 3-2 because they are due so close to the end of the semester.

Working with Other Teams

For certain of the milestones, you may work together with other team(s) if you wish. (These are indicated in the list of milestones above.) However, each team must produce and turn in its own artifacts.

Quizzes

I may choose to give a quiz in class on any due date for any milestone, focussing on what you needed to know/understand for that milestone. The quiz grade will impact the individual grades for the milestone.

I will give a more comprehensive quiz, focussing on the overall project, on the due date for milestone 2-3. This quiz will ask you 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. The project folder contains a README file that you should study **carefully** before starting work on coding!

2. Use Java serialization to provide persistence for data that needs to be saved between program runs. There is no need to provide for multiple data files - therefore, you can "hardwire" a file name into the program.

   a. When the program starts up, it should look for a file with this name and read it in to initialize the stored data. If no file with the specified name exists, it should create a new, empty internal database.

   b. When the manager quits the program, it should first write its internal database to the a file with the specified name.

   c. There must also be a Save option somewhere to allow the manager to save the state of the internal database to the file at any time - it would not do to risk losing an entire day's activities if there is a system crash a few minutes before closing!

   Thus, you will need the traditional File Save and Quit options, but not New, Open, or Save As.

3. The requirements stipulate that the system must restrict some functions to managers. A few years ago, several teams spent an inordinate amount of time trying to build a good password facility - which is really not the point of this exercise. Therefore, when you get to iteration 2 (which is the first place these requirements show up) handle this as follows:

   a. Maintain a boolean variable somewhere that indicates whether the current user is a manager or a clerk. When displaying the GUI, disable (or don't display at all) those options that pertain only to managers if the variable indicates that a clerk is using the system.

   b Provide a menu option to log in as manager if a clerk is currently using the system, or a one to log out as manager if a manager is currently using the system. (Perhaps the same option with different labels). Hardwire a manager password into the program. If log in as manager is chosen, pop up a dialog box asking the user to enter a password (which should echo as bullet characters (•)), and check the password entered against the hardwired value.

   c. When a successful login or logout is done, change the state of the boolean variable appropriately. Have a method that updates the displayed GUI to the current state.

   d. Note that if a menu choice or button is disabled (grayed out), the user can't choose it, so you can assume that if a manager choice is made then the user must be a manager.

   e. Of course, be sure to give me the manager password when you turn the software in for testing!

4. The requirements stipulate that the rental period and rental rate should be set by management, but the functionality to actually do this is in a future iteration. Build the initial system using the following rental periods and rates, but design it in such a way that the system could later be modified to allow the manager to change these:

   Movie DVDs: rental period one day, rental rate $3.00

   Game disks: rental period two days, rental rate $2.50