Travis Allan
CSCE1040
Homework 4
7/14/2014

Library Management System

1. Description of All Data Elements Used

The data elements of this program will include the four classes, as well as their necessary methods and variables, and the elements used in the main of the program.

The first class, “Book”, will represent a single book. Its variables include:

Acquisition ID: keeps track of the book and is the main defining factor

Title: name of the book, isn’t the main defining factor because there could be multiple copies

Author: writer of the book, it isn’t very useful for now, but the program could be expanded to use this in searching for books

Status: state variable declaring whether the book is “in”, “checked out”, “lost”, or “overdue”

CheckTime: tells how long the book can be checked out, and can be used to determine if a book is overdue

PatronPointer: pointer to whichever patron has currently checked out this book

It’s methods are:

EditBook: this method allows the user to update the description of book

The second class, “Patron”, will represent a single user. Its variables include:

Name: holds the users name and can be used to look up information about the user

Patron ID: patron's unique ID, the system will use this as the defining factor of each patron

Address: place of residence, will be used in some print statements, and possibly user inquiry

Phone: phone number, will be used in some print statements, and possibly user inquiry

Email: online mail, will be used in some print statements, and possibly user inquiry

The class's methods are:

editPatron: this allows unique information about the patron to be updated

The third class, "BookList", holds **n** many instances of book, where **n** is the number of books in the library. its variables include:

BookDB: this is the database that will hold all instances of the class book

BookCount: holds the number of books that have been added to the system

The class methods are:

AddBook: this adds a new instance of book to the database

RemoveBook: this will delete or nullify an instance of book from the system

StatusUpdate: changes the status of the book

PrintBook: prints a list of all book titles

PrintOverdue: prints out a list of overdue books along with their patrons and appropriate fines

FindBook: searches the book database for a certain instance of book

SortBook: sorts the books according to acquisition ID

Read DB: loads the database from a file

SaveDB: saves the database to a file

The fourth class, “PatronList”, will hold all of the instances of “Patron”. Its variables are:

PatronDB: database that holds all instances of patron

PatronCount: holds the number of patrons

Its class methods are:

AddPatron: adds a new instance of “Patron”

RemovePatron: deletes or nulifies an instance of “Patron”

PrintPatron: prints a list of all patron names

PrintPatron: prints a list of all patron names who owe fines, as well as their amount due

SortPatron: sorts the database according to last name

ReadDB: loads the database from a file

SaveDB: saves the database to a file

Next, we have the main file, it will have at least these variables:

command: user input that tells the program what to do

tempname: holds the name of a patron or book until it can be assigned

tempID: holds the ID of a person or book until it can be defined

(instance of BookList): instance created in main, so main can use the class

(instance of PatronList): instance created in main, so main can use the class

2. Class Diagram of the Entities of the System

Book - BookList

:BookList
BookDB BookCount AddBook
AddBook RemoveBook StatusUpdate PrintBook PrintOverdue FindBook SortBook ReadDB SaveDB

0…1

1…* Composition Relation

:Book
AcquisitionID Title Author Status CheckTime PatronPointer
EditBook

Patron - PatronList

:PatronList
PatronDB PatronCount
AddPatron RemovePatron PrintPatron PrintFine SortPatron ReadDB SaveDB

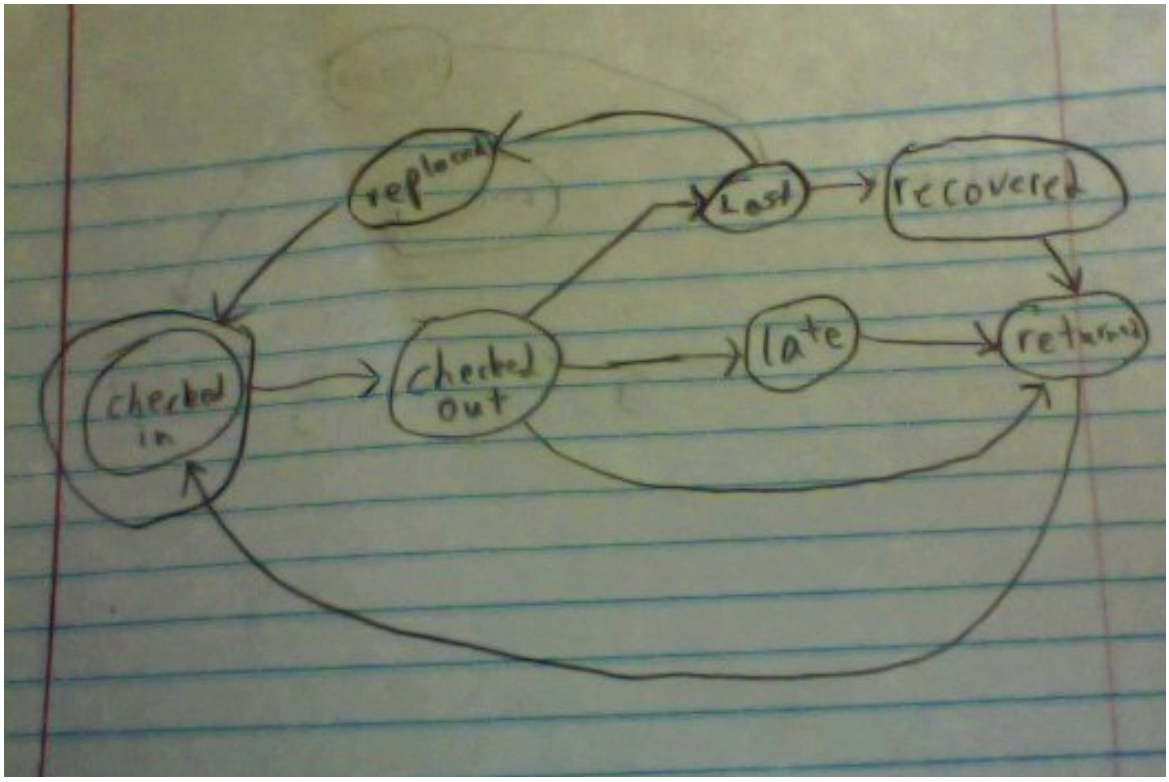
0…1

1…* **Composition Relation**

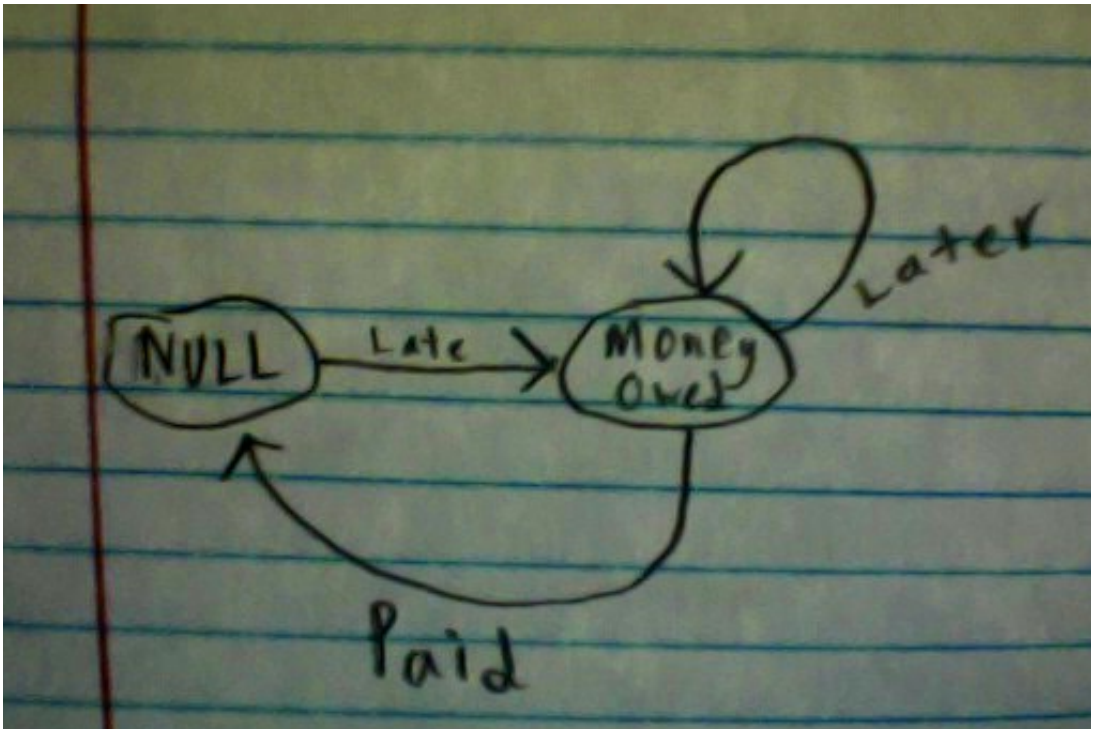
:Patron
Name Address ID Phone Email Balance
EditPatron

3. State Diagram For Relevant State Variables

Book State Diagram



Balance State Diagram



4. Discussion of Approach

I would say that I tried to make this an object oriented system where the classes have all of the power, and the main file just hands off the information. I hope I did well, but here is a basic idea of what a run throughout this program would do.

First, the program would try to the database from a file, if one isn't found then it will start from nothing. Then, it would print out a list of options for the user to choose from. Based on the option chosen it will call one of the class methods to run their course. After that, the program should repeat the prompt until the user is done getting/ giving information to and from the program. Once the user is done and wishes to close the program, it should automatically save all of the new or updated data into a save file. Then, finally the program should close.

5. Questions & Issues With the System

The theoretical person that asked us to build this program didn't specify how many books he would allow each person to check out. So, by default, it is infinite. With that, unless there's a library rule that says the person can't check out all of the books, a person could take all of the books for themselves.

Also, unless we theoretically have a follow up meeting with this person, we won't be able to implement an adequate time to fine function for late returns.

6. Future Implementation Discussion

For future implementation, I really think we should go ahead with the user self serve system. As long as the above stated problems are resolved, we shouldn't(hopefully) have any problems with it. Also, with as much information as we have implemented for the books, it would be a shame to not make some system add on that searches and recommends books for the user.

Also, with the database save to file and load from file methods, this program should be able to support multiple computers. They would just need to be updated after every transaction.