Alan Jin, Jerry Su, Nelson Szeto, Jason Wang

Architecture

Our project code will be designed around the login panel function. From the login panel, users can create or login into precreated/created accounts from the student.txt file, into a student vector. Every time a person registers, its account username and password will be recorded in their respective txt and added into their respective user vector, using the overloaded >> operator of the student class.This will be repeated for the Librarian.txt and Teacher.txt. As for logging in, the program will search through the contents of the user's vector, and try to match it with the user's input for username and password. On the loginPanel function, users also have the choice to exit the program, which will prompt an exit(1) function that exits the program. Since the parameters of the loginFunction are using their addresses, the contents are changing permanently rather than temporarily. Upon startup, the books from the books.txt will also be recorded(method is explained in the comment section of code) into a books vector.In our case we will be working with a vector of the Student/Teacher/Librarian class type and Book class type, which holds a Student type variable in each position, meaning that at each position, they have unique private values for username, password, borrowed books, borrow period, and etc. **We also utilized templates to exceptionally decrease the amount of lines we may need when dealing with teacher, student, and librarian functions.**

After the loginPanel, when the user successfully logs into an account, he/she is then shown the selection screen. Depending whether they're a librarian, teacher or student, they will be given their own selection screen. He/she is only allowed to alter the contents of the logged in account, because when the login is successful, the variable studentNumber, records the position of the logged in account in the student vector, and only changes that position of the array's contents based on the user's selection, unless he/she logs out and logs into another account. Logging out can easily be explained because it will just break out of the switch statement of the loginPanel, resulting in the while function to be run again, prompting the user to login, create an account, or exit the program. For each selection, whether it's borrowing a book, reserving a book, or etc, the input of the user will invoke a switch statement, which will then invoke the selection's corresponding functions. For example, if the student chose to search up a book, then they will invoke the search function in the switch cases and continue from there.Then the search function would do it's searching through the book vector array, to check if the title the user is searching for is part of the library. In general, each selection prompts a separate, relative function to make it easier for us, the coders, to debug, rather than going through every action done in the code, and figuring out which variable is being misused. This method will also eliminate the possibility of us altering a specific value to hold another value by accident, which will take a long process to debug if we do not know where the problem stems from. In essence, if a problem occurs, we can just refer to its corresponding function rather than the entire code.

Specification of Classes

The user class only contains the getter and setter functions for the username and password. This class is included in reader class.

The reader class inherits the user class and contains reserve, borrowPeriod, and books array. They hold the books currently being reserved, the duration that each book is borrowed for, and the books that are currently being borrowed. There are also get and set functions to access the reserve, borrowPeriod, and books array. We also have overloaded functions that can retrieve account information of readers as well as output the information. The reserve array gets multiple functions for adding and removing elements in the array due to needing to be used in different manners (whether its a reader manually adding or removing reserves or a reserve being removed due to the book finally being borrowed by said reader).

The student and teacher class inherits the reader class and are very similar. They contain their basic constructors and overloaded >> and << operators to set user login data and output user information, including what books they are currently borrowing and have reserved. The main difference between the teacher and student class is that the teacher is allowed to borrow more books than the student.

The librarian class is also similar to the student and teacher class. However, in the library management system, the librarian is able to access a different set of functions than other users once logged in.

In the Book class, we created a constructor which included: ISBN, index, category, favor, number of copies, title, and author. Each variable within the constructor can be changed and called using the get and set functions. The books would be put into a vector array from a txt file using an overloaded operator and could be output using its respective overloaded operator as well. The category is kept as an integer rather than a string, due to general neatness and so that the file could be made more concise. The category can be converted from an int variable to its respective string by using the categoryCheck function. Each book also has a vector array that holds its copies. We can add and delete copies of the book using the functions addCopy and deleCopy.

The copy class holds information that was previously held in the book class, such as reader, reserver, ID, startDate, expireDate, and reserveDate. This class only has get and set functions for its variables, since the rest of the information is stored in the book class.

User Class Functions

There are set and get functions for user and pass.

printPassStars

Prints password as ***.

Teacher/Student Class Functions

The teacher and student functions are the same besides the fact that they will create their own respective teacher or student accounts. A teacher can borrow 8 books while students can only borrow 5. They can both reserve 5 books. friend istream& operator >> (istream& input, Student& student); creates the student accounts and if you want the teacher one it's just Teacher& teacher at the end. This is the same for the ofstream part. Student& student for students and Teacher& teacher for teachers.

Reader Class Functions

Overloaded >> operator

Reads the username and password from the class then alters the reader object in the parameter to the obtained username and password.

Overloaded << operator

Prints the information of the reader. It gives the username but not password, all borrowed books, and all reserved books for the reader.

bool CheckBooks

We have the (int accType) as our parameter, which tests to see if the acctype is equal to 2, and if it is it checks the 5 books and if any is empty it will return a false, otherwise it will return true.

get/set BorrowPeriod and Books

It gets and sets BorrowPeriod and Books.

There is a getter function for reserve.

addReserve

It has parameters (string title, int acctype). Checks to see if the acctype is 2, if it is it will go through the array, up to 5, to check for empty slots, if it is it'll set that position as the title of the book. It then sets a flag to true and exits the loop. Does the same thing if acctype is 3, but it checks up to 8 slots instead of 5.

isReserveFull

A for loop to go through the reserve array, if any of the slots are empty, it will return false, otherwise return true.

removeReserve

Checks if the borrowed books and reserve array are not empty as well as checking if a borrowed book is also in the student's reserve array. If the student borrows a book reserved by himself/herself, then the book gets removed from the reserve array and prints out "Reserve successfully removed due to being borrowed by you" while setting the original position of where the book belonged in the reserve back to "", so next time, when we are setting a new book in reserve in that particular position, then it will read it as empty, and insert new reserved book title

setReserve

Checks if the student can reserve any more books. If able to, the book is reserved and placed into the reserve array. Otherwise, an error is printed.

checkReserve

Iterates through the reserve array and checks if the book is reserved already.

printReserve

Prints the reserved books of the student logged in

checkOverdue

Checks if the student has an overdue book, and returns true if the person does have an overdue book, and false otherwise.

Librarian Class Functions

Our librarian are unique as in they are another type of subclass from Reader.
They are given a different selection screen where they can use to manipulate library data
They can

- **Delete Copies of books**
  - They can delete specific copies of books
    - If it's the last copy, the book itself is deleted because it does not have any copies left
- **Add Copies of books**
  - They can add new books or more copies of preexisting books
- **Search User**s
  - They are able to obtain a student/teacher type account information such as borrowed books, reserved books,and username/pass
- **Add Users**
  - They are able to add more student/teacher type accounts
- **Delete Users**
  - They are able to get rid of student/teacher type accounts

Copy Class Functions

The copy class has a **getters/setters** for:
ID, Avail, StartDate, ExpireDate, ReserveDate, Reader, Reserver.
It also has a getter function for Title but no setter function.

BOOK CLASS FUNCTIONS

There are sets and gets for every variable besides the expiration date (which lacks a set due to being set by the starting date).

Default Constructor

Assigns default values to Book class without parameter

Parameterized constructor

Assigns the values in the parameter to private values of the new Book object

Overloaded >> operator

Reads the variables for the book class, inserts the variables into a book constructor and returns the book.

Overloaded << operator

Gets the ISBN, ID, category, title, and author and returns those to be printed. (Some of the variables are left out in this one as they are either irrelevant or not wanted for privacy purposes).

copyVectorSize

Returns size of copies vector of a book object

CategoryCheck

Its only parameter is the integer form of the category, and returns a string corresponding to the integer parameter using a switch case.

addCopy

The addCopy function takes an ISBN and if a book exists with that ISBN, a new copy is added to its copies array. If there is no existing book with that ISBN, the user then inputs category, title, index, and author to create it. The created book is added to the books array and 1 copy is created for it using addCopy.

deleCopy

deleCopy first checks if the book is being borrowed. If not, it then checks if this is the last copy of the book. If there is only 1 copy of the book, the book is erased from the book array. If there are more than 1 copies, the copy of the book with the specified ID is removed.

setCopyReserve

It has (string name) as the parameter, and with this name, we search through the copies vector to see an empty slot, and if it is empty we will input the book's name.

cancelReserve

It also has (string name) as the parameter, and it searches through the copies vector until it finds when the name at a position matches our input name. When it is equal it sets it to blank, cancelling the reservation.

checkAvailableCopies

A boolean to search through the copies' position for anyone reserving the book. If it is empty it will return true, if not it will return false.

Main Class Functions

loginStudent

The parameters for the loginStudent function are (vector<Student>& s, istream& in, bool& login, int& studentNumber). The vector<Student> s, allows the function to call the student vector, which has a different student account at every position. Istream&in allows the function to read inputs from the user and assign it to something, in our case it reads user and pass and sets it to strings. Bool& login is there so we can have a login value that is either true or false and it's default state is false, when it goes through the for loop to check for a student that has the same user and pass as the ones entered, it will set login to true. After going through the for loop, we need int& studentNumber because it will record the position in the vector where the student has logged in, so we can change its content when needed.

loadBooks

The only parameter for loadBooks function is (vector<Book>& books). This function just puts the book information from a text file into the specified vector array. Uses the overloaded operator to input. We also create copies of the book equal to the numCopy specified in the text file using the addCopy function.

checkLogin

The only parameter for checkLogin is bool login. If the bool is true, they are logged in. If the bool is false, we print out an error message.

createStudentAccount

The parameters for createStudentAccount are (vector<Student>& s, istream& in). We keep the information for student logins in a txt file. The function asks the user to input a new username and password, then this is printed into the txt file. The student is also added to the vector array of students. The function also makes sure that the account is unique before being created.

createTeacherAccount

The parameters for createTeacherAccount are (vector<Teacher>& s, istream& in). We keep the information for teacher logins in a txt file. The function asks the user to input a new username and password, then this is printed into the txt file. The student is also added to the vector array of students. The function also makes sure that the account is unique before being created.

createLibrarianAccount

The parameters for createLibrarianAccount are (vector<Librarian>& s, istream& in). We keep the information for librarian logins in a txt file. The function asks the user to input a new username and password, then this is printed into the txt file. The student is also added to the vector array of students. The function also makes sure that the account is unique before being created.

fileAccountStudent

The parameter is just vector<Student> s. It reads the data from a file and it uses push_back to add the accounts into the student vector.

fileAccountTeacher

The parameter is just vector<Teacher> s. It reads the data from a file and it uses push_back to add the accounts into the teacher vector.

fileAccountLibrarian

The parameter is just vector<Librarian> s. It reads the data from a file and it uses push_back to add the accounts into the librarian vector.

searchUsers

This function has parameters vector<user> u, string name, int accType. The user vector is iterated through and if a user name matches the inputted string, the system outputs the type of user they are (student, teacher, or librarian) using the accType.

borrowBook

This function has the parameters of vector<Book>& b, vector<reader>&s, int& useNumber, and int accType. The user types in the ISBN into the console after which the program searches through the book list for a match and then checks for reservations,already borrowed books. After that, it then links a specific copy of the copy to the user by keeping the user's name in the copy and putting the book's title in the user's borrowed book array

returnBook

This function has the parameters of vector<Book>& b, vector<reader>& s, int& userNumber, int accType. This asks the user for the title of the book they wish to return, if the title is found, it replaces the name in both the user's array and the book array.

deleteUsers

This function has parameters vector<user>& u, string name, int accType, int& userNumber. The user inputs the name of the user to be deleted, then the user vector is iterated through until it finds a user with that name. If the user has borrowed books, they can't be deleted. The user is then deleted from the vector and their name is removed from the reservee list of any book they were on.

selectionPanel

The parameters are istream& in, vector<Student>& s, vector<Book>& b, int& studentNumber. As other functions, istream& in is used to read and assign inputs from the user. The student vector and the int, studentNumber, is used to access each student's accounts and its contents. The book vector holds all the books and its current state. This is the base of all processes in this program and it will give 8 different options with a switch that determine what function to perform. At the end of every case it will break; so that it doesn't go to the next case instead it leaves the switch. If 0-7 isn't entered then it will output "invalid choice" and ask them to input another function.

loginPanelpt1

This function asks the user for their account type. This function is used later in loginPanel.

userSearchSelection

This function allows the user to search for books by ISBN, author, title, and category. The user is first asked how they want to search, then asked to input the according ISBN, author, title, or category. The book vector is then iterated through to find a book that matches the search criteria. For category and author, the books found are first ordered by popularity before being printed.

selectionPanelLibrarian

The parameters for selectionPanelLibrarian are istream& in, vector<Librarian>& l,vector<Student>&s, vector<Teacher>& t, vector<Book>& b,int& userNumber. This function allows the librarian to search books, add books, delete books, search users, add users, delete users, change their password, and get their own information. The user inputs a number from 0-8 to choose which action to take, otherwise an error is outputted. This function works very similarly to selectionPanel, but it has functions that only the librarian is allowed to access.

loginPanel

The parameters for loginPanel are istream& in, vector<Student>& s, vector<Book>& b, vector<Teacher> t, vector<Librarian> l,bool& login, int& userNumber. loginPanelpt1 is first used to get the account type of the user. loginStudent is used for the user to log into the system. Based on the user's account type, the appropriate selectionPanel is brought up.

<u>STUDENT AND TEACHER CLASS FUNCTIONS</u>

Default Constructor

Assigns default values to student class without parameter

Parameterized constructor

Assigns the values in the parameter to private values of the new student object

Overloaded >> operator

Reads the username and password from the class then alters the student object in the parameter to the obtained username and password.

Overloaded << operator

Prints the information of the student. It gives the username but not password, all borrowed books, and all reserved books for the student.

The only difference between the teacher and student class is that the teacher is allowed to borrow more books than the student. The teacher is allowed to borrow 8 books and the student is only allowed to borrow 5.

Copy Class Functions

There are just the basic get and set functions for the copy class.
Consists of information of borrowers and will be used to keep track of borrowers and reservees and their time of action.

Highlights

One of the extra functions we've included is the ability to reserve a book. Reserving a book puts the respective student's name as the reader of the book without creating a timer. This allows the user to borrow a book they've reserved while stopping other students/users from borrowing that book. (As of right now, a user can reserve a book forever, but a timer can be added to remove the reserve if needed).

If you're able to reserve a book, you should also be able to cancel said reservation which we've given the ability for a student to do.

Another extra function we have is the myInformation function, which includes the username of the student, the books they have borrowed, how much time they have left to return said borrowed books, and what books the student has reserved

The final extra function we've added is the ability to recommend a book. The function recommends a book depending on what books the student currently has borrowed. It looks at the categories of the books the student has borrowed and recommends the student all books within the same category.