**MSc Object Oriented Programming in Java and UML**

[UML diagram](#_x8533713sfh8)

[Implementation](#_t50slj35ttj1)

[Books](#_aqh5jirpsl71)

[BookList](#_9ih3aqpsvpk3)

[Users](#_u9zq1n2rbis9)

[UserList](#_mfyo3z66ng0u)

[Loan/Loans](#_yl3kcaxjwlqy)

[Loans implementation detail](#_3abm9rw8wfr5)

[method [void borrow(Users users,Books books,String expiryDate)]](#_uhfsz9fk2qzq)

[method [void returnBook(Users users,Books books)]](#_70pzegukam4l)

[method [int countBooksOfUsers(Users users)]](#_e026xkabh942)

[method [ArrayList<OverdueItem> getOverdueItems(String currentDate)]](#_rpei7y3qm289)

[method [ArrayList<Books> getPopularBooks()]](#_ajhj9prgp1nv)

[method [BorrowerInformation getBorrower(Books book)]](#_tmt8328wile1)

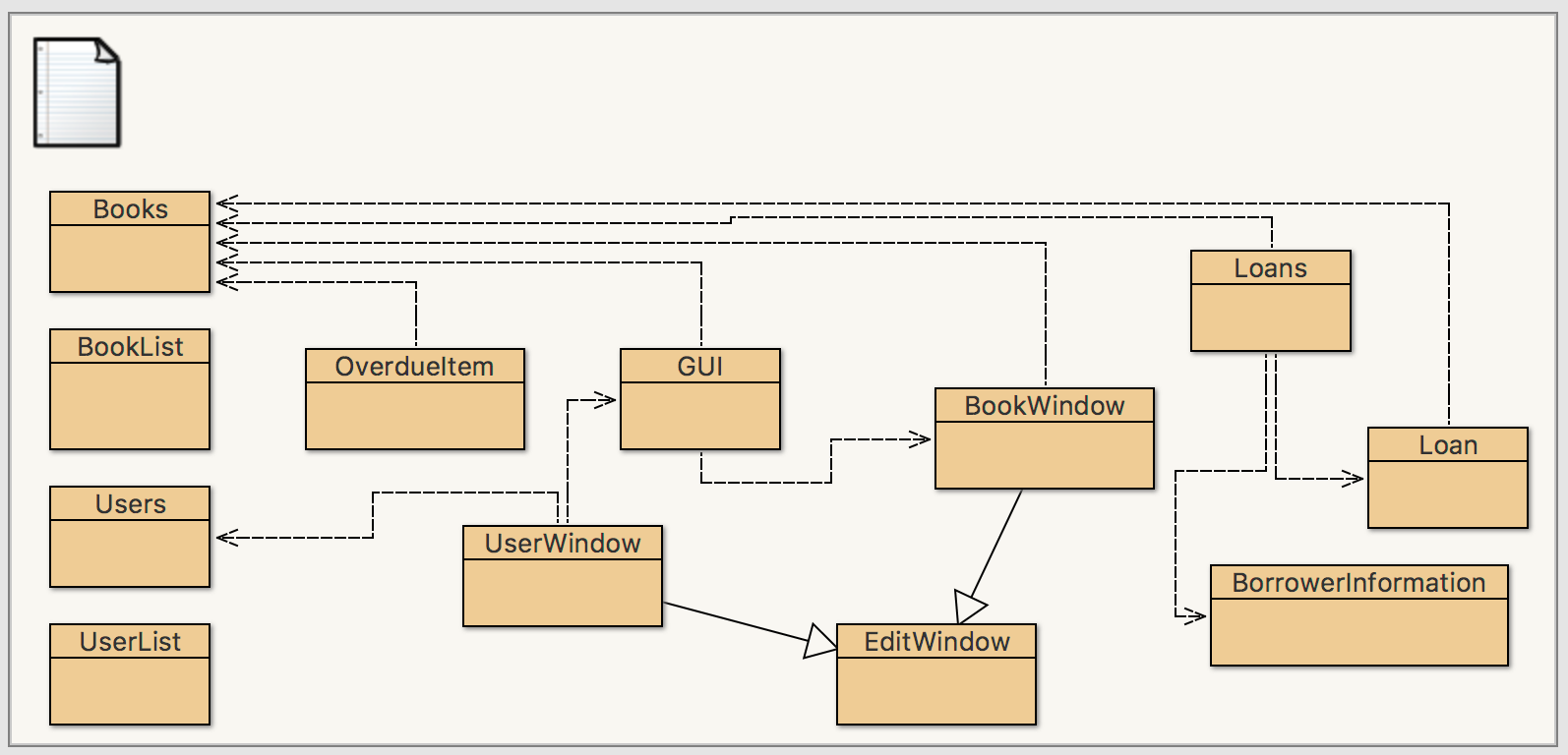
[Extras](#_n2qj624q6jlq)

[Load/Save records From/Into File](#_lr4dbqb4irlo)

[Login Window](#_gn10f1m8u1re)

[Edit Window](#_50kacyhllpa)

# UML diagram



# **Implementation**

## **[Books](file:///Users/gracelo/Desktop/Assignment/Source code/doc/index.html)**

The books class store the information of a book. Besides other fields in the class such as title, author, year, publisher, number of copies and publication date, there is a special field which is called ID. Because ID is randomly generated, I used the (int)(Math.random() \* 1000000000) method that will create an integer between 0 and 1000000000.

One of fields publicationDate that I used the String type of  YYYYMMDD format  to store this, and date in other classes in the assignment that I also used the format.

## **[BookList](file:///Users/gracelo/Desktop/Assignment/Source code/doc/index.html)**

BooksList ;in other words, to hold details of book files. There is a ArrayList<Books> field which is declared as private to store the Books.

[Books search(String keyword)] method in BookList class can search the same book with title, author, publisher or publication date as keyword. If it cannot find, then it will return null. Moreover, if there are multiple Books to match the conditions, then it will return the one which be found firstly.

[void remove(String title)] method in BookList class is used to remove the book which has the title. If there is not a book with the title, then it will throw Exception.

## [**Users**](file:///Users/gracelo/Desktop/Assignment/Source%20code/doc/index.html)

The Users class represents users in a library system. It holds the user records relevant in our context. The class is similar to books.

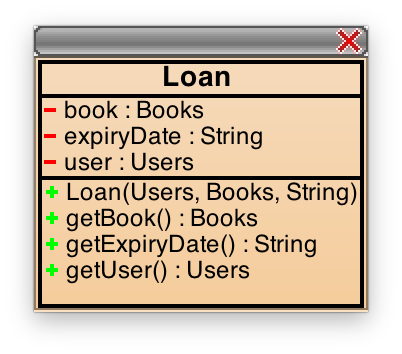
## [**UserList**](file:///Users/gracelo/Desktop/Assignment/Source%20code/doc/index.html)

The UserList class is similar to BookList class, and there is a ArrayList<Users> field that is declared as private to store the user files.

Method [Users search(String username)] is also similar to the search method in BookList, and the  [Users search(String username)] method return the Users if the username match; otherwise, return null.

In UserList class, I wrote a method [void remove(String firstname)]  to remove the Users that has the firstname. However,  if there is not a Users that has the firstname, then it will raise Exception.

## [**Loan**](file:///Users/gracelo/Desktop/Assignment/Source%20code/doc/index.html)**/**[**Loans**](file:///Users/gracelo/Desktop/Assignment/Source%20code/doc/index.html)



The Loan class represents a single loan record in the library system. On the other hand, Loans class is used to provide the function that manage loan records, and the class include ArrayList<Loan> which is used to store all of loan records in the library system.

### **Loans implementation detail**

#### method [void borrow(Users users,Books books,String expiryDate)]

The method is used to add a new Loan into ArayList<Loan>, and the available number of copies of books will change.

#### method [void returnBook(Users users,Books books)]

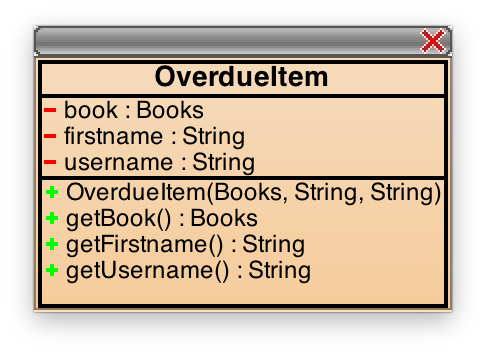
I wrote a method to find a Loan which is corresponding in ArrayList<Loan>, then remove it, and change the available number of copies of books. If it cannot find the Loan record, then will raise Exception.

#### method [int countBooksOfUsers(Users users)]

The method scans to count the total number of books for a user to borrow a book in ArrayList<Loan>, and return the number.

#### method [ArrayList<OverdueItem> getOverdueItems(String currentDate)]

The method is used to find all of Loan that their the expiryDate is more previous than the currentDate. In related to those Loan, I create another class [OverdueItem](file:///Users/gracelo/Desktop/Assignment/Source%20code/doc/index.html) (Including Books, username and firstname), and I put the OverdueItem into ArrayList<OverdueItem>, then return.



Both of the expiryDate and currentDate is the String of YYYYMMDD format, so I used directly [String.compareTo](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html#compareTo(java.lang.String))  to compare the order of  the dates.

#### method [ArrayList<Books> getPopularBooks()]

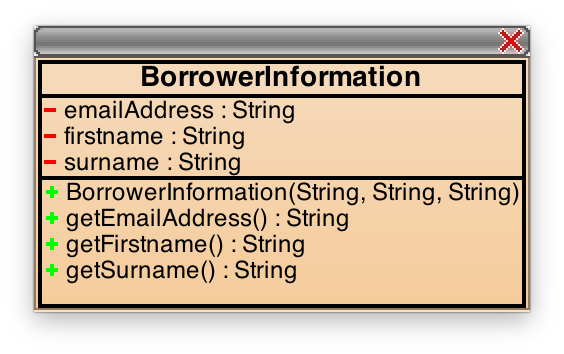
The method returns those books that are borrowed by more than one User. In connection with each Loan in ArrayList<Loan>, I used the for each loop again to judge whether or not there are different the Loan of Users but the same the Books of Loan.

If there are N records of Loan, the execution time of the method is directly proportional to N2.

I have been considered that use the available number of copies of Books to judge whether or not a book which be borrowed by more than one person; however, the function will occur error is that when a person borrowed two books which are the same.

#### method [BorrowerInformation getBorrower(Books book)]

The method returns a corresponding[BorrwerInformation](file:///Users/gracelo/Desktop/Assignment/Source%20code/doc/index.html)according to the Loan in ArrayList<Loan> that is about a book be borrowed firstly. If cannot find, then return null.



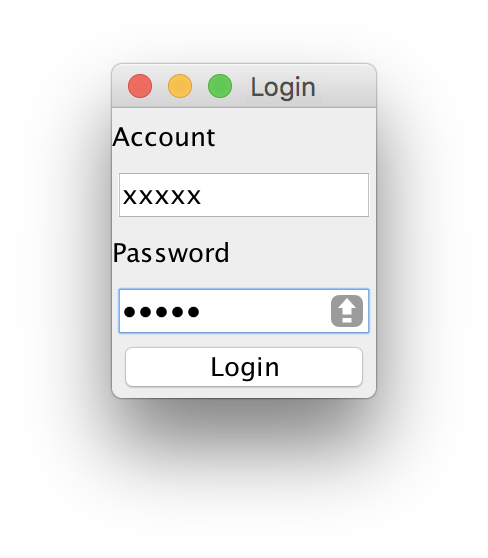
## **Extras**

### **Load/Save records From/Into File**

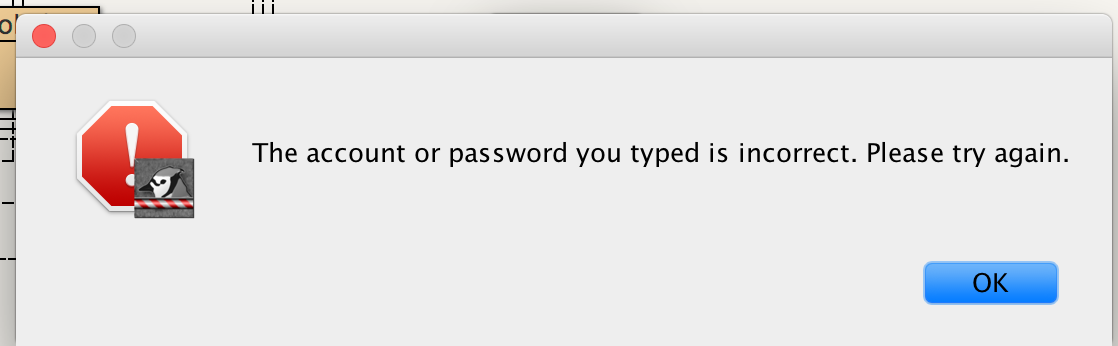
I chose to use File rather than MySQL to store records, because I thought to use File that can reduce trouble to set up and connect with database. I set up a file path which is “data.txt” of current working directory.

I used FileReader+Scanner and FileWriter+PrintWriter to process File I/O. I designed the file which is text file, and B represents some Books in the first line of the file, then there are 6\*B lines. Every six lines represents a Books with title, author, year, publisher, numOfCopies, and publicationDate. There is U represents some Users in the next lines, then there are 8\*U lines. Every eight lines represents an Users with username, firstname, surname, houseNumber, streetName, postcode, emailAddress and dateOfBirth. Above codes in loadFromFile() and saveToFile() in [GUI](file:///Users/gracelo/Desktop/Assignment/Source%20code/doc/index.html) class

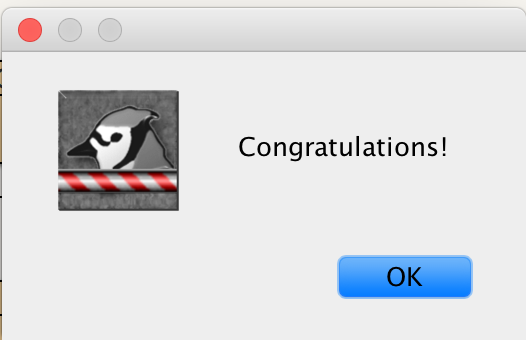
### Login Window



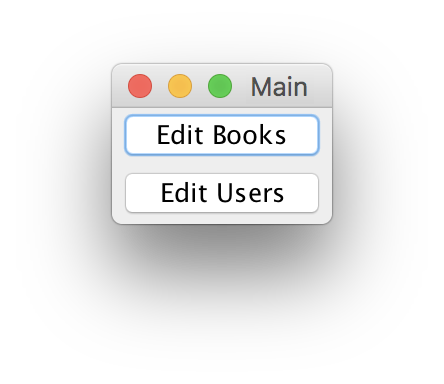
My Login Window use the GridLayou of 5x1, and put  Account Label, Account Text Field, Password Label, Password Field, and Login button,respecitvely. I set up correct account is “xxxxx” and the passwords are “12345”. There is an error message dialog if typed incorrectly. In the following:



If both account and passwords are typed correctly, then the information message dialog will occur. In the following:

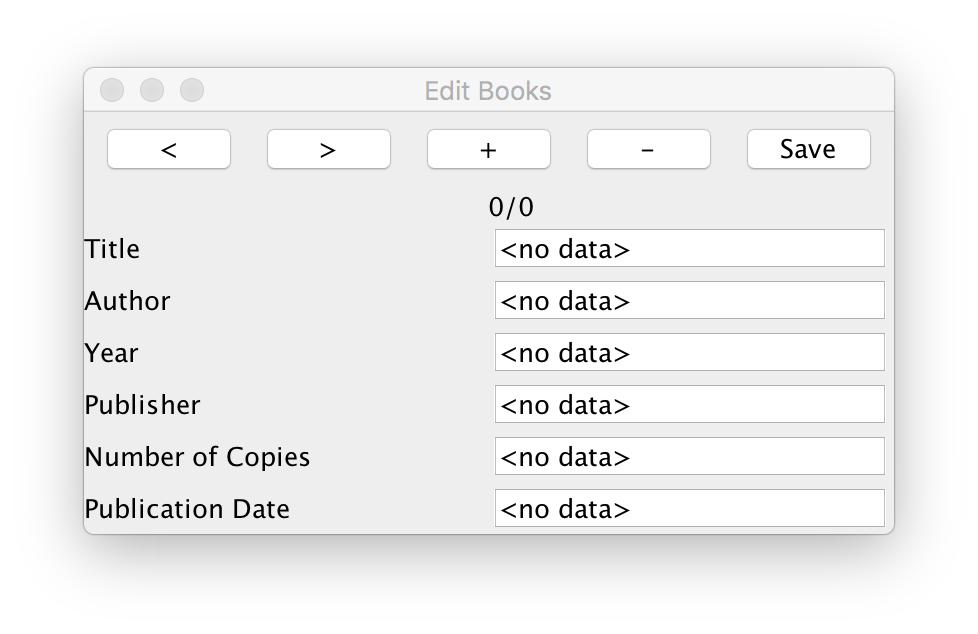


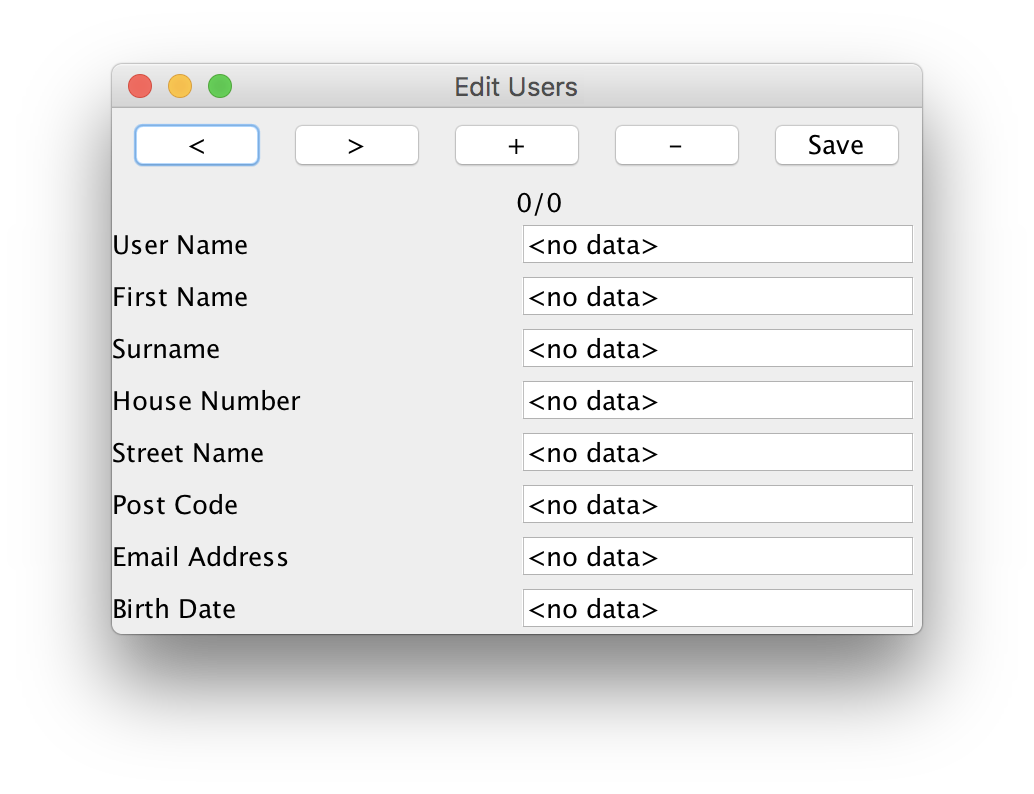
Then login the main screen, and can choose to edit Books or Users.



### **Edit Window**

Because of the function of editing Books is similar to edit Users, I defined a class EditWindow , and insert the same codes into the class, then defined two classes: [BookWindow](file:///Users/gracelo/Desktop/Assignment/Source%20code/doc/index.html) and [UserWindow](file:///Users/gracelo/Desktop/Assignment/Source%20code/doc/index.html)(both of them are extends EditWindow) to insert the codes that are related to edit Books and Users into the two classes.





I used BoxLayout to set buttons area above, index label in the middle and the information column area below in the EditWindow. I used GridLayout in the information column area below, and FlowLayout in buttons area above.

The five functions of the buttons:

 “<”: go to previous record

“>”: go to next record

“+”: add a new record base on the current contents which is finished (It has to notice that need to complete all of contents).

“-”: delete the current record

“Save”: update the current record

The index label in the middle show that the rank of the current record and the total.

**Javadoc link:**

<file:///Users/gracelo/Desktop/Assignment/Source%20code/doc/index.html>