Project Report

Group8

CATALOGUE:

4. reserve

5. notification.

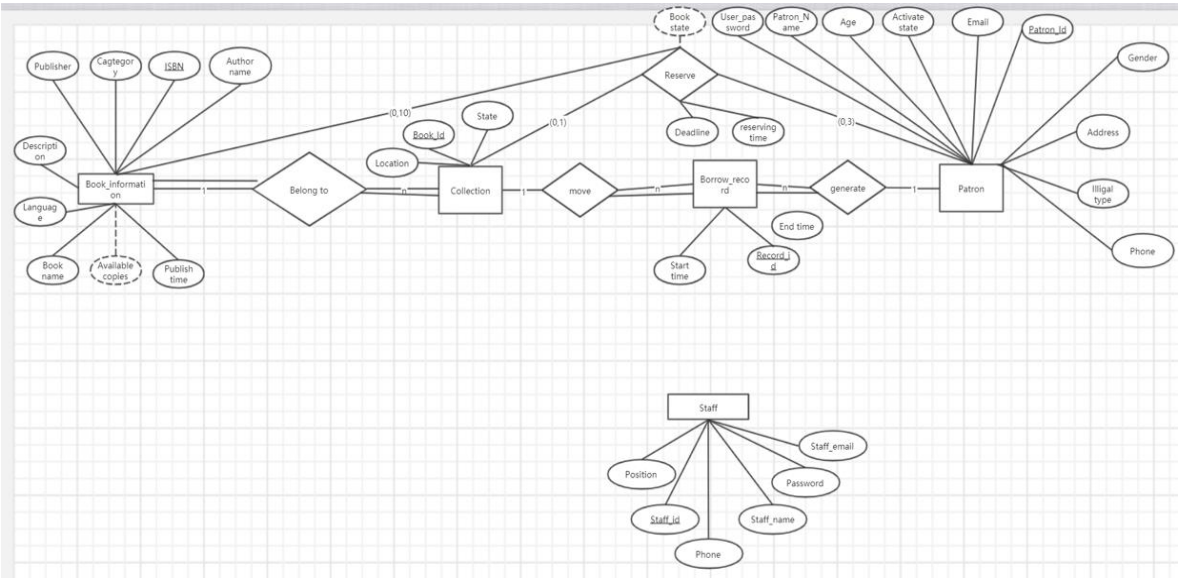
1, Introduction

Nowadays, books have become an indispensable part of people's spiritual world. However, the traditional manual library management system has been difficult to meet people's needs for increasingly higher demand for reading books, and the libraries need more efficient management and effective data storage. For instance, the structure of library staff is complex, the number of staff is limited, and it involves a wide range of aspects. If we still use manual operation to deal with the problem of book borrowing, the work will be very cumbersome, requiring a lot of manpower, physical and financial resources, which is a great waste of resources. Library management includes book information management, borrowing record management, patron management, staff management and so on. And these projects in the past by manual operation, manual recording of these things is not only troublesome, but also often wrong, bring a lot of inconvenience to most users, therefore, the development of such a set of library management system software. Therefore, we designed this intelligent library management system for detailed management of library information, which not only let the administrator convenient management of books and user information, convenient for users to find books but also provides approaches for creating a comprehensive digital library management system in the future.

2, Database design:

(1) Overview—ER diagram

Introduction to main structure:



Here is the basic database structure of the project.

(2) Rational schema

Book Information

ISBN	Author name	Category	Publisher	Description	Language	Book name	Publish time

We create an entity to store the information of the book especially, named “Book information”. We split it out of “Collection of books”, because there may be more than one copy of each book. If we must identify each one, we must store the information many times, which will cause data redundancy. If the content of the books is the same, they have the same ISBN (International Standard Book Number) but have different book IDs to identify each physical book. Note there may be more than one copy of each book, and a book may even be published by different publishers, which means it may happen that two book have the same name but different ISBN. Significantly, it is not book name but ISBN that is the primary key of book information. When handling the searching, we must consider this case. The “available copies” is a derived attribute to calculate how many copies have the same ISBN and are also available to borrow. People can decide whether to borrow or not after knowing this number. This attribute will not be stored in the database, because the maintain cost is high. However, we can drive this attribute through select query plus count.

Collection

Book_id	State	Location	ISBN

The entity of “Collection” has three attributes, “Book_id” is the primary key, “location” shows the special location of a book, and “State” reflects whether the book is available or not.

Patron

Patron_id	User_Password	Patron_Name	Age	Activity_State
E_mail	Gender	Address	Illegal _Type	Phone

The entity “Patron” has a PATRON_ID as the primary key. The attributes are about the basic information of each patron. The “ Activate_state ” is about whether the account of the patron is active or not, and the “illegal type” records why the account is inactive. “Illegal type” includes damaging a book, didn’t return books after a specific period passes, and so on. This design meets feature two, “The ability to deactivate a patron’s account if he/she does not return books after a specific period of time passes.” We also record the email and phone number of the patron. If the desired book becomes available and reminders that a book should be returned to the library, emailing, calling, and sending messages on the page when the patron logs in are good choices. There meets feature four,” Notifications when the desired book becomes available and reminders that a book should be returned to the library. Both could be when the patron logs in to the LMS.”. In normal case the illegal type will always be null, if break the rule, the system will set the activate state to false, and set the illegal type at the same time. Only when the user gets the authority back will the system delete the record.

Analysis: We also can analyze the information of the patrons to summary whether the people of the same type have the same reading preferences and recommend books to others automatically. Like the children like what kinds of books, women like which books. The attributes like “Age”, “Gender” can be helpful.

Borrow_Record

Record_id	Book_id	Patron_id	Start_time	End_time

The “Borrow_ Record” entity is also important. Every patron can borrow 0~3 books; every book should be borrowed or not; The “Borrow” table can record the start and end times of the books, which is helpful for features three (Records of books checked out as well as placed on hold (i.e., “reserved” by a patron to make sure the book is there when he/she gets to the library to check it out).) and four. The start time decides the deadline. If the end time is earlier than the deadline, this borrowing is successful; if not, the account of the patron will be inactive. Of course, the system will send an email to remind the patron to return books if the time, which depends on the start time, is suitable. If the borrow ends, the end time will be set to the time when the user returned the book. If the borrowing is in progress, the end time will be set to null. The maximum borrow period is 100 days, if the time is over, the system will regard it as overdue.

Analysis: For” Borrow_Record” table, we can find the peak hour by analyzing “Start_time”. If too many people want to borrow books at the same time, the management could arrange enough manpower at this time to serve patrons better.

Reserve

Book_id	Patron_id	Book state	Deadline	ISBN

We also design a “Reserve” relationship especially. It is also a trinary relationship. Every patron can just reserve 0~3 books; One unique book can just be recorded as placed on hold in only one specific record. We use the trinary relationship because if the patron wants to borrow a book, which copy is not important. He or she just wants to borrow one book having the same ISBN. So, the relationship to connect to “Book_information” is to make the patrons reserve one kind of book. But if one specific book is available, among the patron who reserved the book, we will inform the patron who first login to the system if he/she wants to get the book, this is for feature 3.

Two different reserve types:

Available reserve: when the related ISBN have available copies.

Function: Be used when there exists available copy, if so, we can lock a copy for the

user, and wait for two days.

2.unavailable reserve: When no available copy at the reserve time.

Function: Available copy does not exist, if so, we will wait until the available copy appear (The maximum waiting period is 100 days), Or if the available copy appears, we will inform the user to transform the unavailable reservation to be available. If the time exceeds the maximum waiting period (100 days) or the user transforms it to an available reservation, our system will delete the records automatically.

Design point: It can deal with different situations in real library system applications, if no available copy exists, the user can still reserve the book they want, and can always be the first to get the latest information of the book they have reserved.

Staff

Staff_id	Password	Staff_name	Position	Phone	Email

We also have an entity named" staff." The staff have unique IDs, and the staff can help to manage the records or update the information of any book.

(3) BCNF analysis

"Book Information" is BCNF. All the underlying domains contain atomic values. All the non-key values are fully dependent on the candidate key. However, they are not transitively dependent on either. The left-handed side of the FDs is a candidate key, fulfilling the condition that BCNF is only if every determinant (left-hand side of an FD) is a candidate key.

"Staff" is BCNF. All the underlying domains contain atomic values. All the non-key values are fully dependent on the candidate key. They are not transitively dependent on either. Also, the left-handed side of the FDs is a candidate key, fulfilling the condition that BCNF if every determinant (left-hand side of an FD) is a candidate key.

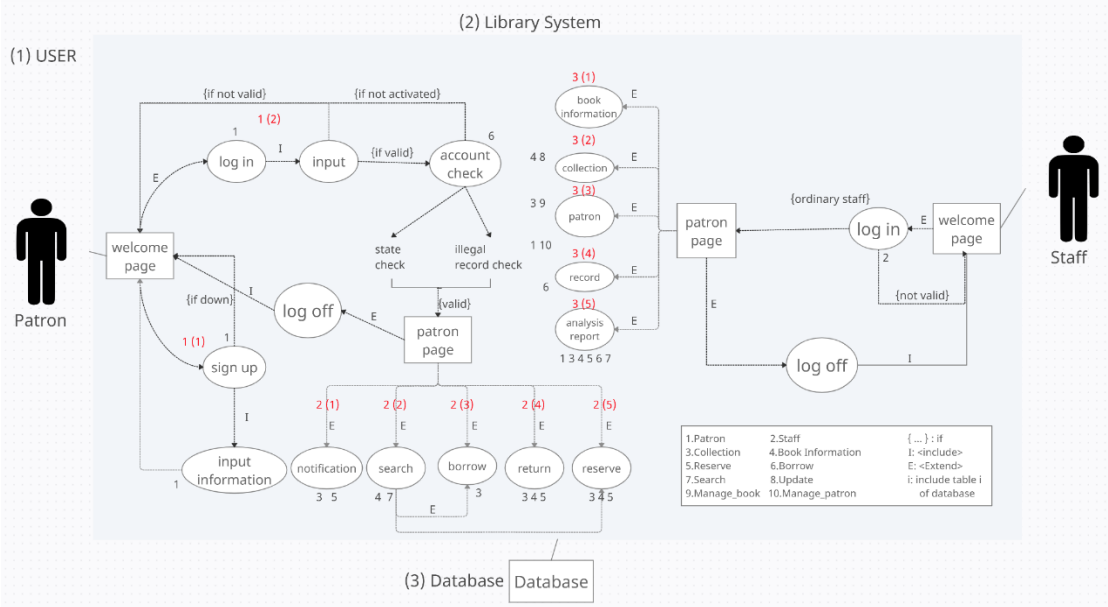
"Reserve" is 3NF.All the underlying domains contain atomic values. All the non-key values are fully dependent on the candidate key. They are not transitively dependent on either. However, the left-handed side of the FDs is not a candidate key, it does not fulfill the condition that BCNF is only if every determinant (left-hand side of an FD) is a candidate key.

"Collection" is 2NF.All the underlying domains contain atomic values. All the non-key values are fully dependent on the candidate key. However, they are transitively dependent on either. The ISBN is a foreign key and there is transitive dependent

existing.
“Borrow Record” is 3NF.All the underlying domains contain atomic values. All the non-key values are fully dependent on the candidate key. They are transitively dependent on either. The Patron_id and Book_id are foreign keys and there is transitive dependent existing.
“Patron” is BCNF. All the underlying domains contain atomic values. All the non-key values are fully dependent on the candidate key. They are not transitively dependent on either. Also, the left-handed side of the FDs is a candidate key, fulfilling the condition that BCNF if every determinant (left-hand side of an FD) is a candidate key.

3, library system design:

Overview:



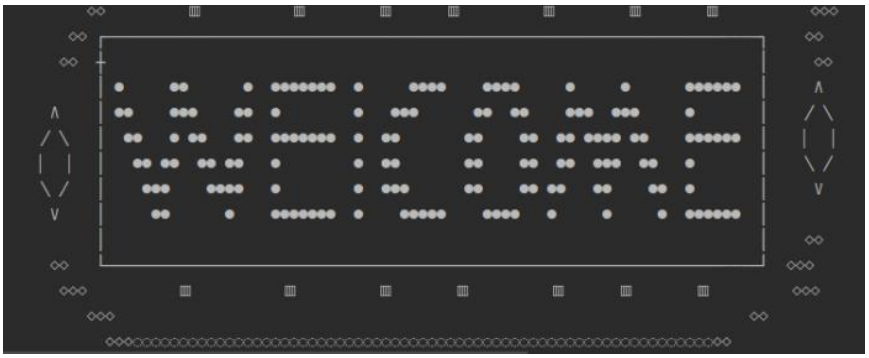
The users can login as patron or staff, both should login through the welcome page. When the user chooses to login as a patron, they will have two choices, one is login, and the other is sign up, when the choose to login, our system will check whether their account has some illegal record, and the state of the account. If they passed these two checks, they could go to the patron page to use the five functions (Notification, search, borrow, return, reserve) provided by the System. Otherwise, if they choose to sign up, they can create a new account, and the system will store the information about it.
If the user chooses to login as a staff member, the system will check if the login is valid, if so, they can use the functions provided on the staff page to manage book information, collection, patron, record, analysis report.

[Welcome Page]

In our welcome page, we let user enter an integer to make the choose (enter 0 to quit):
1 is log in as patron ,2 is sign up as patron ,3 is log in as staff:

(1) Log in as patron

First, there are some situations that lead to unsuccessful operations.

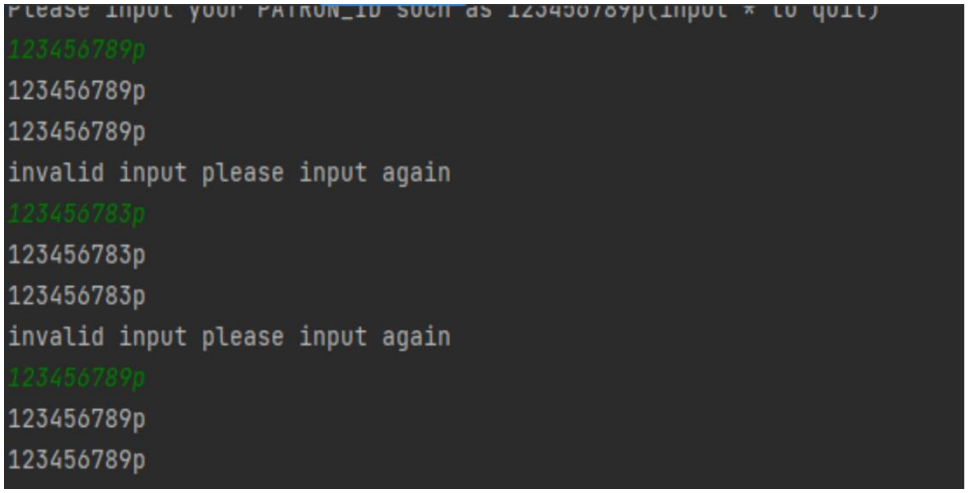


If we input 0, then we quit and leave the welcome page.

1.when we choose 1, then if enter a patron id that is not exists,
then we cannot log in.

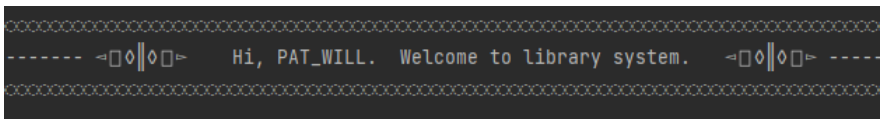
Worse still, if enter an inactivated patron id, then we cannot log in, either.

Every user can only enter the password 3 times.



If a user has entered a password 3 times and all the attempts are not correct, then he cannot log in. Otherwise, he can successfully log in.

This is a successful situation.



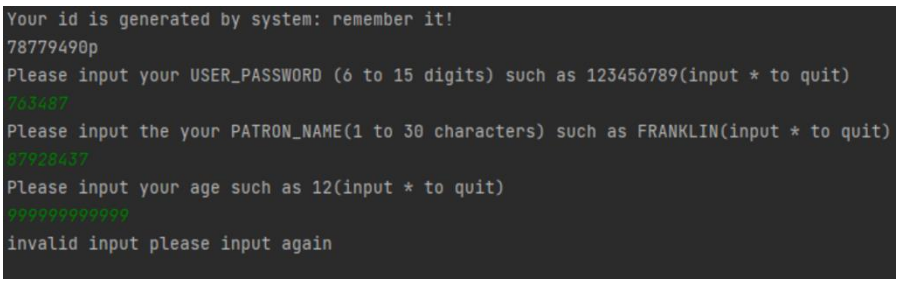
We should enter the patron's name and the password according to the database.

1	64273878p	8216YH7501	PAT_WILL	23	1	ZHOUZHOU343@163.COM	MALE	GARDEN_ROAD_14TH	(null)	71645899
2	21769256p	TT742187	COOK_BUSH	9	1	COOKFOREVER@163.COM	MALE	MERDER_STREET_92RD	(null)	51276812
3	72647867p	KK19770202	BEN_JOHN	78	0	BIGBEN@163.COM	MALE	PRINCE_ROAD_33RD	(null)	73678291
4	83748773p	TV2505050	LILY_WHITE	93	0	LILYBEAUTY@163.COM	FEMALE	HANBERGER_STREET_17TH	(null)	47920184
5	82647821p	MM7671885	MAX_BLACK	102	1	MAXBCD7778163.COM	MALE	MARY_ROAD_87TH	(null)	82163410
6	90278775p	DONGS99887	JOHN_SMITH	44	1	JSJS@163.COM	MALE	GOODMAN_ROAD_25TH	(null)	84217593
7	71264897p	12345999JJ	EMMA_JOE	46	1	EMMAGOOD11@163.COM	FEMALE	PHILIPHIES_STREET_88TH	(null)	82417983
8	82762993p	8765432155	CATHRINE_MOMO	119	1	DUHONGHUA@163.COM	FEMALE	KING_ROAD_45TH	(null)	09428384
9	65488722p	LOG123456	MERGE_HANS	98	1	SIHAFG656@163.COM	MALE	PKU_ROAD_98TH	(null)	48297193
10	27482165p	PPT123490	AUTHOR_BARBECUE	27	1	BSGFIYUYI@163.COM	MALE	FORG_STREET_33RD	(null)	73647821

(2) Sign up as patron

If we choose 2,

That is the patron sign-in part If a user has entered the same item repeatedly, then the sign up is not successful.



Additionally,

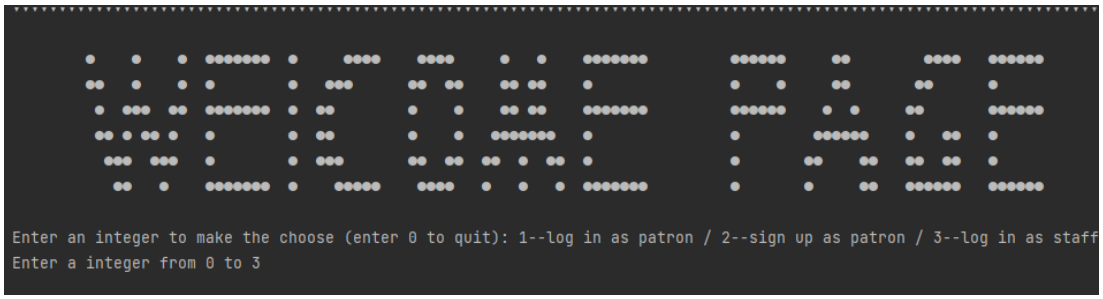
If the user withdraws halfway, then the sign-up is also unsuccessful. Otherwise,

the sign up is successful.

```
Your id is generated by system: remember it!
17054832p
Please input your USER_PASSWORD (6 to 15 digits) such as 123456789(input * to quit)
123456789
Please input the your PATRON_NAME(1 to 30 characters) such as FRANKLIN(input * to quit)
PAT_WILL
Please input your age such as 12(input * to quit)
*

Leave patron_sign_up
```

Below is the situation that we successfully signed up for. The user will go back to the welcome page and now he or she can do the log in operation.

```
Sccessfully sign up !

Leave patron_sign_up
```



THE CHANGE OF SQL DATABASE IS SHOWED BELOW.

1	64273878p	8216YK7501	PAT_WILL	23	1	ZHOUZHOU343@163.COM	MALE	GARDEN_ROAD_14TH	(null)	71645899
2	21789256p	TT742187	COOK_BUSH	9	1	COOKFOREVER@163.COM	MALE	MERDER_STREET_92RD	(null)	51276812
3	72647867p	KK19770202	BEN_JOHN	78	0	BIGBEN@163.COM	MALE	PRINCE_ROAD_33RD	(null)	73678291
4	83748773p	TV2505050	LILY_WHITE	93	0	LILYBEAUTY@163.COM	FEMALE	HANBERGER_STREET_17TH	(null)	47920184
5	82647821p	MM7671885	MAX_BLACK	102	1	MAXBCD777@163.COM	MALE	MARY_ROAD_87TH	(null)	82163410
6	90278775p	DONGS99887	JOHN_SMITH	44	1	JSJS@163.COM	MALE	GOODMAN_ROAD_25TH	(null)	84217593
7	71264897p	12345999JJ	EMMA_JOE	46	1	EMMAGOOD11@163.COM	FEMALE	PHILIPHIES_STREET_88TH	(null)	82417983
8	82762993p	8765432158	CATHRINE_MOMO	119	1	DUHONGHUA@163.COM	FEMALE	KING_ROAD_45TH	(null)	09428384
9	65488722p	LOG123456	MERGE_HANS	98	1	SIHAFG656@163.COM	MALE	PKU_ROAD_98TH	(null)	48297193
10	27482165p	PPT123490	AUTHOR_BARBECUE	27	1	BSGFIYUYI@163.COM	MALE	FORG_STREET_33RD	(null)	73647821

Above is the former page and below is the page after a new patron Franklin has signed up.

1	08548129p	123456789	FRANKLIN	12	1	1234567@POLYU.EDU.HK	MALE	HK	(null)	88888888
2	64273878p	8216YK7501	PAT_WILL	23	1	ZHOUZHOU343@163.COM	MALE	GARDEN_ROAD_14TH	(null)	71645899
3	21789256p	TT742187	COOK_BUSH	9	1	COOKFOREVER@163.COM	MALE	MERDER_STREET_92RD	(null)	51276812
4	72647867p	KK19770202	BEN_JOHN	78	0	BIGBEN@163.COM	MALE	PRINCE_ROAD_33RD	(null)	73678291
5	83748773p	TV2505050	LILY_WHITE	93	0	LILYBEAUTY@163.COM	FEMALE	HANBERGER_STREET_17TH	(null)	47920184
6	82647821p	MM7671885	MAX_BLACK	102	1	MAXBCD777@163.COM	MALE	MARY_ROAD_87TH	(null)	82163410
7	90278775p	DONGS99887	JOHN_SMITH	44	1	JSJS@163.COM	MALE	GOODMAN_ROAD_25TH	(null)	84217593
8	71264897p	12345999JJ	EMMA_JOE	46	1	EMMAGOOD11@163.COM	FEMALE	PHILIPHIES_STREET_88TH	(null)	82417983
9	82762993p	8765432158	CATHRINE_MOMO	119	1	DUHONGHUA@163.COM	FEMALE	KING_ROAD_45TH	(null)	09428384
10	65488722p	LOG123456	MERGE_HANS	98	1	SIHAFG656@163.COM	MALE	PKU_ROAD_98TH	(null)	48297193
11	27482165p	PPT123490	AUTHOR_BARBECUE	27	1	BSGFIYUYI@163.COM	MALE	FORG_STREET_33RD	(null)	73647821
12	50673348p	QWERTYUIOP	FRANKLIN	12	1	1234567@POLYU.EDU.HK	male	afafafafafaf	(null)	99999999

(3) Log in as staff

If we choose 3, that is the staff log in part, it is like 1. when we choose 3, then if enter a staff id that is not exists, then we cannot log in.

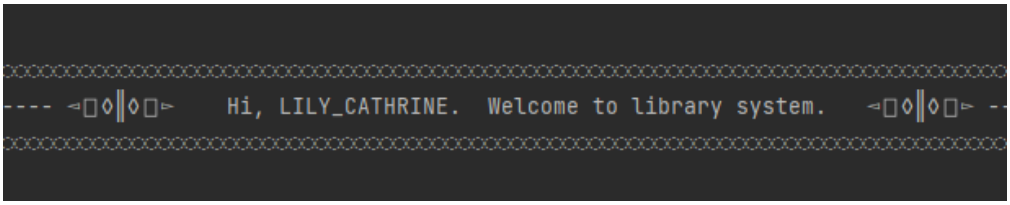
```
input the STAFF_ID (a string with 9 character e.g. 12345678s):[input * quit]
777
the input is invalid, please input again:(input * quit)
12345678s
The STAFF_id is not valid. We cannot find the account in system. Input again or quit.
input the STAFF_ID (a string with 9 character e.g. 12345678s):[input * quit]
```

Every user can only enter the password 3 times. If a user has entered password 3

times and all the attempts are not correct, then he cannot log in. Otherwise, he can successfully log in.

```
input the STAFF_ID (a string with 9 character e.g. 12345678s):[input * quit]
12345678s
The STAFF_id is not valid. We cannot find the account in system. Input again or quit.
input the STAFF_ID (a string with 9 character e.g. 12345678s):[input * quit]
12345679s
The STAFF_id is not valid. We cannot find the account in system. Input again or quit.
input the STAFF_ID (a string with 9 character e.g. 12345678s):[input * quit]
12456789s
The STAFF_id is not valid. We cannot find the account in system. Input again or quit.
input the STAFF_ID (a string with 9 character e.g. 12345678s):[input * quit]
```

Then we can successfully enter the system.



We should enter the staff name and the password according to the database.

1	82647821s	1657AMA1657	LILY_CATHRINE	MANAGER	31729270	DUZILONG@163.COM
2	38741897s	OPG842175833	ALICE_CABBAGE	LIBRARIAN	56276470	DDECHR798@163.COM
3	64273878s	ILOVE520	PETER_S_HARRY	BOSS	12647216	KAIYUANN@163.COM
4	21789256s	M1234567M	LUCAS_BUSH	PROGRAMMER	76412458	WEINANMING88@163.COM
5	72164737s	ABC817761	PETER_JESON	BOSS	12647216	GGJJTING29@163.COM
6	81743988s	17632546ST	KIM_ROBBINS	LIBRARIAN	82147574	KONGDONG1988@163.COM
7	27164767s	RES123456	JIM_BIDEN	VICE_BOSS	23164567	BANGKOKMAN@163.COM
8	12874662s	BBC542290	MARY_BEAUTY	PROGRAMMER	37674882	KULIIISIKI@163.COM

[Patron Page]

sample

Under the patron page we have 4 main functions, these functions allow the patron to search book information, borrow books, return books and reserve books. Here we will introduce them

one by one.

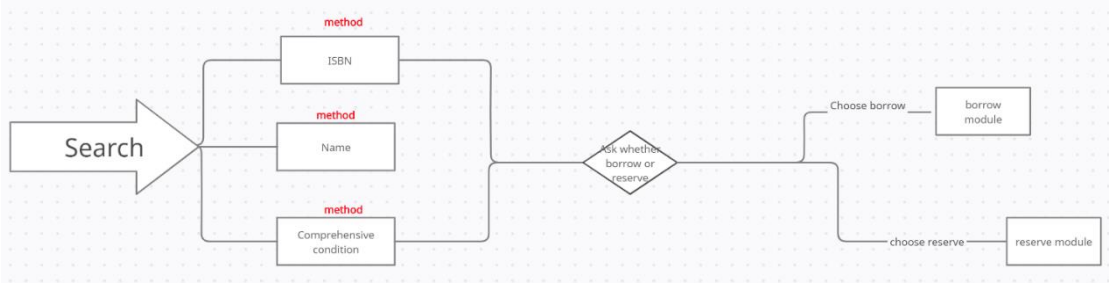
[Core Function List]

```
Enter an integer to make the choose (enter 0 to quit):
1--search for book information / 2--borrow the book
3--return the book / 4-- reserve a book
```

[Notification list]

Every time a patron logs in, he or she will first receive a notification from the system, which will first remind the books which the patron is in progress borrowing.

1. Search for book information



Under the search for book information function, we have three different operations. The following is the description of them.

1.1 search by book name

```
input the BOOK_NAME (a string have 1 to 50 characters e.g. Harry Potter):[input * quit]
BIG_CUCUMEMBER
The information of book BIG_CUCUMEMBER are as follows
BOOK_INFORMATION
    ISBN          BOOKNAME             AVAILABLE_COPY  CATEGORY                        AUTHOR_NAME
(1)_____________________________________________________________________________________________________
    9365827626587  BIG_CUCUMEMBER       1               HORROR                          LILY_BLACK
```

Search sample

This function requires the patron to input a BOOK_NAME and will return a list of the book's information which includes ISBN, BOOK_NAME, AVAILABLE_COPY, CATEGORY, AUTHOR_NAME,
PUBLISHETR, LANGUAGE, PUBLISH_TIME, DESCRIPTION. (The sample only cover a few of the information)

After the search, the system will ask the patron whether he or she may want to borrow or reserve the book.

```
Do you want to borrow or reserve a book from these books?  1--yes, 0--no
Enter a integer from 0 to 1
```

Remind sample

Sometime the system will display a list of books, with the same BOOK_NAME but different ISBN, it is because the books may come from different publishers or editions. (This condition is more like reality.)

```
input the BOOK_NAME (a string have 1 to 50 characters e.g. Harry Potter):[input * quit]
HARRY_POTTER_7
The information of book HARRY_POTTER_7 are as follows
BOOK_INFORMATION
    ISBN           BOOKNAME             AVAILABLE_COPY  CATEGORY                      AUTHOR_NAME
(1)_____________________________________________________________________________________________________
    9787810441063  HARRY_POTTER_7       3               TEENAGER_FANTASY              J_K_ROLIN
(2)_____________________________________________________________________________________________________
    2475947892847  HARRY_POTTER_7       0               TEENAGER_FANTASY              J_K_ROLIN
(3)_____________________________________________________________________________________________________
    8573692876859  HARRY_POTTER_7       0               TEENAGER_FANTASY              J_K_ROLIN
(4)_____________________________________________________________________________________________________
    4298748728748  HARRY_POTTER_7       0               TEENAGER_FANTASY              J_K_ROLIN
```

Search sample

1.2 search by ISBN

```
input the ISBN (a string with 13 number character e.g. 1234567890123):[input * quit]
9787810441063
The information of book with ISBN 9787810441063 are as follows
BOOK_INFORMATION
    ISBN           BOOKNAME             AVAILABLE_COPY  CATEGORY                      AUTHOR_NAME
(1)_____________________________________________________________________________________________________
    9787810441063  HARRY_POTTER_7       3               TEENAGER_FANTASY              J_K_ROLIN
```

Search sample

This function requires the patron to input an ISBN and will return a list of the book's information which includes ISBN, BOOK_NAME, AVAILABLE_COPY, CATEGORY, AUTHOR_NAME, PUBLISHETR, LANGUAGE, PUBLISH_TIME, DESCRIPTION. (The sample only cover a few of the information)

After the search, the system will ask the patron whether he or she may want to borrow or reserve the book.

```
Do you want to borrow or reserve a book from these books?  1--yes, 0--no
Enter a integer from 0 to 1
```

Also, if the input ISBN is wrong, it will also give a reminder information.

```
input the ISBN (a string with 13 number character e.g. 1234567890123):[input * quit]
1234567890123
Sorry, we cannot find the book.
```

Remind sample

1.3 search by comprehensive condition

```
Your requirement: books that (Author name contains J_K_ROLIN)  OR (Author name contains BENJAMIN_LUCAS)  AND (Publish time between 1899-09-09 and 2022-11-19)
The books which meet your requirement are as follows
BOOK_INFORMATION
    ISBN          BOOKNAME          AVAILABLE_COPY  CATEGORY                AUTHOR_NAME             PUBLISHER                LANGUAGE
(1)-------------------------------------------------------------------------------------------------------------------------------------
    4641233494645 STAR_WARS         0               SCIENCE_FICTION         BENJAMIN_LUCAS          USA_CENTRAL_PUBLISHER    SPANISH
(2)-------------------------------------------------------------------------------------------------------------------------------------
    7231648727464 STAR_WARS         0               SCIENCE_FICTION         BENJAMIN_LUCAS          USA_CENTRAL_PUBLISHER    SPANISH
(3)-------------------------------------------------------------------------------------------------------------------------------------
    8573692876859 HARRY_POTTER_7    0               TEENAGER_FANTASY        J_K_ROLIN               U_K_PUBLISHER            ENGLISH
(4)-------------------------------------------------------------------------------------------------------------------------------------
    4298748728748 HARRY_POTTER_7    0               TEENAGER_FANTASY        J_K_ROLIN               U_K_PUBLISHER            ENGLISH
(5)-------------------------------------------------------------------------------------------------------------------------------------
    9787810441063 HARRY_POTTER_7    3               TEENAGER_FANTASY        J_K_ROLIN               U_K_PUBLISHER            ENGLISH
(6)-------------------------------------------------------------------------------------------------------------------------------------
    57387T5782577 STAR_WARS         5               SCIENCE_FICTION         BENJAMIN_LUCAS          USA_CENTRAL_PUBLISHER    SPANISH
(7)-------------------------------------------------------------------------------------------------------------------------------------
    2475947892847 HARRY_POTTER_7    0               TEENAGER_FANTASY        J_K_ROLIN               U_K_PUBLISHER            ENGLISH
(8)-------------------------------------------------------------------------------------------------------------------------------------
    8723657843675 STAR_WARS         0               SCIENCE_FICTION         BENJAMIN_LUCAS          USA_CENTRAL_PUBLISHER    SPANISH

Leave search by comprehensive condition
-------------------------------------------------------------------------------------------------------------------------------------
```

Search sample

(Comprehensive condition: books from J_K_ROLIN or BENJAMIN_LUCAS and time from 1899-09-09 and 2022-11-19)

This function is a quite important searching function in our system, it allows the patron to search by comprehensive condition each link with “and” or “or”. The input sample will be displayed、 below.

Step 1:

```
input the AUTHOR_NAME(a string have 1 to 30 characters CAO WEN XUAN,YANG JIANG):(input * quit)
J_K_ROLIN

-----------------------------------------------------  You can add more contition  ---------
Enter an integer to make the choose (enter 0 to quit): 1--And 2--Or
Enter a integer from 0 to 2
2
```

Choose to search by AUTHOR_NAME and set the condition into “or”

Step 2:

```
input the AUTHOR_NAME(a string have 1 to 30 characters CAO WEN XUAN,YANG JIANG):(input * quit)
BENJAMIN_LUCAS

-----------------------------------------------------  You can add more contition  ---------
Enter an integer to make the choose (enter 0 to quit): 1--And 2--Or
Enter a integer from 0 to 2
1
```

Choose to search by AUTHOR_NAME and set the condition into “and”

Step 3:

```
Please input the START_TIME such as 2021-1-12(input * to quit) (input # to represent today)
1899-09-09
Please input the END_TIME such as 2021-1-25(input * to quit) (input # to represent today!)
#

------------------------------------------------------  You can add more contition  -------
Enter an integer to make the choose (enter 0 to quit): 1--And 2--Or
Enter a integer from 0 to 2
0
```

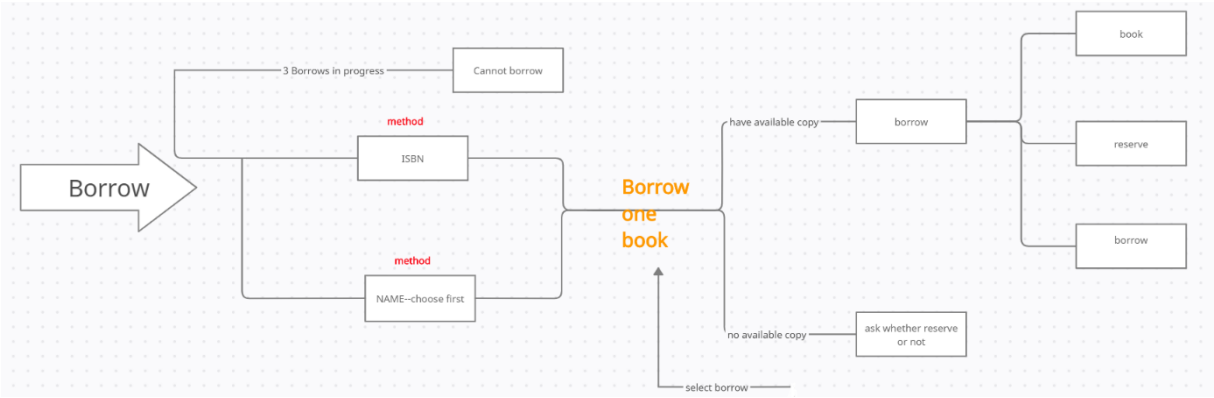
Choose to search by PUBLISH_TIME (it requires the patron input the START_TIME and END_TIME, "#" represented today) and start to search. (Input 0 to stop add condition)

Also like the functions shown above, it allows the patron to reserve or borrow one of the listed books after searching.

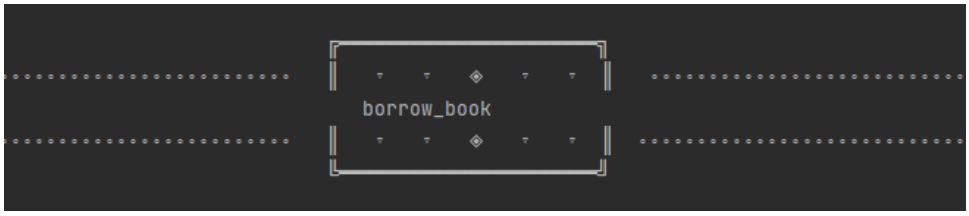
```
Do you want to borrow or reserve a book from these books?  1--yes, 0--no
Enter a integer from 0 to 1
```

Remind sample

2. Borrow the book



Under the borrow book function, we have three different operations. The following is the description of them.



Before the patron borrows any book, this function will first check whether the patron has already borrowed 3 books and has not returned, if he or she is, the patron will not allow to borrow any more books.

```
Sorry, you are 3 in progress borrow now. Our library only allow 3 borrow at a time.
```

Remain sample

2.1 borrow by ISBN

```
input the ISBN (a string with 13 number character e.g. 1234567890123):[input * quit]
9787810441063
The information of book with ISBN 9787810441063 are as follows
BOOK_INFORMATION
    ISBN           BOOKNAME              AVAILABLE_COPY  CATEGORY         AUTHOR_NAME
(1)-------------------------------------------------------------------------------
    9787810441063  HARRY_POTTER_7        3               TEENAGER_FANTASY J_K_ROLIN
Are you sure to choose book HARRY_POTTER_7?  1--yes 0 no
Enter a integer from 0 to 1
1
Your reserve for this book have finished.
Successfully borrow the book.
```

4	0282726605	817491084782184	64273878p	9787810441063	2022-11-19	2022-11-19
5	4572235172	317643546137654	64273878p	9787810441063	2022-11-19	2022-11-19
6	6718467810	716473876473876	64273878p	7218475981205	2019-09-22	(null)

4	0282726605	817491084782184	64273878p	9787810441063	2022-11-19	2022-11-19
5	4572235172	317643546137654	64273878p	9787810441063	2022-11-19	2022-11-19
6	6153096328	736738123764782	64273878p	9787810441063	2022-11-19	(null)
7	6718467810	716473876473876	64273878p	7218475981205	2019-09-22	(null)

Borrow sample

This function allowed the patron to borrow a book by ISBN, after the ISBN is input, it will show the information about the book wait to be borrowed, (information include, ISBN, BOOKNAME, AVAILABLE_COPY, CATEGORY, AUTHOR_NAME, PUBLISHER, LANGUAGE, PUBLISH_TIME, DESCRIPTION) and the patron can decide to cancel or confirm the book.

There is noticeable thing that when the patron borrows the book by ISBN, the system will randomly choose a collection in the library, the state of that book will change (from “1” to “0”)

3	273487236476219	57387T5782577	0	KANGTERROOM_15TH_FLOOR
4	374628376479283	57387T5782577	1	PKULIBRARY_13RD_FLOOR

3	273487236476219	57387T5782577	0	KANGTERROOM_15TH_FLOOR
4	374628376479283	57387T5782577	0	PKULIBRARY_13RD_FLOOR

Change sample

If the book has been reserved by the patron before, the function will automatically delete the reserve recode.

	BOOK_ID	ISBN	PATRON_ID	DEADLINE	BOOK_STATE
1	817491084782184	9787810441063	64273878p	2022-11-09	1

	BOOK_ID	ISBN	PATRON_ID	DEADLINE	BOOK_STATE
1	716481174119742	57387T5782577	21789256p	2022-05-05	1

After the borrow, the first recode have been changed

Also, when the input ISBN is wrong, there will be a remind sentence return.

```
input the ISBN (a string with 13 number character e.g. 1234567890123):[input * quit]
1234567890123
Sorry, we cannot find the book.
```

Remind sentence

2.2 borrow by BOOK_NAME

```
input the BOOK_NAME (a string have 1 to 50 characters e.g. Harry Potter):[input * quit]
STAR_WARS
The information of book STAR_WARS are as follows
BOOK_INFORMATION
    ISBN           BOOKNAME              AVAILABLE_COPY  CATEGORY         AUTHOR_NAME
(1)-------------------------------------------------------------------------------------------
    57387T5782577  STAR_WARS             4               SCIENCE_FICTION  BENJAMIN_LUCAS
(2)-------------------------------------------------------------------------------------------
    7231648727464  STAR_WARS             0               SCIENCE_FICTION  BENJAMIN_LUCAS
(3)-------------------------------------------------------------------------------------------
    8723657843675  STAR_WARS             0               SCIENCE_FICTION  BENJAMIN_LUCAS
(4)-------------------------------------------------------------------------------------------
    4641233494645  STAR_WARS             0               SCIENCE_FICTION  BENJAMIN_LUCAS
Enter the index (start from 1)of book  that you want. If you want to cancel the borrow, enter 0.
Enter a integer from 0 to 4
1
Successfully borrow the book.
```

Borrow sample

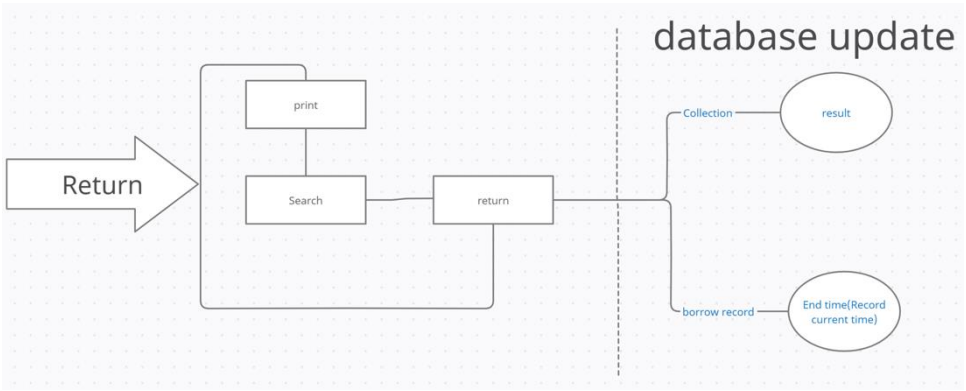
Like the ISBN borrow, this function requires the patron to input the BOOK_NAME and return a list of the books with the same name. This function is the same as the function above.

One thing that needs to be pointed out is that when there is no available copy, the patron can still borrow, and the function will call the reserve function which allows the patron to reserve it.

```
input the ISBN (a string with 13 number character e.g. 1234567890123):[input * quit]
9787810441063
The information of book with ISBN 9787810441063 are as follows
BOOK_INFORMATION
    ISBN           BOOKNAME              AVAILABLE_COPY  CATEGORY         AUTHOR_NAME
(1)-------------------------------------------------------------------------------------------
    9787810441063  HARRY_POTTER_7        0               TEENAGER_FANTASY J_K_ROLIN
Are you sure to choose book HARRY_POTTER_7?  1--yes 0 no
Enter a integer from 0 to 1
```

sample

3. Return the book




```
••••••••••••••••••••••••••••••••••••••••••••••••  You have 2 in progress borrow.  ••••••••••••••••••••
1,  book name: STAR_WARS   book id: 374628376479283  start from 2022-11-19
2,  book name: NARNIA_LEGEND_2   book id: 819427981847657  start from 2022-11-19
Please enter the index(start from 1) of book that you want to return. Enter 0 to cancel this operation
Enter a integer from 0 to 2
```

Below displays the change after the function is called, in the borrow record, the end time change from null to the day the book is returned.

3	6718467810	716473876473876	64273878p	7218475981205	2019-09-22	(null)

3	6718467810	716473876473876	64273878p	7218475981205	2019-09-22	2022-11-21

This function will first display all the books the patron has borrowed, patron is allowed to input the index to return it. After the return, the patron can still return more books.

```
Enter a integer from 0 to 2
1
The book have successfully returned.
Do you want to return another book? 1-- yes 0--no
Enter a integer from 0 to 1
1
••••••••••••••••••••••••••••••••••••••••••••••••  You have 1 in progress borrow.  ••••••••••••••••••••
1,  book name: NARNIA_LEGEND_2   book id: 819427981847657  start from 2022-11-19
Please enter the index(start from 1) of book that you want to return. Enter 0 to cancel this operation
Enter a integer from 0 to 1
```

When all the books have been returned, a reminder sentence will display on the screen.

```
Enter a integer from 0 to 1
1
The book have successfully returned.
All the books have been returned.

Leave return book
```

Remind sentence

4. Reserve the book

Two different reserve types:

1. Available reserve: when the related ISBN have available copies.

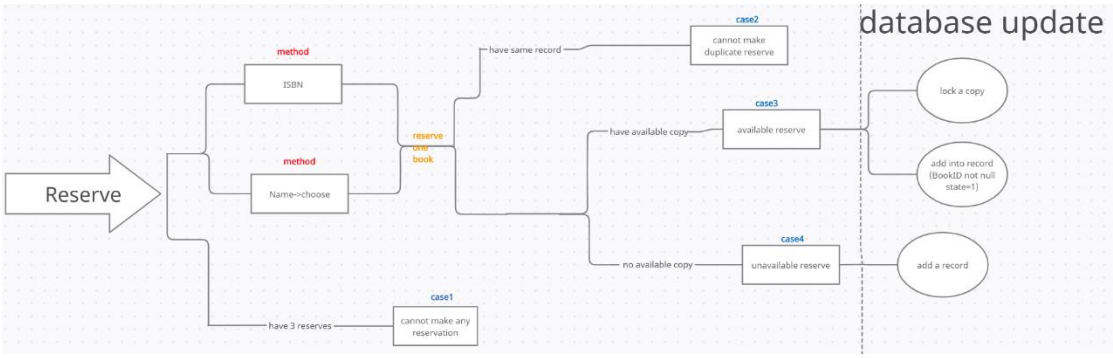
Function: Lock a copy for the user, wait for the user to pick the book, during this period, no one can take this book. The valid period lasts for 2 days, if the user doesn't come after two days, the record will be deleted automatically.

Database: when book_state not true, we will add a copy, when book_id not null, we will set the deadline after 2 days.

2.unavailable reserve: When no available copy at the reserve time.

Function: On waiting state, wait for someone to return the book which they have reserved, if there exists available copy, the user can transform the reserve to available reserve, and go to pick the book in 2 days. The record will be saved for 100 days, if the user transforms it to available reserve or there is no available copy until 100 days later, the record will be deleted automatically.

Database： book state – 0, book_id null, deadline 100 days after reserve day.



4.1 reserve by ISBN

In this function, the patron can input an ISBN to make the reserve, after inputting the ISBN, the function will list out the available reservations, (information contain: ISBN, BOOK_NAME, AVAILABLE_COPY, CATEGORY, AUTHOR_NAME, PUBLISHETR, LANGUAGE, PUBLISH_TIME, DESCRIPTION.) After the patron confirms the reservation, the record of it will be added to the system, the patron will see the record when he or she logs in and will create a new record in the system.

```
input the ISBN (a string with 13 number character e.g. 1234567890123):[input * quit]
9787810441063
The information of book with ISBN 9787810441063 are as follows
BOOK_INFORMATION
    ISBN           BOOKNAME              AVAILABLE_COPY  CATEGORY             AUTHOR_NAME                    PUBLISHER
(1)-------------------------------------------------------------------------------------------------------------------------------------------------
    9787810441063  HARRY_POTTER_7        2               TEENAGER_FANTASY J_K_ROLIN                      U_K_PUBLISHER
Are you sure to choose book HARRY_POTTER_7?  1--yes 0 no
Enter a integer from 0 to 1
1
We will lock the copy for two days for you come to catch. If you do not come to borrow within two days, this reserve will become invalid.
```

When there is an available copy in the collection which meet the needs of the patron's reserve, this function will lock the book for the patron about 2 days (state change from "1" to "0"), after 2 days, if the patron still not come and get the book, the lock will be delete automatically. (state from "0" to "1")

Below is the example of the change of database system after the reserve is successfully performed:

1. Book state change:

26	781467813764732	3698167615698	1	KANSARS_STATE_33RD_FLOOR

26	781467813764732	3698167615698	0	KANSARS_STATE_33RD_FLOOR

2. Add a reserve record:
A record will be created, and the deadline will be set two days later for the reservation to be made.

8	839476930803878	0983246598726	82762993p	2020-10-19	1
9	278676473829763	3698167615698	27482165p	2018-02-01	1

8	839476930803878	0983246598726	82762993p	2020-10-19	1
9	278676473829763	3698167615698	27482165p	2018-02-01	1
10	781467813764732	3698167615698	64273878p	2022-11-27	1

This function related with many conditions which are listed below:
(1) The patron already has three in progress reserves.

By this condition, the patron must first deal with the in-progress reserves. The function will display the reminder sentence.

```
............................................... You have 3 in progress reserves. ...................................
Your reserve for book STAR_WARS    have been canceled because you do not come to catch the book two days after reserve.
The book HARRY_POTTER_7    reserved by you is available now!
The book DISAPPEARING_TOMATO    reserved by you is still unavailable. We will notify you when there is available copy.
```

```
You are 3 in progress reserve. We only allow 3 reserve at most.

Leave reserve the book
```

sample

(2) The patron reserves a same book

```
............................................... You have 3 in progress reserves. ...................................
Your reserve for book STAR_WARS    have been canceled because you do not come to catch the book two days after reserve.
The book HARRY_POTTER_7    reserved by you is available now!
```

(The patron already reserves the book HARRY_POTTER_7)

```
input the ISBN (a string with 13 number character e.g. 1234567890123):[input * quit]
9787810441063
The information of book with ISBN 9787810441063 are as follows
BOOK_INFORMATION
    ISBN           BOOKNAME              AVAILABLE_COPY  CATEGORY              AUTHOR
(1)-------------------------------------------------------------------------------------
    9787810441063  HARRY_POTTER_7        3               TEENAGER_FANTASY J_K_ROLIN
Are you sure to choose book HARRY_POTTER_7?  1--yes 0 no
Enter a integer from 0 to 1
1
You have already made the same reserve.
```

(3) All the books the patron wants to reserve are unavailable.

By this condition, the system will record the patron's action, when other patron give up to reserve or return the book with same name, the system will remind the patron when he or she is log in. (the period will last 100 days)

```
If the book is available, we will lock a collection for you.
```

```
You are supposed to come to borrow that book within two days.
```

```
If the book is not available now, we will notify you until a collection is available.
```

Sample reminds

4.2 reserve by BOOK_NAME

In this function, the patron can input a BOOK_NAME to make the reservation, after inputting the BOOK_NAME, the function will list out all the available

reservations.

```
input the BOOK_NAME (a string have 1 to 50 characters e.g. Harry Potter):[input * quit]
STAR_WARS
The information of book STAR_WARS are as follows
BOOK_INFORMATION
    ISBN           BOOKNAME             AVAILABLE_COPY  CATEGORY             AUTHOR_NAME
(1)-------------------------------------------------------------------------------------------------
    57387T5782577  STAR_WARS            5               SCIENCE_FICTION  BENJAMIN_LUCAS
(2)-------------------------------------------------------------------------------------------------
    7231648727464  STAR_WARS            0               SCIENCE_FICTION  BENJAMIN_LUCAS
(3)-------------------------------------------------------------------------------------------------
    8723657843675  STAR_WARS            0               SCIENCE_FICTION  BENJAMIN_LUCAS
(4)-------------------------------------------------------------------------------------------------
    4641233494645  STAR_WARS            0               SCIENCE_FICTION  BENJAMIN_LUCAS
Enter the index (start from 1)of book  that you want. If you want to cancel the borrow, enter 0.
Enter a integer from 0 to 4
```

List sample

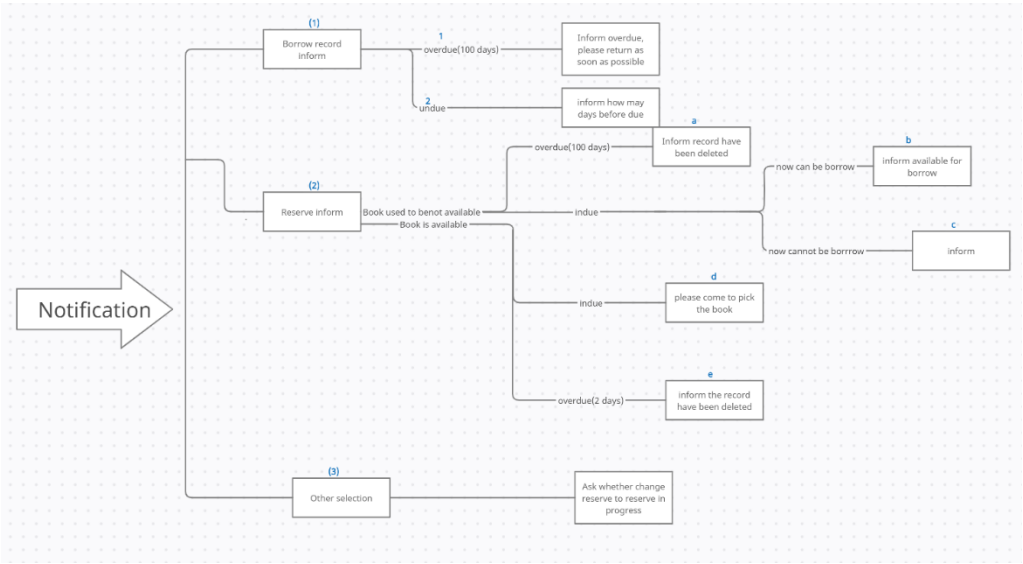
Except the display list, this function has the same idea as the function above.

5, notification

When the patron log in to the system, the system will analyze the in progress borrows and in progress reserves of the patron, then delivers the notification.
There is a sample of the notification.

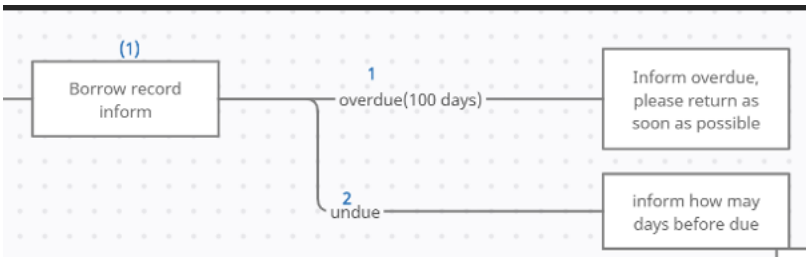
```
Hi, COOK_BUSH.  Welcome to library system.

Notification

You have 2 in progress borrow.
1,  book name: STAR_WARS   book id: 716481174119742  start from 2022-09-11      You still have 25 days left to return the book
2,  book name: HARRY_POTTER_7   book id: 237468723623678  start from 2020-04-11      This borrow is overdue, Please return it now!

You have 3 in progress reserves.
Your reserve for book STAR_WARS    have been canceled because you do not come to catch the book two days after reserve.
The book HARRY_POTTER_7    reserved by you is available now!
The book SAN_LUCIA    reserved by you is available now!
```

The flow chart of the notification function is as follows.



The notification includes two part: message about borrow and messages about reserves.
After delivering the message, the system will interact with patron to deal with a particular condition: the book that reserved by patron is available now.
Let's talk about these three parts one by one.

A, notification about the in progress borrow.

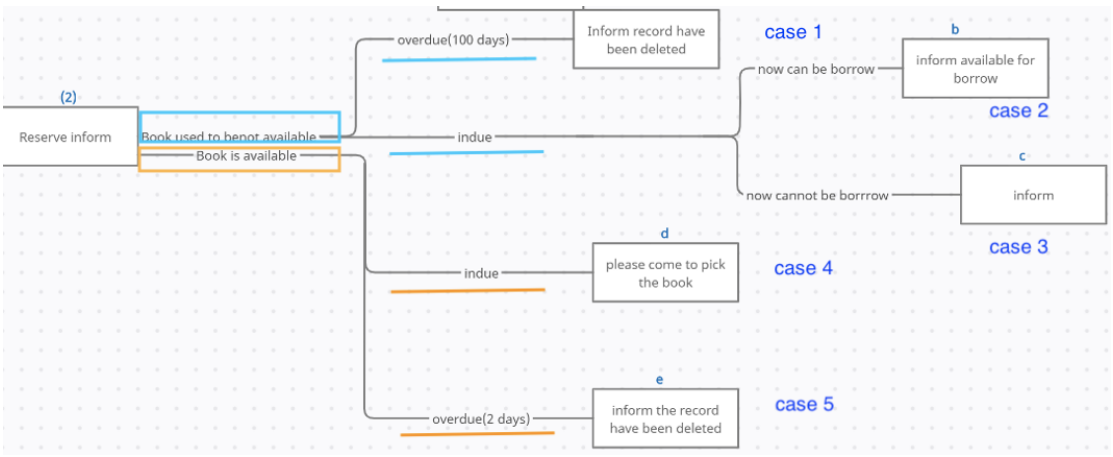


If the patron do not have in progress borrow, the system will just print out " You have no in progress borrow."
Otherwise, the system will analyze every borrow records of that patron and deliver the message about it one by one: There are 2 cases of the in progress borrow:
Case 1: the patron have overdue in progress borrow ---- give a warning and remind to return it as soon as possible.
Case 2: the patron have normal in progress borrow ---- just remind it there are how many days before the deadline.
Two examples:

```
••••••••••••••••••••••••••••••••••••••••••••••••••  You have 2 in progress borrow.  ••••••••••••••••••••••••••••••••••••••••••••••••••
1,  book name: STAR_WARS   book id: 716481174119742  start from 2022-09-11     You still have 25 days left to return the book
2,  book name: HARRY_POTTER_7   book id: 237468723623678  start from 2020-04-11     This borrow is overdue, Please return it now!
```

```
••••••••••••••••••••••••••••••••••••••••••••••••••  You have 2 in progress borrow.  ••••••••••••••••••••••••••••••••••••••••••••••••••
1,  book name: SAN_LUCIA   book id: 716473876473876  start from 2019-09-22     This borrow is overdue, Please return it now!
2,  book name: HARRY_POTTER_7   book id: 127346723876762  start from 2022-09-22     You still have 36 days left to return the book
```

B, messages about in progress reserve
If the patron do not have in progress reserve, the system will just print out " You have no in progress reserve."
Otherwise, the system will analyze every reserve records of that patron and deliver the message about it one by one: There are 5 cases of the in progress reserves: The condition of every case and the corresponding reactions of the system are as follows



There are two examples:

```
You have 3 in progress reserves.                                    case 5
Your reserve for book STAR_WARS    have been canceled because you do not come to catch the book two days after reserve.
The book HARRY_POTTER_7    reserved by you is available now!                                    case 2
Your reserve for HARRY_POTTER_7    have been canceled, because the system will only save the reserve record for 100 days.
Sorry about that!                                                                               case 1
```

```
You have 2 in progress reserves.                                                    case 4
You have a in progress reserve for book HARRY_POTTER_7.    You are supposed to come to catch the book before 2022-11-27
Your reserve for STAR_WARS    have been canceled, because the system will only save the reserve record for 100 days.    case 1
Sorry about that!
```

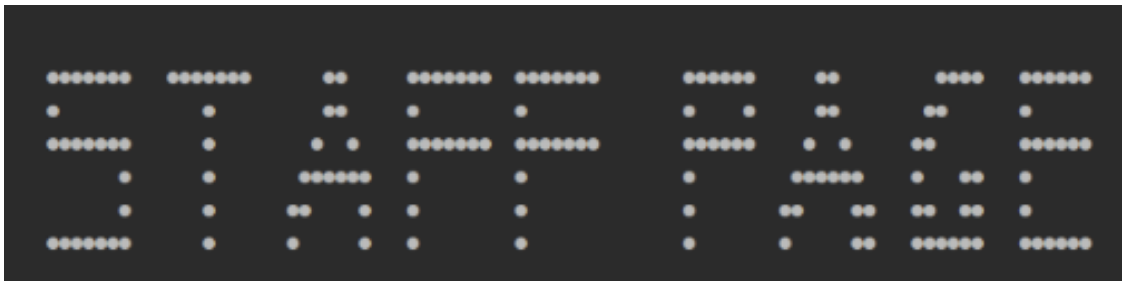
C, deal with the available books

If the book that reserved by patron is available now (case 2), after delivering the notification, the system will ask patron whether they want to come to catch that book.

```
The book HARRY_POTTER_7 is available now. Do you want to come to catch the book in 2 days? 1--yes, 0--no
Enter a integer from 0 to 1
0
The book SAN_LUCIA is available now. Do you want to come to catch the book in 2 days? 1--yes, 0--no
Enter a integer from 0 to 1
0
```

If the patron choose “yes”, the reserve turn to “available reserve”. (case 4)

[Staff Page]



```
Enter an integer to make the choose (enter 0 to quit):
1--manage book information / 2--manage collections of book /
3--manage patrons information / 4--manage borrow records
5--generate analysis reports
```

Under the staff page we have 5 main functions, these functions allow the staff to manage book information, collection of books, patron information, borrow records, generate analysis reports. Here we will introduce them one by one.

[manage book information]

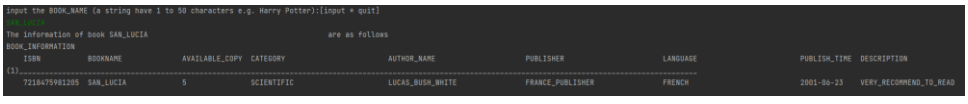
Under the manage book information function, we have three different operations. The following is the description of them.

```
Enter an integer to make the choose (enter 0 to quit):
1--check the book information / 2--update book information
3--add new book information / 4--delete book information
```

1. Check book information

```
-------------------------------------------------------  You can add several conditions to search for book information  ----------------------------
Enter an integer to make the choose (enter 0 to quit): 1-- search by book name / 2-- search by ISBN  3-- search by comprehensive condition
Enter a integer from 0 to 3
```

1.1 Search by book name.



```
input the BOOK_NAME (a string have 1 to 50 characters e.g. Harry Potter):[input * quit]
SAN_LUCIA
The information of book SAN_LUCIA                                    are as follows
BOOK_INFORMATION
    ISBN          BOOKNAME          AVAILABLE_COPY  CATEGORY              AUTHOR_NAME           PUBLISHER             LANGUAGE              PUBLISH_TIME  DESCRIPTION
(1)-------------------------------------------------------------------------------------------------------------------------------------------------
    7218475981205  SAN_LUCIA         5               SCIENTIFIC            LUCAS_BUSH_WHITE      FRANCE_PUBLISHER      FRENCH                2001-06-23    VERY_RECOMMEND_TO_READ
```

Input one to enter search by name

As it shows in the screenshot, you can see all the information of the book has already displayed on the table; by using this function, the user can know how many copies of that book are available in the library.

1.2 Search by ISBN

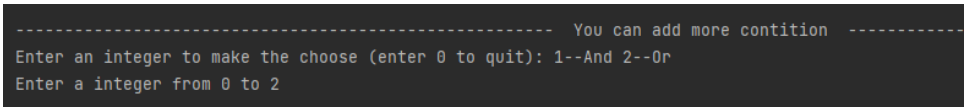
Input two to enter search by ISBN

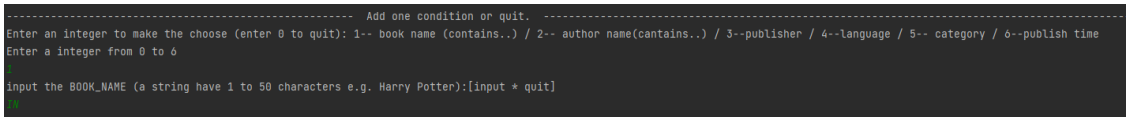
The above is the result after check by ISBN. It is somehow like checking by book name.

1.3 Search by comprehensive condition

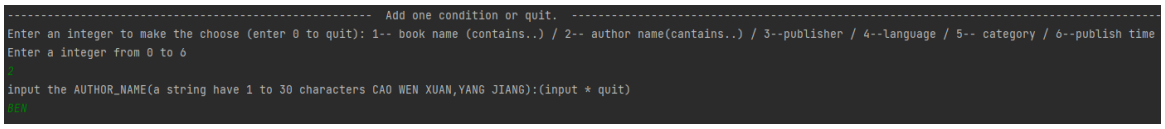
The search by comprehensive condition function supports combining 6 different conditions (book name, publisher, language, category, publish time) to find the book. For book name and author name they are special, you only need to input part of them then it can select those book name/author name, which contains the string that you have input.

Each time you need to first select one of those six conditions to input. Then choose and/or to define if the book you want to select must satisfy all the conditions you have set, or just need to full fill one of them.





Here we first choose to input part of the book name, as it shows in the image above, we select those books which book name contains “IN”.



Then for better demonstration we add a condition that the author's name contains “BEN”.

As here our condition combined is that the book name contains “IN” OR author name contains” BEN”. The results are displayed on the table.

Design point: Here the idea of design comprehensive search comes from the school library, as we have advance search to search for books satisfy multiple conditions, so here this function modeled the advance search.

2. update book information

```
********************************************************************** update the record **********************************************************************
Enter an integer to make the choose (enter 0 to quit): 1-- change the description of a book 2--change category name into another 3--update all information of a book
Enter a integer from 0 to 3
1
```

2.1 Change the description of a book

input 1 to use this function

```
************************************************** Enter the ISBN of book: ****************
input the ISBN (a string with 13 number character e.g. 1234567890123):[input * quit]
9365827626587
The original description is: A_WASTE_OF_TIME
************************************************** Please input the new description: *******
input the DESCRIPTION(a string have 1 to 500 characters:This is a nice book)(input * quit)
IT_IS_A_BOOK
```

By inputting the ISBN of the book, you can find the book which you want to change the ISBN. Here the original description is shows on the screen(“A_WAST_OF_TIME”), then we change it to “IT_IS_A_BOOK”.

```
The information of book with ISBN 9365827626587 are as follows
BOOK_INFORMATION
    ISBN          BOOKNAME            AVAILABLE_COPY  CATEGORY                  AUTHOR_NAME               PUBLISHER                    LANGUAGE                  PUBLISH_TIME  DESCRIPTION
(1)-------------------------------------------------------------------------------------------------------------------------------------------------------------------------
    9365827626587  BIG_CUCUMEMBER      2               SCIENTIFIC                LILY_BLACK                AYSTRILIA_QUEEN_PUBLISHER    POLYNISIAN                2000-11-21    IT_IS_A_BOOK
```

By searching in the library, we can see that the description of the book has already been changed.

2.2 Change the category of the book&&update all the information of a book

These two functions are somehow similar to the change description function, so here we are not going to demonstrate them.

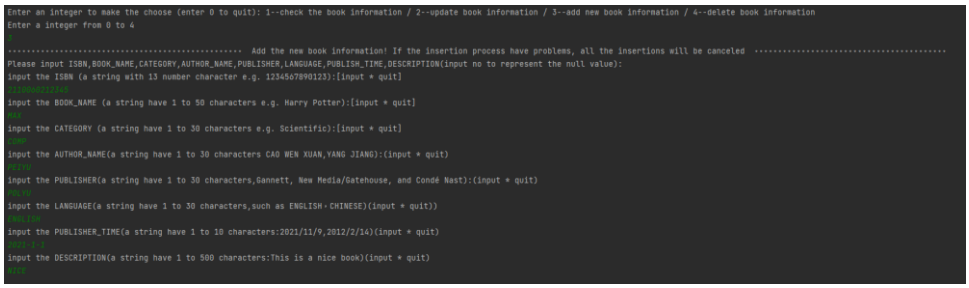
Design point: why we list the change description and category of a book individually, it is because other book information should be stable, as the book’s name, ISBN, publisher e.g., cannot keep update, only description and category can be defined by the users.

3. Add new book information

For add new books the staff needs to input all the information of that book, include e.g., ISBN, BOOK_NAME. We will show the usage of this function in the following part. The image below shows the table before we do the operation.

ISBN	BOOK_NAME	CATEGORY	AUTHOR_NAME	PUBLISHER	LANGUAGE	PUBLISH_TIME	DESCRIPTION
9787810441063	HARRY_POTTER_7	TEENAGER_FANTASY	J_K_ROLIN	U_K_PUBLISHER	ENGLISH	1999-09-09	WORTH_READING
2475947892847	HARRY_POTTER_7	TEENAGER_FANTASY	J_K_ROLIN	U_K_PUBLISHER	ENGLISH	1999-09-09	WORTH_READING
8573692876859	HARRY_POTTER_7	TEENAGER_FANTASY	J_K_ROLIN	U_K_PUBLISHER	ENGLISH	1999-09-09	WORTH_READING
4298748728748	HARRY_POTTER_7	TEENAGER_FANTASY	J_K_ROLIN	U_K_PUBLISHER	ENGLISH	1999-09-09	WORTH_READING
57387T5782577	STAR_WARS	SCIENCE_FICTION	BENJAMIN_LUCAS	USA_CENTRAL_PUBLISHER	SPANISH	1997-08-07	BORING
7231648727464	STAR_WARS	SCIENCE_FICTION	BENJAMIN_LUCAS	USA_CENTRAL_PUBLISHER	SPANISH	1997-08-07	BORING
8723657843675	STAR_WARS	SCIENCE_FICTION	BENJAMIN_LUCAS	USA_CENTRAL_PUBLISHER	SPANISH	1997-08-07	BORING
4641233494645	STAR_WARS	SCIENCE_FICTION	BENJAMIN_LUCAS	USA_CENTRAL_PUBLISHER	SPANISH	1997-08-07	BORING
7218475981205	SAN_LUCIA	HORROR	LUCAS_BUSH_WHITE	FRANCE_PUBLISHER	FRENCH	2001-06-23	VERY_RECOMMEND_TO_READ
8296587296274	NARNIA_LEGEND_2	JOURNAL_FANTASY	ALICE_MENLOW	LUXENBERG_PUBLISHER	CANADIAN	1950-11-19	HAVE_MANY_INTERESTING_SENCE:
0764397506748	INFORMAL	STORY	BUSH_JOHN	SALT_LAKE_PUBLISHER	SPANISH	1877-12-28	CONFUSING
9365827626587	BIG_CUCUMEMBER	HORROR	LILY_BLACK	AYSTRILIA_QUEEN_PUBLISHER	POLYNISIAN	2000-11-21	A_WASTE_OF_TIME
0983246598726	DISAPPEARING_TOMATO	NON_FICTION	P_Q_PERCY	JAPAN_PUBLISHER	JAPANESE	1990-02-25	QUITE_MEANINGFUL
3698167615698	JEEP_IN_DESERT	COURAGE_STORY	JOE_CATON	RUSSIA_MOSCOW_PUBLISHER	RUSSIAN	2006-04-22	NOTHING_EDUCATIONAL

These are the commands we used for adding new books. For easy demonstration we only add one book.



The image below shows the table after the update. We can see that the book “MAX” has already been added.

ISBN	BOOK_NAME	CATEGORY	AUTHOR_NAME	PUBLISHER	LANGUAGE	PUBLISH_TIME	DESCRIPTION
9787810441063	HARRY_POTTER_7	TEENAGER_FANTASY	J_K_ROLIN	U_K_PUBLISHER	ENGLISH	1999-09-09	WORTH_READING
2475947692847	HARRY_POTTER_7	TEENAGER_FANTASY	J_K_ROLIN	U_K_PUBLISHER	ENGLISH	1999-09-09	WORTH_READING
8573692876859	HARRY_POTTER_7	TEENAGER_FANTASY	J_K_ROLIN	U_K_PUBLISHER	ENGLISH	1999-09-09	WORTH_READING
4298748728748	HARRY_POTTER_7	TEENAGER_FANTASY	J_K_ROLIN	U_K_PUBLISHER	ENGLISH	1999-09-09	WORTH_READING
5738775782577	STAR_WARS	SCIENCE_FICTION	BENJAMIN_LUCAS	USA_CENTRAL_PUBLISHER	SPANISH	1997-08-07	BORING
7231648727464	STAR_WARS	SCIENCE_FICTION	BENJAMIN_LUCAS	USA_CENTRAL_PUBLISHER	SPANISH	1997-08-07	BORING
8723657643675	STAR_WARS	SCIENCE_FICTION	BENJAMIN_LUCAS	USA_CENTRAL_PUBLISHER	SPANISH	1997-08-07	BORING
4641233494645	STAR_WARS	SCIENCE_FICTION	BENJAMIN_LUCAS	USA_CENTRAL_PUBLISHER	SPANISH	1997-08-07	BORING
7218475961205	SAN_LUCIA	HORROR	LUCAS_BUSH_WHITE	FRANCE_PUBLISHER	FRENCH	2001-06-23	VERY_RECOMMEND_TO_READ
8296587296274	NARNIA_LEGEND_2	JOURNAL_FANTASY	ALICE_HENLOW	LUXEMBERG_PUBLISHER	CANADIAN	1950-11-19	HAVE_MANY_INTERESTING_SENCES
0764397506748	INFORMAL	STORY	BUSH_JOHN	SALT_LAKE_PUBLISHER	SPANISH	1877-12-28	CONFUSING
9365827626587	BIG_CUCUMEMBER	HORROR	LILY_BLACK	AYSTRILIA_QUEEN_PUBLISHER	POLYNISIAN	2000-11-21	A_WASTE_OF_TIME
0983246598726	DISAPPEARING_TOMATO	NON_FICTION	P_Q_PERCY	JAPAN_PUBLISHER	JAPANESE	1990-02-25	QUITE_MEANINGFUL
3698167415698	JEEP_IN_DESERT	COURAGE_STORY	JOE_CATON	RUSSIA_MOSCOW_PUBLISHER	RUSSIAN	2006-04-22	NOTHING_EDUCATIONAL
2110060212345	MAX	COMP	PEIYU	POLYU	ENGLISH	2021-1-1	NICE

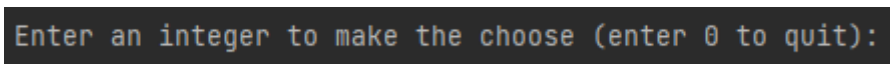
Design point: Here we allow the staff to keep adding books until they finish adding, they can choose to quit. It can benefit them a lot, as the library sometimes imports a lot of books, if they need to login repeatedly to add books, it will be more troublesome.

4. Delete book information:

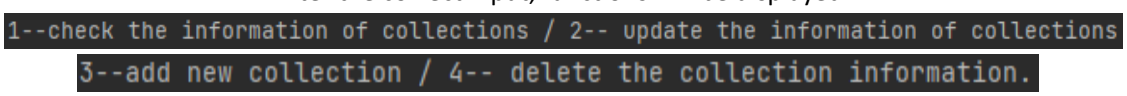
You can delete a book by using this function.
Here we are not going to detail, later, we will demonstrate delete function in the deletion of collection.

[manage collections]

Under this function, we have four main functions to manage the collections:



After the correct input, functions will be displayed



4 core functions

Four functions to manage the collections (1. Check information, 2. Update information, 3. Add collection, 4. Delete collection), also under the four core

functions, there are many subfunctions under the stage to help to detailly search the information we need (e.g., Check the information of collections by ISBN), which will display below.

1. Check information

1.1 Check information of a collection by BOOK_ID

```
input the BOOK_ID (a string with 15 number character e.g. 123456789054321):[input * quit]
716481174119742
The collection's information:
COLLECTION
    ISBN          BOOK_ID         STATE  LOCATION
(1)_____________________________________________________________________
    57387T5782577  716481174119742  false  POLYULIBRARY_2ND_FLOOR
The collection's related book information.
BOOK_INFORMATION
    ISBN          BOOKNAME             AVAILABLE_COPY  CATEGORY                      AUTHOR_NAME
(1)_____________________________________________________________________________________________________
    57387T5782577  STAR_WARS            6               SCIENCE_FICTION               BENJAMIN_LUCAS
```

Search sample

This function requires the staff to input the BOOK_ID to track a unique book and will display the collection information and Book information about it.

1.2 Check information of a collection by ISBN

```
input the ISBN (a string with 13 number character e.g. 1234567890123):[input * quit]
57387T5782577
The book with ISBN 57387T5782577 have 6 copies.
COLLECTION
    ISBN          BOOK_ID          STATE  LOCATION
(1)_____________________________________________________________________
    57387T5782577  673647739963676  false  CITYULIBRARY_12ND_FLOOR
(2)_____________________________________________________________________
    57387T5782577  273487236476219  false  KANGTERROOM_15TH_FLOOR
(3)_____________________________________________________________________
    57387T5782577  374628376479283  false  PKULIBRARY_13RD_FLOOR
(4)_____________________________________________________________________
    57387T5782577  421756784675923  false  LONGLIBRARY_BASEMENT
(5)_____________________________________________________________________
    57387T5782577  716481174119742  true   POLYULIBRARY_2ND_FLOOR
(6)_____________________________________________________________________
    57387T5782577  843175265783746  true   MYTH_LIBRARY_12ND_FLOOR
```

Search sample

In this function, we require the user to input the ISBN to search all the fit collections store in the library, they will display as a list after the correct input which contains ISBN, BOOK_ID, STATE and LOCATION information. (multiple books can have same ISBN)

1.3 Check information of a collection by LOCATION

```
input the LOCATION (a string has 1 to 30 characters e.g. A part 11 line):[input * quit]
CITYULIBRARY_12ND_FLOOR
The book at CITYULIBRARY_12ND_FLOOR        :
COLLECTION
    ISBN           BOOK_ID          STATE  LOCATION
(1)_____________________________________________________________________
    57387T5782577  673647739963676  false  CITYULIBRARY_12ND_FLOOR
```

Search sample

In this function, we require staff to input the BOOK_ID to search for a specific collection (BOOK_ID is unique to every book). It will display after the correct input which conations ISBN, BOOK_ID, STATE and LOCATION information.

1.4 Check unavailable collections

```
There are 5 unavailable books now.
COLLECTION
    ISBN           BOOK_ID          STATE  LOCATION
(1)_____________________________________________________________________
    57387T5782577  673647739963676  false  CITYULIBRARY_12ND_FLOOR
(2)_____________________________________________________________________
    57387T5782577  273487236476219  false  KANGTERROOM_15TH_FLOOR
(3)_____________________________________________________________________
    57387T5782577  374628376479283  false  PKULIBRARY_13RD_FLOOR
(4)_____________________________________________________________________
    57387T5782577  421756784675923  false  LONGLIBRARY_BASEMENT
(5)_____________________________________________________________________
    9787810441063  127346723876762  false  TSINGHUALIBRARY_28TH_FLOOR
```

Search sample

Due to this page is only used for staff, to simplify the operation, after they input “4” to call this function, it will directly display all the unavailable collections. The information shown contains ISBN, BOOK_ID, STATE and LOCATION.

2. Update the information of collections

2.1 Change the location of a book

```
input the BOOK_ID (a string with 15 number character e.g. 123456789054321):[input * quit]
673647739963676
The original location of the book is: CITYULIBRARY_7ND_FLOOR
input the LOCATION (a string has 1 to 30 characters e.g. A part 11 line):[input * quit]
CITYULIBRARY_12ND_FLOOR
Successfully change the location of the book
```

	BOOK_ID	ISBN	STATE	LOCATION
1	673647739963676	57387T5782577	1	CITYULIBRARY_7ND_FLOOR

	BOOK_ID	ISBN	STATE	LOCATION
1	673647739963676	57387T5782577	1	CITYULIBRARY_12ND_FLOOR

Sample

This function requires the staff input the BOOK_ID to track a unique book and will display the original location of it, after inputting a new location, we may check that this function has already changed the LOCATION of this book in the database system.

2.2 Active the state of a book

```
input the BOOK_ID (a string with 15 number character e.g. 123456789054321):[input * quit]
673647739963676
Successfully update the state of book.
```

	BOOK_ID	ISBN	STATE	LOCATION
1	673647739963676	57387T5782577	0	CITYULIBRARY_12ND_FLOOR

	BOOK_ID	ISBN	STATE	LOCATION
1	673647739963676	57387T5782577	1	CITYULIBRARY_12ND_FLOOR

Sample

This function requires the staff to input the BOOK_ID to track a unique book and will activate the available stage of it (from 0 to 1). If the book is already active, the function will return the remind sentence.

```
2734872364762l9
The collection is available now. No need to update.
```

Remind sentence

2.3 Inactive the state of a book

```
input the BOOK_ID (a string with 15 number character e.g. 123456789054321):[input * quit]
673647739963676
Successfully update the state of book.
```

	BOOK_ID	ISBN	STATE	LOCATION
1	673647739963676	57387T5782577	1	CITYULIBRARY_12ND_FLOOR

	BOOK_ID	ISBN	STATE	LOCATION
1	673647739963676	57387T5782577	0	CITYULIBRARY_12ND_FLOOR

Sample

This function requires the staff to input the BOOK_ID to track a unique book and will inactivate the available stage of it (from 1 to 0). If the book is already inactive, the function will return the remind sentence.

```
1273467238762762
The collection is not available. No need to update.
```

Remind sentence

3. Add new collection

```
input the ISBN (a string with 13 number character e.g. 1234567890123):[input * quit]
9365827626587
Please input COLLECTION information (input nothing to represent the null value):

The BOOK_ID have been generated by the system. which is 666877604086341
input the STATE (true / false):[input * quit]
true
input the LOCATION (a string has 1 to 30 characters e.g. A part 11 line):[input * quit]
NEW_YORK_LIBRARY_14TH_FLOOR
Do you want to add more collections ? 1--yes 0 -- no
Enter a integer from 0 to 1
0
Successfully add 1 collections

Leave add collection information
```

Addition sample

This function will receive an ISBN and check whether the ISBN is in the collection, if it does, which means we have copy collections in it, the system will automatically generate a BOOK_ID for it and let the staff set the other two information (STATE and LOCATION). This function also allows the staff to add multiple copies once he calls the function.

If the input ISBN is not in the collection information, which means we have not got any copy of this book, the staff will be required to first add a new ISBN.

```
input the ISBN (a string with 13 number character e.g. 1234567890123):[input * quit]
1234567890123
The book information have not been added to library. Please add the book information about this collection first!
```

Remind sentence

4. Delete the collection information

4.1 Delete by collecting's BOOK_ID

```
input the BOOK_ID (a string with 15 number character e.g. 123456789054321):[input * quit]
781467813764732
The collection information is as follows:
3698167615698  781467813764732  true   KANSARS_STATE_33RD_FLOOR
Are you sure to delete the information of this book? 1--yes, 0--no
Enter a integer from 0 to 1
1
Finish the operation. The information of the book and all its history borrow record have been deleted.
```

22	217465782194672	0764397506748	1	SHANXI_BOOKROOM_7TH_FLOOR
23	781467813764732	3698167615698	1	KANSARS_STATE_33RD_FLOOR

22	217465782194672	0764397506748	1	SHANXI_BOOKROOM_7TH_FLOOR

Delete sample

This function requires the staff to input the BOOK_ID to track a unique book and will delete it from the database system, but before the deletion, it will display all the information (ISBN, BOOK_ID, STATE, LOCATION from left to right). After the staff confirms, he can input “1” to delete the book or input “0” to cancel this deletion.

If the book waits to be deleted is now on loan, the deletion will be canceled automatically.

```
716481174119742
The collection information is as follows:
57387T5782577  716481174119742  false  POLYULIBRARY_2ND_FLOOR
There are reserves on this book in progress. You cannot delete this collection.
```

Remind sentence

4.2 Delete by collections ISBN

This function receives ISBN and will first list all the collections which have the same ISBN with all the information (ISBN, BOOK_ID, STATE, LOCATION). After the confirmation, the staff can input “1” to delete the list or input “0” to cancel the deletion. (Same with delete by BOOK_ID)

If one or more books in the list are now on loan, the deletion will be canceled automatically.

```
There are reserves on this book in progress. You cannot delete this collection.
```

Remind sentence

[manage patron’s information]

Under manage patron's information, we have two functions: [1] check the patron's information: we support the staff to select their id and name and check the inactivated patron account. [2] update the patron's information: we support the staff to record users' violations, activate the user account and inactivate all abnormal borrow records and inactivate the patrons making these borrow. We will demonstrate all these functions in the following part.

1. Check the patron information.

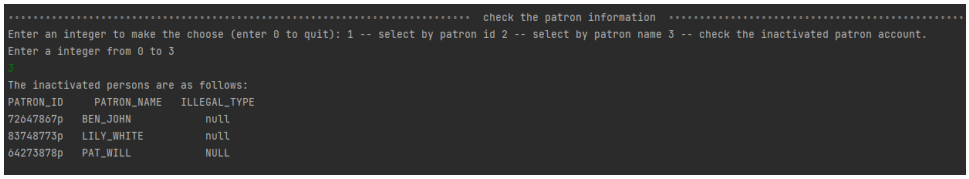
Here we support three ways for searching, we can search by using patron id, name, and check the inactivated patron account.

1.1 select by patron id and 1.2select by patron name

The usage of these two methods is somehow like the previous functions, so here we are not going to detail, just show an example of them.

1.3 check the inactivate patron account

The usage of this function is to check the patron whose state is in activate, in our database system, we use 0 to represent the patron's state which is inactivate.



Here these patrons are inactivating because they did not do anything after they created the account.

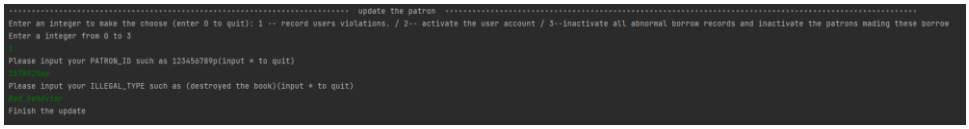
Design point: we built the function to make it more convenient for the staff to check which patron's accounts have been inactivated. In case they may ask our staff to check why their account has been inactivated, or the staff want to activate some patron's account, it will be easy for them to find the user.

2. Update patron's information

Here we provide three kinds of updates: record users violation, activate the user account, inactivate borrow records and inactivate all abnormal borrow records and inactivate the patrons making these borrow.

2.1 record violation

This function is used for the staff to record those patrons who made some illegal behaviors.

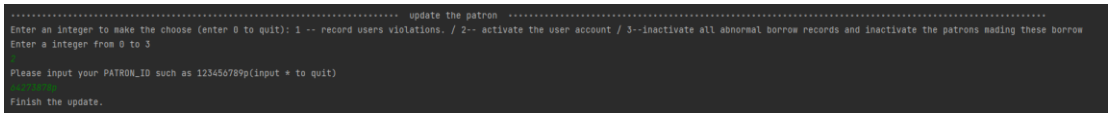


For demonstration we use the user ID to find a patron and change something in his illegal type. As it shows in the following image, by searching the information of the patron, we can see his illegal type has already been changed.

2.2 activate the user account

Now here PAT_WILL's account is inactivated.

In our system we use 1 to represent true value, which means activate, and 0 to represent false, which means inactivate.



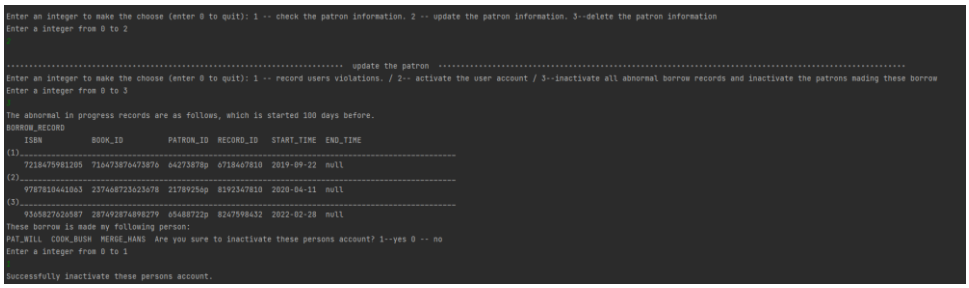
Here we use patron id to find the patron, and change its state to true, which means activate.

To see the change we have made, we can use the search function to find the user. From the image, we can see the state has already been changed, it is activated now.

2.3 inactivate all abnormal borrow records and inactivate the patrons making these borrows

This function is used for checking those users who made illegal borrow records and inactivate their borrow record.

The following image is the usage of it, it will ask you to confirm if you want to inactivate all the user in one go.



The inactivated persons are as follows:

PATRON_ID	PATRON_NAME	ILLEGAL_TYPE
64273878p	PAT_WILL	Overdue borrow
21789256p	COOK_BUSH	Overdue borrow
72647867p	BEN_JOHN	null
83748773p	LILY_WHITE	null
65488722p	MERGE_HANS	Overdue borrow

The following image shows that our change is succeed.

Design point: To create convenience for the staff, we create these three functions, by using the record violation function, we can record the illegal changes that have been made by the user. And by using the third function the staff can check who has an overdue borrow record, and the human design here is that we will ask them whether they want to make the change in case they mistakenly make some changes. If so, still you can use the activate function to change their state bake to normal.

3. delete the patron information

Here the staff can choose to delete the information of the patron. The following shows the usage of it.

Here is the table before deletion.

	PATRON_ID	USER_PASSWORD	PATRON_NAME	AGE	ACTIVITY_STATE	E_MAIL	GENDER	ADDRESS	ILLEGAL_TYPE	PHONE
1	64273878p	8216YH7501	PAT_WILL	23	1	ZHOUZHOU343@163.COM	MALE	GARDEN_ROAD_14TH	(null)	71645899
2	21789256p	TT742187	COOK_BUSH	9	1	COOKFOREVER@163.COM	MALE	MERDER_STREET_92RD	(null)	51276812
3	72647867p	HX19770202	BEN_JOHN	78	0	BIGBEN@163.COM	MALE	PRINCE_ROAD_33RD	(null)	73678291
4	83748773p	TV2505050	LILY_WHITE	93	0	LILYBEAUTY@163.COM	FEMALE	HANBERGER_STREET_17TH	(null)	47920184
5	82647821p	MM7671885	MAX_BLACK	102	1	MAXBCD7778@163.COM	MALE	MARY_ROAD_87TH	(null)	82163410
6	90278775p	DONGS99887	JOHN_SMITH	44	1	JSJS@163.COM	MALE	GOODMAN_ROAD_25TH	(null)	84217593
7	71264897p	1234599YJ	EMMA_JOE	46	1	EMMAGOOD11@163.COM	FEMALE	PHILIPHIES_STREET_88TH	(null)	82417983
8	82762993p	87654321SS	CATHRINE_MOMO	119	1	DUHONGHUA@163.COM	FEMALE	KING_ROAD_45TH	(null)	09428384
9	65488722p	LOG123456	MERGE_HANS	98	1	SIHAFG656@163.COM	MALE	PKU_ROAD_98TH	(null)	48297193
10	27482165p	PPT123490	AUTHOR_BARBECUE	27	1	BSGFIYUYI@163.COM	MALE	FORG_STREET_33RD	(null)	73647821

First, we will ask the staff to use the patron id to find the one they want to delete. Then in case they mistakenly made any operation, we will ask them to confirm the deletion.

After the deletion, the table becomes the image below.

to manage the borrow records:

	PATRON_ID	USER_PASSWORD	PATRON_NAME	AGE	ACTIVITY_STATE	E_MAIL	GENDER	ADDRESS	ILLEGAL_TYPE	PHONE
1	21789256p	TT742187	COOK_BUSH	9	1	COOKFOREVER@163.COM	MALE	MERDER_STREET_92RD	(null)	51276812
2	72647867p	HX19770202	BEN_JOHN	78	0	BIGBEN@163.COM	MALE	PRINCE_ROAD_33RD	(null)	73678291
3	83748773p	TV2505050	LILY_WHITE	93	0	LILYBEAUTY@163.COM	FEMALE	HANBERGER_STREET_17TH	(null)	47920184
4	82647821p	MM7671885	MAX_BLACK	102	1	MAXBCD7778@163.COM	MALE	MARY_ROAD_87TH	(null)	82163410
5	90278775p	DONGS99887	JOHN_SMITH	44	1	JSJS@163.COM	MALE	GOODMAN_ROAD_25TH	(null)	84217593
6	71264897p	1234599YJ	EMMA_JOE	46	1	EMMAGOOD11@163.COM	FEMALE	PHILIPHIES_STREET_88TH	(null)	82417983
7	82762993p	87654321SS	CATHRINE_MOMO	119	1	DUHONGHUA@163.COM	FEMALE	KING_ROAD_45TH	(null)	09428384
8	65488722p	LOG123456	MERGE_HANS	98	1	SIHAFG656@163.COM	MALE	PKU_ROAD_98TH	(null)	48297193
9	27482165p	PPT123490	AUTHOR_BARBECUE	27	1	BSGFIYUYI@163.COM	MALE	FORG_STREET_33RD	(null)	73647821

[manage borrow records]

Under this function, we have three main functions


```
Enter an integer to make the choose (enter 0 to quit):
```

After input, the functions will display

```
1 -- check the record. 2 -- update the record. 3 -- delete the record
```

3 core functions

Three functions to manage the collections (1. Check the record, 2. Update the record, 3. Delete the record), also under the three core functions, there are many subfunctions under the stage to help to detailly search the information we need (e.g., Check the records by RECORDS_ID), which will be displayed below.

1. Check the record

1.1 Check the records by RECORDS_ID

```
Please input the record_ID such as 1234567890(input * to quit)
6718467810
7218475981205  716473876473876  64273878p  6718467810  2019-09-22  null
```

Search sample

This function requires the staff to input the record_ID to track a unique record and will display the information about it (RECURD_ID, BOOK_ID, PATRON_ID, ISBN, START_TIME, END_TIME from left to right).

1.2 Check the records made by patron

(1) search by patron name

```
You can choose to input either the patron id or the patron name:  (0 to quit)  1 -- name  2 -- id
Enter a integer from 0 to 2
1
Please input the your PATRON_NAME(1 to 30 characters) such as FRANKLIN(input * to quit)
PAT_WILL
The record by patron with name PAT_WILL                    is as follows:BORROW_RECORD
    ISBN           BOOK_ID          PATRON_ID  RECORD_ID   START_TIME  END_TIME
(1)_____________________________________________________________________________________________
    7218475981205  716473876473876  64273878p  6718467810  2019-09-22  null
(2)_____________________________________________________________________________________________
    9787810441063  127346723876762  64273878p  8241578940  2022-09-22  null
(3)_____________________________________________________________________________________________
    57387T5782577  421756784675923  64273878p  3485679380  2019-11-22  2020-01-06
```

(2) search by patron id

```
You can choose to input either the patron id or the patron name:  (0 to quit)  1 -- name  2 -- id
Enter a integer from 0 to 2
2
Please input the PATRON_ID such as 12345678p(input * to quit)
21789256p
The record by patron with id 21789256p is as follows:BORROW_RECORD
    ISBN           BOOK_ID          PATRON_ID  RECORD_ID   START_TIME  END_TIME
(1)_____________________________________________________________________________________________
    57387T5782577  716481174119742  21789256p  7182461870  2022-09-11  null
(2)_____________________________________________________________________________________________
    9787810441063  237468723623678  21789256p  8192347810  2020-04-11  null
(3)_____________________________________________________________________________________________
    7218475981205  716473876473876  21789256p  7164871640  2017-08-13  2017-08-28
(4)_____________________________________________________________________________________________
    57387T5782577  716481174119742  21789256p  7164871646  2020-08-13  2020-08-28
(5)_____________________________________________________________________________________________
    57387T5782577  843175265783746  21789256p  7214976574  2021-09-18  2021-11-24
```

This function requires the staff to input the patron information (1. Name 2. ID) to track all the related records and return a list of them. (RECURD_ID, BOOK_ID, PATRON_ID, ISBN, START_TIME, END_TIME from left to right).

There will be an empty list is the patron's name or ID has no relation to all the records.

```
You can choose to input either the patron id or the patron name:  (0 to quit)  1 -- name  2 -- id
Enter a integer from 0 to 2
1
Please input the your PATRON_NAME(1 to 30 characters) such as FRANKLIN(input * to quit)
Franklin
The record by patron with name Franklin is as follows:BORROW_RECORD
    ISBN           BOOK_ID          PATRON_ID  RECORD_ID   START_TIME  END_TIME
```

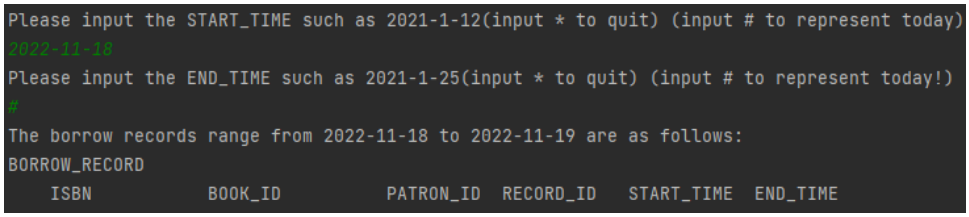
sample

1.3 Check in progress record

```
The in progress borrow records is as follows:
BORROW_RECORD
    ISBN           BOOK_ID          PATRON_ID  RECORD_ID   START_TIME  END_TIME
(1)_____________________________________________________________________________________________
    7218475981205  716473876473876  64273878p  6718467810  2019-09-22  null
(2)_____________________________________________________________________________________________
    9787810441063  127346723876762  64273878p  8241578940  2022-09-22  null
(3)_____________________________________________________________________________________________
    57387T5782577  716481174119742  21789256p  7182461870  2022-09-11  null
(4)_____________________________________________________________________________________________
    9787810441063  237468723623678  21789256p  8192347810  2020-04-11  null
(5)_____________________________________________________________________________________________
    8296587296274  819427981847657  82647821p  6234787367  2022-11-18  null
(6)_____________________________________________________________________________________________
    0764397506748  423976982002757  82647821p  1067487146  2022-09-27  null
(7)_____________________________________________________________________________________________
    3698167615698  278676473829763  65488722p  7246762876  2022-11-06  null
(8)_____________________________________________________________________________________________
    9365827626587  287492874898279  65488722p  8247598432  2022-02-28  null
```

A simple function returns all the in-progress records when called.

1.4 Check the record of a particular time range

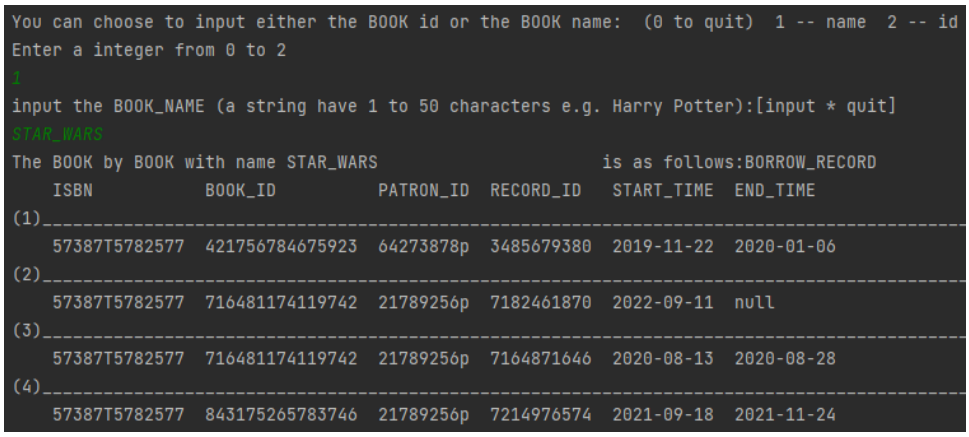


Search sample

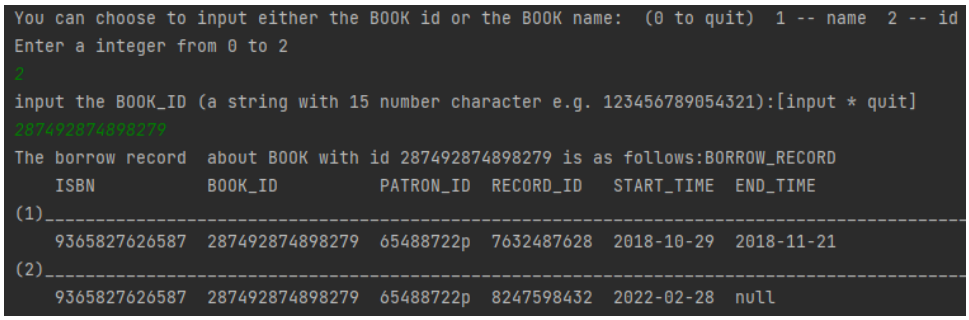
This function required the staff input two certain times (YYYY-MM-DD, "#" to represented today's time). After two times all past the efficiency test, this function will display all the records in the period.

1.5 Check the record about one collection

(1) Search by Book name

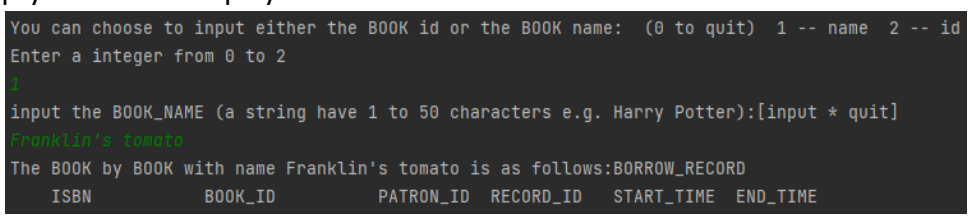


(2) Search by Book ID



Like the search for the patron's information, this function allowed the staff to search the record by Book name OR ID (1. BOOK_NAME 2. BOOK_ID), After the input, all the related records will be displayed. (RECURD_ID, BOOK_ID, PATRON_ID, ISBN, START_TIME, END_TIME from left to right).

Empty list will be displayed when the name or ID cannot be found.



sample

1.6 Check the abnormal in process record

```
The overdue in progress borrow record are as follows:
BORROW_RECORD
    ISBN           BOOK_ID          PATRON_ID  RECORD_ID   START_TIME  END_TIME
(1)_____________________________________________________________________________
    7218475981205  716473876473876  64273878p  6718467810  2019-09-22  null
(2)_____________________________________________________________________________
    9787810441063  237468723623678  21789256p  8192347810  2020-04-11  null
(3)_____________________________________________________________________________
    9365827626587  287492874898279  65488722p  8247598432  2022-02-28  null
```

A simple function returns all the abnormal records when called.

2. update the record

The usage of this function is to terminate the in progress borrow record. The following is the demonstration of its progress. As the end time is null, it is an in-progress record.

Before:

1	6718467810	716473876473876	64273878p	7218475981205	2019-09-22	(null)

Commands:

```
Enter an integer to make the choose (enter 0 to quit): 1 -- check the record. 2 -- update the record. 3 -- delete the record
Enter a integer from 0 to 3
2

************************************************************************ update the borrow record ************************************************
Enter an integer to make the choose (enter 0 to quit): 1 -- end the in progress record.
Enter a integer from 0 to 1
1
Please input the record_ID such as 1234567890(input * to quit)
6718467810
Successfully end the borrow.
```

After:

1	6718467810	716473876473876	64273878p	7218475981205	2019-09-22	2022-11-19

As the image shows the end date has been set to today, which means it has been terminated.

3. delete the record

3.1 delete record before a particular time.

Below is the table of borrowing records before we made the deletion.

	RECORD_ID	BOOK_ID	PATRON_ID	ISBN	START_TIME	END_TIME
1	6718467810	716473876473876	64273878p	7218475981205	2019-09-22	(null)
2	8241578940	127346723876762	64273878p	9787810441063	2022-09-22	(null)
3	3485679380	421756784675923	64273878p	57387T5782577	2019-11-22	2020-01-06
4	7182461870	716481174119742	21789256p	57387T5782577	2022-09-11	(null)
5	8192347810	237468723623678	21789256p	9787810441063	2020-04-11	(null)
6	7164871640	716473876473876	21789256p	7218475981205	2017-08-13	2017-08-28
7	6234787367	819427981847657	82647821p	8296587296274	2022-11-18	(null)
8	1067487146	423976982002757	82647821p	0764397506748	2022-09-27	(null)
9	7246762876	278676473829763	65488722p	3698167615698	2022-11-06	(null)
10	7632487628	287492874898279	65488722p	9365827626587	2018-10-29	2018-11-21
11	8247598432	287492874898279	65488722p	9365827626587	2022-02-28	(null)
12	7216438716	423976982002757	71264897p	0764397506748	2019-02-28	2019-04-22
13	8937629657	343976927667759	82762993p	9365827626587	2018-11-29	2020-03-15
14	4817496165	839476930803878	82762993p	0983246598726	2020-05-24	2020-09-16
15	8275902508	278676473829763	27482165p	3698167615698	2018-10-07	2019-02-03
16	7164871646	716481174119742	21789256p	57387T5782577	2020-08-13	2020-08-28
17	7214976574	843175265783746	21789256p	57387T5782577	2021-09-18	2021-11-24
18	7983641546	082758275764655	82647821p	7218475981205	2021-02-16	2021-04-30
19	8192048767	819427981847657	65488722p	8296587296274	2020-11-04	2020-12-01
20	8213579412	217465782194672	71264897p	0764397506748	2018-04-07	2018-11-03
21	8743816477	198745423657898	82762993p	9365827626587	2019-07-22	2019-04-23
22	8921643718	781467813764732	27482165p	3698167615698	2019-08-19	2021-03-25
23	8364721864	198745423657898	65488722p	8296587296274	2019-12-14	2020-05-09
24	6421784368	278676473829763	27482165p	3698167615698	2019-02-28	2020-04-22
25	1284789724	423976982002757	71264897p	0764397506748	2018-06-17	2018-12-06
26	8731642876	423976982002757	71264897p	0764397506748	2019-08-18	2019-09-27

For easy demonstration, we just delete the records before today, which means deleting all the records except those in progress.

```
delete the borrow record
Enter an integer to make the choose (enter 0 to quit): 1 -- delete record before a particular time. 2 -- delete record by record id 3--delete records about a collection
Enter a integer from 0 to 3

Please input the END_TIME such as 2021-1-25(input * to quit) (input # to represent today!)

Successfully perform the deletion.
```

The following image shows the results after deletion.

	RECORD_ID	BOOK_ID	PATRON_ID	ISBN	START_TIME	END_TIME
1	6718467810	716473876473876	64273878p	7218475981205	2019-09-22	(null)
2	8241578940	127346723876762	64273878p	9787810441063	2022-09-22	(null)
3	7182461870	716481174119742	21789256p	57387T5782577	2022-09-11	(null)
4	8192347810	237468723623678	21789256p	9787810441063	2020-04-11	(null)
5	6234787367	819427981847657	82647821p	8296587296274	2022-11-18	(null)
6	1067487146	423976982002757	82647821p	0764397506748	2022-09-27	(null)
7	7246762876	278676473829763	65488722p	3698167615698	2022-11-06	(null)
8	8247598432	287492874898279	65488722p	9365827626587	2022-02-28	(null)

Design point: this function is used to help the staff keep updating the data in the library, in case there may be a lot of old records in the library system. It is too troublesome to delete them one by one, so by ignoring those records before a particular day, it will be much more convenient.

3.2 delete record by record id

To delete a particular record, the staff needs to use record id. Because it is somehow similar to the previous function, here we are not going into detail. Just show the steps Before:

6718467810	7164738764738 76	64273878p	7218475981205	2019-09-22	(null)
8241578940	127346723876762	64273878p	9787810441063	2022-09-22	(null)
7182461870	716481174119742	21789256p	57387T5782577	2022-09-11	(null)
8192347810	237468723623678	21789256p	9787810441063	2020-04-11	(null)
6234787367	819427981847657	82647821p	8296587296274	2022-11-18	(null)
1067487146	423976982002757	82647821p	0764397506748	2022-09-27	(null)
7246762876	278676473829763	65488722p	3698167615698	2022-11-06	(null)
8247598432	287492874898279	65488722p	9365827626587	2022-02-28	(null)

Commands:

```
delete the borrow record
Enter an integer to make the choose (enter 0 to quit): 1 -- delete record before a particular time. 2 -- delete record by record id 3--delete records about a collection
Enter a integer from 0 to 3
3
Please input the record_ID such as 1234567890(input * to quit)
7164738764738
Successfully perform the deletion.
```

After:

RECORD_ID	BOOK_ID	PATRON_ID	ISBN	START_TIME	END_TIME
8241578940	127346723876762	64273878p	9787810441063	2022-09-22	(null)
7182461870	716481174119742	21789256p	57387T5782577	2022-09-11	(null)
8192347810	237468723623678	21789256p	9787810441063	2020-04-11	(null)
6234787367	819427981847657	82647821p	8296587296274	2022-11-18	(null)
1067487146	423976982002757	82647821p	0764397506748	2022-09-27	(null)
7246762876	278676473829763	65488722p	3698167615698	2022-11-06	(null)
8247598432	287492874898279	65488722p	9365827626587	2022-02-28	(null)

Design point: As there might be some invalids borrow records stored in the system, the staff need to delete it, they can use this function.

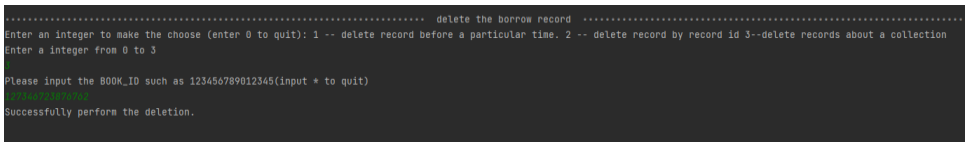
3.3 delete records about a collection

To delete the record of a particular collection, the staff needs to use the collection name. Because it is somehow similar to the previous function, here we are not going into detail. Just show the steps

Before:

	RECORD_ID	BOOK_ID	PATRON_ID	ISBN	START_TIME	END_TIME
1	6718467810	7164738764738 76	64273878p	7218475981205	2019-09-22	(null)
2	8241578940	127346723876762	64273878p	9787810441063	2022-09-22	(null)
3	3485679380	421756784675923	64273878p	57387T5782577	2019-11-22	2020-01-06
4	7182461870	716481174119742	21789256p	57387T5782577	2022-09-11	(null)
5	8192347810	237468723623678	21789256p	9787810441063	2020-04-11	(null)
6	7164871640	7164738764738 76	21789256p	7218475981205	2017-08-13	2017-08-28
7	6234787367	819427981847657	82647821p	8296587296274	2022-11-18	(null)
8	1067487146	423976982002757	82647821p	0764397506748	2022-09-27	(null)
9	7246762876	278676473829763	65488722p	3698167615698	2022-11-06	(null)
10	7632487628	287492874898279	65488722p	9365827626587	2018-10-29	2018-11-21
11	8247598432	287492874898279	65488722p	9365827626587	2022-02-28	(null)
12	7216438716	423976982002757	71264897p	0764397506748	2019-02-28	2019-04-22
13	8937629657	343976927667759	82762993p	9365827626587	2018-11-29	2020-03-15
14	4817496165	839476930803878	82762993p	0983246598726	2020-05-24	2020-09-16
15	8275902508	278676473829763	27482165p	3698167615698	2018-10-07	2019-02-03
16	7164871646	716481174119742	21789256p	57387T5782577	2020-08-13	2020-08-28
17	7214976574	843175265783746	21789256p	57387T5782577	2021-09-18	2021-11-24
18	7983641546	082758275764655	82647821p	7218475981205	2021-02-16	2021-04-30
19	8192048767	819427981847657	65488722p	8296587296274	2020-11-04	2020-12-01
20	8213579412	217465782194672	71264897p	0764397506748	2018-04-07	2018-11-03
21	8743816477	198745423657898	82762993p	9365827626587	2019-07-22	2019-04-23
22	8921643718	781467813764732	27482165p	3698167615698	2019-08-19	2021-03-25
23	8364721864	198745423657898	65488722p	8296587296274	2019-12-14	2020-05-09
24	6421784368	278676473829763	27482165p	3698167615698	2019-02-28	2020-04-22
25	1284789724	423976982002757	71264897p	0764397506748	2018-06-17	2018-12-06
26	8731642876	423976982002757	71264897p	0764397506748	2019-08-18	2019-09-27

Commands:



After:

	RECORD_ID	BOOK_ID	PATRON_ID	ISBN	START_TIME	END_TIME
1	6718467810	716473876473876	64273878p	7218475981205	2019-09-22	(null)
2	3485679380	421756784675923	64273878p	57387T5782577	2019-11-22	2020-01-06
3	7182461870	7164811741119742	21789256p	57387T5782577	2022-09-11	(null)
4	8192347810	237468723623678	21789256p	9787810441063	2020-04-11	(null)
5	7164871640	716473876473876	21789256p	7218475981205	2017-08-13	2017-08-28
6	6234787367	819427981847657	82647821p	8296587296274	2022-11-18	(null)
7	1067487146	423976982002757	82647821p	0764397506748	2022-09-27	(null)
8	7246762876	278676473829763	65488722p	3698167615698	2022-11-06	(null)
9	7632487628	287492874898279	65488722p	9365827626587	2018-10-29	2018-11-21
10	8247598432	287492874898279	65488722p	9365827626587	2022-02-28	(null)
11	7216438716	423976982002757	71264897p	0764397506748	2019-02-28	2019-04-22
12	8937629657	343976927667759	82762993p	9365827626587	2018-11-29	2020-03-15
13	4817496165	839476930803878	82762993p	0983246598726	2020-05-24	2020-09-16
14	8275902508	278676473829763	27482165p	3698167615698	2018-10-07	2019-02-03
15	7164871646	7164811741119742	21789256p	57387T5782577	2020-08-13	2020-08-28
16	7214976574	843175265783746	21789256p	57387T5782577	2021-09-18	2021-11-24
17	7983641546	082758275764655	82647821p	7218475981205	2021-02-16	2021-04-30
18	8192048767	819427981847657	65488722p	8296587296274	2020-11-04	2020-12-01
19	8213579412	2174657821946 72	71264897p	0764397506748	2018-04-07	2018-11-03
20	8743816477	198745423657898	82762993p	9365827626587	2019-07-22	2019-04-23
21	8921643718	781467813764732	27482165p	3698167615698	2019-08-19	2021-03-25
22	8364721864	198745423657898	65488722p	8296587296274	2019-12-14	2020-05-09
23	6421784368	278676473829763	27482165p	3698167615698	2019-02-28	2020-04-22
24	1284789724	423976982002757	71264897p	0764397506748	2018-06-17	2018-12-06
25	8731642876	423976982002757	71264897p	0764397506748	2019-08-18	2019-09-27

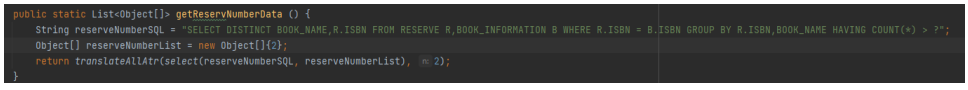
Overall design of deletion: We won't generate error if you input wrong id or time, that is for better experience for the staff.

Analysis report

Our LMS also can provide analysis reports to management to review the system. We designed 5 different reports, if the staff type 5 in the staff page, the staff can see the analysis report.

I. Book information analysis

a. Analyze which are the popular books



```
public static List<Object[]> getReservNumberData () {
    String reserveNumberSQL = "SELECT DISTINCT BOOK_NAME,R.ISBN FROM RESERVE R,BOOK_INFORMATION B WHERE R.ISBN = B.ISBN GROUP BY R.ISBN,BOOK_NAME HAVING COUNT(*) > ?";
    Object[] reserveNumberList = new Object[]{2};
    return translateAllAtr(select(reserveNumberSQL, reserveNumberList), n: 2);
}
```

```
List<Object[]> reserveNumberData = GeneralQuery.getReservNumberData();
System.out.println("Statistical Table the Popular Books");
System.out.println("Book name                      ISBN");
System.out.println("------------------------------ ------------");
for (Object[] array : reserveNumberData) {
    System.out.printf("%-30s %-13s\n", array[1], array[0]);
}
System.out.println();
```

Many patrons may want to borrow the same books at the same time, but the books are not enough. So according to the borrow records, we want to analysis the which are popular books. By selecting the books which will be borrowed twice, we get this report. Of course, 2 times is too small because we don't have too much data, but the staff can set any number in the real situation. Book names and ISBNs are both shown in this report.

```
Book information analysis

Statistical Table the Popular Books
Book name                      ISBN
------------------------------ ------------
7218475981205                  SAN_LUCIA
```

This report shows which books are popular during these years and helps to decide whether our library can buy more of these books or organize some promotions for these books. There are many choices to use these data.

II. **Book analysis**

```
Vector<COLLECTION> unavailable = COLLECTIONTable.select_unavailable_COLLECTION();
Vector<COLLECTION> available = COLLECTIONTable.select_available_COLLECTION();
int numOfUnavailable = unavailable.size();
int numOfAvalaible = available.size();
int total = numOfAvalaible  + numOfUnavailable;
System.out.println("The library has " + total + " books in total");
System.out.println(numOfAvalaible + " of them are available now. count for " + percentage( number: numOfAvalaible/(double) total));
System.out.println(numOfUnavailable + " of them are unavailable. count for " + percentage( number: (numOfUnavailable) / (double) total));
System.out.println();
```

First, we write some codes to count what percentage books are available and unavailable. The sample is shown below, not too many books in this example.

```
book analysis

The library has 27 books in total
15 of them are available now. count for 55.56%
12 of them are unavailable. count for 44.44%
```

a. **Count the number of the copies**

```
public static List<Object[]> getCopyCountAndName () {
    String copySQL = "SELECT DISTINCT BOOK_NAME,C.ISBN,COUNT(BOOK_ID) FROM COLLECTION C,BOOK_INFORMATION B WHERE C.ISBN = B.ISBN GROUP BY C.ISBN,BOOK_NAME ORDER BY COUNT(BOOK_ID) DESC";
    Object[] copyList = new Object[]{};
    return translateAllAtr(select(copySQL, copyList), n: 3);
```

```java
List<Object[]> copyData = GeneralQuery.getCopyCountAndName();

System.out.println("Statistical Table of the Number of Copies of Books");
System.out.println("Book Name                    ISBN          The Number of Copies");
System.out.println("---------------------------- ------------ --------------------");
for (Object[] array : copyData) {
    System.out.printf("%-30s %-13s %-20s\n", array[0], array[1], array[2]);
}
System.out.println();
```

There are maybe many books having the same ISBN. Because the data updates frequently, and to avoid data redundancy, we don't set an attribute to store the number of copies. However, we always need this kind of data. By group by ISBN and count the number, we can get this report. Book names, ISBNS and the numbers of the copies are shown in descending order.

```
Statistical Table of the Number of Copies of Books
Book Name                    ISBN          The Number of Copies
---------------------------- ------------ --------------------
STAR_WARS                    57387T5782577 7
SAN_LUCIA                    7218475981205 5
HARRY_POTTER_7               9787810441063 5
BIG_CUCUMEMBER               9365827626587 3
JEEP_IN_DESERT               3698167615698 2
INFORMAL                     0764397506748 2
NARNIA_LEGEND_2              8296587296274 2
DISAPPEARING_TOMATO          0983246598726 1
```

This report can help the staff to browse the overall number of books and make decisions about the purchase status later.

b. Analyze which are the old books which aren't welcomed

```java
List<Object[]> oldBookData = GeneralQuery.getOldBookData();

System.out.println("Statistical Table of the Information of the Old Books Which Aren't Welcomed");
System.out.println("Book name                    ISBN          Publish Time");
System.out.println("---------------------------- ------------ ------------");
for (Object[] array : oldBookData) {
    System.out.printf("%-30s %-13s %-12s\n", array[0], array[1], array[2]);
}
System.out.println();
```

Maybe some books are old but are important to use to do some research. However, there are still many old books having the younger versions and no need to check the old ones. This report is set to get these book names, ISBNs and the publish times of the old ones. The time can be adjusted at any time, but in our example, we analysis

the records after 2020-01-01, and the publish time is before 2010-01-01.

```
Statistical Table of the Information of the Old Books Which Aren't Welcomed
Book name                      ISBN          Publish Time
------------------------------ ------------- ------------
HARRY_POTTER_7                 2475947892847 1999-09-09
STAR_WARS                      7231648727464 1997-08-07
STAR_WARS                      8723657843675 1997-08-07
HARRY_POTTER_7                 8573692876859 1999-09-09
HARRY_POTTER_7                 4298748728748 1999-09-09
STAR_WARS                      4641233494645 1997-08-07
```

If the staff want to update the version or remove these old books which didn't be read these days, these data can be used.

III. Patron analysis

```
Vector<PATRON> activateP = PATRONTable.select_good_patron();
Vector<PATRON> inavtivateP = PATRONTable.select_error_patron();
int num1 = activateP.size();
int num2 = inavtivateP.size();
int total1 = num1 + num2;
System.out.println("There are " + total1 + " signed up patrons in total. ");
System.out.printf("%d of them are activated. count for %s\n" , num1, percentage( number: num1/(double)total1));
System.out.printf("%d of them are inactivated. count for %s\n", num2, percentage( number: num2/(double)total1) );
System.out.println();
```

First, the percentage of the activated and inactived patrons will be shown, the staff can see this information of the patrons.

```
There are 10 signed up patrons in total.
8 of them are activated. count for 80.00%
2 of them are inactivated. count for 20.00%
```

a. Show the age composition of the patrons in our library

```
public static List<Object[]> getAge () {
    String agePatronSQL = "SELECT AGE FROM PATRON";
    Object[] ageList = new Object[]{};
    return translateAllAtr(select(agePatronSQL, ageList),  n: 1);
}
```

```
System.out.println("Statistical Table of Patrons of Different Age Groups");
System.out.println("Age Groups       quantity     percentage");
System.out.println("---------------- --------     ----------");
System.out.println(" 0  ~ 12        " + ageData[0] + "              " + percentage( number: ageData[0]/ (double) total1));
System.out.println(" 13 ~ 18        " + ageData[1]+ "              " + percentage( number: ageData[1]/ (double) total1));
System.out.println(" 19 ~ 30        " + ageData[2]+ "              " + percentage( number: ageData[2]/ (double) total1));
System.out.println(" 31 ~ 40        " + ageData[3]+ "              " + percentage( number: ageData[3]/ (double) total1));
System.out.println(" 41 ~ 60        " + ageData[4]+ "              " + percentage( number: ageData[4]/ (double) total1));
System.out.println("   > 60         " + ageData[5]+ "              " + percentage( number: ageData[5]/ (double) total1));
System.out.println();
```

This part just selects the ages of different patrons and counts how many patrons are in each group. We set 6 different groups, 0~12, 13~18, 19~20 31~40, 41~60, and larger than 60. The sample is shown as below：

```
Statistical Table of Patrons of Different Age Groups
Age Groups        quantity      percentage
----------------- --------      ----------
 0  ~ 12          1              10.00%
 13 ~ 18          0              0.00%
 19 ~ 30          2              20.00%
 31 ~ 40          0              0.00%
 41 ~ 60          2              20.00%
   > 60           5              50.00%
```

The report can help the staff to see which age group has more patrons, and which age group has fewer patrons. When new books come out, we can buy more books suitable for patrons of a large number of age groups. For example, if the number of patrons between 13 and 18 is largest, we can get more fictions or tutorial materials for them and let them have more choice.

Then the analysis report will show the gender groups of the patrons. The sample is below.

```
Statistical Table of Patrons of Different gender Groups
8 are MALE. Count for 72.73%
3 are FEMALE. Count for 27.27%
```

b. Count active patrons

```
public static List<Object[]> getActivePatronData () {
    String activePatronSQL = "SELECT DISTINCT P.PATRON_NAME, B.PATRON_ID FROM PATRON P,BORROW_RECORD B WHERE P.PATRON_ID = B.PATRON_ID AND START_TIME >= '2020-01-01' GROUP BY B.PATRON_ID,PATRON_NAME HAVING COUNT(*) > 2";
    Object[] activePatronList = new Object[][2];
    return translateAllAtr(select(activePatronSQL, activePatronList), 2);
}
```

```
System.out.println("Statistical Table information of the active patron who frequently borrowed the books these days.");
System.out.println("Patron name                    patron ID");
System.out.println("------------------------------ ---------");
for (Object[] array : activePatronData) {
    System.out.printf("%-30s %-9s\n", array[0], array[1]);
}
```

Some patrons may always come to our library and really enjoy borrowing some books from us, we can give them some benefits. In our report, we count the patrons whose the number of books borrowed after 2020-01-01 is larger than 2. We set a start time because borrowing records that are too old cannot translate the reader's current borrowing status. Of course, 2 is too small because our system doesn’t have too much data. The staff can set a larger boundary in the real situation. Patron names and IDs

are shown in this part.

```
Statistical Table information of the active patron who frequently borrowed the books these days.
Patron name                    patron ID
------------------------------ ---------
MAX_BLACK                      82647821p
COOK_BUSH                      21789256p
MERGE_HANS                     65488722p
```

The report can show who are the active patrons, the library can give them some benefits like some borrowing discounts to encourage them to keep reading and post a compliment to encourage other readers to learn from them.

IV. Borrow records analysis

First, the overdue situation and the sitaution of the history records will show

```
3 overdue records in total. The overdue in progress borrow record are as follows:
BORROW_RECORD
    ISBN           BOOK_ID          PATRON_ID  RECORD_ID   START_TIME  END_TIME
(1)-------------------------------------------------------------------------------
    7218475981205  716473876473876  64273878p  6718467810  2019-09-22  null
(2)-------------------------------------------------------------------------------
    9787810441063  237468723623678  21789256p  8192347810  2020-04-11  null
(3)-------------------------------------------------------------------------------
    9365827626587  287492874898279  65488722p  8247598432  2022-02-28  null
```

```
The system saves 18 history records. The time range of these history records are as follows
within 1 week: 0 records. Count for 0.00%
1 week to 1 month: 0 records. Count for 0.00%
1 month to 1 quarter: 0 records. Count for 0.00%
1 quater to 1 year: 0 records. Count for 0.00%
1 year to 3 year: 17 records. Count for 94.44%
3 year to 10 year: 1 records. Count for 5.56%
10 years ago : 0 records. Count for 0.00%
```

Then, we analysis the popular books and popular categories. It is a little different from the first one. This one analysis all the borrow records but this first just get part.

```
According to the borrow records. The most popular 5 books in the library is as follows.
Book Name                      ISBN          number of borrow records
------------------------------ ------------- -----------------------
INFORMAL                       0764397506748 5
STAR_WARS                      57387T5782577 4
BIG_CUCUMEMBER                 9365827626587 4
JEEP_IN_DESERT                 3698167615698 4
SAN_LUCIA                      7218475981205 3

According to the borrow records. The most popular 3 categories of books is as follows.
Category                         number of borrow records
-----------------------------    -----------------------
HORROR                         7
STORY                          5
SCIENCE_FICTION                4
COURAGE_STORY                  4
JOURNAL_FANTASY                3
```

V. in progress reserve analysis

This report will analyze the situation of the reserve. Because the reserve records will be deleted after finishing, we can just analyze how many reserves are in progress and how many patrons are waiting for some books.

```
in progress reserve analysis

There are 9 in progress reserves
4 patrons are waiting for the books.
RESERVE
    ISBN           BOOK_ID     PATRON_ID  DEADLINE    BOOK_STATE
(1)
------------------------------------------------------------------------------------------
    9787810441063  81749108478218421789256p  2022-09-08  false
(2)
------------------------------------------------------------------------------------------
    7218475981205  85768293857685921789256p  2022-10-08  false
(3)
------------------------------------------------------------------------------------------
    57387T5782577  71648117411974264273878p  2022-10-06  false
(4)
------------------------------------------------------------------------------------------
    7218475981205  23897483927483864273878p  2022-11-04  false
```

4, implementation:

1. SQL

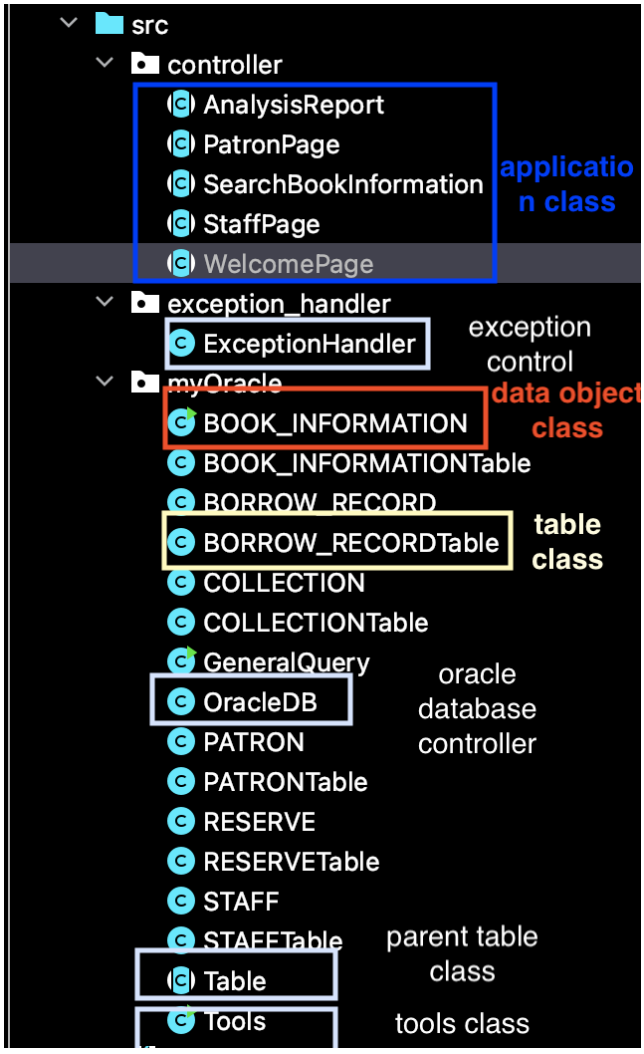
2. **JAVA:**

1'High level analysis: basic structure of the program & sample code (oop) ,
2' main advantages (security, encapsulation, expandability) & disadvantages.
3' interesting functions implement analysis: (such as comprehensive condition search)

Code design analysis:

1, overview:

(1) Main structure:



There are several kind of classes in the library management system.

The most important three kinds of class is application class, data object class and table class.

The table class refer to every table in the oracle database, providing the interface to handle queries. All the jsbc details and SQL query is encapsuled in the methods of this kind of classes.

The data object class is the representation of every tables' tuples in java.

The application class implement the process of functions.

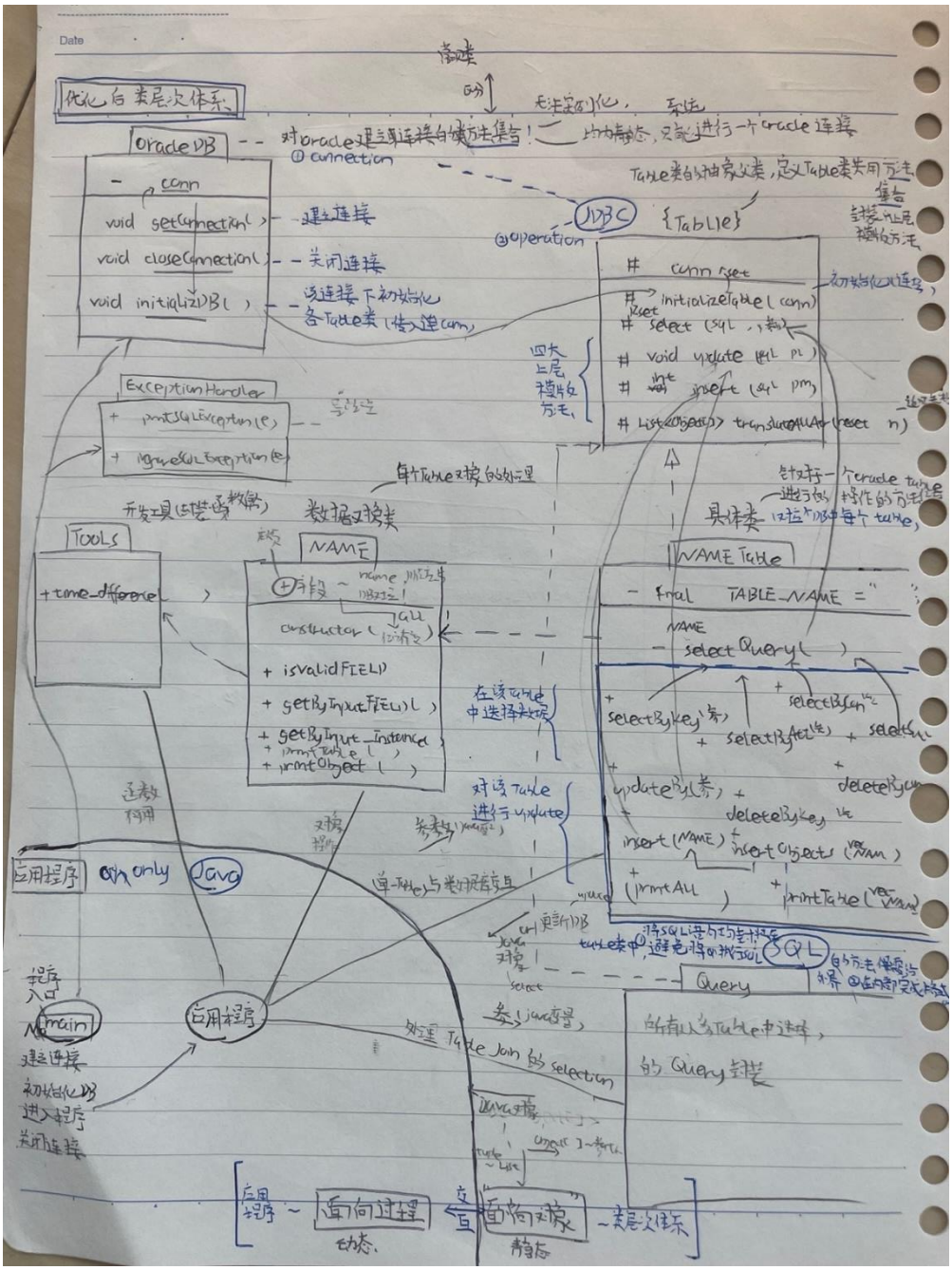
Other class have the different mechanisms .

OracleDB is the class to deal with the database connections. We use this class to connect to the database and initialize all the table classes.

Table class is the abstract parent class of all the specific table classes, providing several template of jdbc operations.

Tools class contains all of the tools methods the program need to use, such as different print methods, input check methods.

The draft of the class diagram is as follows. Although it contains same Chinese characters and is a little chaotic, you can get a basic impression of the system design from it.



Now let's explain more about the three main kind of classes – table class, data object class, and application class. This part is the core of the system. We will take the operations on borrow records as example to explain the relationship of these three kinds of classes.

a, data object class

The data object class is the representation of database records in java. It is related to every table in database. The fields of object refer to the attributes of table one by one.

representation in database:

CREATE TABLE B**ORROW_RECORD(RECORD_ID** CHAR(10),B**OOK_ID** CHAR(15), P**ATRON_ID** CHAR(15), I**SBN** CHAR(10),**START_TIME** CHAR(10),**END_TIME** CHAR(10));

representation in java.

```
73 usages
public class BORROW_RECORD {
    6 usages
    public String RECORD_ID;
    9 usages
    public String BOOK_ID;
    6 usages
    public String PATRON_ID;
    public String ISBN;
    11 usages
    public String START_TIME;
    9 usages
    public String END_TIME;
```

The data object class also provides the methods that handle the operations on this object.

Input check (format)

```
1 usage
public static boolean isValidRECORD_ID(String RECORD_ID) {
```

Get the input.

```
3 usages
public static String getFromInput_RECORD_ID() {
```

Print object and table.

```
2 usages
public static void printObject(BORROW_RECORD o) {
```

```
8 usages
public static void printTable(Vector<BORROW_RECORD> ivec) {
```

The data object class is an abstraction of database records. Because of this design, the application layer can just regard the database records an java object, ignoring the details of the database design.

b, table classes

The table classes refer to every table in the database, performing sql operations on that table. These class provides 4 kinds of operations: select, update, insert, and delete, which will be discussed in details later.

1'Select operations:

Select by key

```
2 usages
public static BORROW_RECORD selectByRECORD_ID(String RECORD_ID) throws SQLException
```

Select by attribute

```
1 usage
public static Vector<BORROW_RECORD> selectByPATRON_ID(String PATRON_ID) throws SQLException
```

Select by condition

```
3 usages
public static Vector<BORROW_RECORD> select_in_progress_borrow() throws SQLException
```

2'Update operation:

```
2 usages
public static void endTheBorrowByRECORD_ID(String id)
```

3'Insert operations:

Insert one object

```
2 usages
public static void insertTuple (BORROW_RECORD o) throws SQLException
```

Insert several objects.

```
1 usage
public static void insertObjects (Vector<BORROW_RECORD> objects) throws SQLException
```

4'Delete operations:

```
1 usage
public static void deleteBR_before(String date)
```

The select operations in the class will translate the result set to java objects, and only return the java object.

```
BORROW_RECORD r ;
while (rset.next()) {
    r = new BORROW_RECORD();
    r.RECORD_ID = rset.getString( columnIndex: 1);
    r.BOOK_ID = rset.getString( columnIndex: 2);
    r.PATRON_ID = rset.getString( columnIndex: 3);
    r.ISBN = rset.getString( columnIndex: 4);
    r.START_TIME = rset.getString( columnIndex: 5);
    r.END_TIME = rset.getString( columnIndex: 6);
    ivec.add(r);
}
```

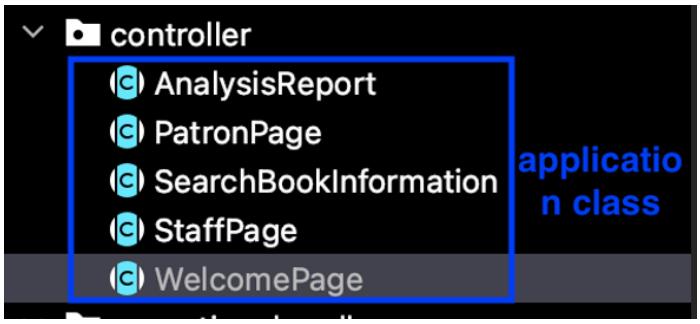
All of the jdbc and sql expression is hidden in the methods of this kinds of class. The application program only need to invoke the methods in table class, and the sql operations will be performed automatically.

```
3 usages
public static Vector<BORROW_RECORD> select_in_progress_borrow() throws SQLException
```

```
2 usages
public static void insertTuple (BORROW_RECORD o) throws SQLException
    String insertQuery;
```

C, application class.

The application classes implement the logic of application functions. This part is process-oriented (The class cannot be instanced and all of the methods is static methods.)



There are 3 main pages in the program: welcome page, staff page, and patron page. The functions in every page are as follows.

a, welcome page:

1, patron log in, 2, patron sign up, 3, staff log in.

b staff page:

1, manage book information 2, manage patrons 3, manage collections 4, manage borrow records 5, analysis report

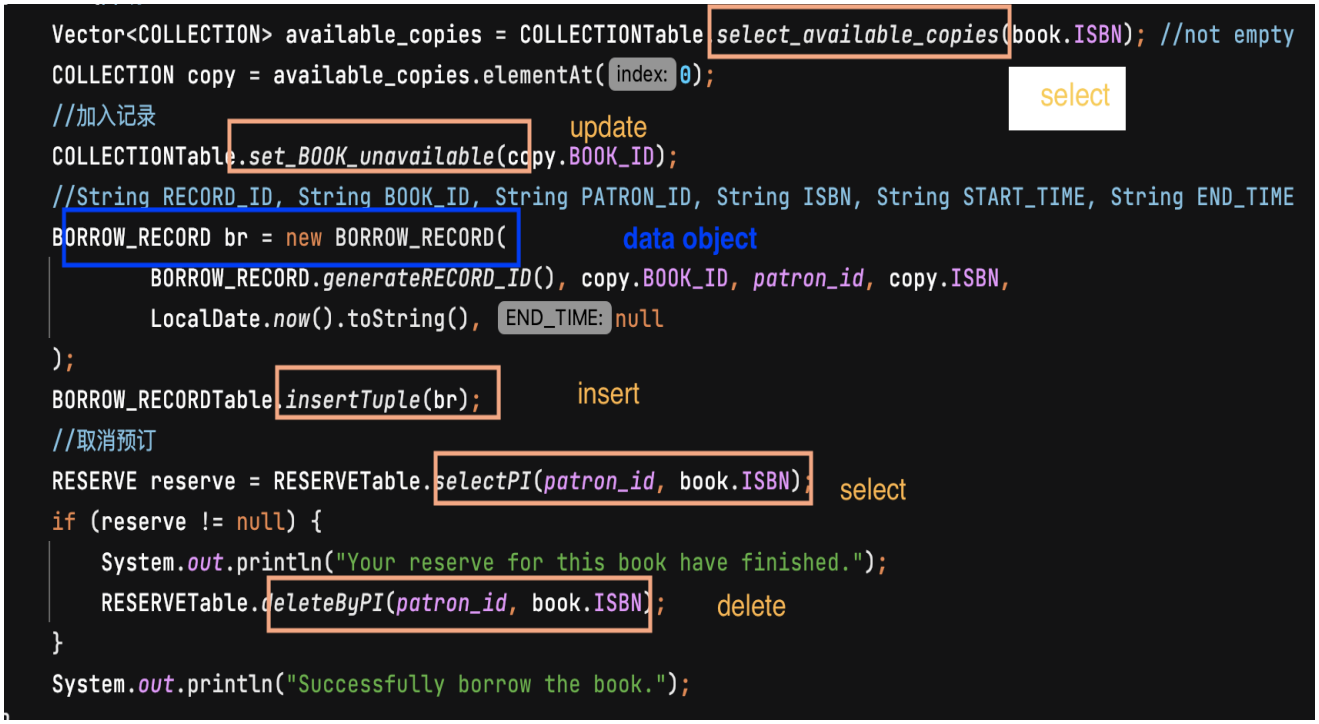
c patron page:

1, search 2, borrow 3, return 4, reserve 5, notification.

Because the search module of staff and patrons is similar, we use a search book information class to implement the function once.

Because of the encapsules of above two classes, the code in the application classes do not have any jdbc and sql expression, only focusing on the logic of functions.

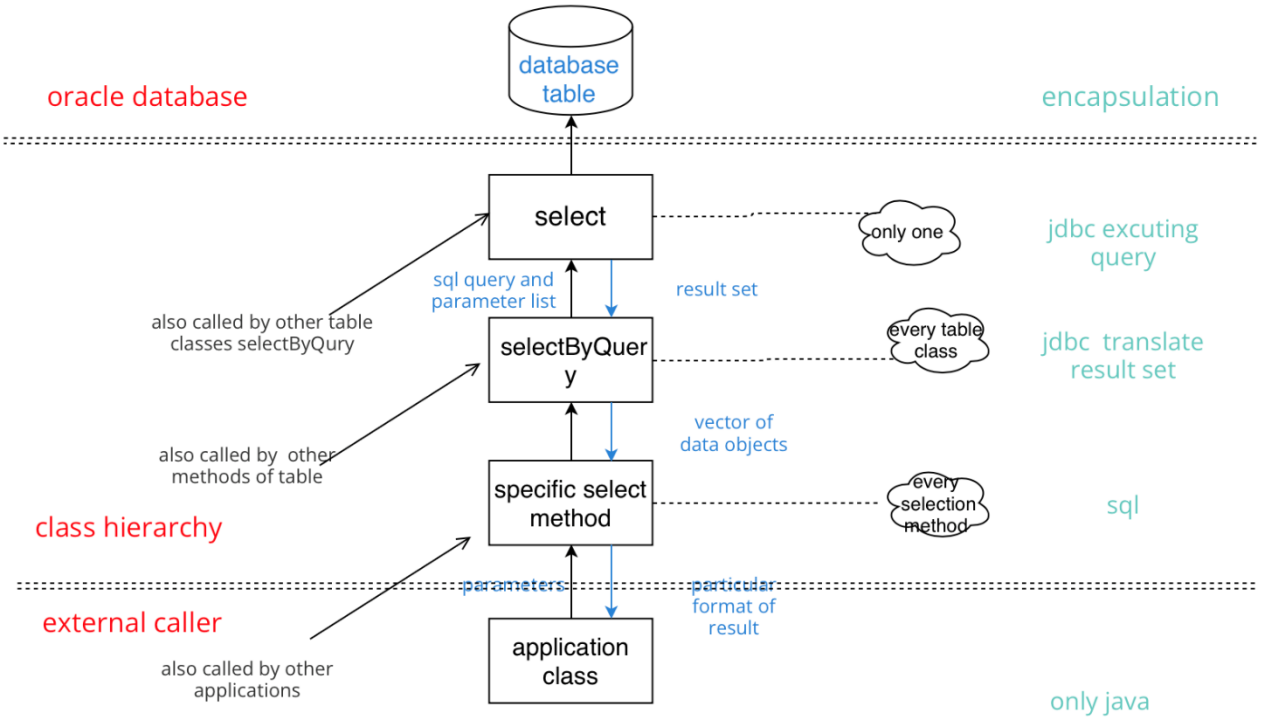
Here is a screenshot of a piece of borrow function. You can see that the logic is quite clear. Although this function perform several sql operations, this piece of code do not contains any sql expression or jdbc details.



2, JDBC sample program

Now, I will use 4 samples to explain how the program performs the jdbc operations. This may give you a clearer view about the encapsulation of the program.

(1) Select operation:



The execution of every select operation includes four layers of function invoke. The system provides three levels of encapsulation. The application class (outer invoker) only invoke the select methods of a particular table and get the java object as return result.

Every table class have a selectByQuery method, which handle the translation from result set to related data object. All the specific select method will invoke this method to excite the sql query.

The parent class of all the table classes – Table – define a methods called select. Every selectByQuery method of the table classes will invoke this method. This method execute the jdbc operations, using prepared statement to retrieve data from database.

The following paragraph explain the process of executing an selection operation. The function is to select the borrow records with the end time in a particular range. The application layer will first ask user to input a range of time. After checking the input, the records satisfying the conditions will be retrieved from the database.
The red rectangle refer to the invoke of next level methods.

A, first level : application layer

```
Tools.printInputHint("Check the borrow records that end time between a particular time range.");
boolean valid = false;
String startTime = null;
String endTime =  null;
while (!valid) {
    startTime = BORROW_RECORD.getFromInput_START_TIME();
    endTime = BORROW_RECORD.getFromInput_END_TIME();
    valid = startTime.compareTo(endTime) < 0 || startTime.equals("*") || endTime.equals("*");
    if (! valid) {
        System.out.println("The input is valid! This section do not allow quit and end time must later than start time");
        break;
    }
}
Vector<BORROW_RECORD> ivec2 = BORROW_RECORDTable.select_records_of_time_range(startTime, endTime);
System.out.println("The borrow records range from " + startTime  + " to " +  endTime + " are as follows:");
BORROW_RECORD.printTable(ivec2);
break;
```

get the time range

select the borrow records

print the results

Second level : selece records of time range method in BORROW_RECORDTable class.

```
1 usage
public static Vector<BORROW_RECORD> select_records_of_time_range(String startDate, String endDate) throws SQLException {
    String selectQuery = "SELECT * FROM BORROW_RECORD WHERE " + "END_TIME > ? AND END_TIME < ?";
    Object[] paramList = new Object[]{startDate, endDate};
    return selectByQuery(selectQuery, paramList);
}
```

output: java objects

input: parameters

sql query

invoke general select operations template

Prepare the sql query and parameter list and invoke the selectByQuery method of this class.

Third level: select by query

```
9 usages
private static Vector<BORROW_RECORD> selectByQuery (String selectQuery, Object[] paramList) throws SQLException {
    Vector<BORROW_RECORD> ivec = new Vector<>();
    ResultSet rset = select(selectQuery, paramList);
    try {
        BORROW_RECORD r ;
        while (rset.next()) {
            r = new BORROW_RECORD();
            r.RECORD_ID = rset.getString( columnIndex: 1);
            r.BOOK_ID = rset.getString( columnIndex: 2);
            r.PATRON_ID = rset.getString( columnIndex: 3);
            r.ISBN = rset.getString( columnIndex: 4);
            r.START_TIME = rset.getString( columnIndex: 5);
            r.END_TIME = rset.getString( columnIndex: 6);
            ivec.add(r);
        }
    } catch (NullPointerException e) {
        System.out.println("\n\nNullPointerException in selectQuery of table BORROW_RECORD.");
        e.printStackTrace();
        System.out.println("The return vector has length: " + ivec.size());
    }
    return ivec;
}
```

output: vector of java objects

input: sql query, parameter list

invoke general jdbc select template

translate the result set to java object

deal with the null pointer exception

Invoke the select method and translate the result set to java objects.

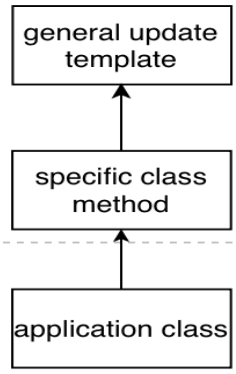
Last level: select method of class Table.


```
protected static ResultSet select (String sql, Object... paramList) {
    ResultSet rs = null;
    try  {
        PreparedStatement ps = conn.prepareStatement(sql);

        if (paramList != null) {
            for (int i = 0; i < paramList.length; ++i) {
                ps.setObject( parameterIndex: i+1, paramList[i]);
            }
        }
        rs = ps.executeQuery();

    } catch (SQLException e){
        System.out.println("Error in executing the sql code: " + sql);
        while(e != null){
            System.out.println("message: " + e.getMessage());
            e = e.getNextException();
        }
    }

    return rs;
}
```

Retrieve data from the database and return the result set.

As you can see, every level orients towards different problems and handles different exception. Because the encapsulation, every method will be invoked by several higher level methods, which decrease the repeatability of the codes. What's more, the logic of every method is quite clear.

(2) Update operation:



The design pattern of the update operation is similar to the select operations. The only difference is that, because update operation do not have the return value we do not need to translate the result. The level that performing this translation is no need.

Now I will use a example to explain more about it. The following code is a piece of codes in the manage patrons selection of staff page. It shows the in progress overdue borrows to the staff, allowing the staff to inactivate the accounts of these patrons.

The manageAbnormalBR method is invoked here. It will select the overdue borrow records, then print the overdue borrow records as well as the patrons holding this records. The screenshot of this method is for you reference, if you do not want to know the specific logic of this method, just skip it.

```
1 usage
private static Vector<BORROW_RECORD> manageAbnormalBR () throws SQLException {
    Vector<BORROW_RECORD> records = BORROW_RECORDTable.select_in_progress_borrow();
    String today = LocalDate.now().toString();
    HashSet<String> patrons = new HashSet<>();
    Vector<BORROW_RECORD> abnormal = new Vector<BORROW_RECORD>();
    for (BORROW_RECORD record : records) {
        if (Tools.time_difference(record.START_TIME, today) > 100) {
            abnormal.add(record);
            PATRON p = PATRONTable.selectByPATRON_ID(record.PATRON_ID);
            if (p != null) {
                patrons.add(p.PATRON_NAME.strip());
            }
        }
    }
    System.out.println("The abnormal in progress records are as follows, which is started 100 days before.");
    BORROW_RECORD.printTable(abnormal);
    //待优化： 统一输出（查分printTable） or 在类体内实现
    System.out.println("These borrow is made my following person:");
    for (String name : patrons) {
        System.out.print(name + "  ");
    }
    //
    return abnormal;
}
```

First

```
case 3:
    Vector<BORROW_RECORD> ivec = manageAbnormalBR();
    System.out.println("Are you sure to inactivate these persons account? 1--yes 0 -- no");
    int choose2 = Tools.getIntByInput( n: 1);
    if (choose2 == 1) {
        for (BORROW_RECORD record : ivec) {
            PATRONTable.inactivateThePatron(record.PATRON_ID,  illegal_type: "Overdue borrow");
        }
        System.out.println("Successfully inactivate these persons account.");
    } else {
        System.out.println("Cancel the operation.");
    }
    break;
```

get the overdue records

inactivate the patrons

invoke the update methods


```
case 3:
    Vector<BORROW_RECORD> ivec = manageAbnormalBR();
    System.out.println("Are you sure to inactivate these persons account? 1--yes 0 -- no");
    int choose2 = Tools.getIntByInput( n: 1);
    if (choose2 == 1) {
        for (BORROW_RECORD record : ivec) {
            PATRONTable.inactivateThePatron(record.PATRON_ID,  illegal_type: "Overdue borrow");
        }
        System.out.println("Successfully inactivate these persons account.");
    } else {
        System.out.println("Cancel the operation.");
    }
    break;
```

get the overdue records

inactivate the patrons

invoke the update methods

Second level : specific update methods

Inactivate the accounts of patrons, and record why the account is activated.

```
2 usages
public static void inactivateThePatron(String patron_id, String illegal_type) {
    String updateQuery = "UPDATE " + TABLE_NAME + " SET " + "ACTIVITY_STATE = 0 WHERE PATRON_ID = ?";
    Object[] paramList = new Object[]{ patron_id};
    update(updateQuery, paramList);

    String updateQuery1 = "UPDATE " + TABLE_NAME + " SET " + "ILLEGAL_TYPE = ? WHERE PATRON_ID = ?";
    Object[] paramList1 = new Object[]{illegal_type, patron_id};
    update(updateQuery1, paramList1);
}
```

input: parameters

sql query

excute the query

Third level: execute the update query

```
protected static void update(String sql, Object... paramList){
    try{
        PreparedStatement ps = conn.prepareStatement(sql);
        if(paramList != null) {
            for (int i = 0; i < paramList.length; ++i) {
                if(paramList[i] != null)
                    ps.setObject( parameterIndex: i+1, paramList[i]);
                else
                    ps.setObject( parameterIndex: i+1, Types.NULL);
            }
        }
        ps.executeUpdate();
    } catch (SQLException e){
        System.out.println("Error in executing the sql code: " + sql);
        while(e != null){
            System.out.println("message: " + e.getMessage());
            e = e.getNextException();
        }
    }
}
```

input: sql query and parameter list

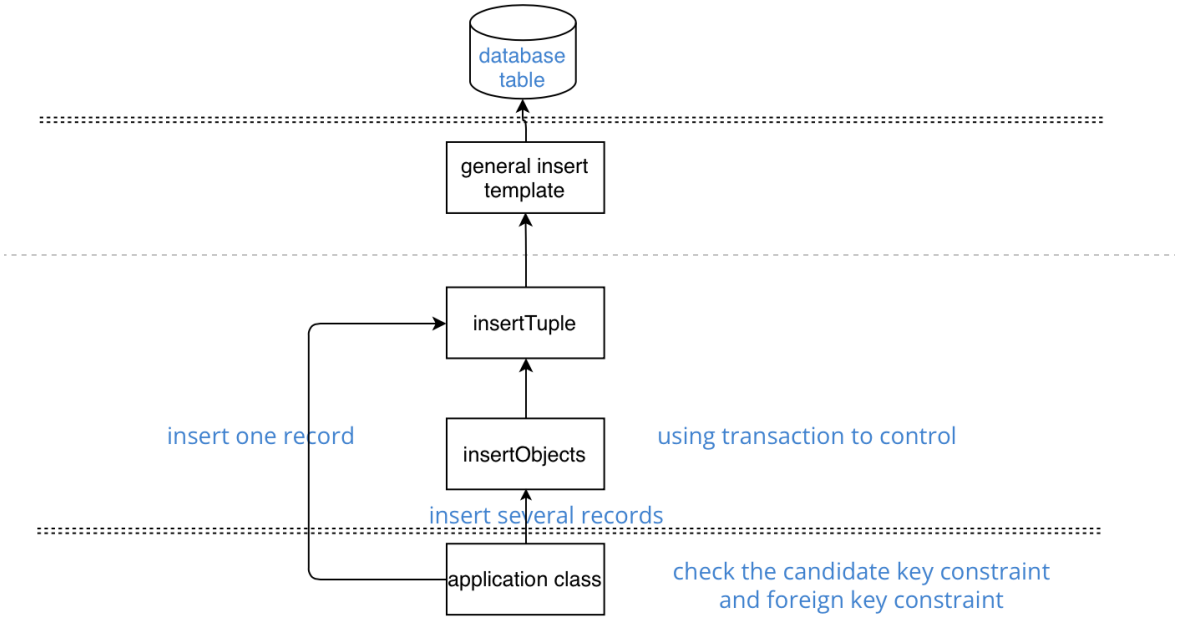
prepared statement

set the wildcards

execute the query

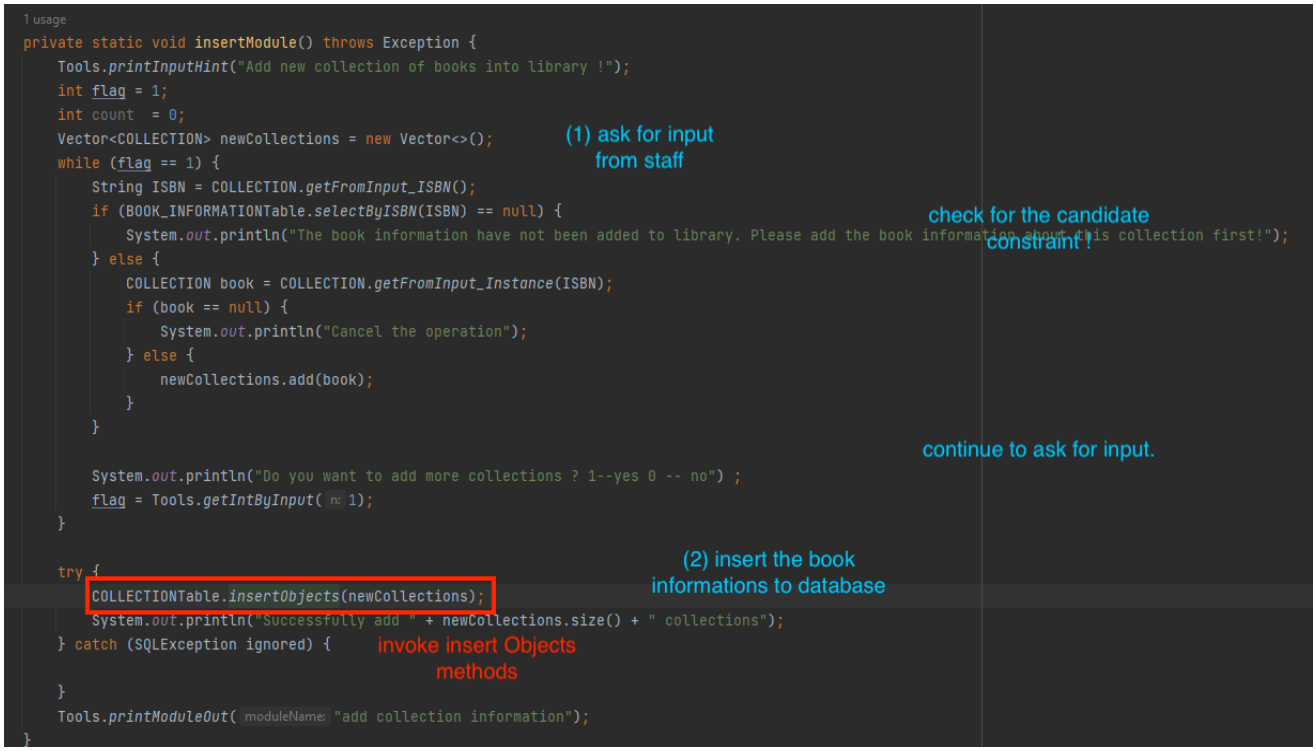
deal with the sql exception

(3) Insert operation



Insertion operation has two main differences from the above two operations: Firstly, we should check for the candidate key constraint and foreign key constraint before adding new records. Secondly, when inserting a set of records, the program use transaction to make sure that all of these records will either successfully inserted or fail together.

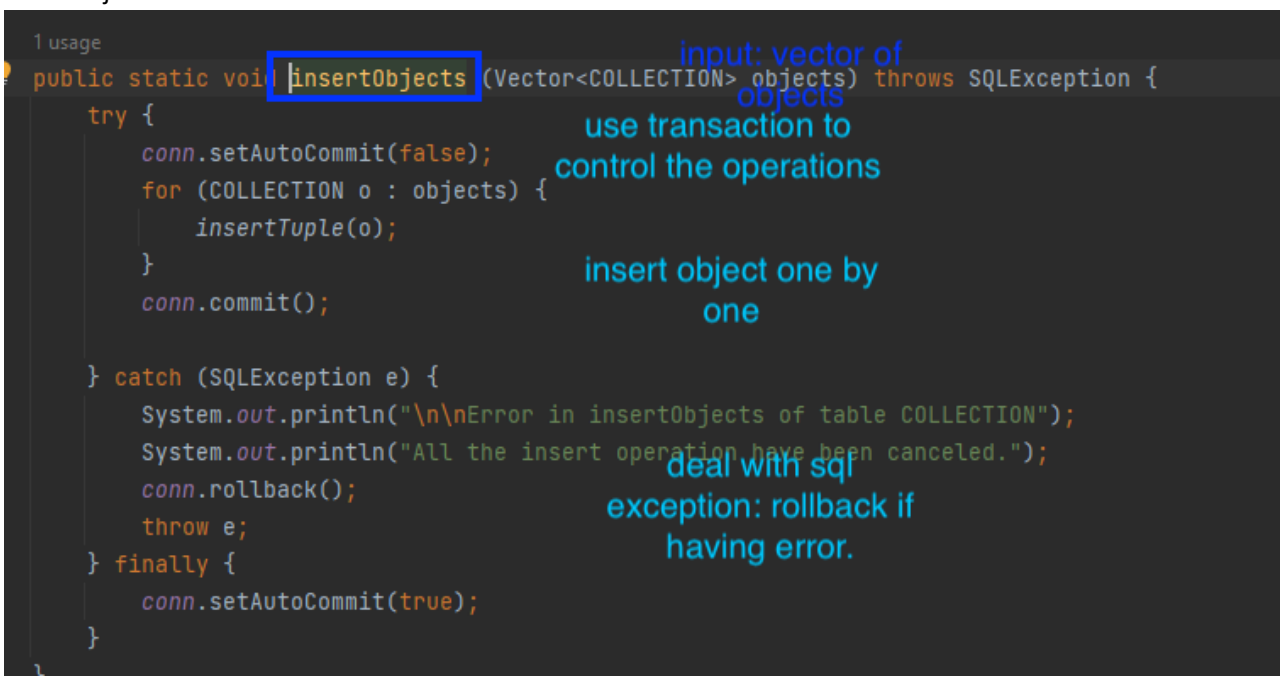
The following content give an example about insert operation.



This function is prepared for staffs to add new book into library, which is used for book update. The staff can input more than one book information and add them to database at once.

Second level:

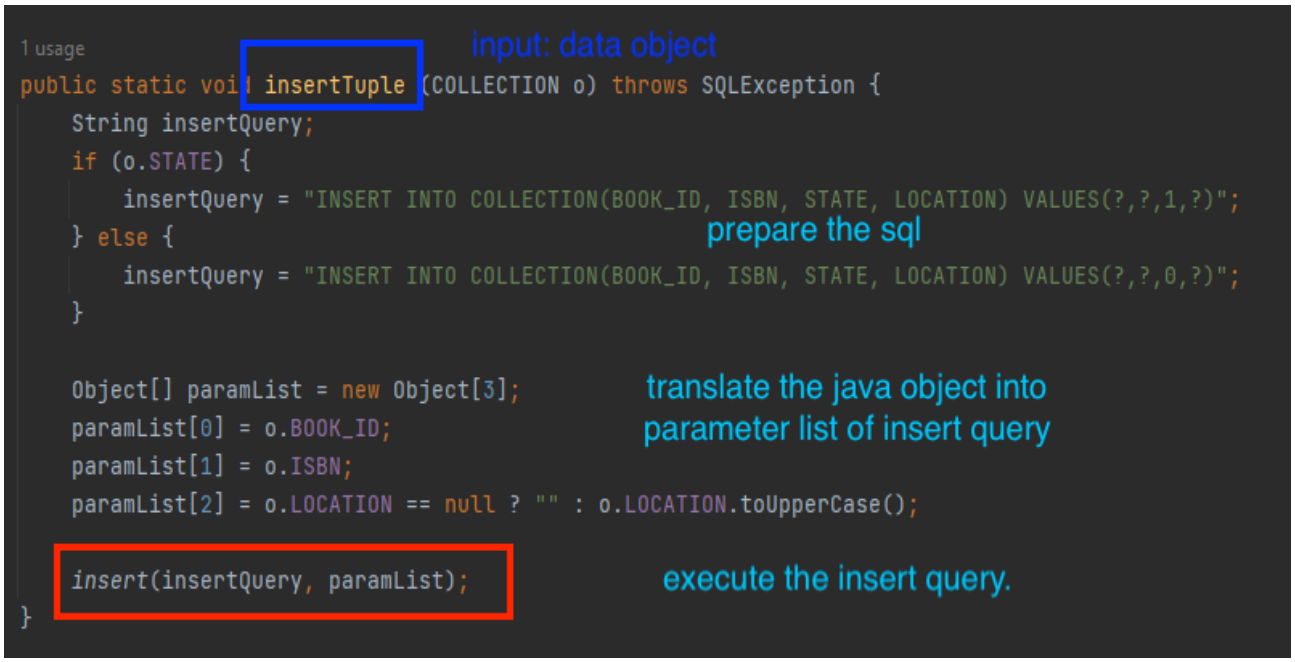
InsertObjects methods



If one of the insert operation fails, all of the records will not be inserted.

Third level :

insertTuple methods




```
1 usage
public static voi insertTuple (COLLECTION o) throws SQLException {
    String insertQuery;
    if (o.STATE) {
        insertQuery = "INSERT INTO COLLECTION(BOOK_ID, ISBN, STATE, LOCATION) VALUES(?,?,1,?)";
    } else {
        insertQuery = "INSERT INTO COLLECTION(BOOK_ID, ISBN, STATE, LOCATION) VALUES(?,?,0,?)";
    }

    Object[] paramList = new Object[3];
    paramList[0] = o.BOOK_ID;
    paramList[1] = o.ISBN;
    paramList[2] = o.LOCATION == null ? "" : o.LOCATION.toUpperCase();

    insert(insertQuery, paramList);
}
```

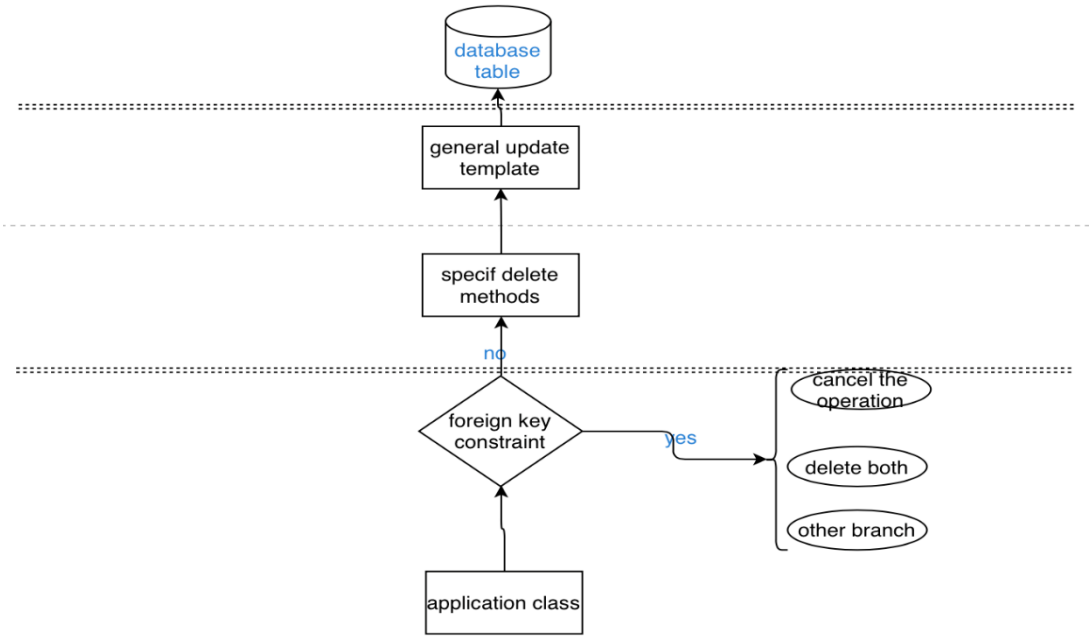
Translate the java data object into parameter list and insert the new records to database.

Last level:

Insert method in class Table.


```
protected static int insert(String sql, Object... paramList) throws SQLException {
    try {
          String[] id_name = {(String) paramList[0]};

        PreparedStatement ps = conn.prepareStatement(sql);
        for (int i = 1; i < paramList.length+1; ++i) {
            if(paramList[i-1] != null)
                ps.setObject(i, paramList[i-1]);
            else{
                ps.setNull(i, Types.INTEGER);
            }

        }

        return ps.executeUpdate();



    } catch (SQLException e){
        SQLException e2 = e;
        System.out.println("Error in executing the sql code: " + sql);
        while(e2 != null){
            System.out.println("message: " + e2.getMessage());
            e2 = e2.getNextException();
        }
        throw e;
    }
}
```

(4) Delete operation:

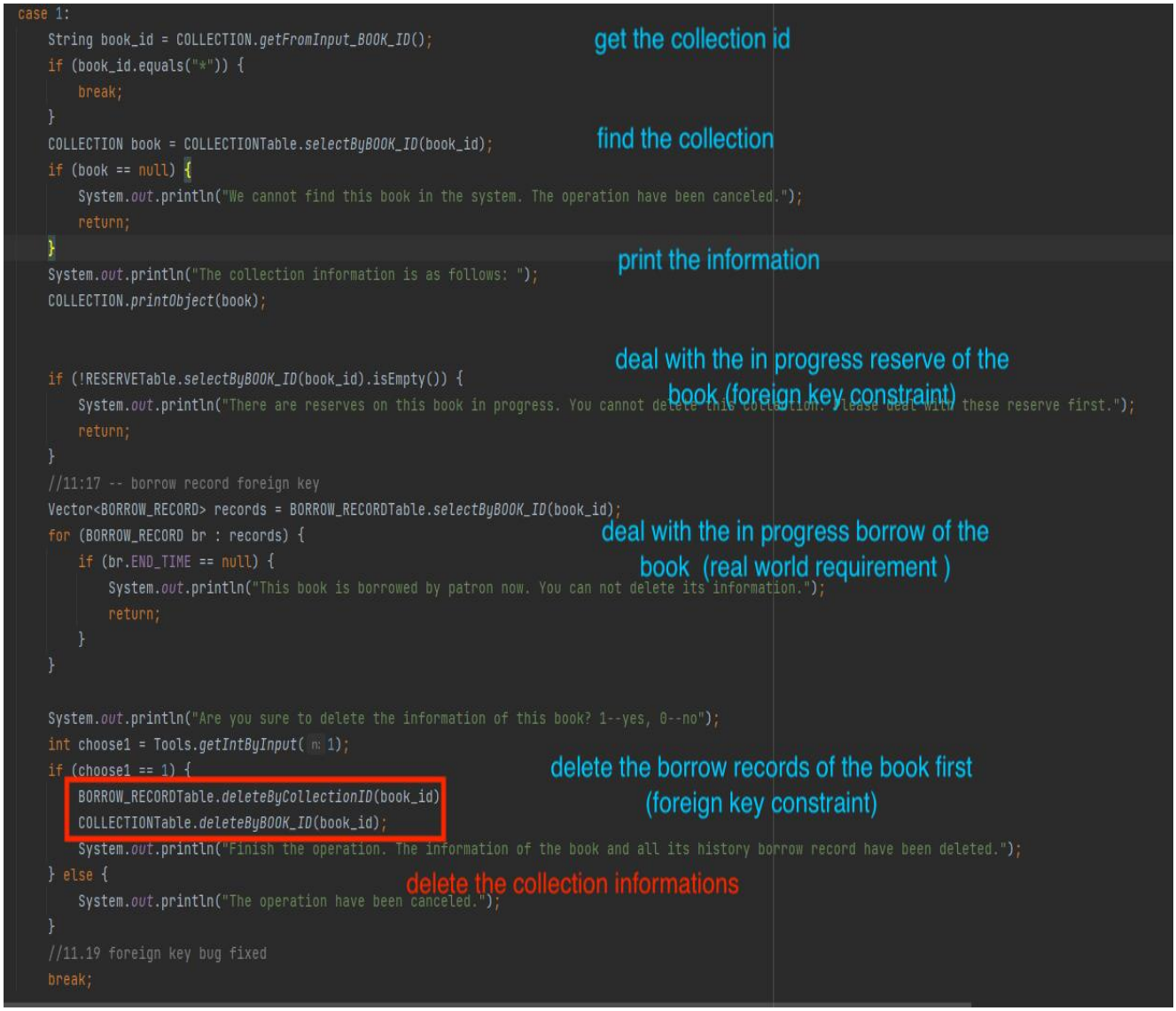


The delete operation is similar to insert operation. We should check for the foreign key constrain before delete the record. If the record is referenced by the records on other tables, The program will perform the following actions: 1, refuse to delete the record. 2, delete the records together 3, provide other branches to handle the contradictory. That depends on the real world requirement.

The following give an example of delete operation. The function is to delete the information about a book. There is one thing that need to pay attention to : When deleting the information about the book, the system may also record the in progress borrow and in progress reserve for this book. What's more, The borrow records table may hold the history borrow record of this book, which will also violate foreign key constraint.

First level :

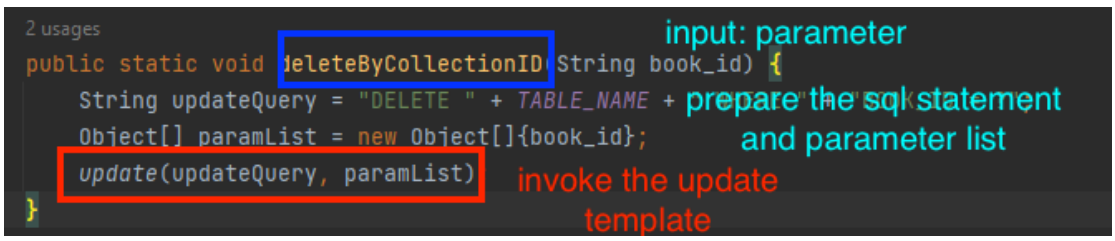
Application layer.



As you can see, it is not that easy to deal with the different cases that may violate foreign key constraint.

Second level:

Delete method in COLLECTIONTable class.



Prepare the sql query and just invoke the update method.

Third level:

General update method in Table class (same to update operation)


```
protected static void update(String sql, Object... paramList){
    try{
        PreparedStatement ps = conn.prepareStatement(sql);
        if(paramList != null) {
            for (int i = 0; i < paramList.length; ++i) {
                if(paramList[i] != null)
                    ps.setObject( parameterIndex: i+1, paramList[i]);
                else
                    ps.setObject( parameterIndex: i+1, Types.NULL);
            }
        }
        ps.executeUpdate();
    } catch (SQLException e){
        System.out.println("Error in executing the sql code: " + sql);
        while(e != null){
            System.out.println("message: " + e.getMessage());
            e = e.getNextException();
        }
    }
}
```

The above functions are the examples of four databse operations (select, update, insert, delete), which is used to explain the basic design pattern. They are just the tip of the iceberg to the whole program. Actually, the system contains more than 1 hundred times of jdbc operations execution, and it is impossible to explain in details about every operation.

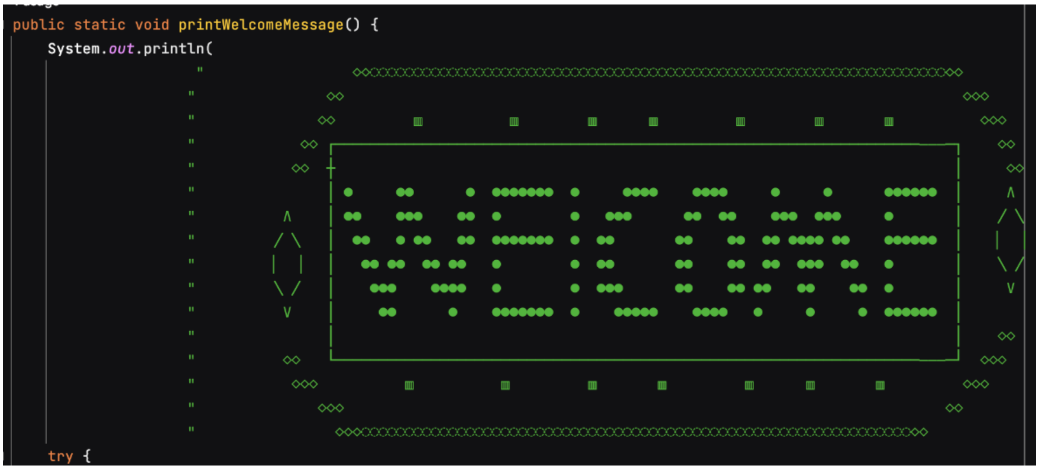
In the next part, we will explain the implementation of some interesting functions in the system.

3, library system functions implementation

This part will introduce some the implementation of some functions provided by the system. We will only show the most typical functions to explain the logic of application.

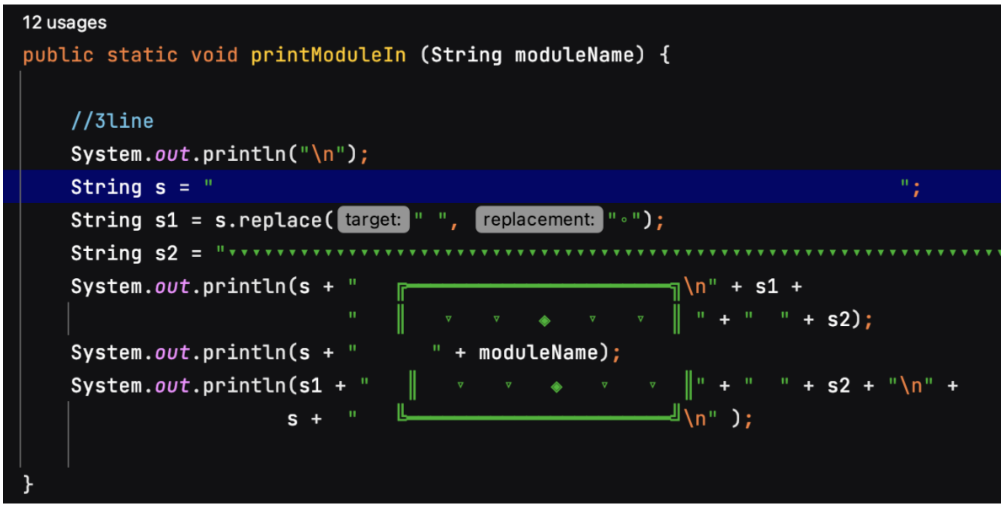
(1) Interaction with users

The system use traditional methods to interact with users ---- command line. Because the time is limited and the system provide too much functions, We do not have the chance to develop GUI. It is a pity. And I try to provide more intuitive way to interact with user using command line. Therefore, I use ASCLL character to draw some graph and use them to interact with user. What's more, the thread sleep is used to span the execution of every function.

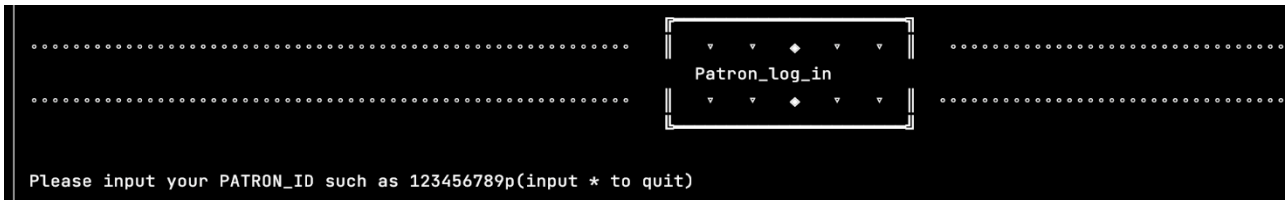


What’s more, I define plenty of print methods in Tools class, which helps to provide more intuitive interaction with users.

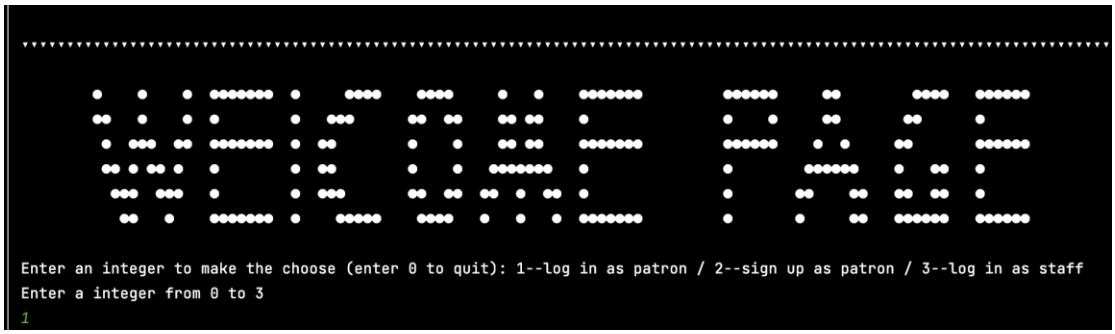
Like this one: module in



----effect



Welcome and leave message:



To be honest, design this interaction pattern is very funny.

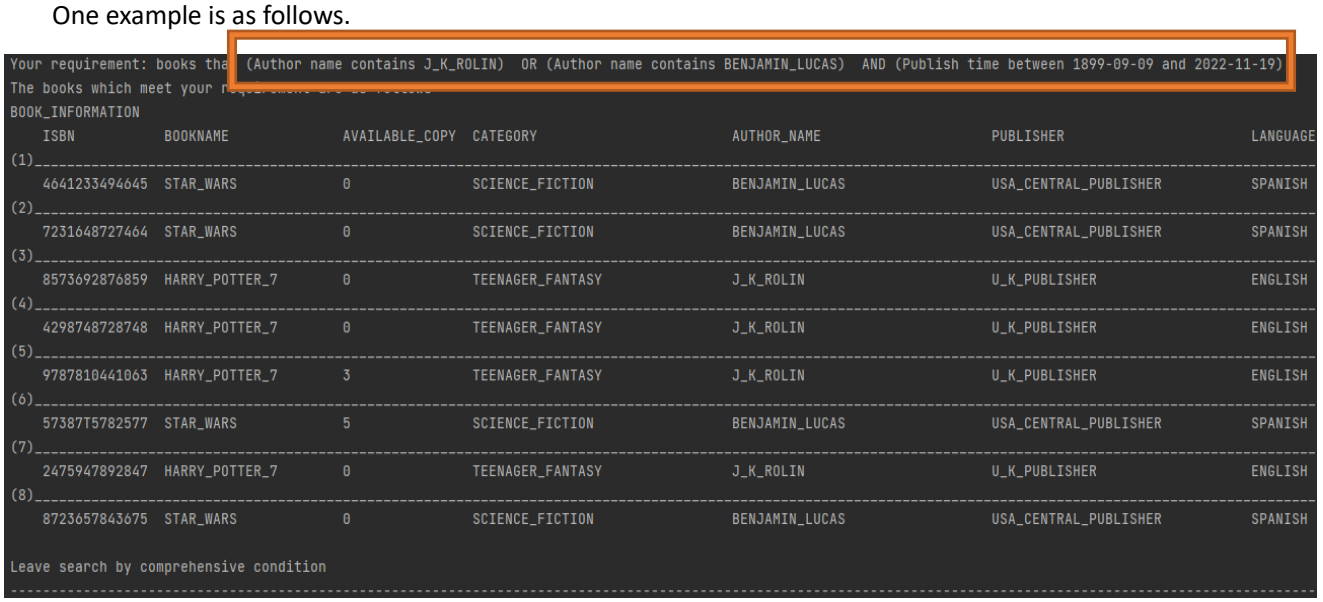
(2) Search by comprehensive conditions function

This part implement the function of searching book information by comprehensive conditions, which means patron and staff can add as much condition as they want when searching for the book information.

There are 6 atomic conditions, which all are not the key condition like search by ISBN. Users can choose to add more conditions and connect them using “AND” or “OR” .

After searching , the system will show the books that satisfy the comprehensive conditions to users.

One example is as follows.



There are three function invoking layers.
1, search by comprehensive condition – the main logic of the function
2, search by multiple condtion. – implement the logic of doing one search.
3, search by particular condition – implement the logic of specific search condition.
Let's talk about it one by one , from top to bottom.

1, search by comprehensive condition method:
Initialization

```
3 usages
public static Vector<BOOK_INFORMATION> searchByComprehensiveCondition() throws Exception {
    Tools.printInputHint("Search book by comprehensive condition");

    int flag = 1;
    Vector<BOOK_INFORMATION> ivec = new Vector<>();
    HashSet<BOOK_INFORMATION> resultSet = new HashSet<>();
```

empty hash set

Main part: perform the search.
The condition join is implemented using hash set in java.

```
while (flag == 1) {
    //build the condition
    condition = "";
    //
    flag = 0;
    //1, operation
    //first condition
    resultSet.addAll( searchByMultipleCondition());
    //other condition
    int flag1 = 1;
    while (flag1 == 1) {
        Tools.printChooseHint( name: "", description: "You can add more contition", choice: "1--And 2--Or");
        int choose = Tools.getIntByInput( n: 2);
        switch (choose) {
            case 1 :
                condition += "  AND ";
                resultSet.retainAll(searchByMultipleCondition());
                break;

            case 2 :
                condition += "  OR ";
                resultSet.addAll(searchByMultipleCondition());
                break;

            default : {
                flag1 = 0;
            }
        }
    }
```

initialize the condition string

do the first search

add more condtion

and — hash set's intersection

or— hash set's union

Terminate: print the result

```
        //translate and print out -- condition, table.
        System.out.println("Your requirement: books that " + condition);
        System.out.println("The books which meet your requirement are as follows");
        //转型可能出错
        for(BOOK_INFORMATION element:resultSet){
            ivec.add(element);
        }
        BOOK_INFORMATION.printTable(ivec);
    }
    Tools.printModuleOut( moduleName: "search by comprehensive condition");
    return ivec;
}
```

print out the condition string

print out the result

2, search by multiple conditions

This method is invoked by the above searchByComprehensiveCondition method. It implement the process of doing one search.

```
3 usages
public static Vector<BOOK_INFORMATION> searchByMultipleCondition () throws Exception {
    String choice = "1-- book name (contains..) / 2-- author name(cantains..) / 3--publisher / 4--language / 5
    Tools.printChooseHint( name: "",  description: "Add one condition or quit.", choice);
    int choose = Tools.getIntByInput( n: 6);
    while (choose == 0) {
        System.out.println("Sorry, you can not leave here please finish the condition.");
        choose = Tools.getIntByInput( n: 6);
    }
    return switch (choose) {
        case 1 -> searchByBN();
        case 2 -> searchByAN();
        case 3 -> searchByPublisher();
        case 4 -> searchByLanguage();
        case 5 -> searchByCategory();
        case 6 -> searchByTime();
        default -> new Vector<BOOK_INFORMATION>();
    };
}
```

return the result of search

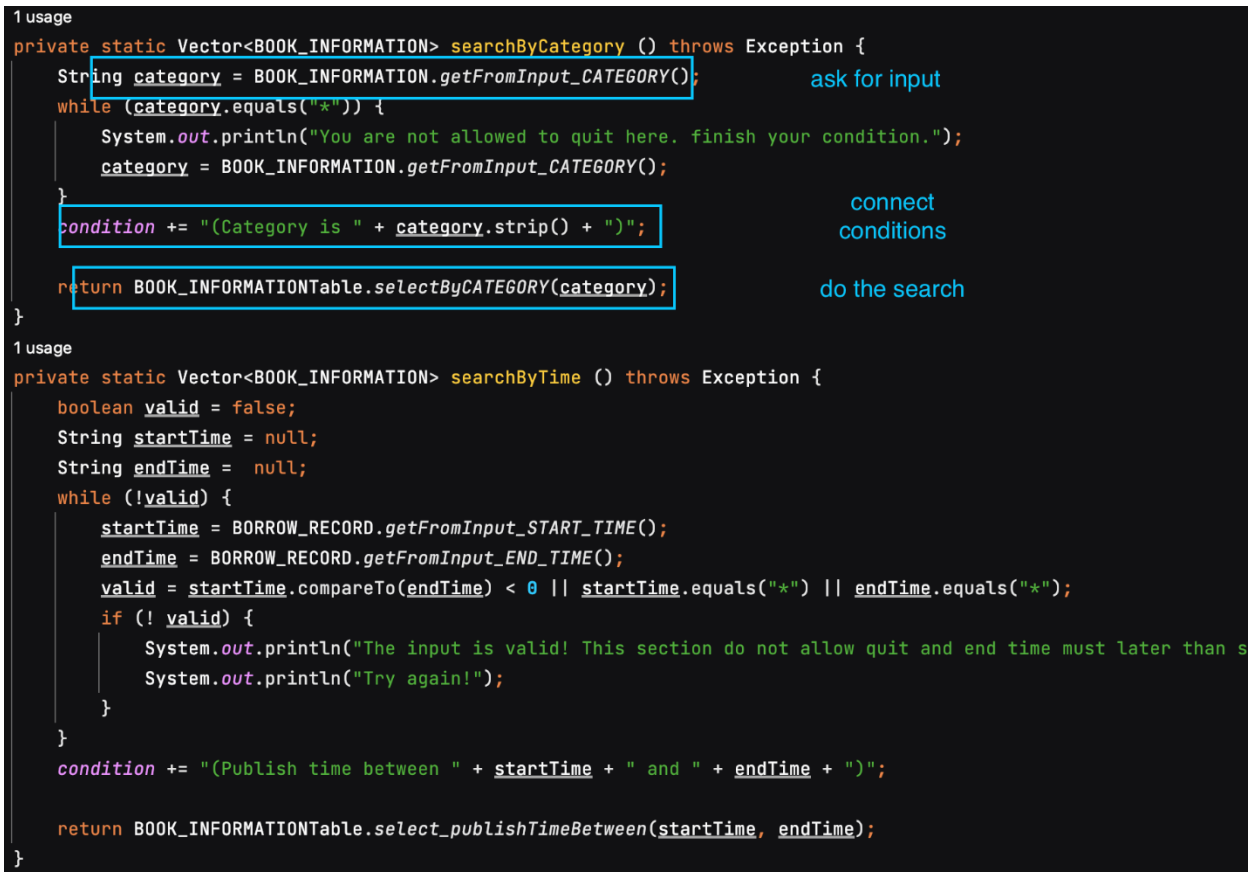
ask for choice

invoke specific method
to do the search.

3, search by specific method.

This level is the implementation of specific search, which will interact with user, do the search and return the search result to upper level invoker.

There are 6 functions in total, related to the 6 kinds of conditions that provided. Two of them are as follows.




```
1 usage
private static Vector<BOOK_INFORMATION> searchByCategory () throws Exception {
    String category = BOOK_INFORMATION.getFromInput_CATEGORY();
    while (category.equals("*")) {
        System.out.println("You are not allowed to quit here. finish your condition.");
        category = BOOK_INFORMATION.getFromInput_CATEGORY();
    }
    condition += "(Category is " + category.strip() + ")";

    return BOOK_INFORMATIONTable.selectByCATEGORY(category);
}
1 usage
private static Vector<BOOK_INFORMATION> searchByTime () throws Exception {
    boolean valid = false;
    String startTime = null;
    String endTime =  null;
    while (!valid) {
        startTime = BORROW_RECORD.getFromInput_START_TIME();
        endTime = BORROW_RECORD.getFromInput_END_TIME();
        valid = startTime.compareTo(endTime) < 0 || startTime.equals("*") || endTime.equals("*");
        if (! valid) {
            System.out.println("The input is valid! This section do not allow quit and end time must later than s
            System.out.println("Try again!");
        }
    }
    condition += "(Publish time between " + startTime + " and " + endTime + ")";

    return BOOK_INFORMATIONTable.select_publishTimeBetween(startTime, endTime);
}
```

Actually, to implement this function, I have two choices. The first one is using string connection to generate a sql query with comprehensive condition. The second one is to using atomic condition to do the selection, and implement the main logic in java. Obviously, the first one is easier to implement, but I choose to use the second method.

here are two main reasons:

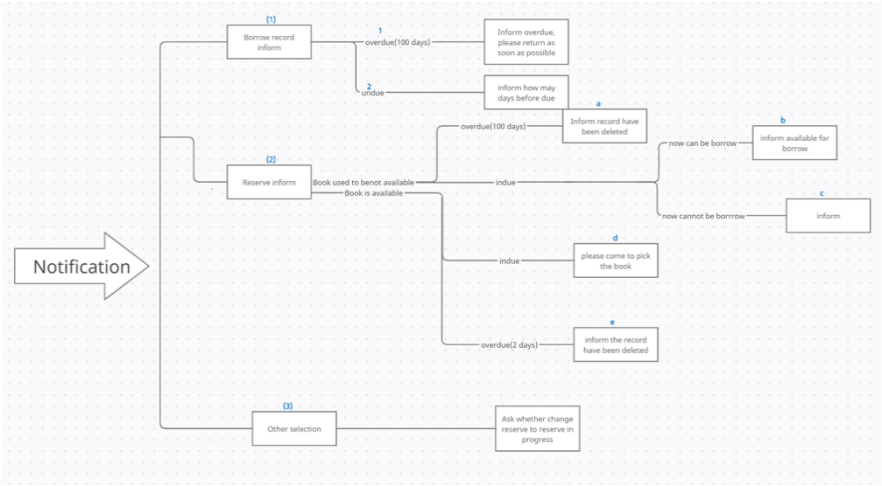
First of all, in the system, all of the jdbc interface and sql query is encapsuled in Table class hiarechy. This means application layer developer do not have the chance to invoke general sql execution function and the user can not do the sql injection attack. This design pattern lead to a great security of the system and database. If I use string connection method to implement the function, I have to make some dangerous interface expose to outside layer, which will break the previous security pattern design. Secondly, use java method to do the implementation is challenging, and I have more motivation to do this.

(3) Notification generate

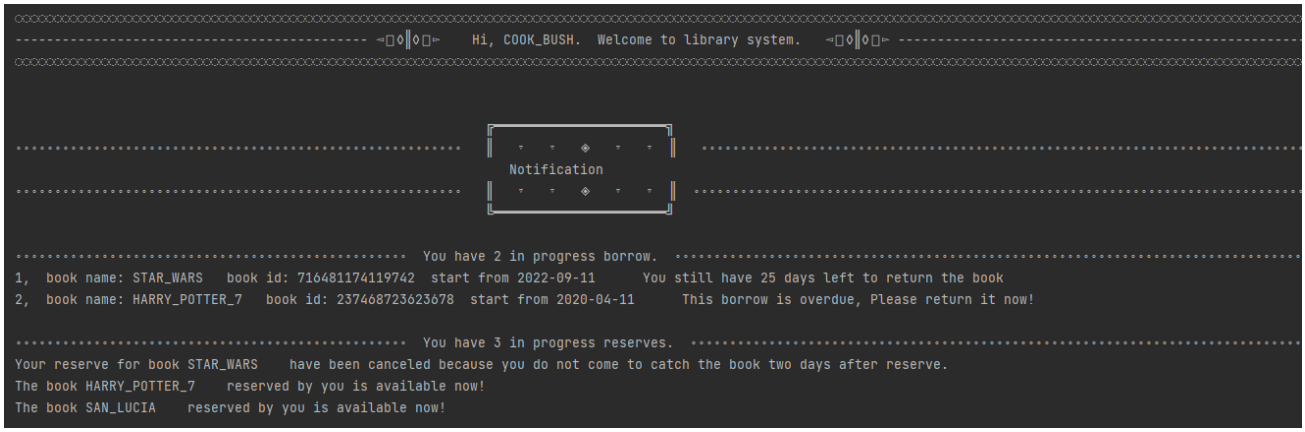
There are 5 functions that oriented towards patrons: notification, search for book information, borrow the books , return books and reserve books. In these functions, the most complicated one is the notification. This part will take notification as example

to explain how the service functions is implemented in the system.

The notification function is invoked when the patron enter the system. The program do the analysis based on the patron's behavior, generate a notification and display to the patron.



here are two examples about the notification.



A, main logic about the notification

The notification.main() function is invoked when the patron log in, and turn to the patron page for the first time.

The notification include three part : notification about in progress borrow, notification about in progress reserve, and choice about the available reserved book. I will talk about It one by one.

```
//module
1 usage
public static void main(String[] args) throws Exception {
    Tools.printModuleIn( moduleName: "Notification");
    //
      Tools.notification("Notification", "Here is a notification for you:");
                                                                          (1)
    Vector<BORROW_RECORD> borrows = get_borrow_record_message();
    Vector<RESERVE>  now_available_reserves = get_reserve_message();  (2)

      dealWithBorrow(borrows);
    System.out.println();
    Tools.printDownLine();
    //

                                                                  (3)
    dealWith_now_available_reserve(now_available_reserves);
}
```

B, display the notification about in progress borrow

(The implementation about the get_borrow_record_message method.)

There are three cases, which will lead to different results:

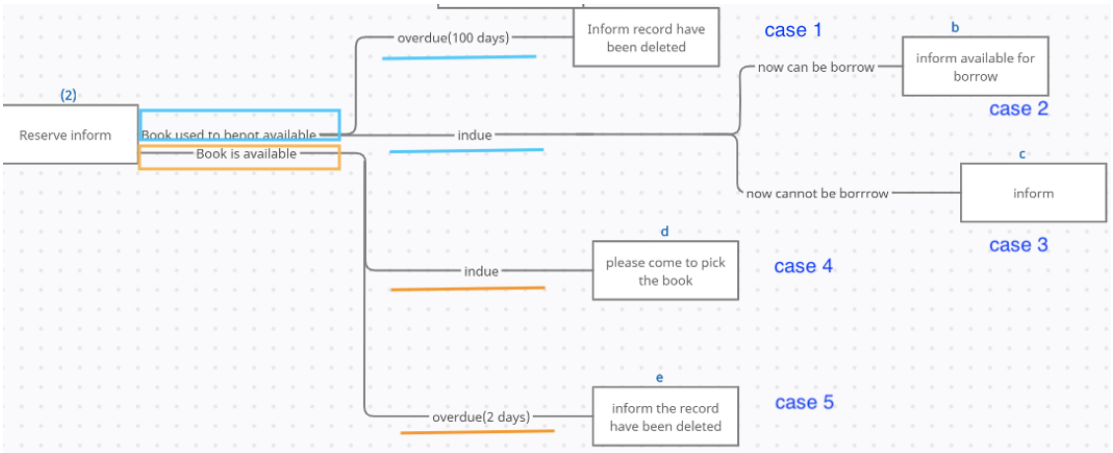
Case 1: the patron have no in progress borrow records. ---- ignore it.

Case 2: the patron have overdue in progress borrow ---- give a warning and remind to return it as soon as possible.

Case 3: the patron have normal in progress borrow ---- just remind it there are how many days before the deadline.

```
1 usage
private static Vector<BORROW_RECORD> get_borrow_record_message() throws Exception{   get the inprogress borrow record of the patron
    Vector<BORROW_RECORD> inProgressBR = BORROW_RECORDTable.select_in_progress_borrow_ofPerson(patron_id);
    int numberOfBR = inProgressBR.size();
    if (numberOfBR == 0) {            case1: no in progress borrow
        System.out.println("You do not have any in progress borrow.");
        return inProgressBR;
    }
    //print
    Tools.printInputHint("You have " + numberOfBR + " in progress borrow.");
    int count = 1;                                                  print the message according to the type of borrow records
    for (BORROW_RECORD br : inProgressBR) {
        String book_name = BOOK_INFORMATIONTable.selectBNAME_from_BOOK_ID(br.BOOK_ID);
        int days = 100 - Tools.time_difference(br.START_TIME, LocalDate.now().toString());
        if (days < 0) {              case2: overdue—warning
            System.out.printf("%d,  book name: %s   book id: %s  start from %s      This borrow is overdue, Please return it now!\n",

        } else {                     case3: in due - - remind the deadline
            System.out.printf("%d,  book name: %s   book id: %s  start from %s      You still have %d days left to return the book\n"
        }
    }
    return inProgressBR;
}
```

C, notification about the in progress reserve.
Before go into the details about the implementation of this function, let's review this function quickly.



If the patron have no in progress reserve, the system will just print out the message: "You have no in progress reserve".
If the patron have in progress reserve, system will check the type of reserve one by one and deliver the corresponding message.
There are 5 cases in total. The 5 cases and the corresponding system reaction is showed on the above graph.

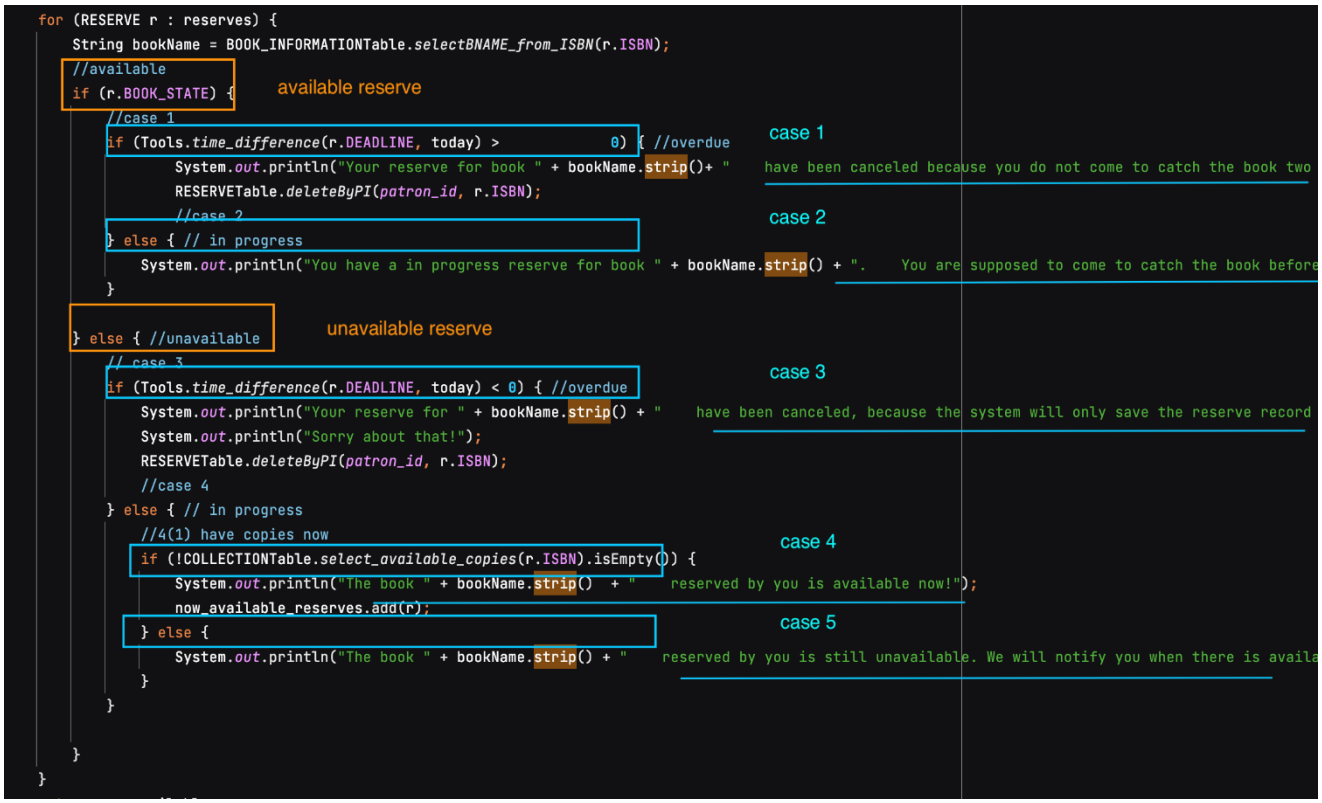
The implementation of this part is as follows:

```
1 usage
private static Vector<RESERVE> get_reserve_message() throws Exception{
      Tools.printLine();
    System.out.println();


    Vector<RESERVE> reserves = RESERVETable.selectByPATRON_ID(patron_id);
      System.out.println(reserves.size());

    Vector<RESERVE> now_available_reserves = new Vector<>();
    if (reserves.isEmpty()) {
        System.out.println("You do not have in progress reserve now.");
        return now_available_reserves;
    }
    //deal with the available reserve: case1 overdue -- remind and delete the record. case 2: in progress --
    //deal with the unavailable reserve: case3 overdue -- remind and delete the record  case4 : in progress
    String today = LocalDate.now().toString();
    Tools.printInputHint("You have " + reserves.size() + " in progress reserves.");
```

get the in progress reserve of the particular patron

get the in progress reserve of the particular patron

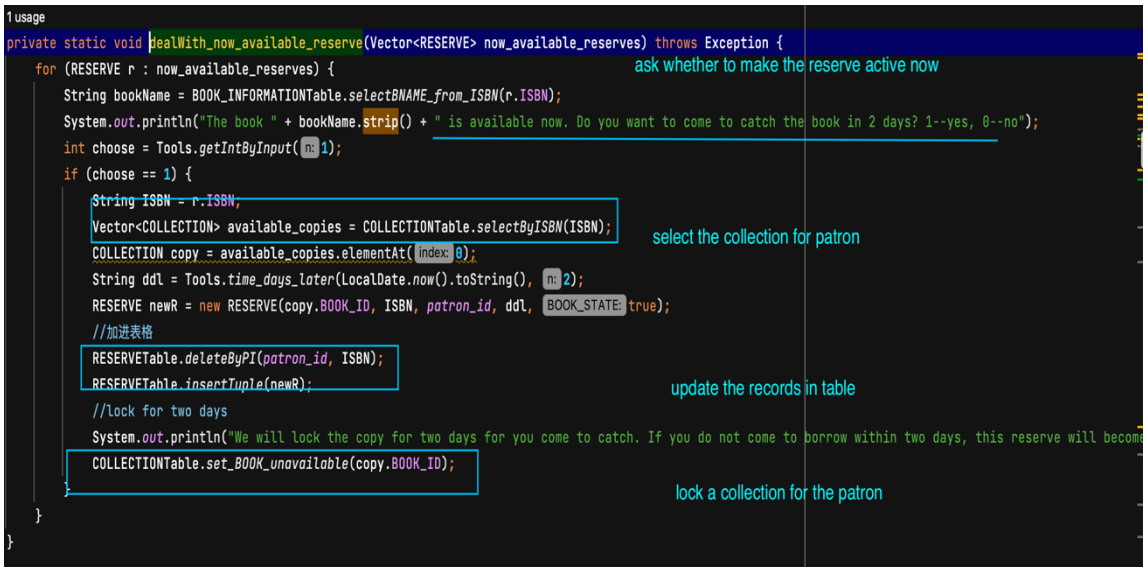
For every in progress reserve, do the corresponding operation to deliver the message.



C, deal with the available book.

If there are the books that reserved by patron before is available now (case 5 on above demonstration) , the program provide the interaction with user to ask whether the user want to make the reserve active now. If the answer is yes, the reserve translate from unavailable reserve to available reserve.

The implementation of this part is as follows:



4, implementation analysis

(1) Advantages:

A, encapsulation

There are mainly three types of classes: table class, data object class and application class.

All of the jdbc and sql expression is hidden in the methods of this kinds of class. What's more, we use OOP design pattern, regarding the records as objects. All of the operations on the objects is implemented in the class methods of data object class.Therefore, when develop the application, we do not to need to deal with this tedious details and can only focus on the implementation of system functions.

Besides the interaction between class hierarchy （OOP） and application layer （procedure-oriented）, the program have a good design of function relationship. The system makes full use of the relationship between functions, and encapsulates the repetitive code as much as possible, thus achieving better encapsulation and reusability.

every level orients towards different problems and handles different exception. Because the encapsulation, every method will be invoked by several higher level methods, which decrease the repeatability of the codes. What's more, the logic of every method is quite clear.

B, expandability

This encapsulation design pattern makes the program easy to expand.

The system has a complete underlying structure -- class hierarchy system. In the function development， many functions and tools methods are encapsulated. If there are the need of adding new functions, the developers only need to express the basic business logic. Through the use of classes and the original function calls, the difficulty of development can greatly reduced.

What's more, The system is encapsulated into plenty of different levels. Every level orients towards different problems and handles different exception. every method will be invoked by several higher level methods, which decrease the repeatability of the codes. This means if we want to change the original code, only the parent function

need to be changed. In addition, because the encapsulation, the logic of every method is quite clear, making us possible to expand the program easily.

C, security

Because all sql operations are predefined in the body of the class, the risk of sql injection attacks is avoided. Moreover, the application layer cannot call the methods of general sql statements performing operations, which provides good security for the database.

For example, the destroying operations like “DROP” can be avoided.

(2) Weakness:

A, efficiency can be improved

The weakness of the system is mainly related to efficiency. To perform the better encapsulation and security, some efficiency (time) requirements is sacrificed.

The first considerations is that we use OOP pattern to deal with the records. For all most all of the select operation, we use select “*” and translate it to the java object before return. Actually, in some cases, this translation can be avoided. For example, we just want to retrieve the borrow record id of a patron, it is unnecessary to retrieve all of the attributes.

What’s more, the SQL query is raw without any optimization. That is because the time is limited and we do not learn how to optimize query when we start to develop the program. No query optimization and no index creation.

B, lack of concurrency control.

Our system is a bit complex and providing too much functions. It is hard for us to develop concurrency control and add it into every functions. What’s more, our understanding of concurrency control is quite naïve, which make it difficult for us to develop this pattern.

5, review:

Timeline….

Maybe some books are old but are important to use to do some research. However, there are still many old books having the younger versions and no need to check the old ones. After 2020-01-01, if the books which are published before 2010-01-01 never be borrowed, the book names, ISBNs and the publish times of these old ones will be shown.

By these data, the staff can decide whether to remove these books from the library or update them to the latest version or not.