**User Class design**

/\*\*

\* Find user object by borrow card ID

\* @param int cardID

\* @return User

\* @throwException if no user's founded

\*/

public User getUser(int cardID) {

User user = User.findBy(cardID);

throw IOException("User's not founded") if user is null;

return user;

}

**Book Class Design**

/\*\*

\* Find book object by book name

\* @param string bookName

\* @return Book

\* @throwException if no book's founded

\*/

public Book getBook(string bookName) {

Book book = Book.findBy(bookName);

throw IOException("Book's not founded") if book is null;

return book;

}

/\*\*

\* Update book info

\* @param Array infoArray

\* @return void

\* @throwSQLException if update errors

\*/

public void updateBook(Array infoArray){

try {

this.updateNumberOfValidCopy(infoArray);

} catch SQLException(e){ throw IOException(e) }

}

**BookCopy Class Design**

private int bookID;

/\*\*

\* Find book copy from book copy ID

\* @param int bookCopyID

\* @return BookCopy

\* @throwException if no book copy's founded

\*/

public BookCopy getBookCopy(int bookCopyID) {

BookCopy bookCopy = BookCopy.findBy(cardID);

throw IOException("Book Copy's not founded") if bookCopy is null;

return bookCopy;

}

/\*\*

\* Update book copy status

\* @param string copyBookStatus

\* @return void

\* @throwException if update errors

\*/

public void updateBookCopy(string copyBookStatus) {

try {

this.updateCopyBookStatus(copyBookStatus);

} catch SQLException(e){ throw IOException(e) }

}

/\*\*

\* Find book record of this book copy

\* @return Book

\*/

public Book searchBook() {

return Book.getBook(this.bookID);

}

**LentBookHistory Class Design**

private int userID;

private int bookCopyID;

/\*\*

\* Find lent book history from borrow card ID and book's name

\* @param int cardID

\* @param string bookName

\* @return LentBookHistory

\* @throwException if no LentBookHistory's founded

\*/

public LentBookHistory getLentBook(int cardID, string bookName) {

LentBookHistory lentBookHistory = BookCopy.findBy(cardID, bookName);

throw IOException("Lent book history's not founded") if lentBookHistory is null;

return lentBookHistory;

}

/\*\*

\* Find book copy borrowed of this lent book history

\* @return BookCopy

\*/

public searchBookCopy() {

return BookCopy.getBookCopy(this.bookCopyID);

}

/\*\*

\* Find borrower of this lent book history

\* @return User

\*/

public searchUser() {

return User.getUser(this.userID);

}

/\*\*

\* Update lent book history

\* @param Array returnInfo

\* @return void

\* @throwException if update errors

\*/

public updateLentBook(Array returnInfo) {

try {

this.updateLentBookStatus(returnInfo);

} catch SQLException(e){ throw IOException(e) };

this.searchBookCopy()

.updateBookCopy(newBookCopyStatus);

}

**ReturnBookController Class Design**

//

**ReturnBookForm Class Design**

//