**----- class BookBorrowRegistrationForm-----**

/\*\*\* Submit new borrow registration information\* @param username a string representation of user’s username.  
\* @param password a string representation of user’s password.  
\* @param cardID a long value represent ID number of borrower card  
\* @param copies the list of book copies the borrower wants to borrow   
\* @throws IllegalArgumentException if the parameter is of wrong type  
\* @throws IOException if any other part of the process fails.  
\*/  
public static void submitBorrowRegistrationInfo(String username, String password, long cardID, List<BookCopy> copies)  
throws IllegalArgumentException, IOException {

BookBorrowRegistrationController.registerToBorrowBooks(username, password, cardID, copies);

}

**----- class BookBorrowRegistrationController -----**

/\*\*\* Register a new book borrow.\* @param username a string representation of user’s username.  
\* @param password a string representation of user’s password.  
\* @param cardID a long value represent ID number of borrower card  
\* @param copies the list of book copies the borrower wants to borrow   
\* @throws SQLException if the connection encounters an error during an interaction with a data source  
\*/  
public static void registerToBorrowBooks(String username, String password, long cardID, List<BookCopy> copies)  
throws SQLException {

User currentUser = find a user with username from database

if currentUser.getPassword() != password

return;

BorrowingCard borrowingCard = find a card with cardID from database;

if borrowingCard == null

return;

if currentUser.getCardID() != cardID

return;

if borrowingCard.getExpiredDate() <= current date time

return;

List<BorrowInfo> borrowInfos = find borrow records with borrowingCard from database;

for info in borrowInfos {

if info.getBorrowStatus() == ‘borrowed’

return;

}

for copy in copies {

if copy.getBookCopyStatus() != ‘available’

return;

}

if copies.lenght > 5

return;

BorrowInfo newInfo = new BorrowInfo(borrowingCard, copies);

save newInfo to database

}

**----- Class User -----**

/\*\*  
\* Get password of an user  
\* @return the password of an user.  
\*/  
public String getPassword(){

return this.password;

}

/\*\*\* Get card ID of an user  
\* @return the card ID of an user  
\*/  
public long getCardID(){

return this.cardID;

}

**----- Class BookCopy -----**

/\*\*\* Get the status of a book copy (available, referenced, borrowed or lent)  
\* @return the status of a book copy  
\*/  
public String getBookCopyStatus(){

return this.copyStatus;

}

**----- Class BorrowingCard -----**

/\*\*  
\* Get CardID of the borrowing card  
\* @return the cardID of the borrowing card  
\*/  
public long getBorrowingCardID(){

return this.cardID;

}

/\*\*  
\* Get expired date of the borrowing card  
\* @return the expired date of the borrowing card  
\*/  
public Date getExpiredDate(){

return this.expiredDate;

}

**----- Class BorrowInfo -----**

/\*\*  
\* Get status of the a book borrow (‘requested’, ‘borrowed’, ‘returned’)  
\* @return the status of the a book borrow  
\*/  
public Date getBorrowStatus(){

return this.status;

}

/\*\*\* Constructor to create new BorrowInfo object\* @param borrowerCard an object represent the borrowing card of user.  
\* @param copies list of book copy that borrower want to borrow  
\*/  
BorrowInfo(BorrowingCard borrowerCard, List<BookCopy> copies){

this. borrowerCard = borrowerCard;

this.copies = copies;

this.borrowedDate = the current date time;

this.expectedReturnDate = the current date time + 2 weeks;

this.status = ‘requested’;

}