Library Management System Design

First step is to clean the data provided in plain text CSV files. The borrowers file is well-formatted while the books file is tab-delimited values file which is to be converted. I write a python file named converter to transfer the books file to a file named data which has a good shape in CSV format. Another change between books and data is that I divide the multivalued column named Author into multiple rows and each row has exactly one author name. I will use data.csv and borrowers.csv file to insert into my database.

Next step is to construct a database named library. I create 6 tables in my schema which is different from schema provided by professor. The BOOK table is the same, but I do not create BOOK_AUTHORS table. By the way, the identity of book I use is ISBN13 which is a 13 digits' number. The book table contains 3 attributes: Isbn (primary key), Title, Name. The Author table in my schema contains Aid, ISBN and Name which is also different from the doc. The BORROWER, LOAN and FINES table are the same. The attribute Bname in borrower table is the full name (first name and last name) of a borrower. So, I concatenate two columns (first_name and last_name) of input file to one attribute and insert into borrower table. The Loan table has already been inserted with data. That is inserting all ISBN numbers in it with automatically increment id and the card_id is set to be 0, other fields are null. When a new record of Loan is inserted, the card_id is updated with non-zero value. Other fields are also updated with not null value. And I create a table named PAYMENT which is used to store the payment calculated by the library system. It contains only two attributes: Card_id (primary key) and Amount. After creating the databases, I just need to insert data using previous files.

I use Tomcat server and Spring-Hibernate framework. Hibernate can connect my database through JDBC driver. Since I have 6 tables, I will create 6 entities in the entity package corresponding to 6 tables in the database. Another 3 packages are controller, service, DAO package. The controller package is used to receive request and map it to front page. The service package is used to connect controller package and DAO package. The manipulation on database used by controller is implemented in DAO package. The controller package has only one class named BookController. The Service package contains 5 interfaces and 5 implementation classes. That is BookService, BorrowerService, FineService, LoanService and PaymentService. The 5 implementation classes are BookServiceImpl, BorrowerServiceImpl, FineServiceImpl, LoanServiceImpl and PaymentServiceImpl. The DAO package is delegated by the Service package which contains 5 interfaces and 5 implementation classes. They are BookDAO, BorrowerDAO, FineDAO, LoanDAO, PaymentDAO interfaces and BookDAOImpl, BorrowerDAOImpl, FineDAOImpl, LoanDAOImpl and PaymentDAOImpl. As to the front-end, I create two folders: resources and view. The resources folder contains css folder which contains add-book-style.css, app.css and style.css. The view folder has add-form.jsp, borrower-list.jsp, check-books.jsp, check-confirm.jsp, check-form.jsp, check-in.jsp, error-page-borrower.jsp, error-page.jsp, fines-search.jsp, list-books.jsp, loan-search.jsp, payment-list.jsp, search-form.jsp and a starter page named index.jsp. Also, there are two configuration file called spring-mvc-crud-demo-servlet.xml and web.xml.

Basically, there is a search form for text or number input. After typing keywords, just enter search and it will turn to the book list to show results. The search support substring matching. You can search by any substring of book's title, book's author and book's ISBN. After locating books, you can check in one of them and you will be prompted to input a card_id. The rule is that each borrower is permitted a maximum of 3 books and each book can be checked out only once before being checked in. There are another 3 buttons, called Add Borrower, Update Fines and Show Payments. After clicking the Add Borrower button, you will be prompted to input your SSN, full name and address. All these fields cannot be null and SSN must be correct format. I use regular expression to constrain the input of SSN. After the correct input of these fields, the system will automatically generate a card_id for you which is automatically incremented starting from 1001 since we already have 1000 borrowers' information in our database. By clicking Update Fines button, the system will automatically insert data into fines table and you will see a search form to locate the fine (search by either card_id or ISBN number) you want to know. Also, you can check in book after locating the fine. After check in, you can click the button Show Payment, which will show you payments grouped by card_id. Also, the payment information in database will be inserted or updated after clicking the button, then you can see the information after the query from database. The amount attribute in payment table means all the books' payments are in sum with same card_id. The rule for loan amount is that it will increase 0.25 dollars by day. In my system, once you check out a book, the system will automatically generate the date_out value by using current date. And once you check in a book, it will also automatically generate the date_in value by using current date. If the book is returned, the Paid attribute will be 1. Otherwise, it will be 0. The amount for payment is calculated by 0.25 * (date_in – date_out) whether the book is checked in. Otherwise, it will replace the date_in with current date until it is check in. The key point is that the GUI for user must be in correspondence to the library database structure.