Library Book Management System (LBMS)

OO Design Specification Document

Team C

Rochester Institute of Technology

Golisano College of Computing and Information Sciences

Teams:

NAME	ROLE	NOTES
Ryan Tytka	Developer	
Mikayla Wishart	Developer	
Bryan Wang	Developer	
Yug Patel	Developer	

Date: 11-17-2020

Version #: LBMS-DSD-11172020

Virsion History

VIRSION	AUTHOR(S)	CHANGE DESCRIPTION	DATE	NOTES
1.0	Ryan Tytka	Incomplete implementation	10/23	
1.1	Ryan Tytka	Finished initial implementation	11/9	Release 1
2.0	Ryan Tytka	Added GUI elements and improved design	11/15	Release 2

Table of Content

1 Overview

This design document will provide details about our design and implementation of the Library Book Management System (LBMS). It contains information about the requirements of the project and how those requirements have been met.

1.1 Goals and Objectives

Our team was tasked with designing and implementing the LBMS. The LBMS is Book Worm Library's (BWL) system for providing book information to users, tracking library visitor statistics for a library statistics report, tracking checked out books, and allowing the library inventory to be updated. It is the server-side system that provides an API used by client-side interfaces that BWL employees use.

1.2 Software Context

This application can be used by businesses such as libraries to help manage their collection of books, as well as track their visitors and checkout information. This system can also be used by the user to perform functions such as pay fines or check out books. Adding a user interface will make this application even more intuitive and user friendly.

2 Functional Descriptions

1. The LBMS shall use text-based requests and responses.
    A. An LBMS exchange consists of one text string sent by a client followed by one text string sent by the system.
    B. The system shall receive requests from a client as text strings.
        i. A client shall terminate request strings with a semicolon (;) character.
        ii. If the exchange is a partial client request, i.e. does not end with a termination character, the system response shall indicate that it received the partial client request and the system shall wait to receive the remainder of the request in the next one or more exchanges.
        iii. If the exchange completes a client request, the system shall perform the requested operation, and provide a response for the request according to the LBMS server reply format specification.

2. The LBMS will require that first time visitors to the library register.
    A. The system will store the following information for each visitor: first name, last name, address, and phone number, and a visitor ID (a unique 10-digit ID generated the first time a visitor registers)

3. The LBMS shall keep track of visits by visitors.
    A. The system shall keep track of the time each visitor spends at the library during each visit. (Information will be used for statistical data in library reports.)
        i. The system will store the following data of a visitor encounter: visitor's unique ID, date of visit, time of arrival, and time of departure (will be stored when a visitor is departing).
    B. The library opens at 08:00 every day.
    C. All visits in progress are automatically ended when the library closes at 19:00 when visitors remaining in the library are asked to leave. Visits do not extend over multiple days.
    D. The library will not allow the following transactions while closed (between the hours of 19:00 and 08:00 the following day):
        i. Starting a new visit by a visitor.
        ii. Checking out a book.

4. The LBMS shall respond to queries for book information.
    A. The system shall store book data for all books currently in BWL's possession. Book data shall consist of: isbn, title, author (can be multiple authors), publisher, published date, page count, number of copies, and number of copies currently checked out
    B. The system shall respond with all information matching the provided search parameters in the order requested in the query. The client can request an ordering by title, publish date, total number of copies, and the number of available copies (i.e. not checked out).
    C. The system shall respond with an empty string when there are no books matching the query.

5. The LBMS shall track books checked out by visitors.
    A. The system shall allow visitors to check out books from the library for a number of days with a max of 7 days.
        i. Visitors may check out a total of 5 books at a time.
    B. The system will store the data of the book check out transaction with the following information: isbn, visitor ID, date checked out, due date.

C. The system shall allow users to return checked out books to the library.
    i. If a book is returned after the due date there is a fine applied to the transaction.
        a. The initial fine for 1 day after the due date is $10.00 (added to the library's balance).
        b. After the first week the fine is increase incrementally by $2.00 each week following up to a total fine (including the initial fine) of $30.00

6. The LBMS shall respond to queries for an informational report of the library.
    A. The system shall respond to a statistical query with the following information about a queried month at the library:
        i. The number of books currently owned by the library.
        ii. The number of visitors of the library.
        iii. The average amount of time spent at the library for a visit.
        iv. The books purchased for the specified month.
        v. The amount of money collected through checked out book fines

7. For testing and simulation purposes, the LBMS shall support a feature to track and advance time.
    A. On initial startup the LBMS system will record the date.
    B. The LBMS system shall track the number of days that have passed.
    C. The LBMS system shall allow users to move the date forward by a specified number of days and/or hours.
    D. Upon each date change the system will generate a report of any overdue books (checked out by users past the due date).
    E. If the time is advanced such that the library closes and/or opens, this should be handled appropriately by the system (e.g. visits ended, commands enabled/disabled, etc.)

8. The LBMS system shall provide a mechanism for a "clean" shutdown of the system.
    A. The LBMS system shall end any visits in progress at system shutdown.
    B. The LBMS shall persist all data at system shutdown.

9. The LBMS system shall restore persistent state on startup.
    A. Any state previously stored will be restored on startup.

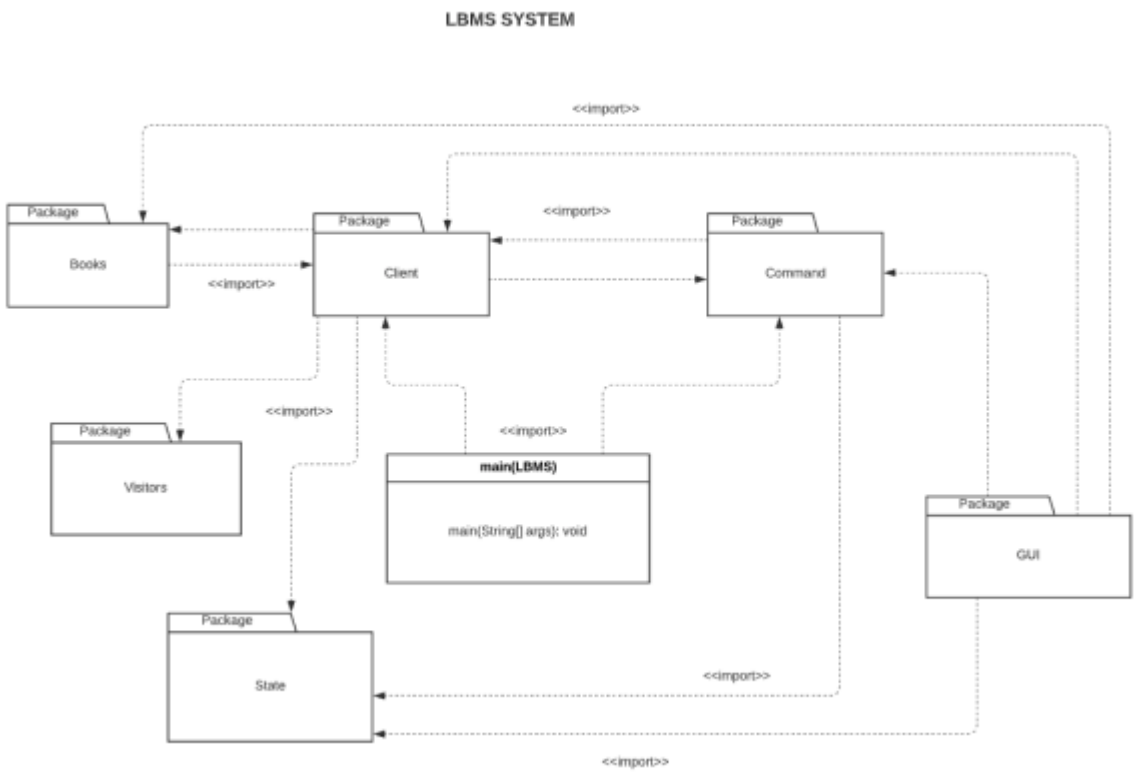
---

3 Non-Functional Descriptions

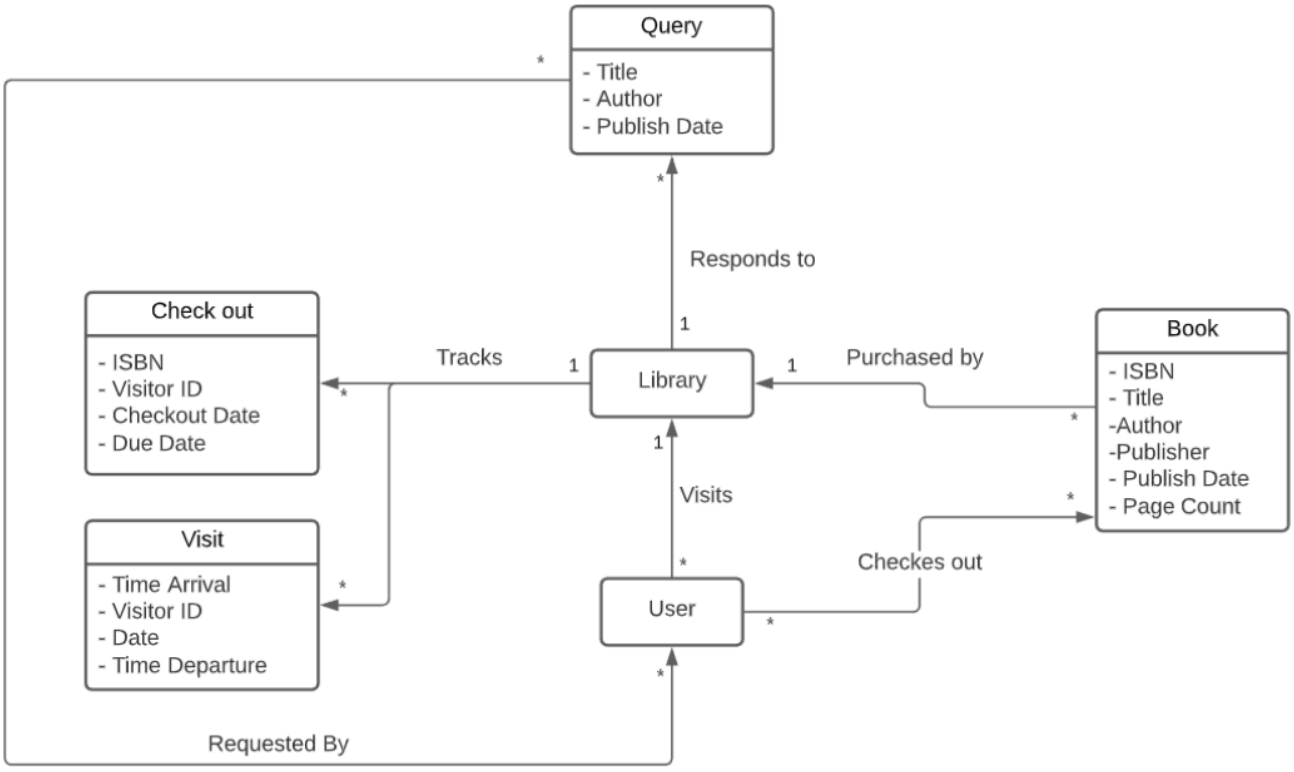
1. Since the system architects have designed the server API to be text-based, there is no need for a graphical user interface. All testing will be done from standard input and output. Your design should consider that a future LBMS release may add a graphical user interface as a requirement. Isolating all view-related components in a view subsystem should be a primary design consideration
2. At the beginning of the simulation, the library's collection of books will be empty. Books purchased by employees will be added to the library's collection.
3. The books available for purchase are included in the file books.txt and are given in the following format:
    A. isbn, title, authors, publisher, published-date, page-count, book-status
4. For the purposes of this simulation you may ignore the data regarding 'book-status' (i.e. checked out or borrowed).

4 System Architecture

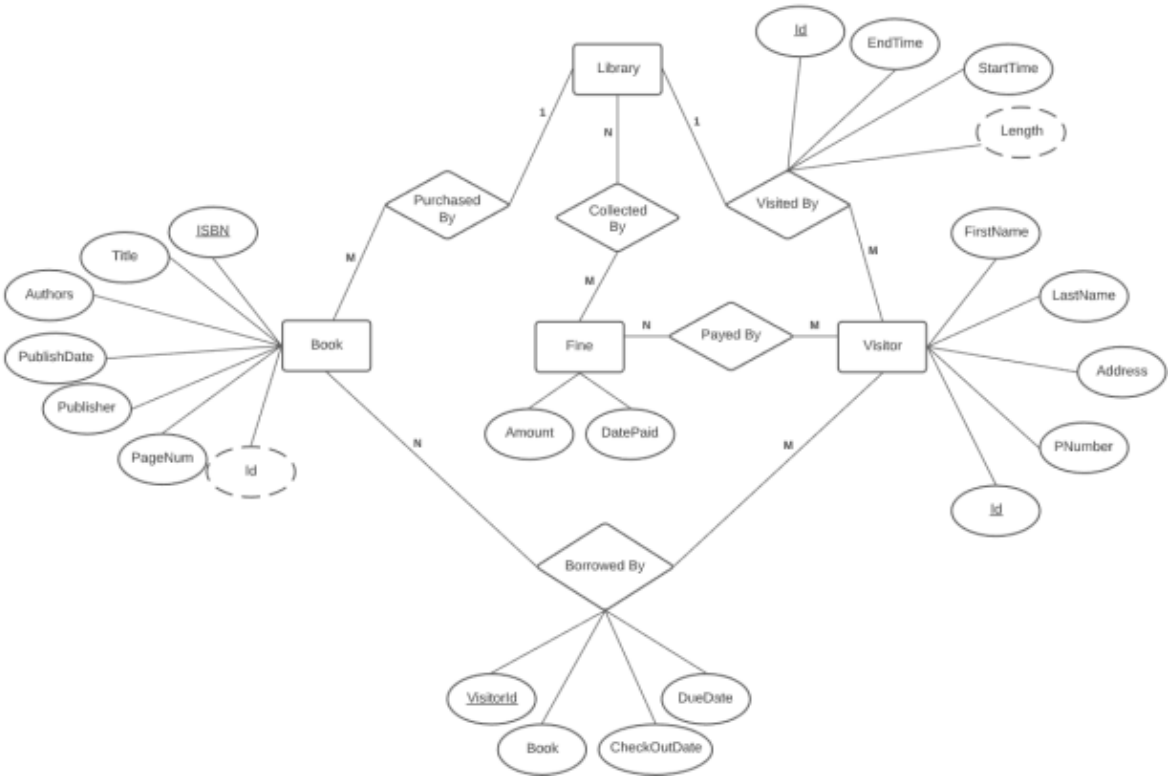
4.1 Package Diagram



4.2 Domain Model



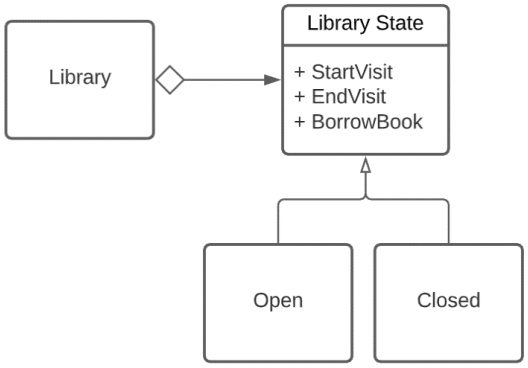
4.3 Data Architecture



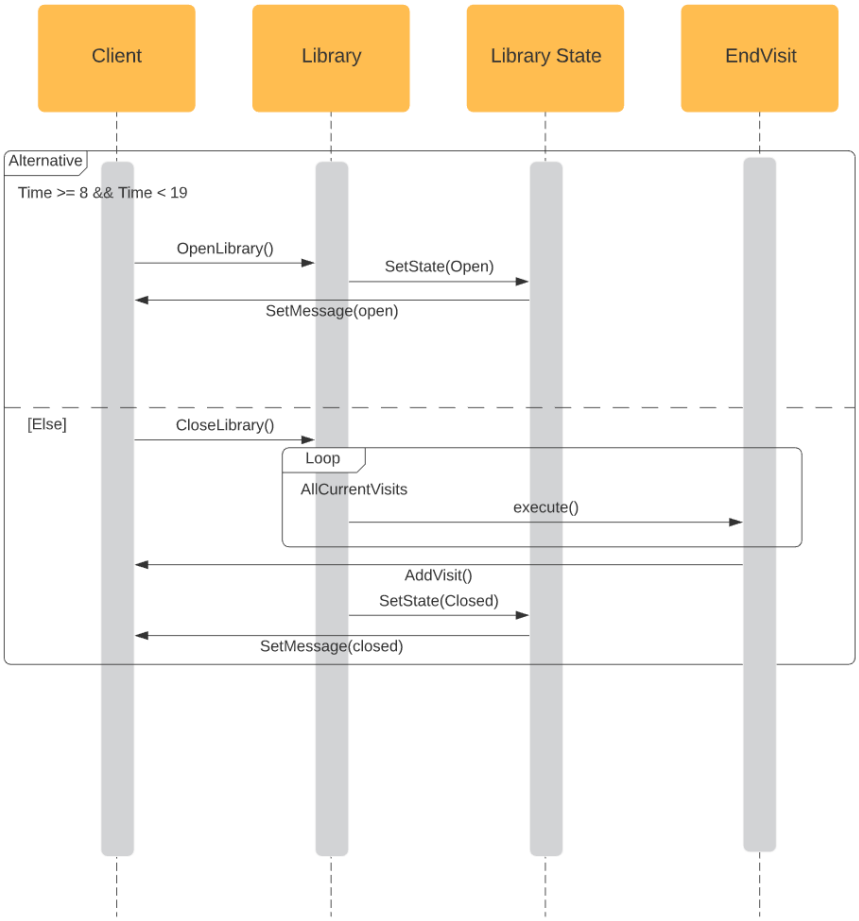
5 Detailed Architecture

5.1 Library State

5.1.1 Class Diagram



5.1.2 Interaction Diagram

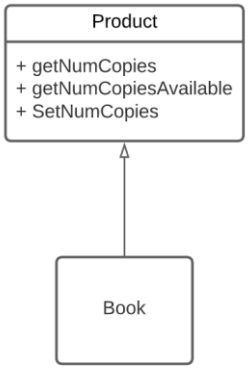


5.1.3 Detailed Design

The library state change functionality uses the most basic form of the State Pattern. There are two states that the library object will switch between. The change of states are dependent on the calendar object in the Client class. Everytime there is a change in time or day, the method class in Client checks for the current state of the library and changes it through library methods if needed. In the Open state, the library will allow the visitors to start and end the visit, or allow the user to borrow books that the library has purchased. In the Closed state, all the current visits are ended by creating EndVisit commands. The library will not allow visitors to start or end visits, or allow the user to borrow any books from the library.

5.2 Library Product

5.2.1 Class Diagram

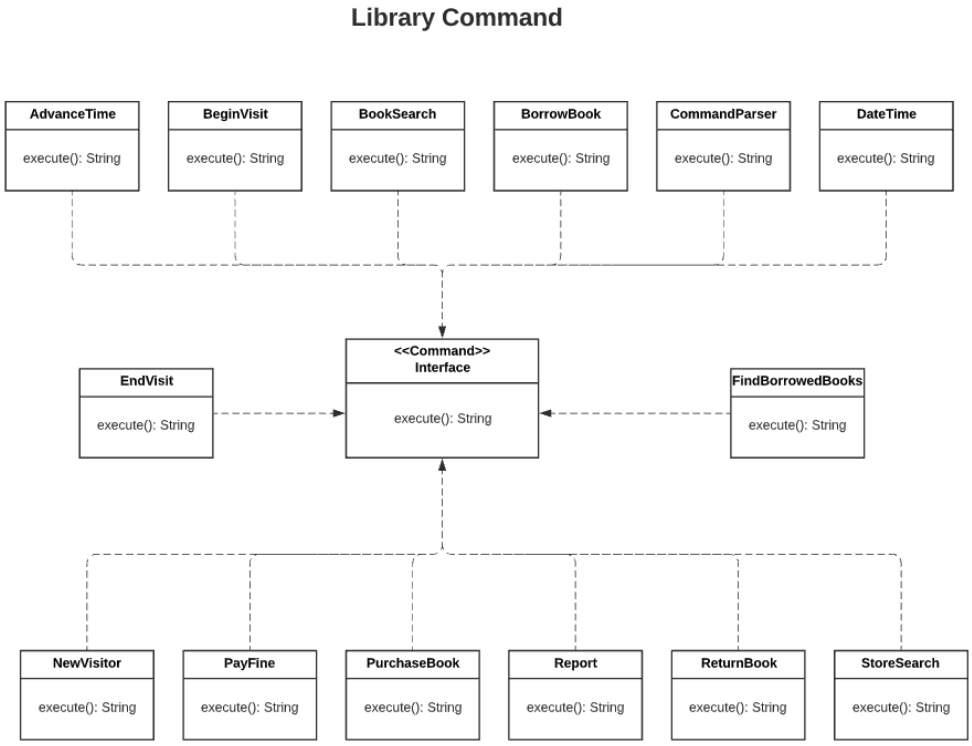


5.2.2 Detailed Design

The library product functionality has a product interface that any current or future products in a library. They may have the same behaviors, which will make it easier to perform requests. This would also allow for easy addition of new types of products, such as magazines or videos.

5.3 Library Command

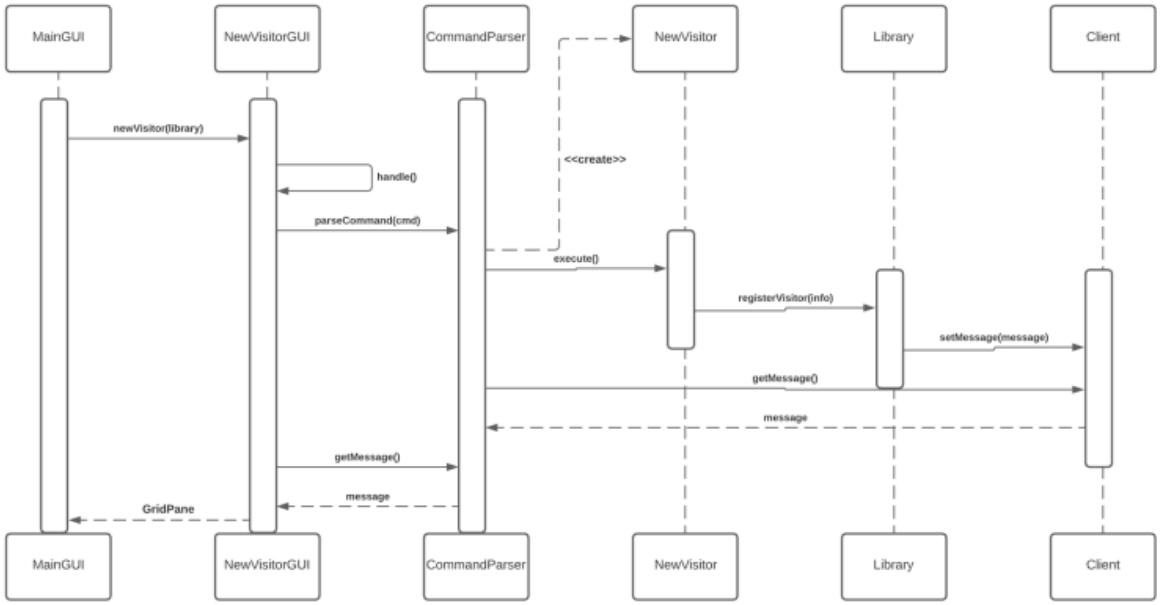
5.3.1 Class Diagram



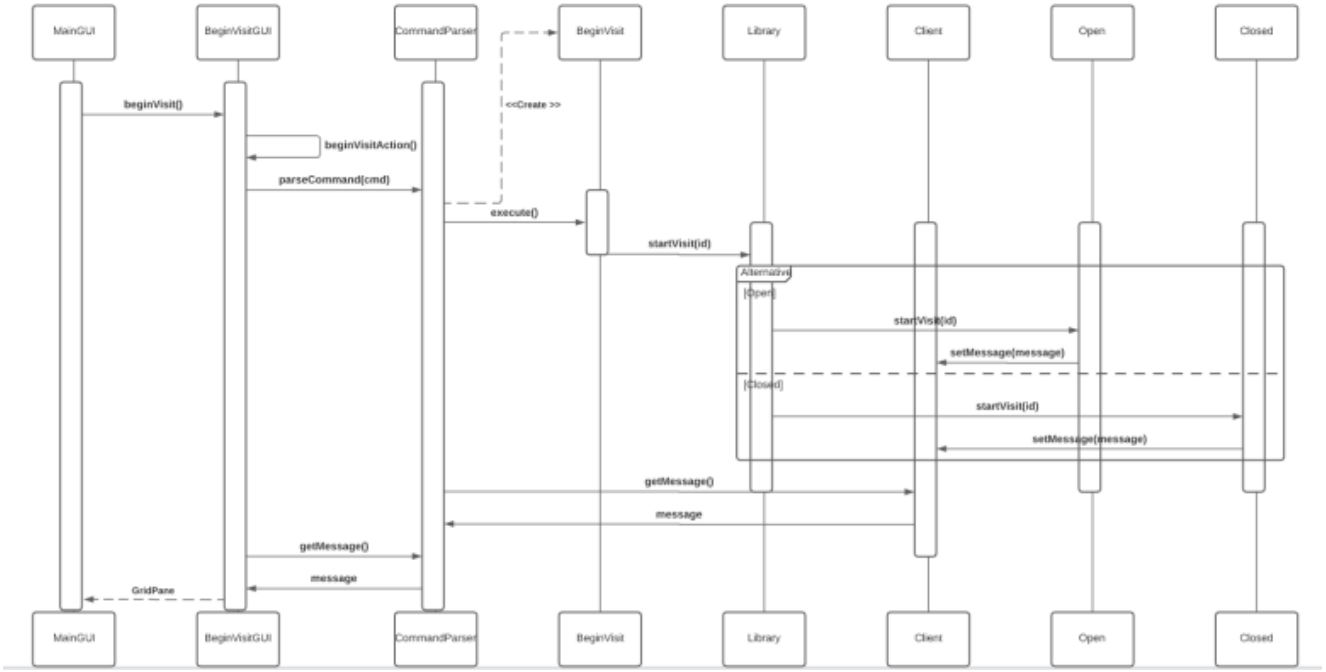
5.3.2 Interaction Diagrams

Presented below are the sequence diagrams that use the commandParser class to parse and execute each of the different commands.

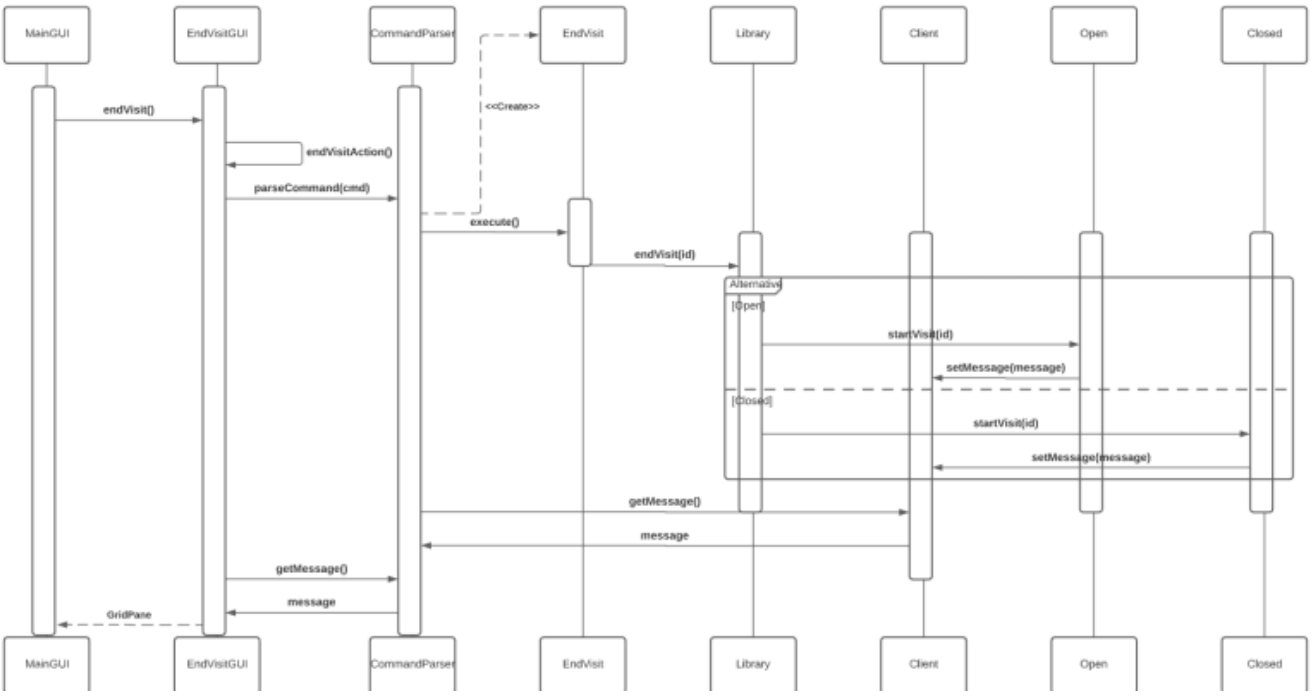
New Visitor:



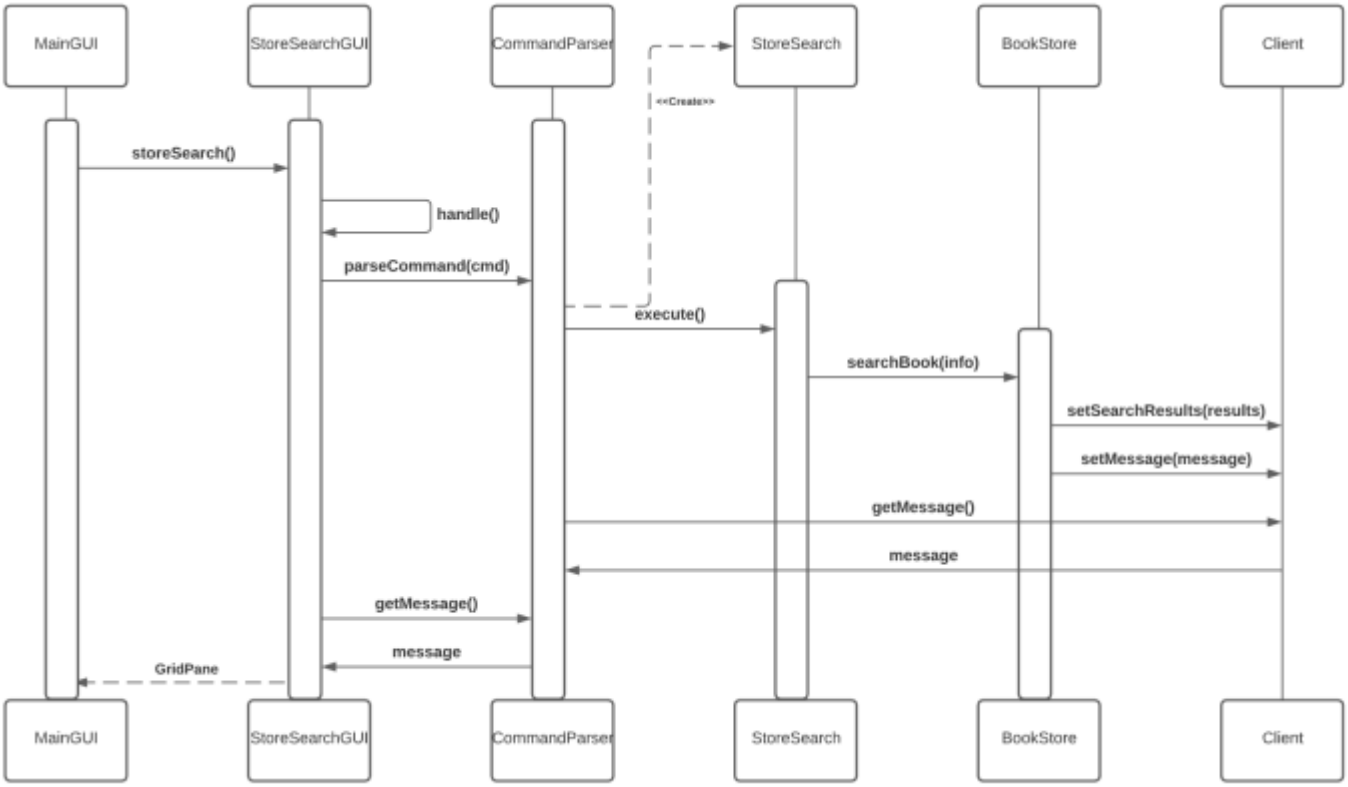
Begin Visit:



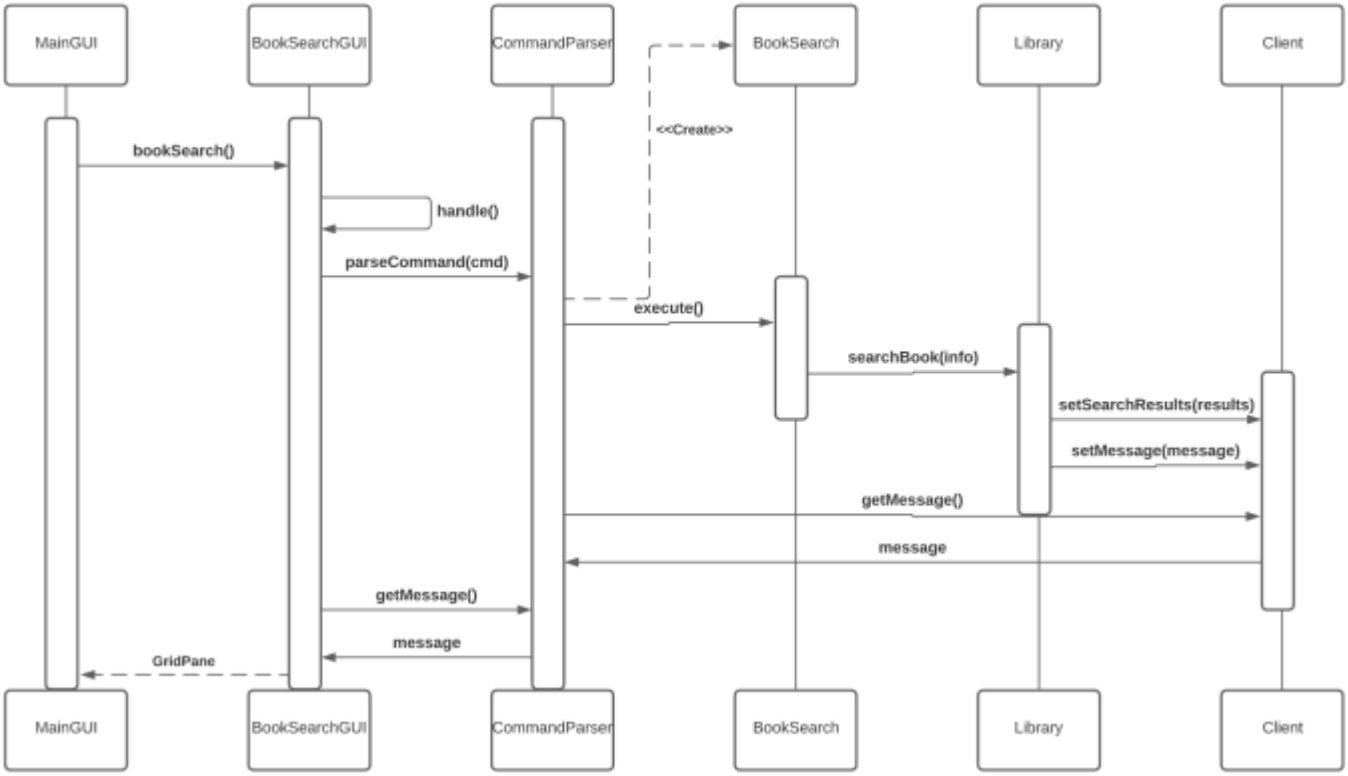
End Visit:



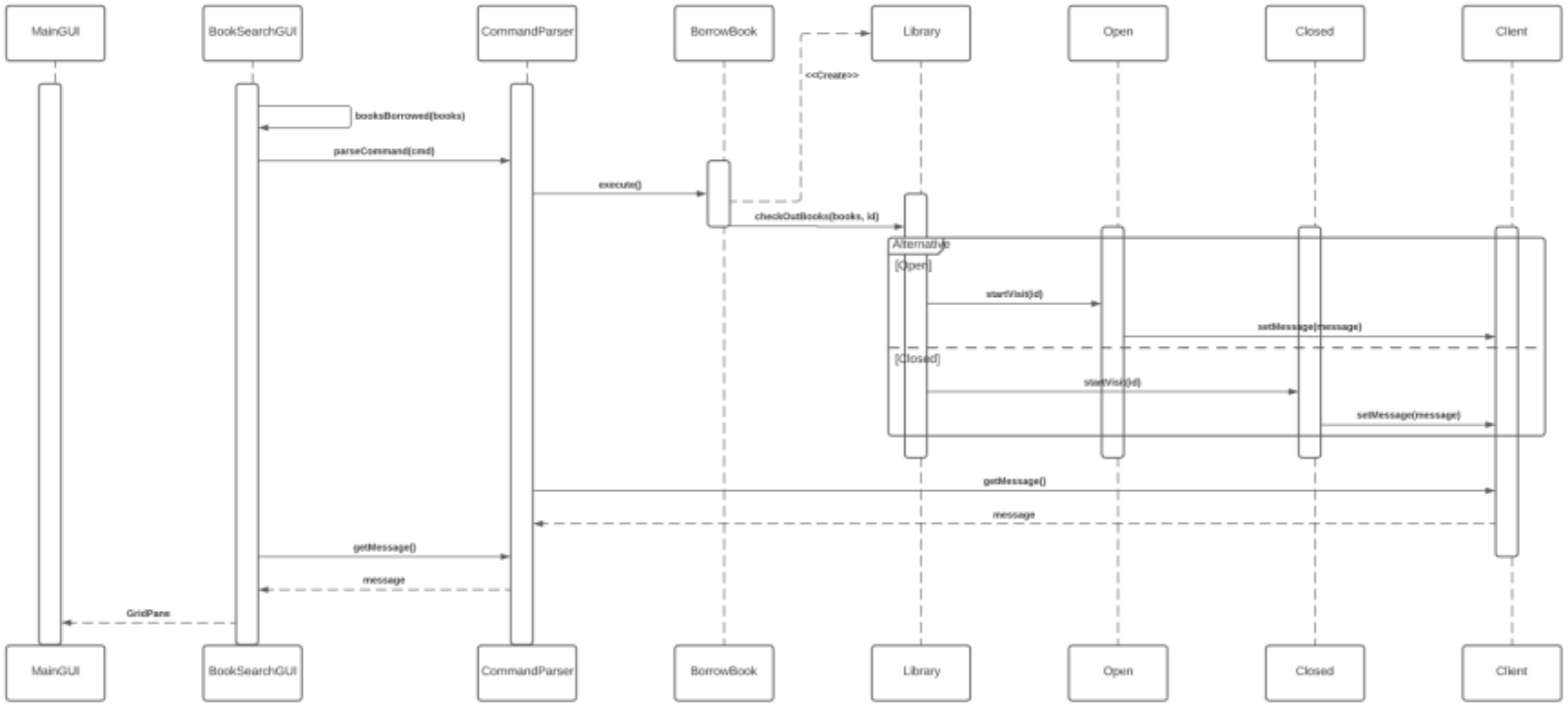
Store Search:



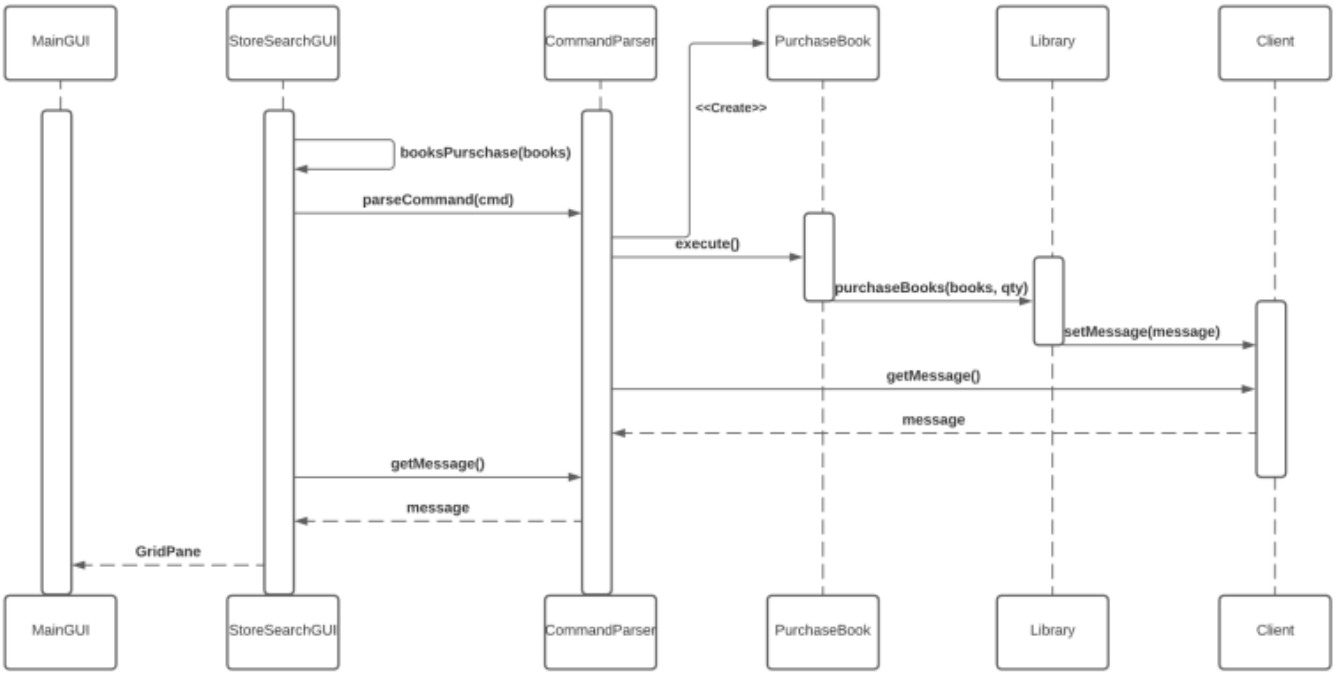
Book Search:



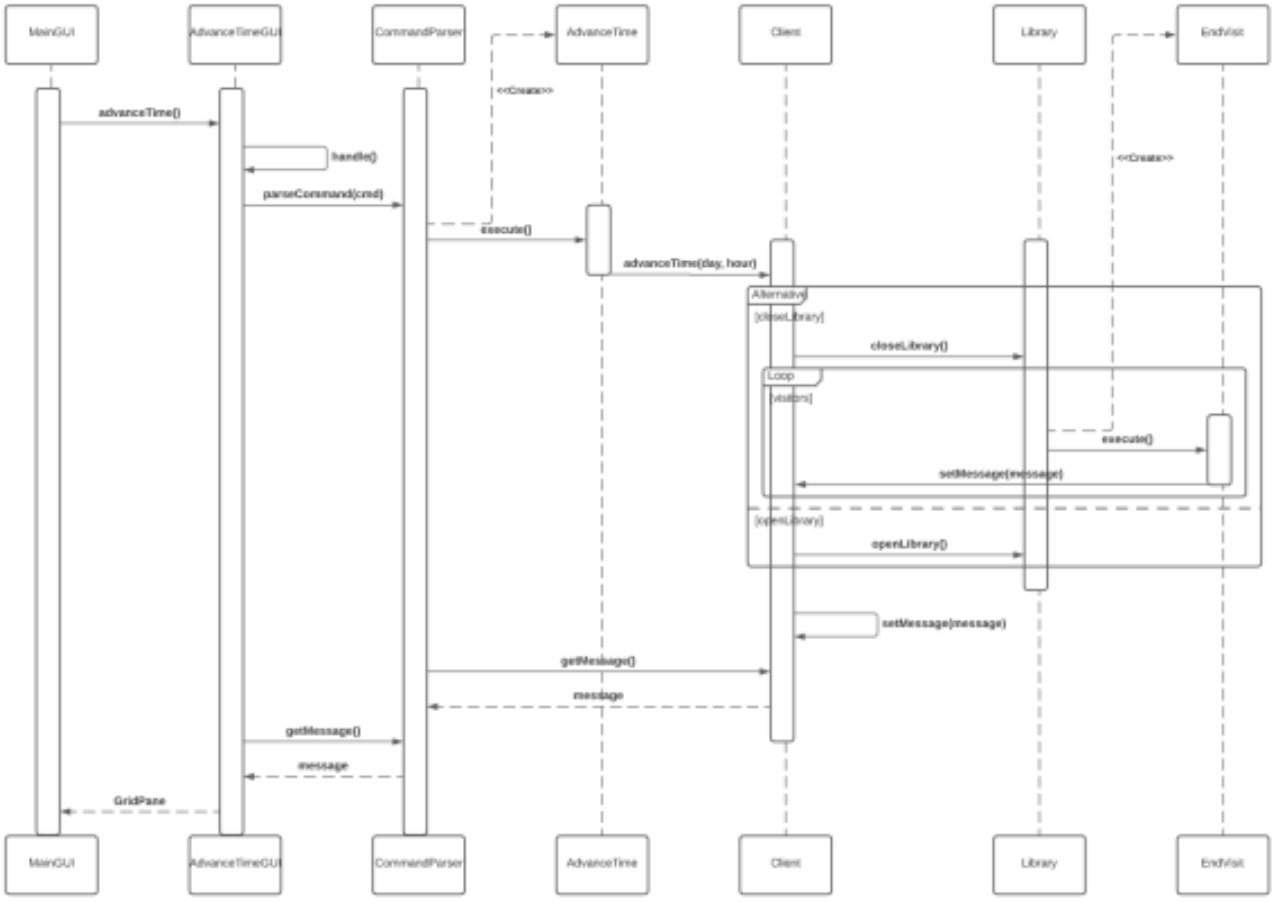
Borrow Book:



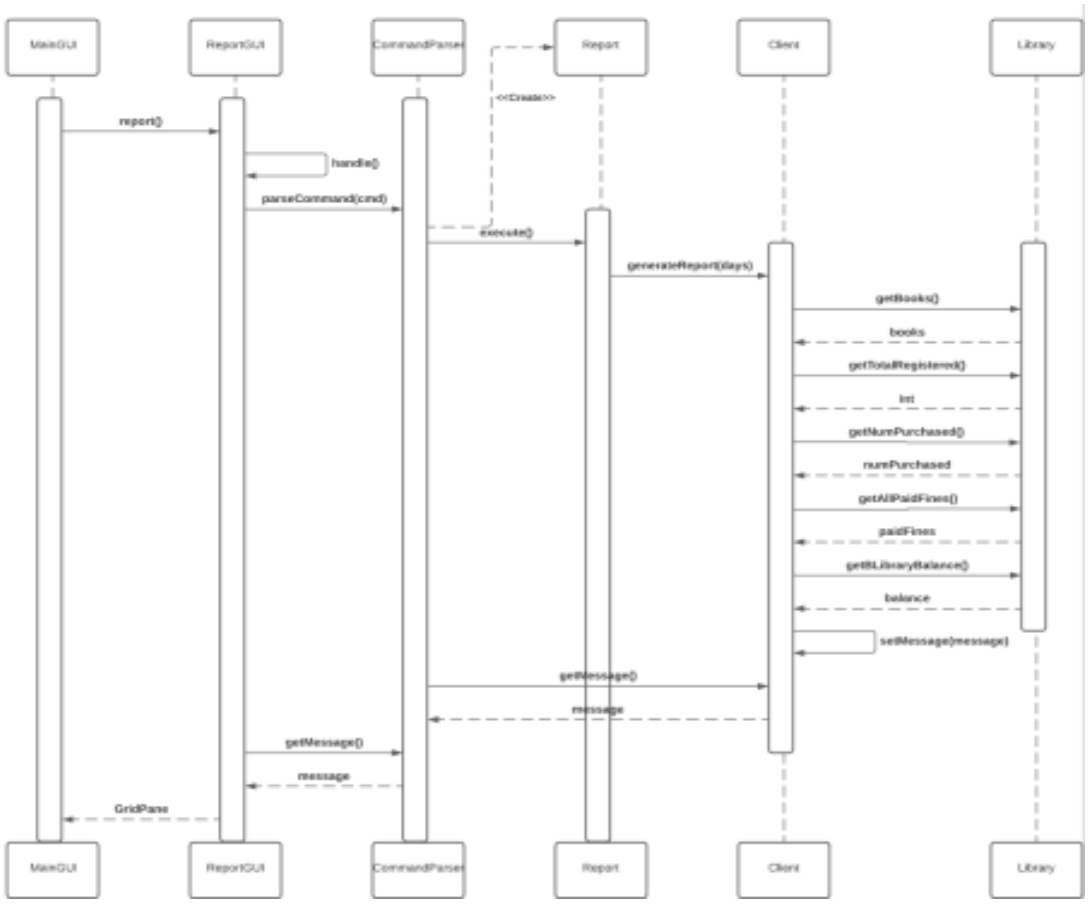
Purchase Book:



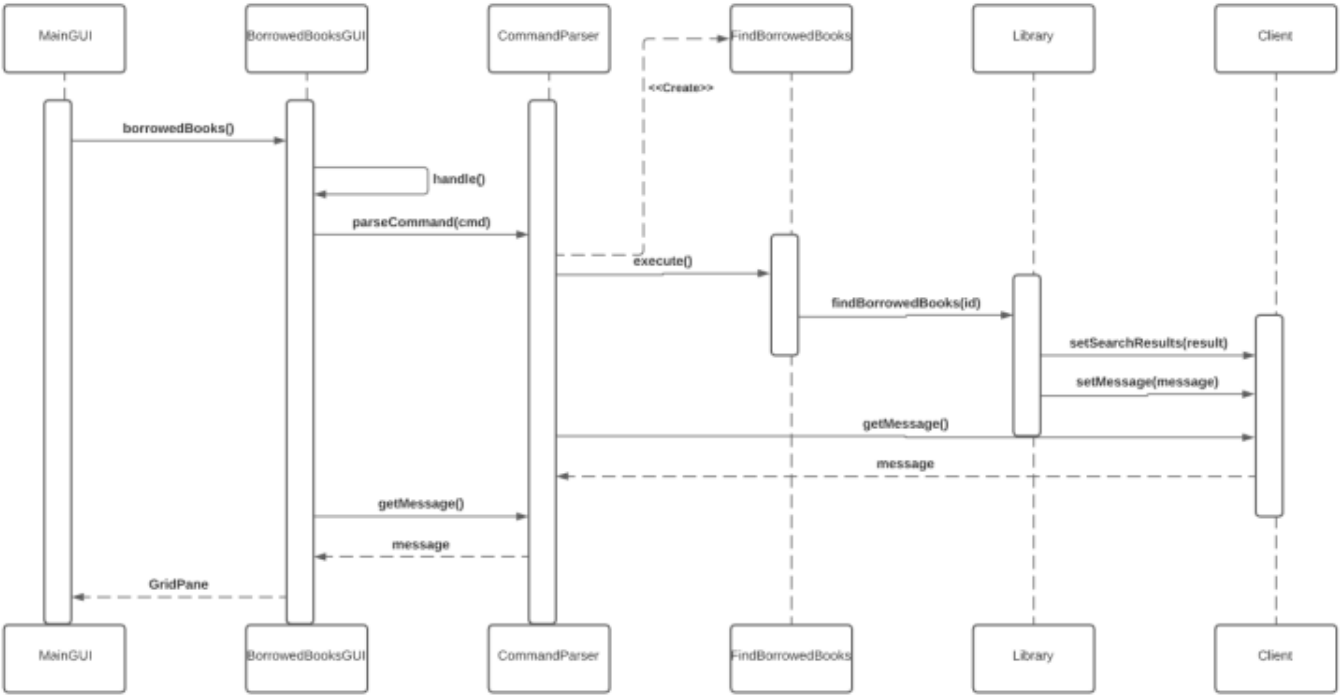
Advance Time:



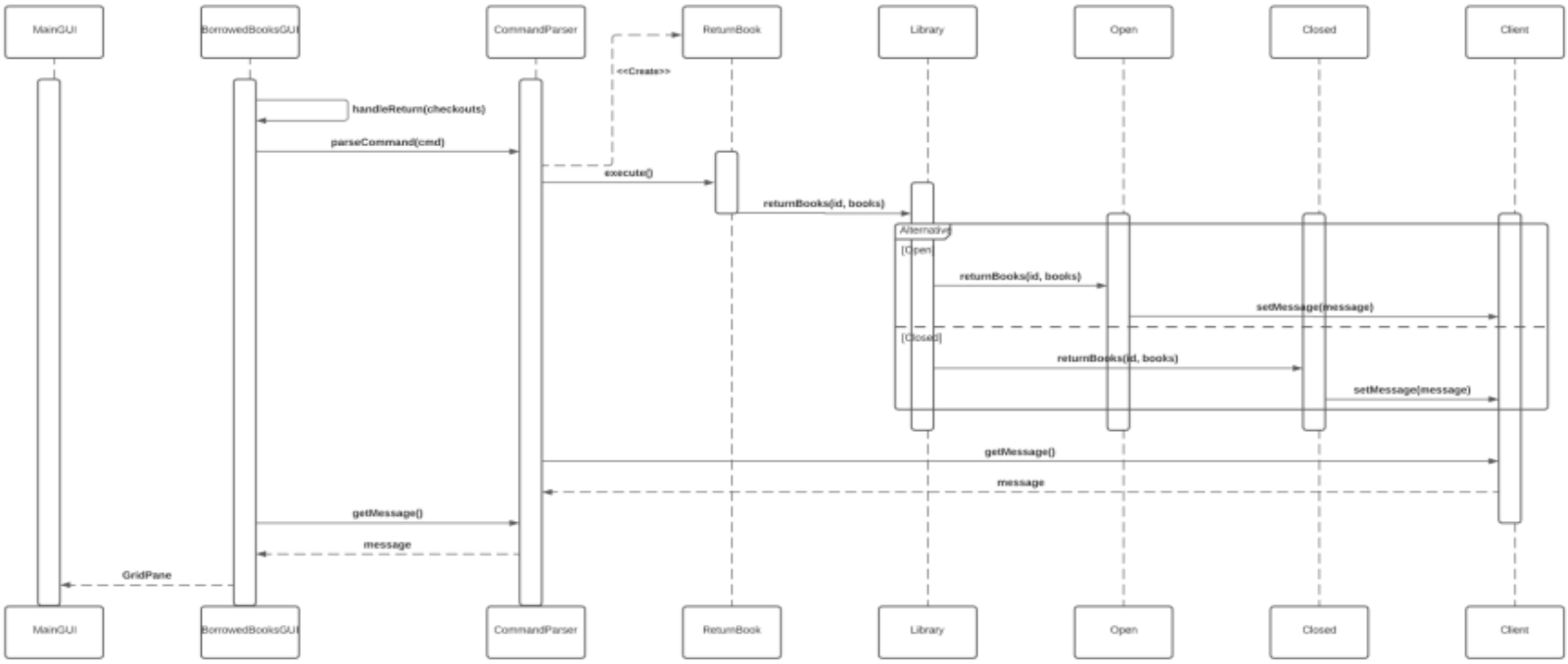
Report:



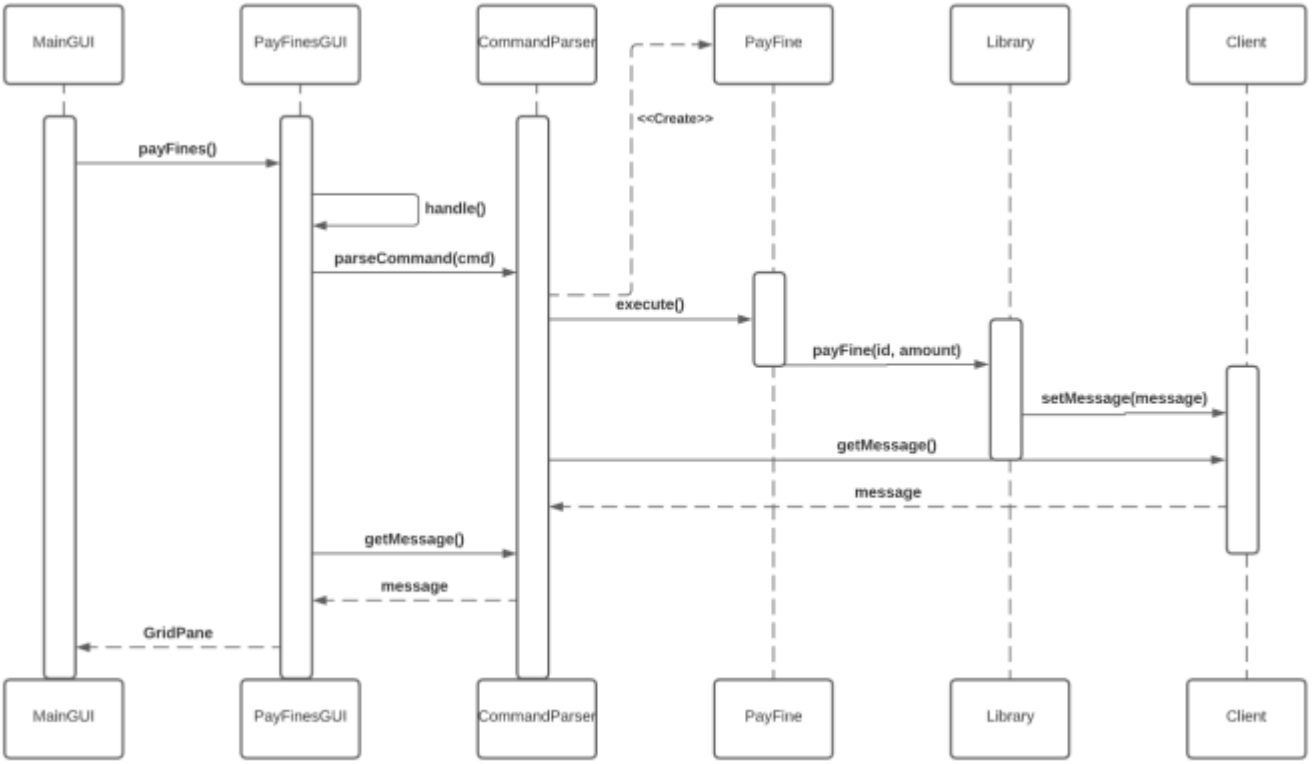
Find Borrowed Books:



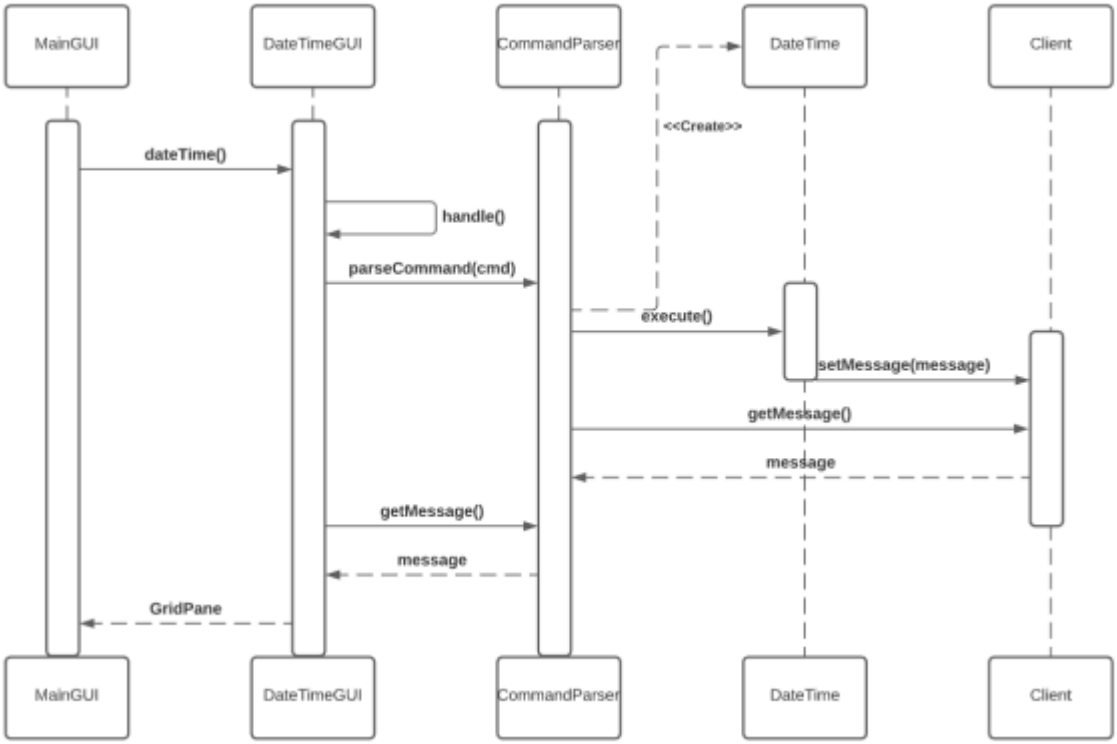
Return Book:



Pay Fine:



Date Time:



5.3.3 Detailed Design

Library command implements the command design pattern. Command, the only interface that is contained to the command package has an execute() method that returns a String. We need to set up classes by requesting String, and store the data in the command. When the execute() is eventually called, the command will interact with LBMS and receive or modify its data to generate a response string. The request format must be proper, otherwise it will respond a correct format that you need to input.

5.4 Design Patterns

An overview of the incorporated design patterns used in our implementation of the LBMS system.

Name of the used Pattern (GoF pattern): State Pattern	
Purpose: The library will not allow the following transactions while closed	
Participants	
Class	**Role in pattern**
Library	Library class holds all the information about purchased books, registered visitors, current visitors; and handles all the methods that may access them.
Library State	LibraryState acts as an interface for the 2 states of the library.
Open	Open class holds all methods that are exclusive to the open state.
Closed	Closed class includes all methods that are exclusive to the closed state.

Name of the used Pattern (GoF pattern): Strategy Pattern	
Purpose: The system shall store book data for all books currently in BWL's possession, The system will store the data of the book check out transaction	
Participants	
Class	**Role in pattern**
Book store	BookStore holds all the products in the system through parsing a file.
Library	Library holds all the products that I has purchased.
Product	Product acts as an interface for all the products that are available now or in the future.
Book	Book uses the Product interface to represent a single book in the system.

Name of the used Pattern (GoF pattern): Command	
Purpose: The library system takes the command and generates a response string that follows the specific format.	
Participants	
Class	**Role in pattern**
Command<<interface>>	The template for the concretes.
AdvanceTime	Defines the current time by days, hours and minutes.
BeginVisit	Defines a visitor begin to visit.
BookSearch	This class defines the visitor to search the books.
BorrowBook	This class defines the visitor to borrow the books.
CommandParser	This class parses all the commands that corresponds to each class inside of the Command package.
DateTime	This class defines the current date and time.
EndVisit	This class defines the visitor who leaves the library and ends the visit.
FindBorrowedBooks	This class defines the books that were borrowed by specific visitor.
NewVisitor	This class defines a new coming visitor and register in order to begin visit.
PayFine	This class defines the amount of fine that a visitor needs to pay.
PurchaseBook	This class defines the amount of books that need to purchase by a visitor.
Report	This class generates a report of the library.
ReturnBook	This class defines the action of returning books.
StoreSearch	This class defines the visitor to search the book in the bookstore.

5.5 Human interface

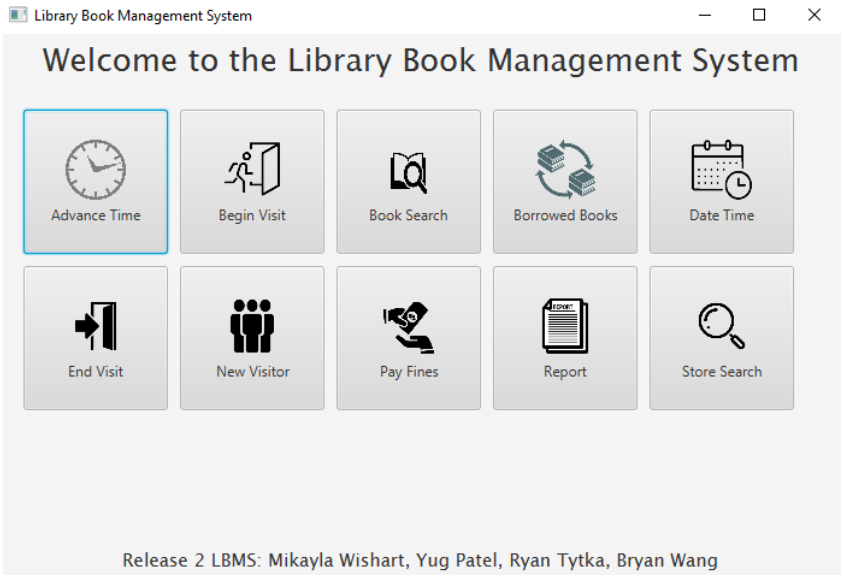
5.5.1 User interface design

We are using JavaFX to design and implement our Graphical User Interface.

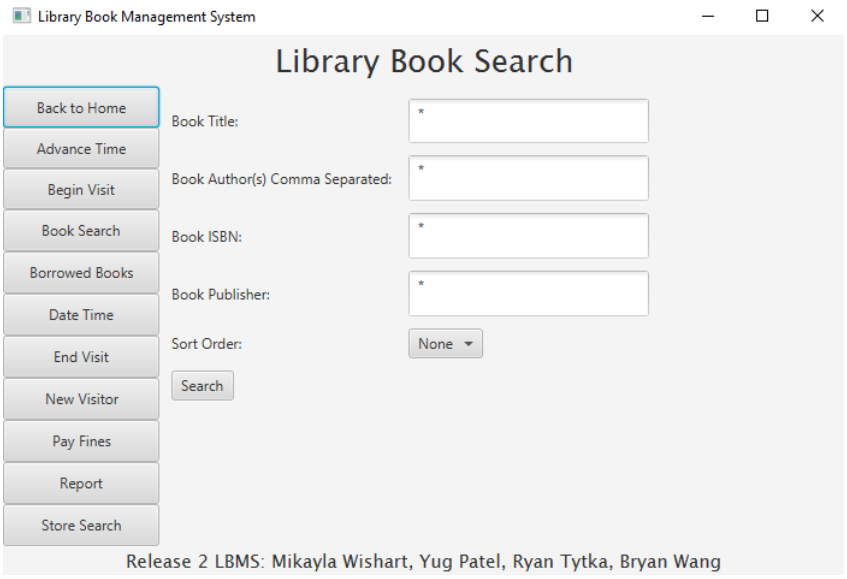
5.5.2 Description of the user interface

The application will have a home screen where the user can access the multiple different functions from. By clicking a button for a specific function, the user will go to a different screen with controls that will let the user carry out that function. There will be a sidebar for quick navigation between menus.

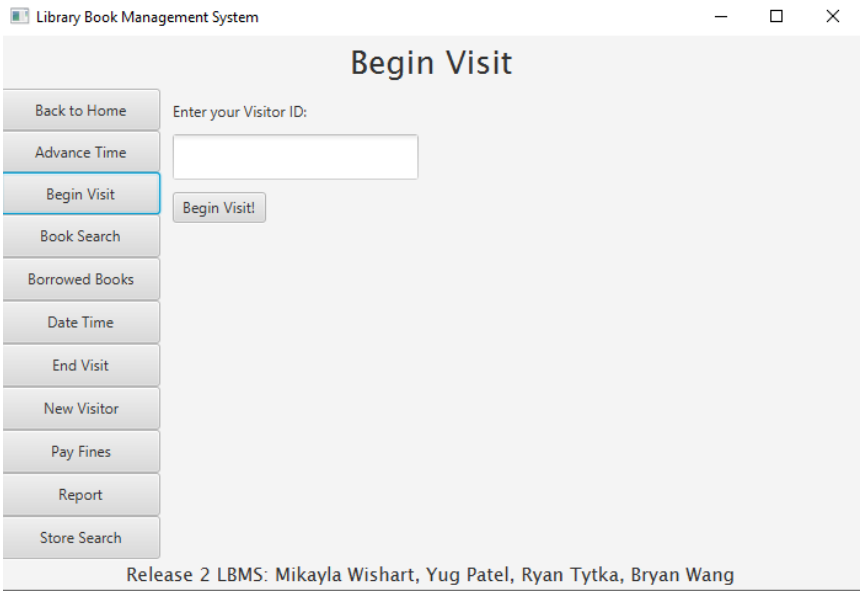
Home Menu Interface



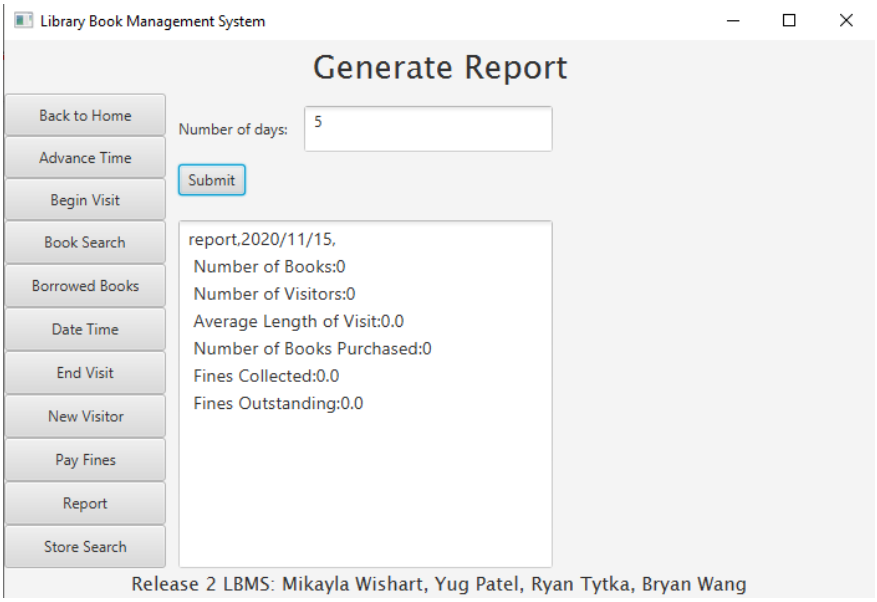
Book Search Interface



Begin Visit Interace



Generating a Report Interface



6 Implementation Issues & Challenges

Using Java's Calendar class was a challenge, as it was new to us and we had to learn how to format the dates and manage the Calendar objects. We also struggled a bit when trying to place checkboxes into the GUI using JavaFX.

7 Supporting Systems Requirements

7.1 Design Hardware Requirements

Our application runs on Mac or Windows.

7.2 Design Software Requirements

At a high-level this project will be source controlled on GitHub and implemented in Java as a desktop application. It will be written using the Java 13 SDK and JavaFX 4.

8 Conclusions & Future Extensions

In conclusion, this project was a valuable learning experience that allowed us to understand and implement different design patterns. Future additions to the project could include adding new types of materials to purchase, such as magazines or videos.

9 Related Materials

Listed below are the textbooks used in class to learn about design patterns.

- "Design Patterns: Elements of Reusable Object-Oriented Software", Gamma, Helm, Johnson, Vlissides (The Gang of Four [GOF])
- "Head First Design Patterns", Eric Freeman & Elisabeth Freeman

10 Appendix

10.1 Class Diagram