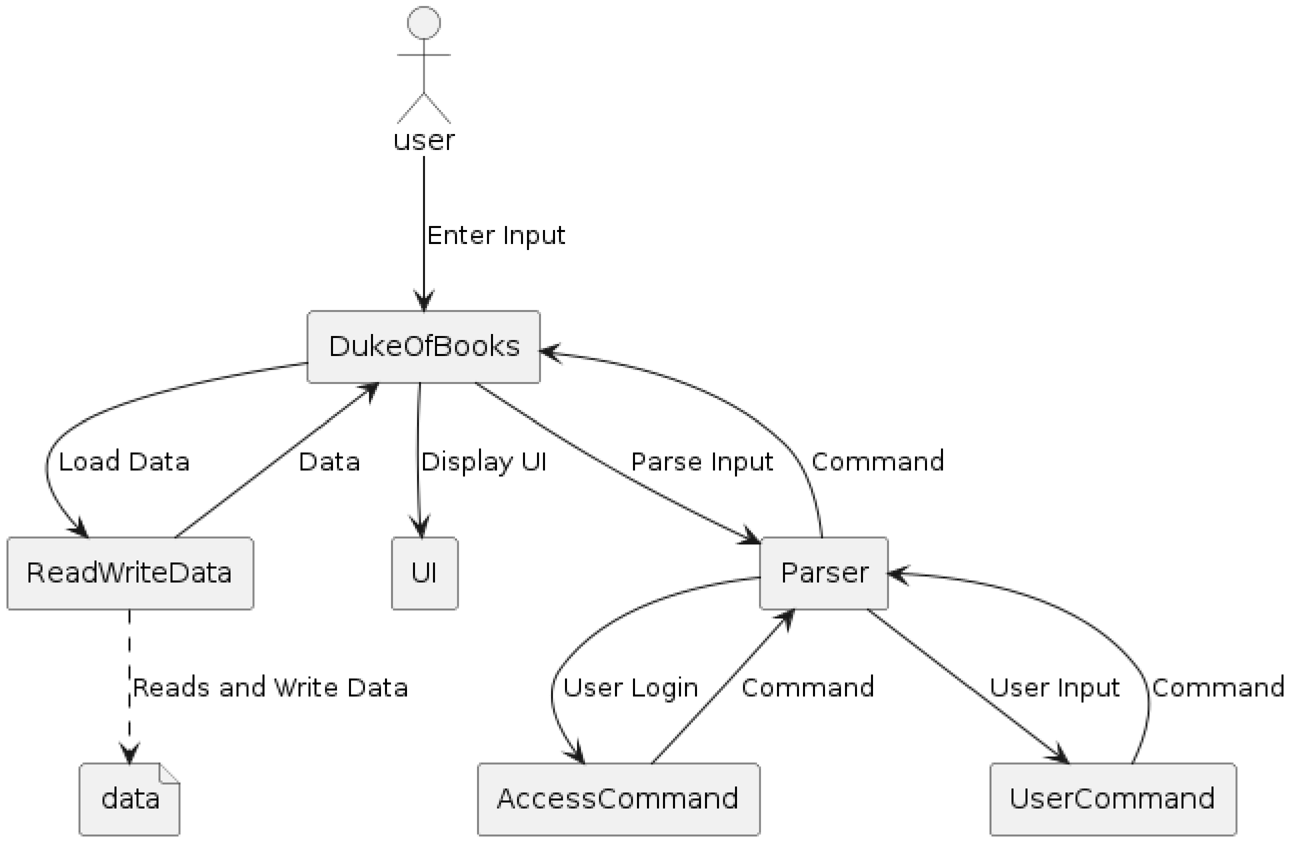
Developer Guide

Acknowledgements

- NUS SC2113 Duke project
- NUS SC2113 tp

Design & Implementation



Access commands related to accounts are fetched by the `getUserCommand()` method in the *TextUi* class. These commands are parsed by the `parseAccessCommand()` method in the *AccessCommandParser* class, which is then parsed to the respective command classes responsible for executing the relevant features.

Upon execution, access commands return an AccessResponse object containing a User object and a message. If a log in or sign up is unsuccessful, the User object will be null. If successful, the AccessResponse will contain the signed-in User object. The accompanying message will display a relevant message to the user.

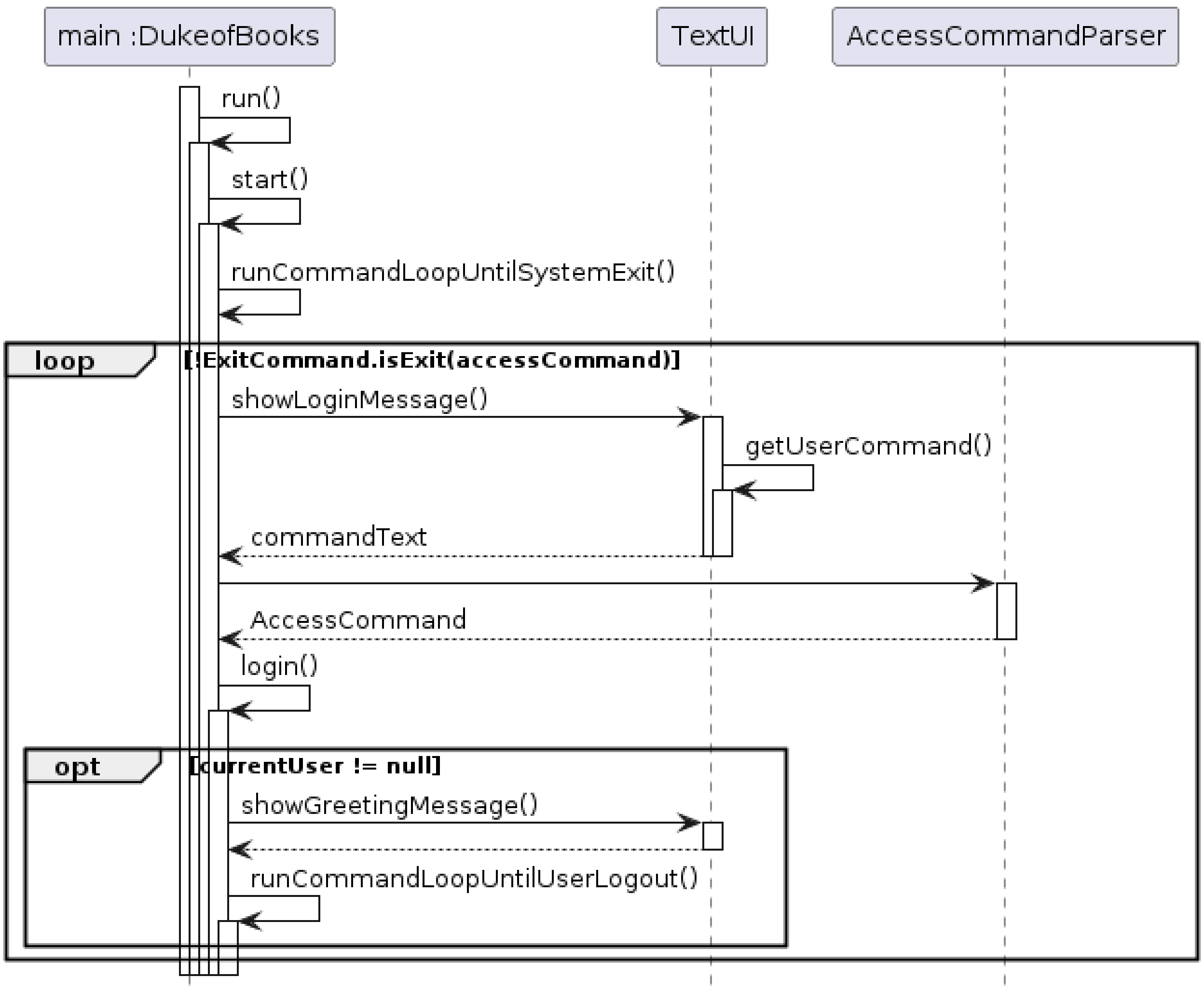
Regardless of whether the password is successfully changed, the command will return a null User object. However, the message will vary depending on the result.

User data is stored in a HashMap mapping unique usernames to *User* objects, which store personal information as well as the username and the hashed password. Passwords are not explicitly stored for security reasons.

There is a default superuser whose username is `root` and password is also `root`.

Users can only execute user commands, such as checking their borrowing history or borrow books, after successfully logging into the system. They must log out before exiting the application.

Log in



Description: Log in to the account using saved username and password.

Format: `login -username USERNAME -password PASSWORD`

Example:

- `login -username me -password mypassword`

When a user successfully logs into their account using their saved username and password, the application will display a welcome message and allow the user to execute user commands such as borrowing and returning books. The application only allows for an exact match.

If the user is unable to log in, the application will wait for the next access command (either login or signup).

Sign up

Description: To use the system, users who do not have an account need to sign up with a unique username, a password, and a name. The username and password can only contain letters and numbers, but the name can contain spaces.

Format: `signup -username USERNAME -password PASSWORD -NAME FULL_NAME`

Example:

- `signup -username me -password apassword -name my name`

If the username entered by the user is unique in the database and the password is valid, the account creation process will be successful. However, if the username is already present in the database or the username/password contains invalid characters such as '$' or '^', the signing up process will be unsuccessful.

Change Password

Description: Existing users can change their passwords by providing their username, old password and new password.

Format: `password -username USERNAME -old OLD_PASSWORD -new NEW_PASSWORD`

Examples:

- `password -username joe123 -old 123456 -new 654321`

The password changing is successful only if the username exist in the database, the old password matches with the corresponding username, and new password is considered valid (i.e., it does not contain invalid characters).

Design & Implementation for Users Commands

All user commands will be fetched by the `getUserCommand()` method in the *TextUi* class. The `parseCommand()` method in the *Parser* class will then parse the command and pass it to the respective command classes. The command classes will be responsible for executing the relevant features of the application.

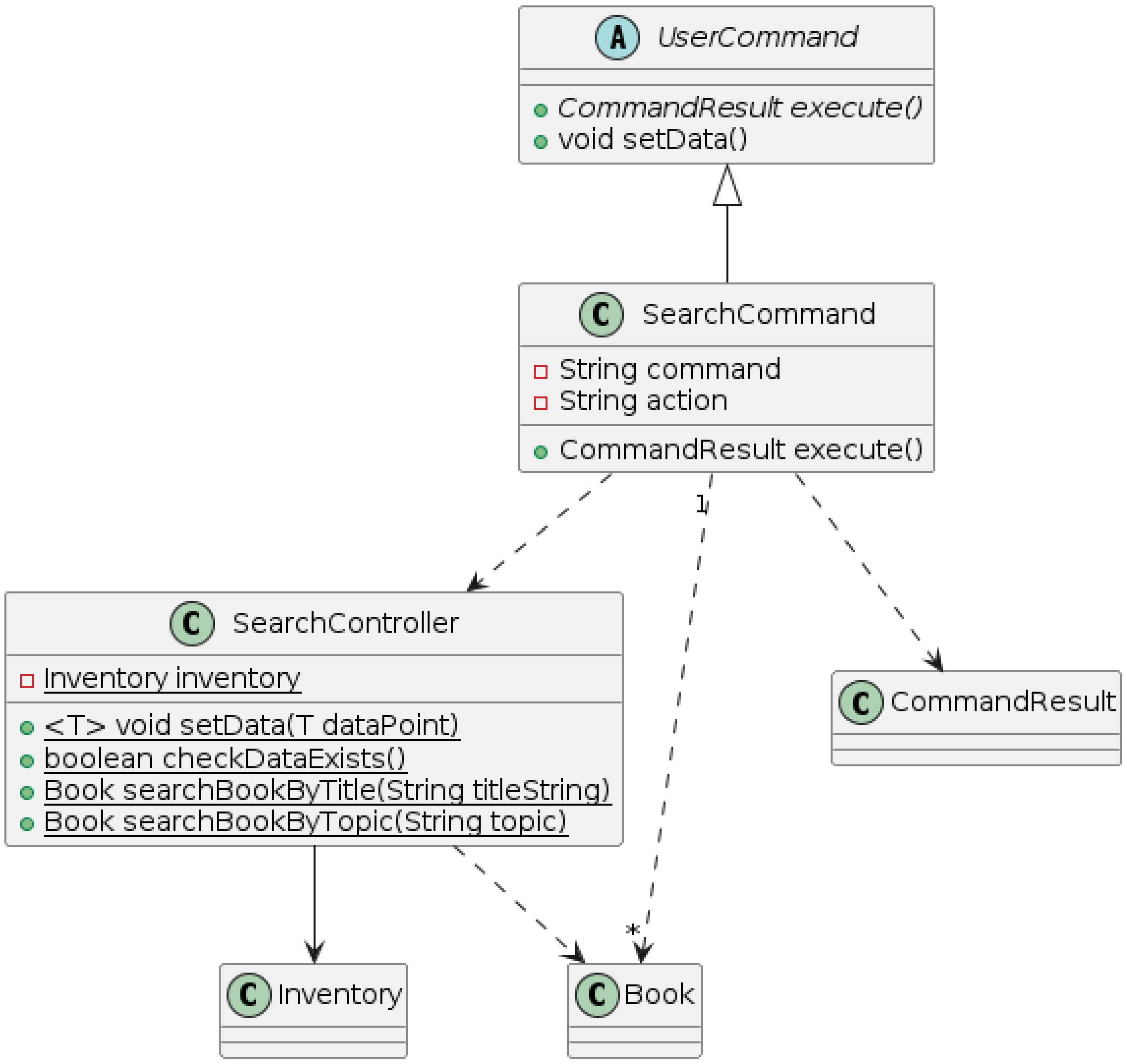
Search book by title

Description: Searches for a book using its title.
Format: `search -title TITLE`
Example:

- `search -title Python Programming`

Successful searches will result in the program outputting the relevant book details, including ISBN, title, author, and topic. If the search is unsuccessful, the program will output a message indicating that there is no match in the inventory for the input title.

Search book by topic



Description: Searches for a book by its topic.
Format: `search -topic TOPIC`
Example:

- `search -topic Business`

Successful searches will result in the program outputting the relevant book details, including ISBN, title, author, and topic. If the search is unsuccessful, the program will output a message indicating that there is no match in the inventory for the input title.

Check book availability

Description: Check if a book is available for borrowing.
Format: `check -title TITLE`
Example:

- `check -title Python Programming`

The program will check whether the book is available for borrowing and return the borrowing status (borrowed or not borrowed). It will also handle the case where the book does not exist in the inventory.

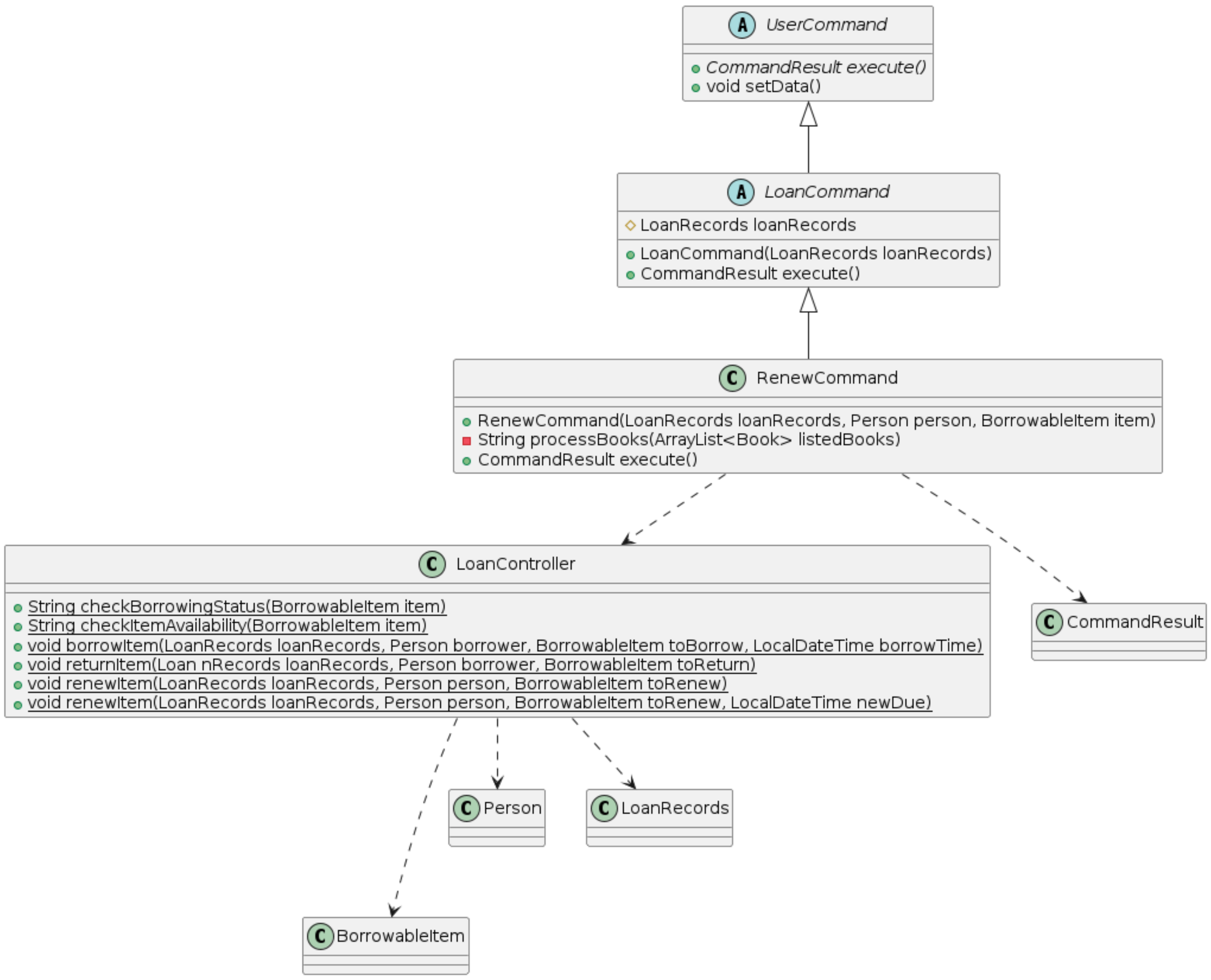
Borrow book

Description: Borrow a book from the library.
Format: `borrow -title TITLE`
Example:

- `borrow -title Python Programming`

For successful borrowing, the program will output a message indicating that the action was successful and mark the book as borrowed in the system. For unsuccessful borrow requests, the program will either output a message stating that there is no such book in the inventory, or a message indicating that the book is already on loan at the time of the borrow request.

Renew borrow period of book

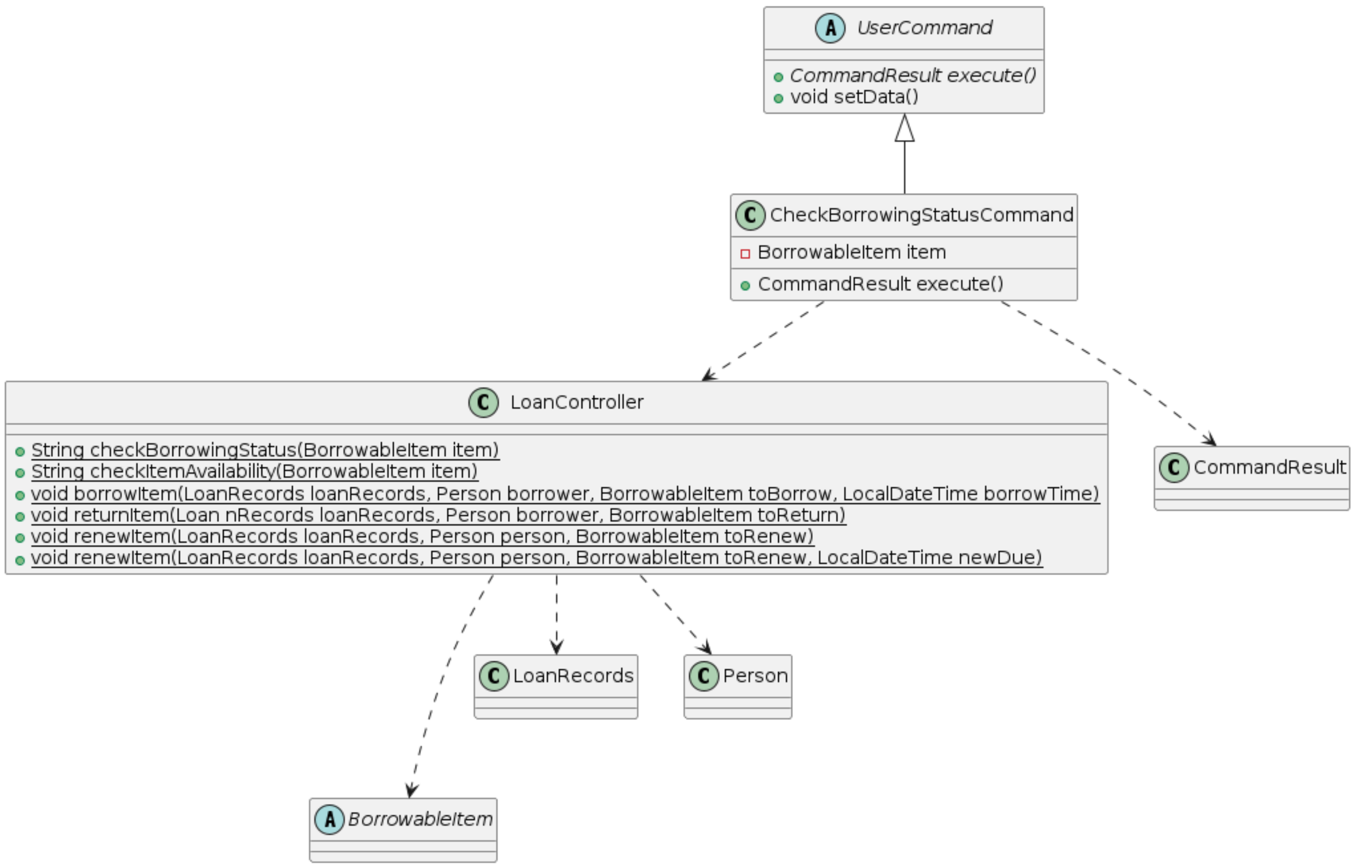

UserCommand
CommandResult execute()
void setData()
LoanCommand
LoanRecords loanRecords
LoanCommand(LoanRecords loanRecords)
CommandResult execute()
RenewCommand
RenewCommand(LoanRecords loanRecords, Person person, BorrowableItem item)
String processBooks(ArrayList<Book> listedBooks)
CommandResult execute()
LoanController
String checkBorrowingStatus(BorrowableItem item)
String checkItemAvailability(BorrowableItem item)
void borrowItem(LoanRecords loanRecords, Person borrower, BorrowableItem toBorrow, LocalDateTime borrowTime)
void returnItem(Loan nRecords loanRecords, Person borrower, BorrowableItem toReturn)
void renewItem(LoanRecords loanRecords, Person person, BorrowableItem toRenew)
void renewItem(LoanRecords loanRecords, Person person, BorrowableItem toRenew, LocalDateTime newDue)
CommandResult
Person
LoanRecords
BorrowableItem

Description: Renew borrowing of books for a fixed duration.
Format: `renew -title TITLE`
Example:

- `renew -title C++Primer`

For successful renewal of books, the program will output a string showing that the action is successful and also change the due period of borrow in the system. The program will handle error cases such as incorrect titles provided or books not available for renewal.

Check borrowing status


UserCommand
CommandResult execute()
void setData()
CheckBorrowingStatusCommand
BorrowableItem item
CommandResult execute()
LoanController
String checkBorrowingStatus(BorrowableItem item)
String checkItemAvailability(BorrowableItem item)
void borrowItem(LoanRecords loanRecords, Person borrower, BorrowableItem toBorrow, LocalDateTime borrowTime)
void returnItem(Loan nRecords loanRecords, Person borrower, BorrowableItem toReturn)
void renewItem(LoanRecords loanRecords, Person person, BorrowableItem toRenew)
void renewItem(LoanRecords loanRecords, Person person, BorrowableItem toRenew, LocalDateTime newDue)
CommandResult
LoanRecords
Person
BorrowableItem

Description: Check status of borrowed book.
Format: `status -title TITLE`
Example:

- `status -title C++Primer`

The program will output the details of the relevant book being borrowed and also show the due date of the loan. The program will handle cases where there is no such book in the borrow history.

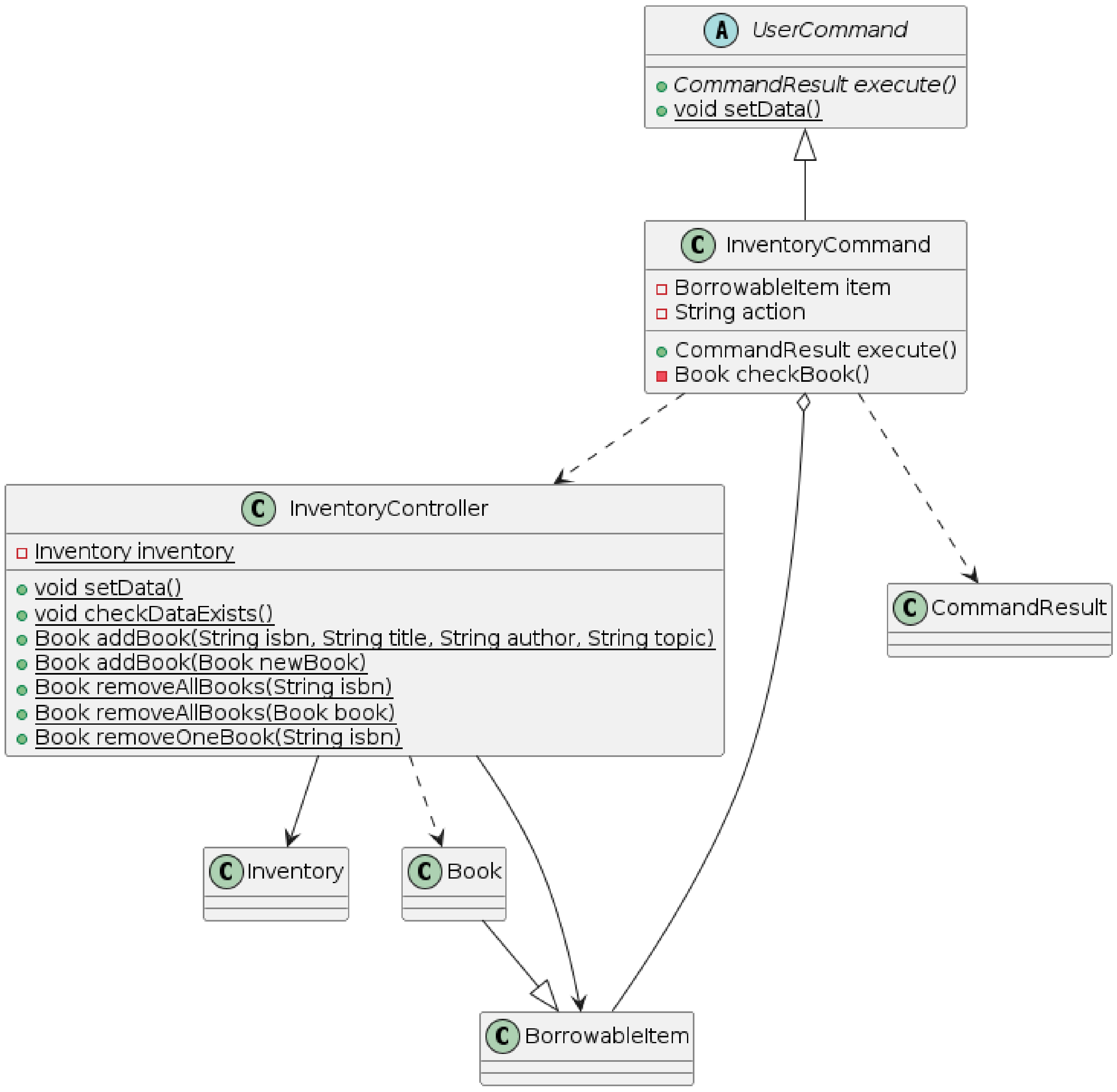
Return book

Description: Return a book to the library.
Format: `return -title TITLE`
Example:

- `return -title C++Primer`

The program should output a success message upon successful book returns and mark the book as available for borrowing by other users in the system. It also handles cases of incorrect title inputs or unsuccessful returns.

Add book



Description: Add new books into the system. Format: `librarian -title TITLE -topic TOPIC -author AUTHOR -isbn ISBN -action add`
Example:

- `librarian -title C++Primer -topic Programming -author James -isbn 12345 -action add`

This feature is only applicable for the admin. Upon successful addition of a new book, the program will output a message to inform the librarian. The new book will also be added to the inventory of the system, allowing users to borrow the book from the library. Error inputs and missing inputs will be handled accordingly.

Delete book

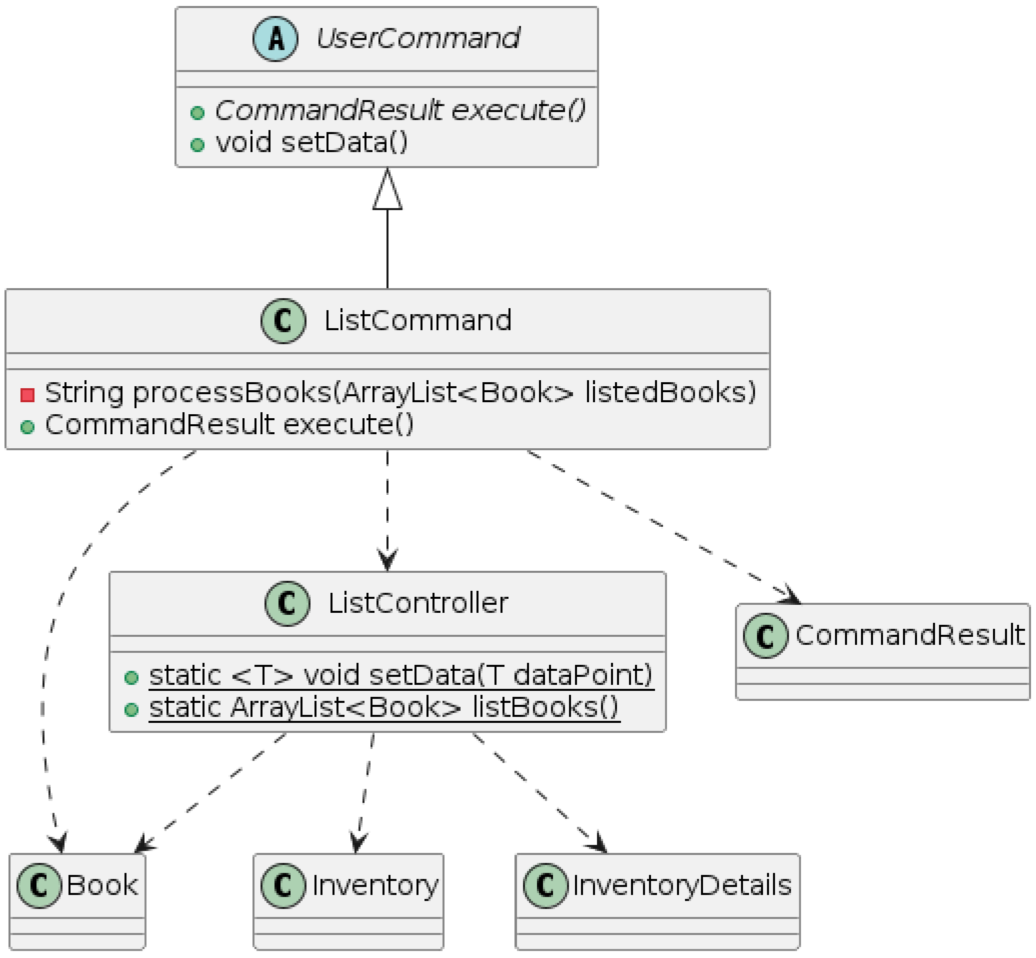
Description: Remove books from the system.
Format: `librarian -title TITLE -topic TOPIC -author AUTHOR -isbn ISBN -action`

`delete`
Example:

- `librarian -title C++Primer -topic Programming -author James -isbn 12345 -action delete`

This feature is only applicable for the admin. For successful deletion of a book, the program will output a message to inform the librarian. The book will also be removed from the inventory of the system so that it will not be borrowable by users anymore. Error inputs and missing inputs will be handled accordingly.

List book



Description: List all the books in the library. Format: `list`
Example:

- `list`

The program will list out all the books in the library inventory. If the inventory is empty, the program will output a message indicating that there is currently no book in the library system.

View history

Description: The program includes a feature to check the borrow history of books, which has two versions - admin and user. The admin version permits the administrator to access the entire borrow history, while the user version only permits the user to view their own borrow history.

Format: `history`
Example:

- `history`

The program will output the borrow history of books according to the accessibility mentioned in the description above. The output will include the book title and other book details, the status of the book (whether it is available for borrowing or has been borrowed), and details of the loan, which will include the date and time of borrowing and returning.

Make payment

Description: Make payment for overdue loans.
When returning item(s) that are overdue, the user will be prompted to make a payment. Failure to do so will result in the returning action being unsuccessful.

Example:

```
|| Enter command: return -title Introduction to Algorithms
|| Payment method:
|| 1.PayNow
|| 2.Debit/Credit card
|| 3.NETS
|| 4.Cash
|| 5.Cancel Payment
2
|| Payment Successful! Transaction ID: 978-0262033848202304090809
|| Item has been returned!
```

Product Scope

Target user profile

This program is designed for National University of Singapore (NUS) Computer Science (CS) students who wish to borrow and read CS related books.

Value proposition

CS students are often busy, so a command line interface (CLI) program without a GUI can make finding the books they need quick and efficient. This software can also help them keep track of loans and return dates.

User Stories

Version	As a ...	I want to ...	So that I can ...
v1.0	user	search by title	borrow a book to read
v1.0	user	search by topic	borrow a book to read
v1.0	libriarian	add or remove book	change the books in the inventory

Version	As a ...	I want to ...	So that I can ...
v1.0	user	return book i borrowed	remove the borrow status taged to my account
v2.0	libriarian	pay my fine	continue to borrow more books and see my exam results
v2.0	user	create an account	use the system
v2.0	user	change password	log in with a new secure password
v2.0	user	renew book	continue reading the book for a longer period
v2.0	user	list books	know what are the books in the library
v2.0	user	search by topic and title	filter the results further

Non-Functional Requirements

1. Data should be stored in text file so that information like borrowed books and inventory are not lost
2. System should run on Java 11

Glossary

- *librarian* - Admin user
- *user* - Normal user

Instructions for Manual Testing

1. Download the jar file *tp.jar* in release v2.0
2. Launch the jar file using the command `java -jar tp.jar`