Project Summary: Library Management System by Emi Herdman

Introduction

For my project, I decided to create a project that is based on a Library Management System, in which you can loan or return books and receive a transaction receipt

Classes, Inheritance, and Object Associations

For this I created an ABSTRACT class LibraryInv.java which is contained in the package of books.

LibraryInv contains attributes common to a book, such as ISBN, release date, and the book's name.

A Book.java class is also contained the package extends this class as a super, and adds attributes that is specific to a book such as an Author and whether it is loaned or not, this class also implements the LibraryItem.java Interface which I will explain later

Hyperlink to location of use: LibraryInvClassAbstract and BookClass

Constructors and Constructor Chaining

Book.java uses a constructor which uses a super to take attributes from the ABSTRACT class of LibraryInv.java which uses the ISBN, release date and the book's name as attributes alongside the attribute author and BookLoaned which is contained in Book.java

Hyperlink to location of use: BookClass

Properties and Visibility Modifiers

Throughout the project I demonstrated the use of visibility modifiers to enhance the security of the program, I used the protected modifier on setters to ensure that values could not be set where they should not be, I did this in LibraryInv and Books.java.

For private visibility I integrated primarily in the LibraryUser class for the methods handleLoan and handleReturn

Hyperlink to location of use: BookClass, LibraryInvClassAbstract and LibraryUserClass

Accessor Methods (Getters and Setters)

I implemented methods like 'getIsbn()', 'getBookName()', and 'getDateReleased()' in the LibraryInv Class to allow controlled access to public fields. These methods allow me to extract data from the Objects,

Hyperlink to location of use: BookClass and LibraryInvClassAbstract

Packages

The project is organized into five packages, which are packaged into a singular package to follow the guidelines set out by Oracle for example miniproject.books/inputoutput

Hyperlink to location of use: booksPackage, exceptionsPackage, inputoutputPackage, interfacesPackage, menuPackage and miniprojectPackage

Enumerations

I used an enumeration (Enum) in the BookLoaned class to represent the loan status of books. The two values can be set as is LOANED or AVAILABLE.

Hyperlink to location of use: BookLoanedClassEnum

Static vs Instance

In my project I used static methods throughout the project in multiple classes such as:

The String variable filename, and the class Storage

Hyperlink to location of use: StorageClass

Class Hierarchies (Abstract and Final Classes and Methods)

The 'LibraryInv' class is abstract because it contains common functionality for all library items. It cannot be instantiated directly, but it serves as a base class for other objects like 'Book'. I used 'final' for methods where overriding was not needed to prevent further changes.

Hyperlink to location of use: LibraryInvClassAbstract

Interfaces

I created the 'LibraryItem' interface to define the basic behavior of library items like loaning and returning. Both the 'Book' class and other potential library items can implement this interface, ensuring that they follow the same basic structure.

Hyperlink to location of use: LibraryItemClassInterface

Exceptions

To manage errors and exceptional cases, I implemented custom exceptions such as 'InvalidFileException' and 'InvalidInputException'. These exceptions are thrown when there are issues with reading the book file or when the user provides invalid inputs.

Hyperlink to location of use: InvalidFileExceptionClass, InvalidInputExceptionClass

Conclusion

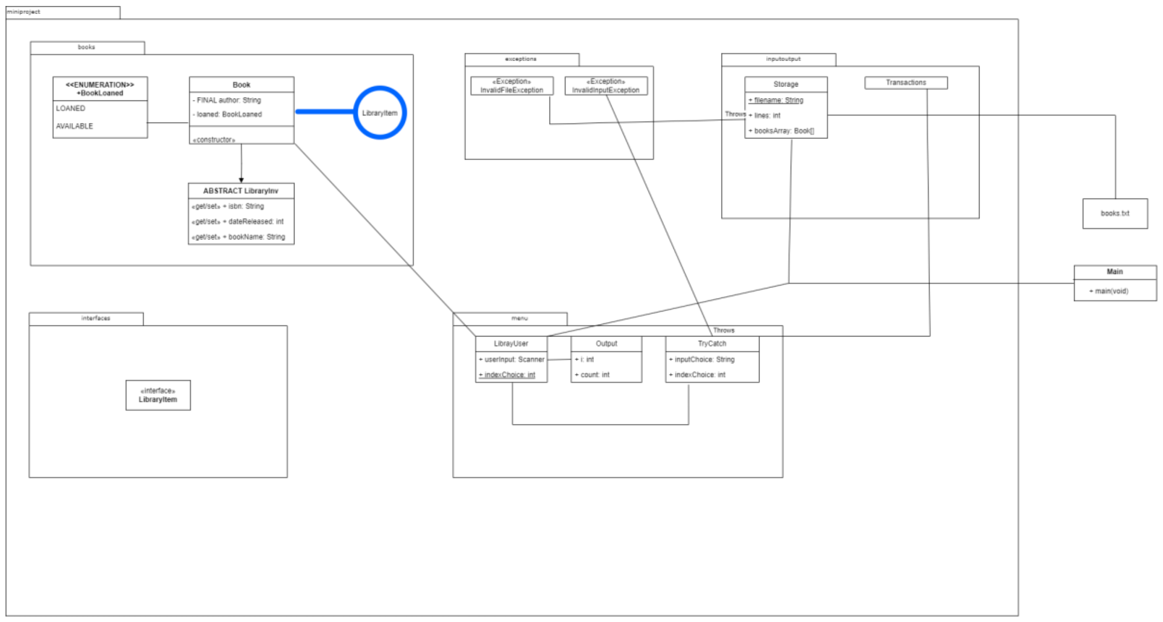
In summary, this project demonstrates my understanding of object-oriented programming concepts such as classes, inheritance, constructors, encapsulation, interfaces, and exceptions. The system allows users to interact with library books by loaning and returning them, with file persistence ensuring that changes are saved. I have successfully applied these concepts in a real-world scenario, providing a functional and organized solution.

File Structure

- miniproject (hyperlink: miniprojectPackage)
  - books (hyperlink: booksPackage)
    - Book.java (hyperlink: BookClass)
    - BookLoaned.java (hyperlink: BookLoanedClassEnum)
    - LibraryInv.java (hyperlink: LibraryInvClassAbstract)
  - exceptions (hyperlink: exceptionsPackage)
    - InvalidFileException.java (hyperlink: InvalidFileExceptionClass)
    - InvalidInputException.java (hyperlink: InvalidInputExceptionClass)
  - inputoutput (hyperlink: inputoutputPackage)
    - Storage.java (hyperlink: StorageClass)
    - Transactions.java
  - interfaces (hyperlink: interfacesPackage)
    - LibraryItem.java (hyperlink: LibraryItemClassInterface)
  - menu (hyperlink: menuPackage)
    - LibraryUser.java (hyperlink: LibraryUserClass)
    - Output.java
    - TryCatch.java

  Main.java

UML


miniproject
books
<<ENUMERATION>> +BookLoaned
LOANED
AVAILABLE
Book
- FINAL author: String
- loaned: BookLoaned
«constructor»
LibraryItem
ABSTRACT LibraryInv
«get/set» + isbn: String
«get/set» + dateReleased: int
«get/set» + bookName: String
exceptions
«Exception» InvalidFileException
«Exception» InvalidInputException
inputoutput
Storage
+ filename: String
+ lines: int
+ booksArray: Book[]
Throws
Transactions
books.txt
Main
+ main(void)
interfaces
«interface» LibraryItem
menu
Throws
LibrayUser
+ userInput: Scanner
+ indexChoice: int
Output
+ i: int
+ count: int
TryCatch
+ inputChoice: String
+ indexChoice: int