Report of book administration system

Yiyang Zuo, Zhenghan Wang

For the project-design requirement of the course SE111 in semester 2302, our group are scheduled to design and develop a book administration system, which can be widely used in library management, bringing such convenience to the staff of the library. This report will mainly concentrate on summarizing the details and development process. The first group member's part is to implement Book, Booklist, Add, Borrow, Delete, Exit, Admin, Normal classes and operation interface. Another one's part is to implement Frame, Find, Return, Show and User classes.

Firstly, from the perspective of the operators of this system, it divides them into normal users and administrators, and they gain different authorities to use and control the system respectively. For normal users, they are allowed to find the specific book they want and decide whether to borrow it or not. Also, they will be required to return the books finally if they have borrowed the book. For administrators, the extra authorities they have compared with the normal users are they can add and delete some books according to their needs and check the details of all books in the inventory, meanwhile delete the "borrow" mode.

Thus, to establish the whole system, we need to implement all the modes mentioned above respectively, and we will use an App interface to run the system specifically. To simplify the report, not to make it redundant, we just classify the implementation of each mode.

The first operation "find". The core algorithm of implementing this operation is to compare every book's name with the book that the user or administrator wants to find. If their names are the same one, which means that you have found the specific book. so we just use a single loop and the "equals" method in class "String" to implement it. After finding it, the system will give the hint that the operator has successfully found the book. Also, we should notice that if the comparison fails after a loop, it means that we cannot find the given book, so we also need to give a hint that "cannot find the book".

The second operation "borrow". The only difference is that we need to mark the book the user wants to borrow with a boolean variable "isborrowed". Only if the book's name in the inventory is the same as the given book and its status is "unborrowed"(the value of "isborrowed" is false), the book can be borrowed.

The third operation “delete”. We need to give a parameter with a concrete object “Book” and a specific position to the public method “setBooks” to implement this operation. Firstly, to implement “setBooks” method, we also use a single loop to judge if we can set a book in the inventory or not, depending on its position. Secondly, based on the premise that the book can be set in the inventory, the core algorithm to implement this operation is to change the position of the rest of the books after deleting a specific book. The remanent books after the book that was deleted should be moved. (The details has been shown in the source code.) More importantly, we need to set the book whose position is the last one before making “delete” operation null.

For the fourth operation “show”, we just need to use a single loop to browse all the books in the inventory and print them out, and it is simple to be implemented.

The fifth operation “return”, it is actually a reversible operation to “borrow”. A single loop is needed also, and we check the status of the specific book we want to return. If it is “borrowed”, we can return this book successfully; otherwise, it fails.

The sixth operation ”add”, it is also a reversible operation to “delete”, but the difference is that we add the book behind of the last book in the inventory. What is noticeable is that the initial index of the book is 0.

In conclusion, this book administration system is a light software and still exists some shortages, however, it is a first step for our students who are freshmen in Software Engineering.