SAAD Final

Chris Hughes

March 26, 24

Contents

1 Initial code review **1**

2 Recommendations **1**

3 Diagrams **2**

4 Implementation **3**

5 Final Thoughts **5**

1 Initial code review

As it stands the code currently doesn't do anything. There are functions for adding, updating and deleting books, but they are not called in the main switch statement. The switch statement runs once, and then simply exist regardless of input. None of the setter methods in the `Book` class have any validation and will accept any string or integer regardless of whether or not it is appropriate. This makes the setter methods no better than simply making the attributes public. The Book constructor has no validation or no logic for creating a book beyond simply taking the attributes as methods. There is currently no testing of any kind. There is virtually no documentation. The main function is in the `BookManager` class. I think it would make more sense to move the switch and main method into a Main class to make the entry point clearer. Names should be split into given, middle and last for better storage in a database.

2 Recommendations

Improve the switch statement

- Call the relevant functions in the switch statement.
- Put the switch statement inside a loop so the user can perform repeated tasks.
- Move the switch and the main function to a Main class.
- This makes the code more modular. For example a "Magazines" class could be instantiated so users could interact with other kinds of media

Book creation and validation

- Create methods for creating and validating books.
- This should be managed in the Book class, possibly though constructor overloading.

- Improve validation in setter methods.
- Add confirmation to delete a book to prevent mistakes
- Remove the public setters.

Documentation

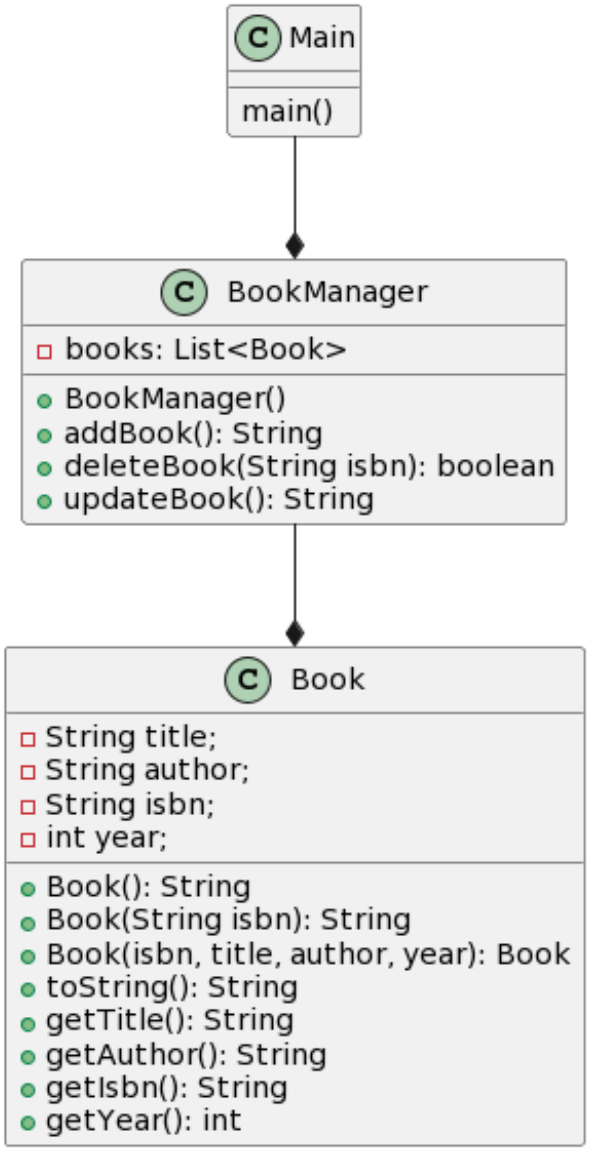
- Write javadoc strings for the classes and methods.
- Generate a javadoc

Testing

- Add jUnit testing to validate books are added correctly

3 Diagrams

Preliminary UML diagram

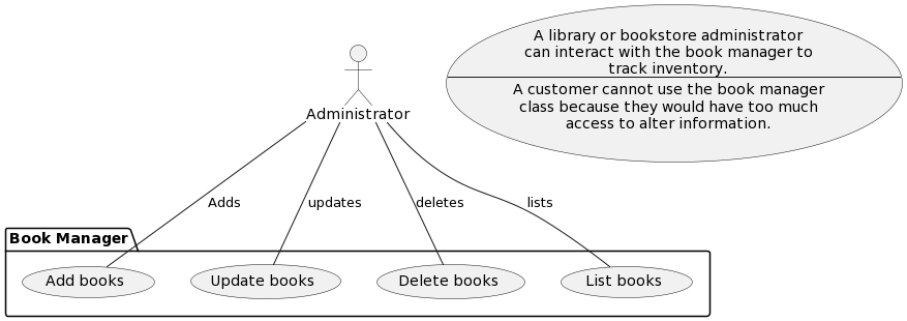


Preliminary Use Case Diagram

User Stories

- An administrator must be able to add books, so that they can be tracked.
- An administrator must be able to delete books, so that out of print books can be handled.
- An administrator must be able to update existing books, so that errors can be corrected.
- An administrator must be able to list books, so that the inventory can be searched and tracked.

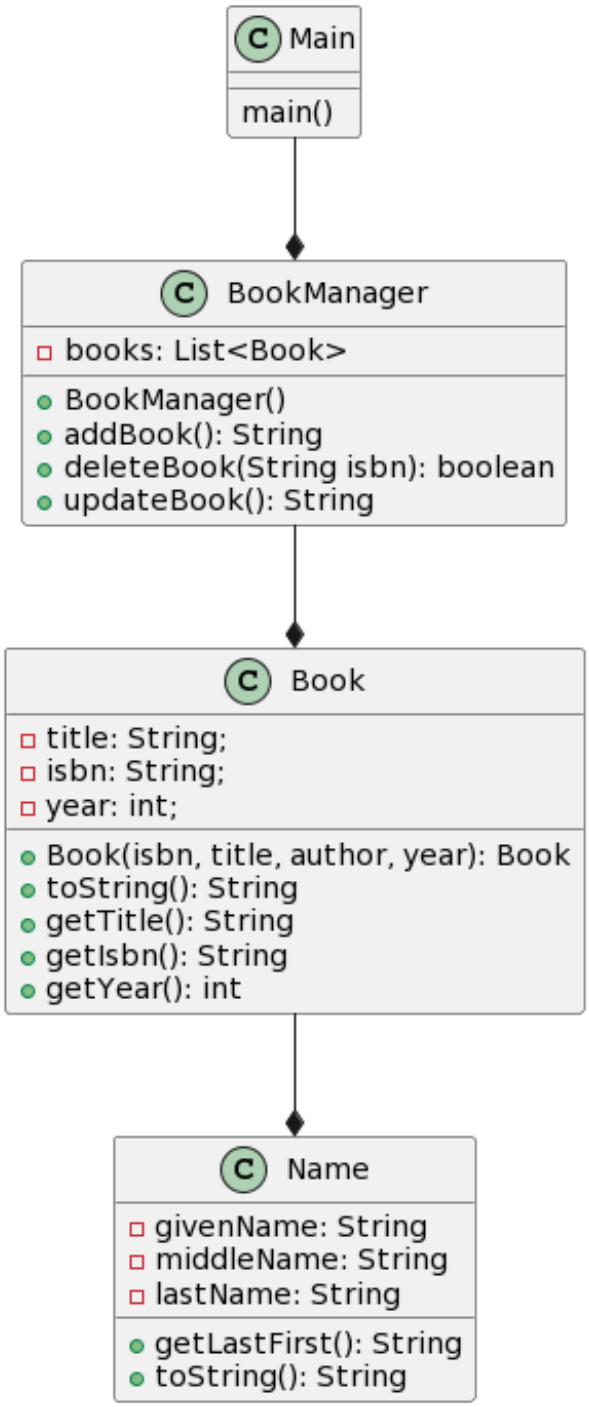
Diagram



4 Implementation

upon starting to code I decided that a name could be more nicely displayed as a separate class using composition. This will store names in a format that is more appropriate to later put into a database and making the change now will mean that if changes need to be made to how names are stored later, only the name class will need to be changed as long as the method names and return types remain the same. Furthermore, since the model would allow for any of the name fields to be blank, it handles the case where all names are blank by throwing an exception. Furthermore, I decided to scrap the idea of method overloading the Book constructor in place of validating data within the `BookManager` class. While it makes sense for the logic needed to validate a book should be within the `Book` class, the actual implementation is overly complex.

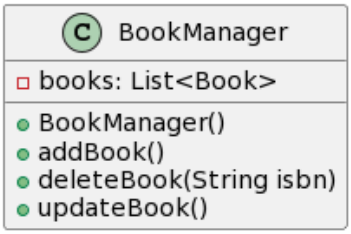
I have updated the UML diagram to reflect this change:



Return value changes

After trying to implement the return values of the add/delete/update methods, I decided that it made more sense to print the results in the same place that the validation was done. Doing it the other way would make it `BookManager` more modular perhaps, but it would be an easy change to make later.

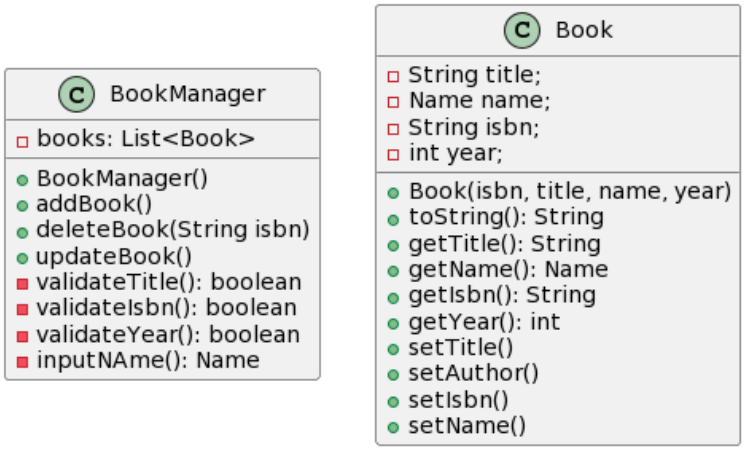
I updated the class diagram to reflect this change:



Updating books

I decided in the end to put the setter functions back into `Book` class and then and then validate them within the `BookManager`. The setter functions don't do anything, but leaving them in future proofs the application in case there was a need to do further internal processing of the attributes within the book class.

The updated `BookManager` and `book` class are as follows:



Testing

One issue that I discovered with the app is that, after trying to do tests with Junit, its very hard to run automated tests on the application because so much of it is directly reliant on user input. Perhaps this something to take into further consideration when creating apps in the future. I could make I f

5 Final Thoughts

The code works as expected right now, and it should be ready to be added to. Mainly the main interface could import a magazine or movie manager that could be added to the main loop and would not effect the implementation of the book manager. As it stands, I think the major issue with the application revolves around testing. I would like to be able to test the validation methods, but I can't currently because they are private methods. I also can't test any of the input methods because they rely on private methods. I think the solution would have to be to move all of the user input methods to a `Input` class and then test from there by calling public `BookManager` methods from there.