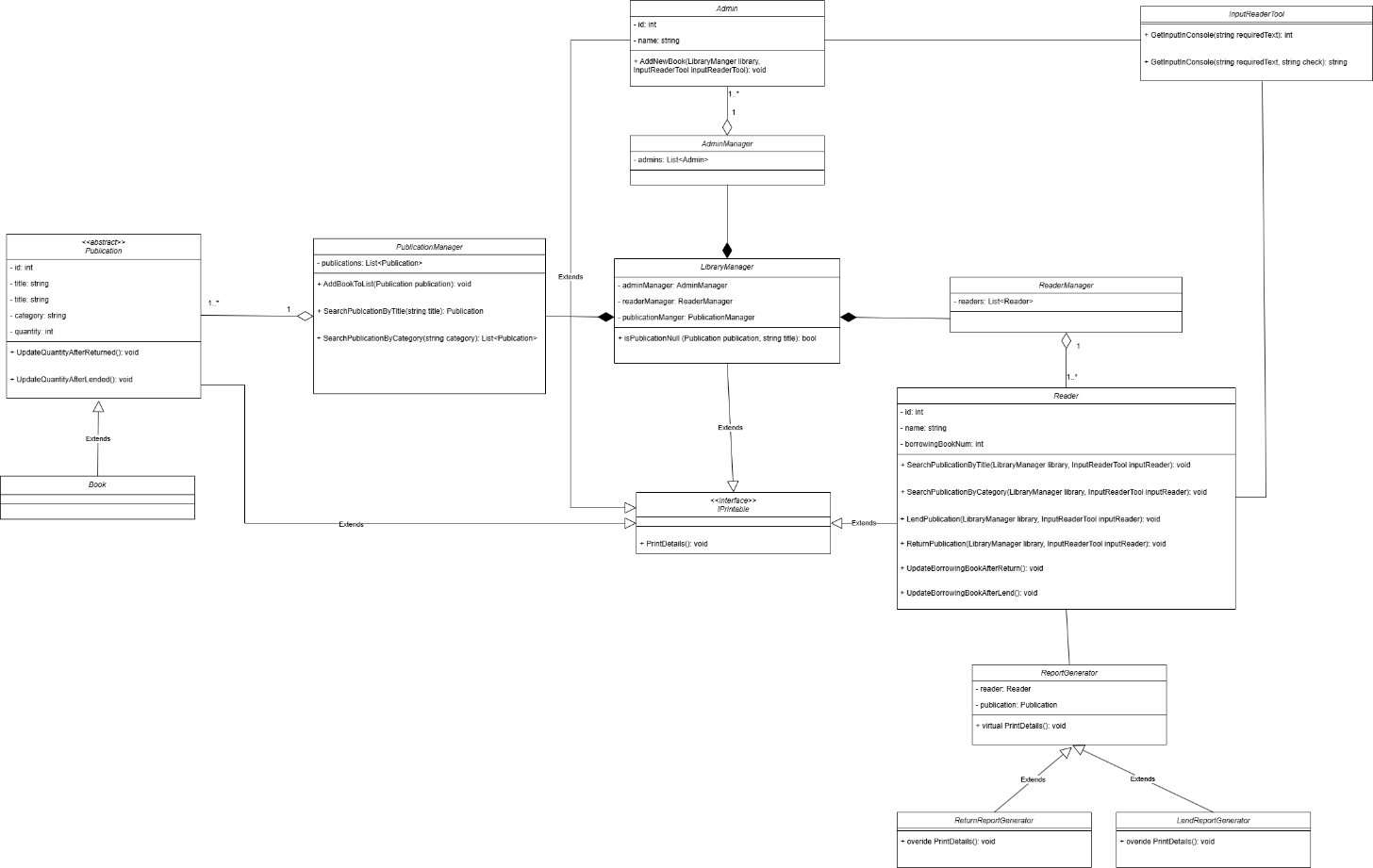
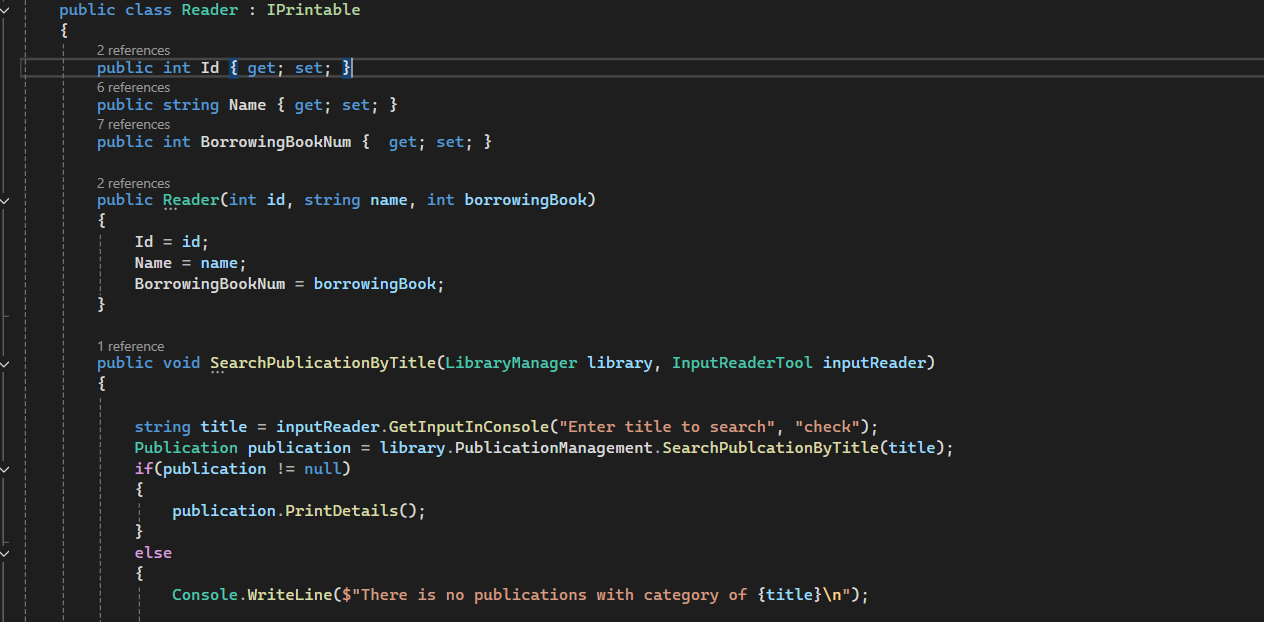
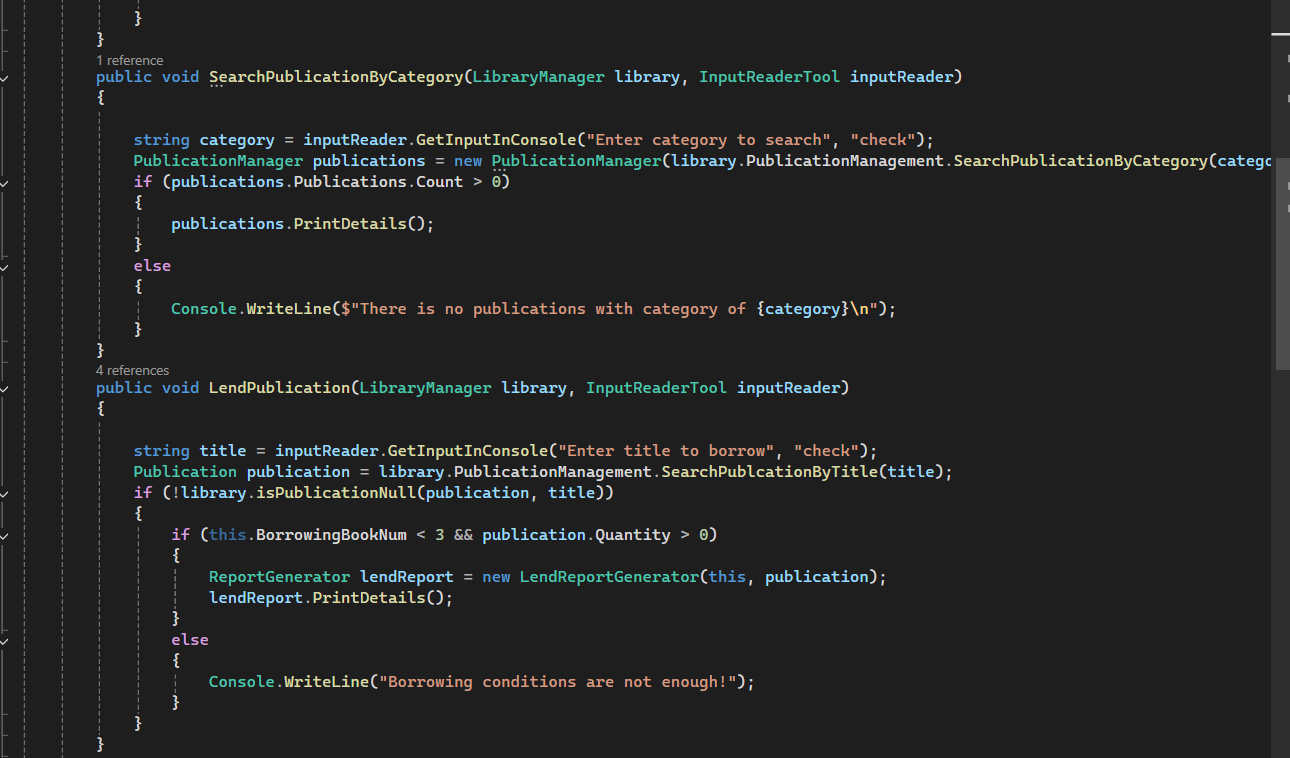
Lab 4 – Report – SOLID in Library Management System

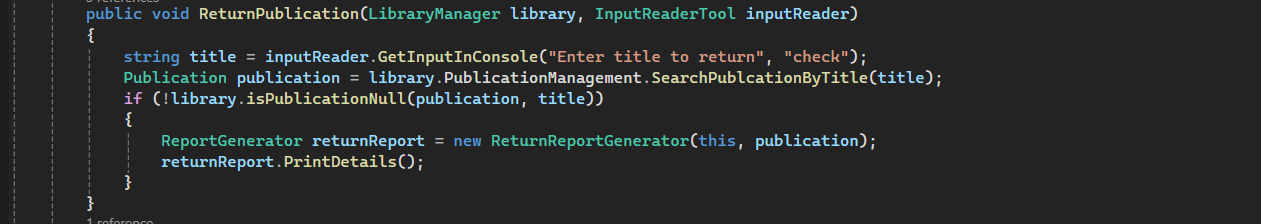
1. Class Diagram.



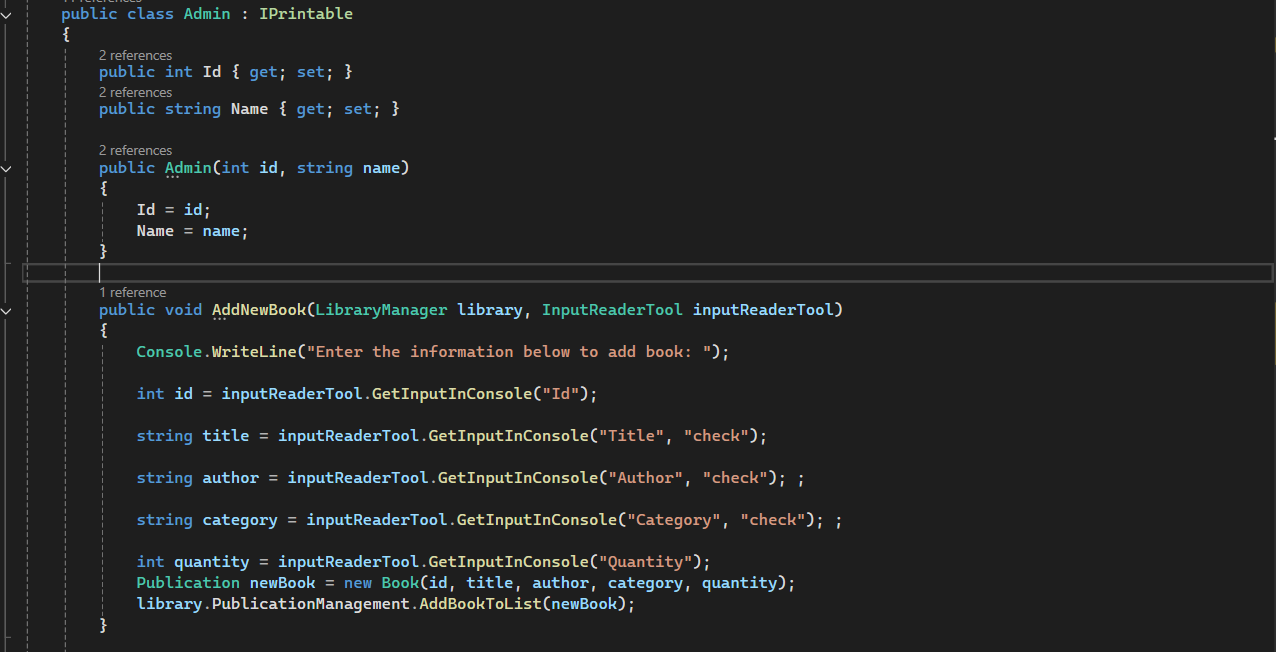
1. Explain SOLID.
2. Single Responsibility Principle: Each object has vital functionalities that it should have.







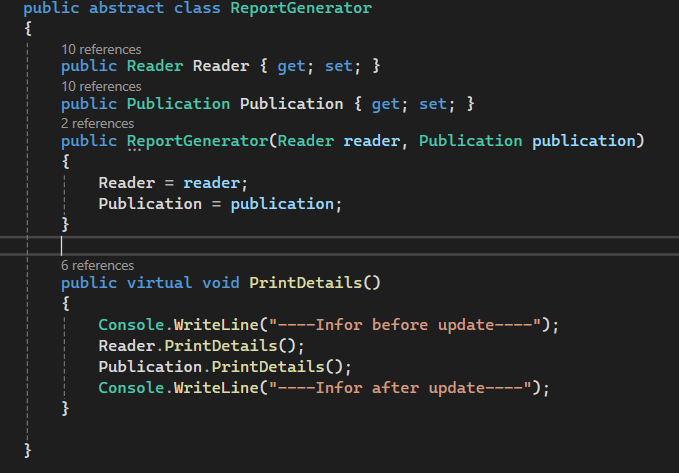
Reader has simple functionalities such as: search for a book, lend books and return book.



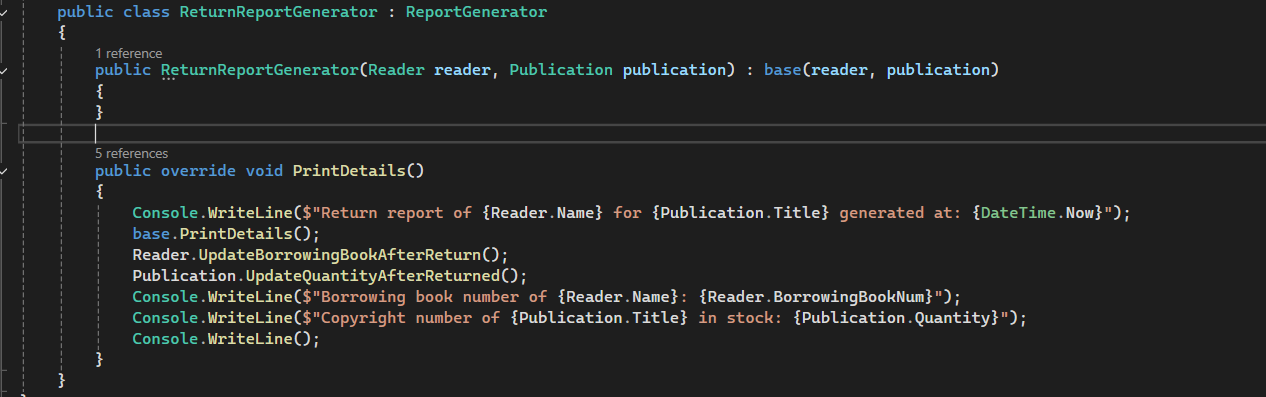
Admin object has simple functionalities such as: add new books into the library.

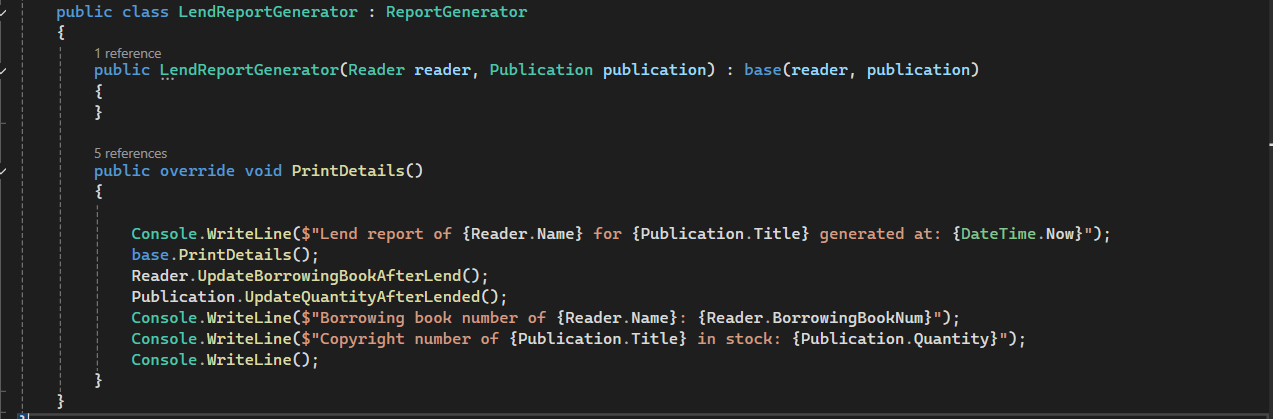
1. Open/Closed Principle: we can extend a class’s behavior without modifying it.

I create a abstract class for ReportGenerator so that, we can generate different types of reports later.



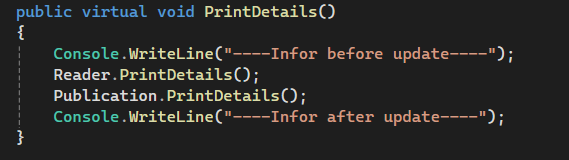
This ReturnReportGenerator extends the ReportGenerator to so information when readers return books:

 This LendReportGenerator extends the ReportGenerator to so information when readers lend books:

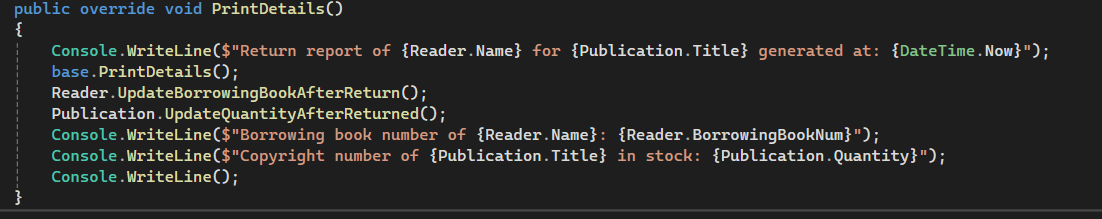


1. Liskov Substitution Principle: my subclass can utilize the superclass without causing problems

Superclass:

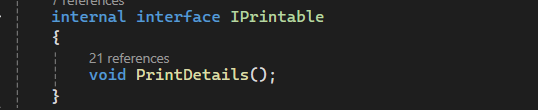


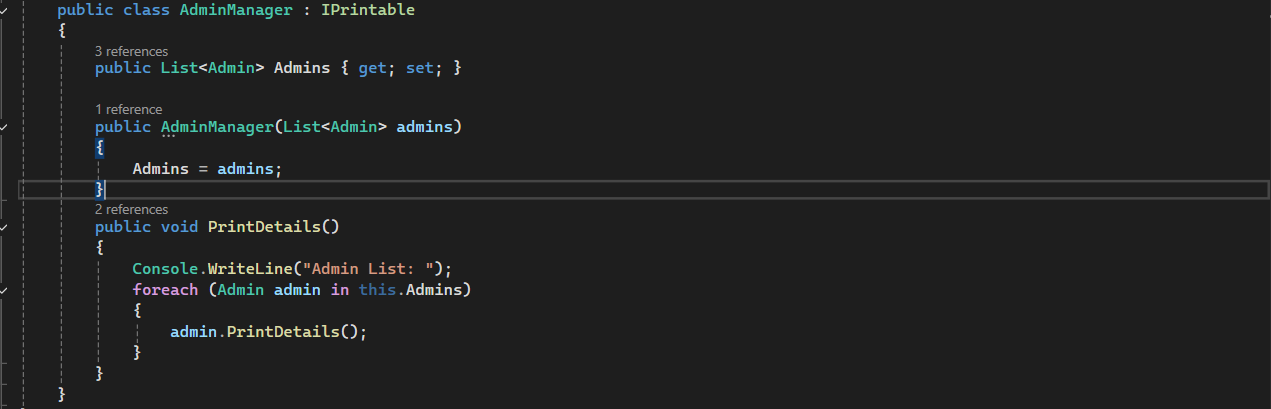
Subclass:



I utilize the superclass to minimum coding lines the subclass instead of changing the superclass to serve my subclass.

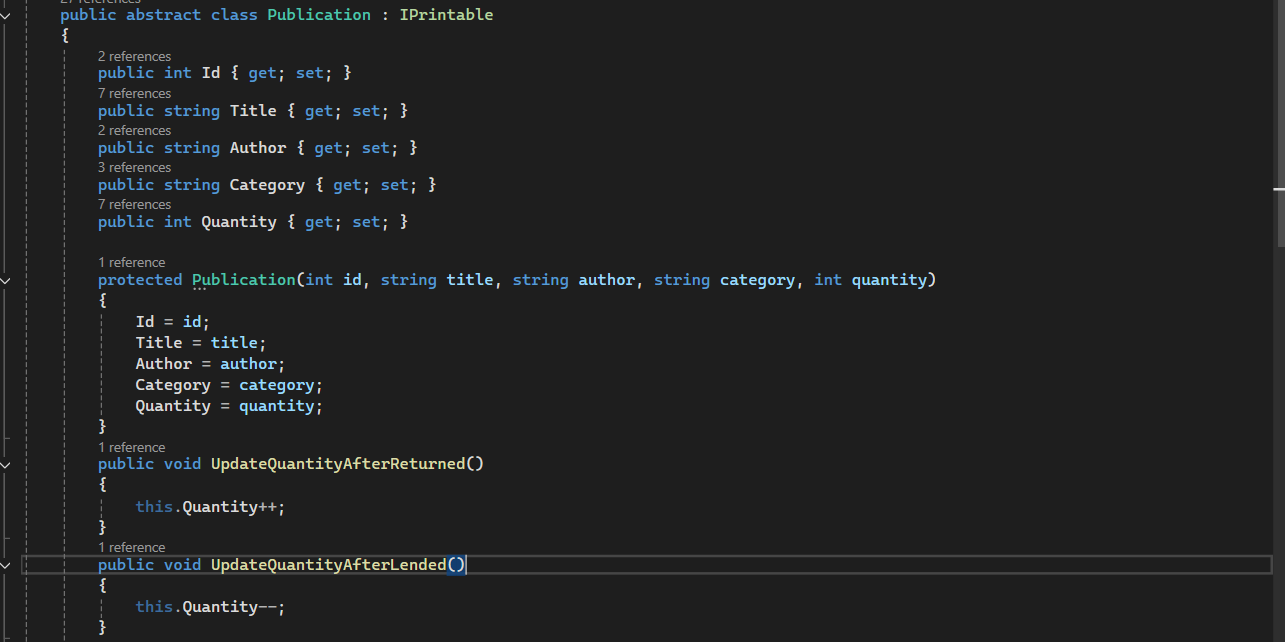
1. Interface Segregation Principle: I generate Interface to serve my implemented class, so that it do not create any redundant factors in my subclass:

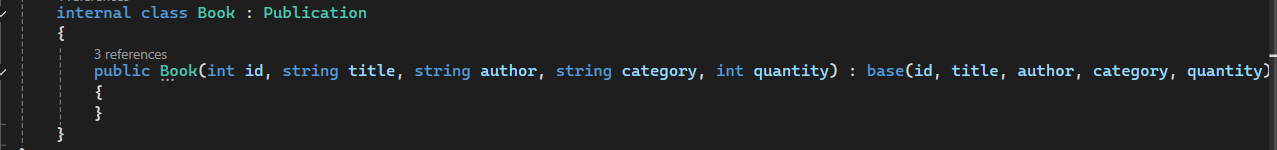




1. Dependency Inversion Principle:

High-level modules should not depend on low-level modules. Both should depend on abstractions. Abstractions should not depend on details. Details should depend on abstractions.

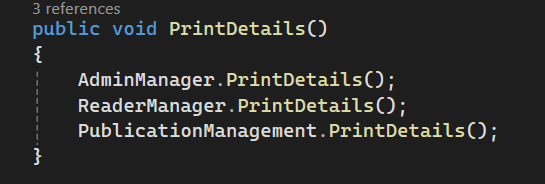


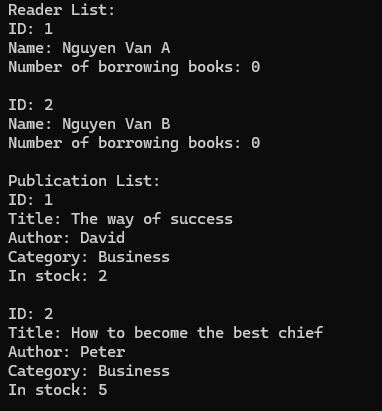


By doing this I can generate different types of publication later.

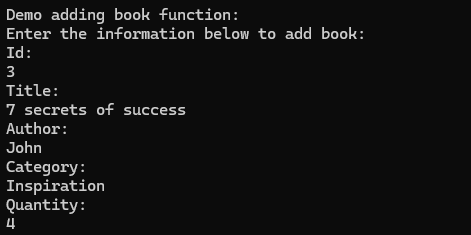
1. Demo functionalities.

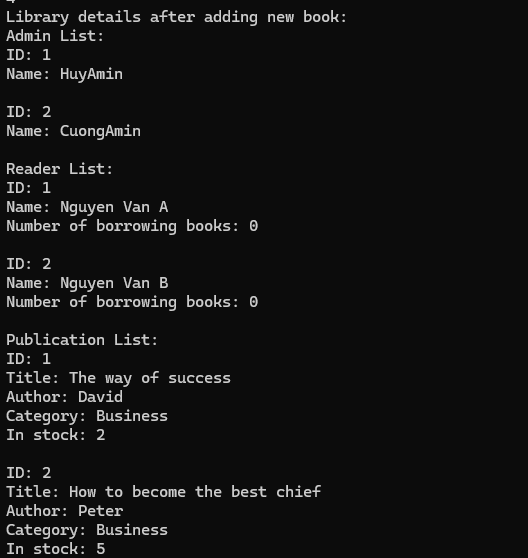
Library can show all of fields:

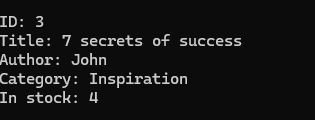




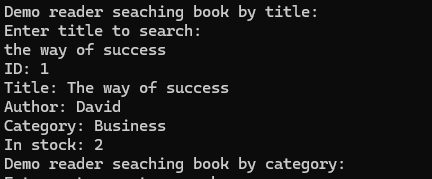
Admin can add new book into the system:



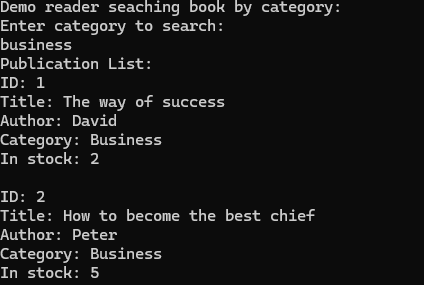




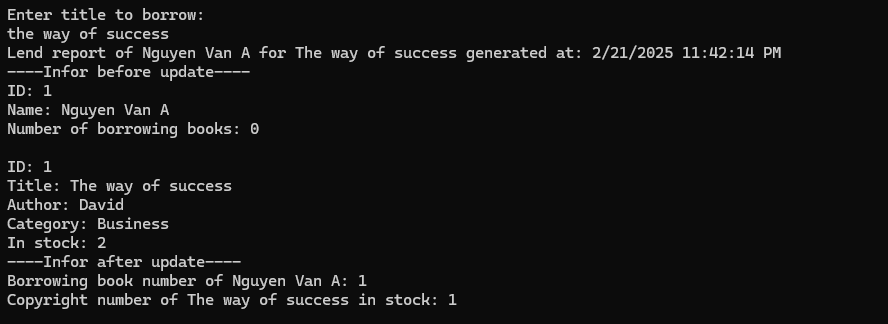
Reader can search book by title:



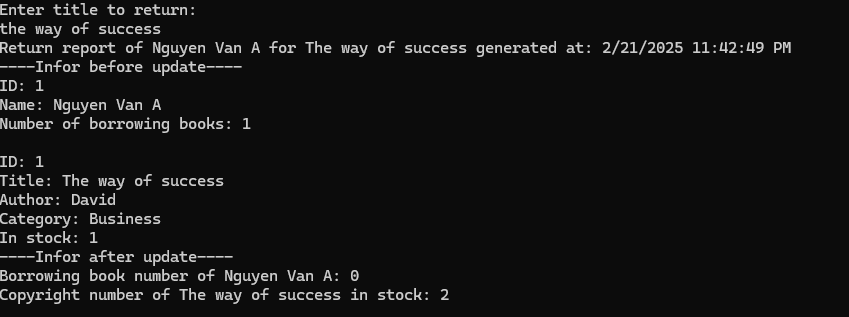
Reader can search book by category:



Reader can borrow book and the report will be generated:



Reader can return book and the report will be generated:

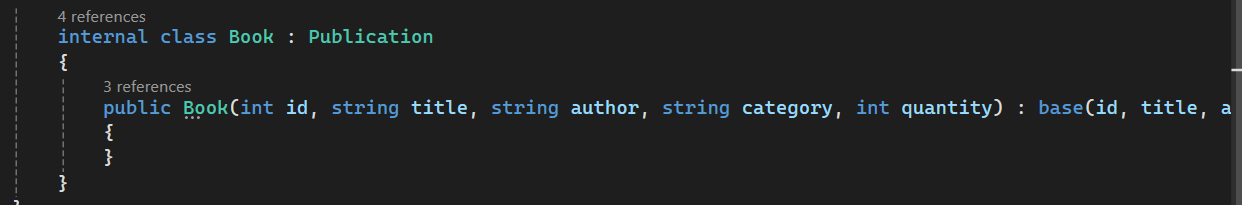


1. Propose ways to extend the system.

By creating abstract publication class:



We can create different types of books in the future without any problems such as Book, eBook, Magazine, Newspaper,…:



We also can apply this system to manage many libraries later.

We can generate different types of reports when needed because we can extends the abstract class ReportGenerator:

