ISSN No. 0976-5697

Volume 8, No. 3, March – April 2017

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

Improving Design of Library Management System using Design Patterns

Pankhuri Jain, Sourav Shaw, and Manjari Gupta*
Department of Computer Science, Institute of Science,
Banaras Hindu University, Varanasi, India

Abstract: Designing a system is perhaps the most critical factor affecting the quality of the software. No design methodology reduces the process of design to a sequence of mechanical steps. It only provides guidelines to aid the designer during the design process. Fortunately, GOF [2] and other researchers proposed many design patterns that give us solution of general design problems. Design decisions of experienced designers are recorded in form of design patterns. Thus each design pattern focuses on a particular object-oriented design problem or issue. If these design patterns are used by a novice designer, the obtained system design would be much better. In this paper, we designed a very simple library management system using object oriented approach without design patterns. All the defects are shown and this design is improved using few design patterns. We used three design patterns: Factory, Facade and Template Design Patterns in this study.

Keywords: System Design, Design problems, Design Patterns, Façade, Factory, Template

I. INTRODUCTION

Demand of software has been increasing day by day. Therefore, there is more demand for software development paradigm that improves quality and productivity of software development. A **Design pattern** [1] is a general repeatable solution to a commonly occurring problem in software design. A design pattern isn't a finished design that can be transformed directly into code. It is a description or template for how to solve a problem that can be used in many different situations. They guarantee the creation of transparent structures which allow software to be easily understood, extended, and reused. The description of design patterns [2] provides information about the structure, the participant's roles, the interaction between participants and, above all, the intent for which they should be used. In this paper, we will be implementing design patterns in library management system to improve its design that is obtained without using design patterns. We choose the Library management system as a domain because of its simplicity. Further we designed the system for few requirements to show the objective of use of design patterns in design for improvement.

In the next section, we will explain the system under study- Library Management System, its design without using design patterns and defects in this design. In section 3, we will describe three design patterns: Factory, Facade and template design patterns that are used in this work to improve the design. How the design proposed in section 2 is improved using these three design patterns are shown in section 4. Lastly we conclude in section 5.

II. LIBRARY MANAGEMENT SYSTEM

Problem Statement and Analysis: We have taken a very simplified version of library management system that manages the catalog of a library. It performs functions such as managing book transaction and creating user. It also include function that enable users to search their resources in library. To analyse and understand the system under study we will develop a model using object-oriented approach.

"Fig. 1" shows the static structure of the program to be developed using this design. Whenever MemberRecord class requests for a book, Librarian class instantiates an instance of Book class, issuing it to the requesting MemberRecord. Also, MemberRecord pays the Bill created by Librarian.

The system design shown in "Fig. 1" is quite complex and unreadable. Class diagram of Library management system creates a lot of duplication as multiple instance of Book class are instantiated with same property and different attributes such as Textbook, Journal and Magazine. Further, Librarian need to know details of Book class, hence there is tight coupling between these two classes creating inflexibility. Also, MemberRecord is exposed to complex details of the system further reducing modifiability. Hence, design pattern would be utmost solution for handling arrangement of system code, but it would also make code reusable. So, we would use design patterns to make it flexible and more understandable. We identified the scope of use of three design patterns described in the next section. In the book class multiple classes such as magazine, journal and textbook are required to be created thus it would be better to use here factory design pattern. We can use facade design patterns to reduce the exposed complexity to MemberRecord class. Further to differentiate different types of members like students, teachers, non teaching staffs we can use template design pattern. In the next section these three patterns are described in little detail.

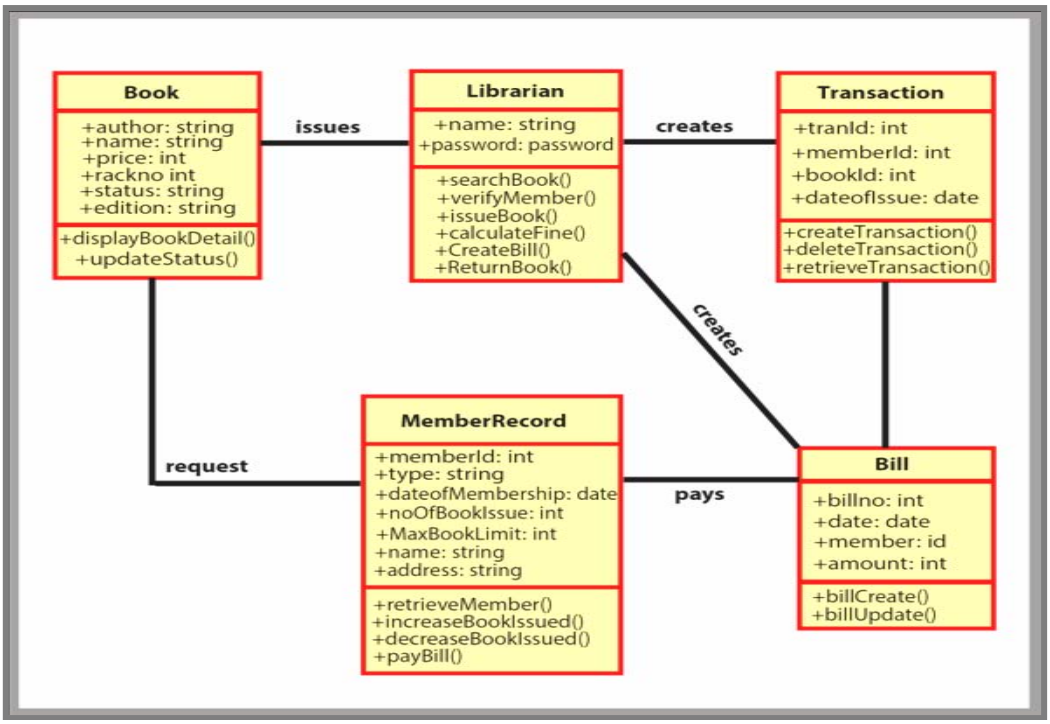

Fig.1 Class diagram of Library Management System

III. DESIGN PATTERNS: FACTORY, FACADE AND TEMPLATE

The factory pattern is one of the patterns that is frequently used in Java programming. It is categorized as creational pattern i.e the pattern involving in creating objects. The factory pattern creates object without exposing the creation logic to the user and it will also refer to the new created object using interface [5]. The factory pattern is used in the book class because multiple classes such as magazine, journal and textbook need to be created and also it prevents the code in the subclasses to be written everywhere in the different classes repeatedly. UML diagram of factory pattern is shown in "Fig. 2".

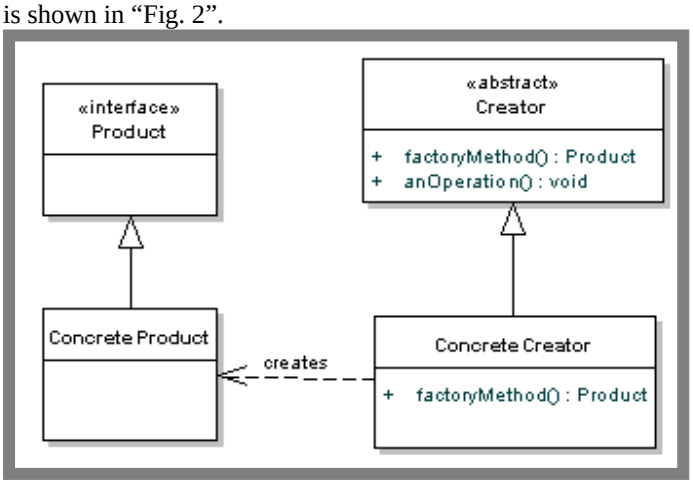

Fig.2 Factory Design Pattern [3]

Facade is one of the easiest patterns to implement. Typically this pattern is used to either hide the implementation of the underlying classes it is presenting an interface for, or to simplify the underlying implementation of something that may be complex. A facade may present a simple interface to the outside world, but under the hood do things like create instances of books, manage transactions -- all stuff that you can be shielded from by the simplified interface [6]. UML diagram of Façade design pattern is shown in "Fig. 3".

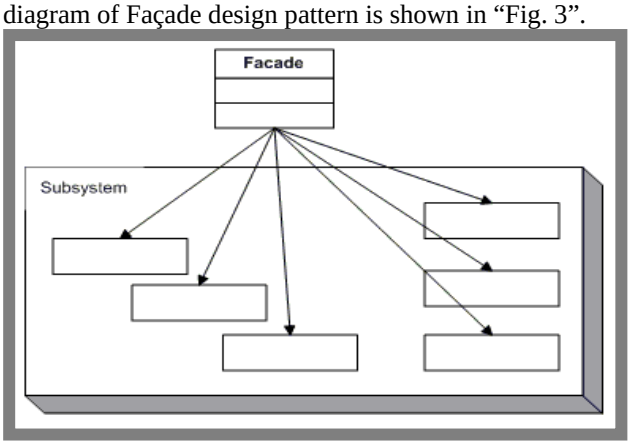

Fig.3 Façade Design Pattern [4]

Template pattern is a behavioural pattern. It defines skeleton of an algorithm in an operation, and defer some steps to subclasses. Template pattern defines a main MemberRecord template and this main MemberRecord template has sub classes. UML diagram of Template design pattern is shown in "Fig. 4".

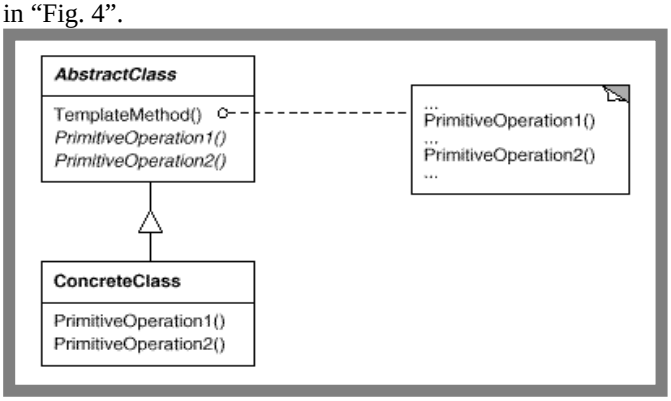

Fig.4 Template Design Pattern [2]

IV. IMPROVING THE EARLIER PROPOSED DESIGN USING DESIGN PATTERNS

First, the book class is modified to become an abstract class and a set and get method is written in the class for every instance declared. A function named displayBookDetail() is declared in the book class as well for the factory class to access. Then, other subclasses such as magazine, journal and textbook class is inheriting to it. Next, a factory class named "BookFactory" is created and a method named retrieveBook with a String parameter is declared. Therefore, whenever the librarian class need to obtain the information from the book class, it needs to request from the factory class first. The "Fig. 5" shown refined class diagram.

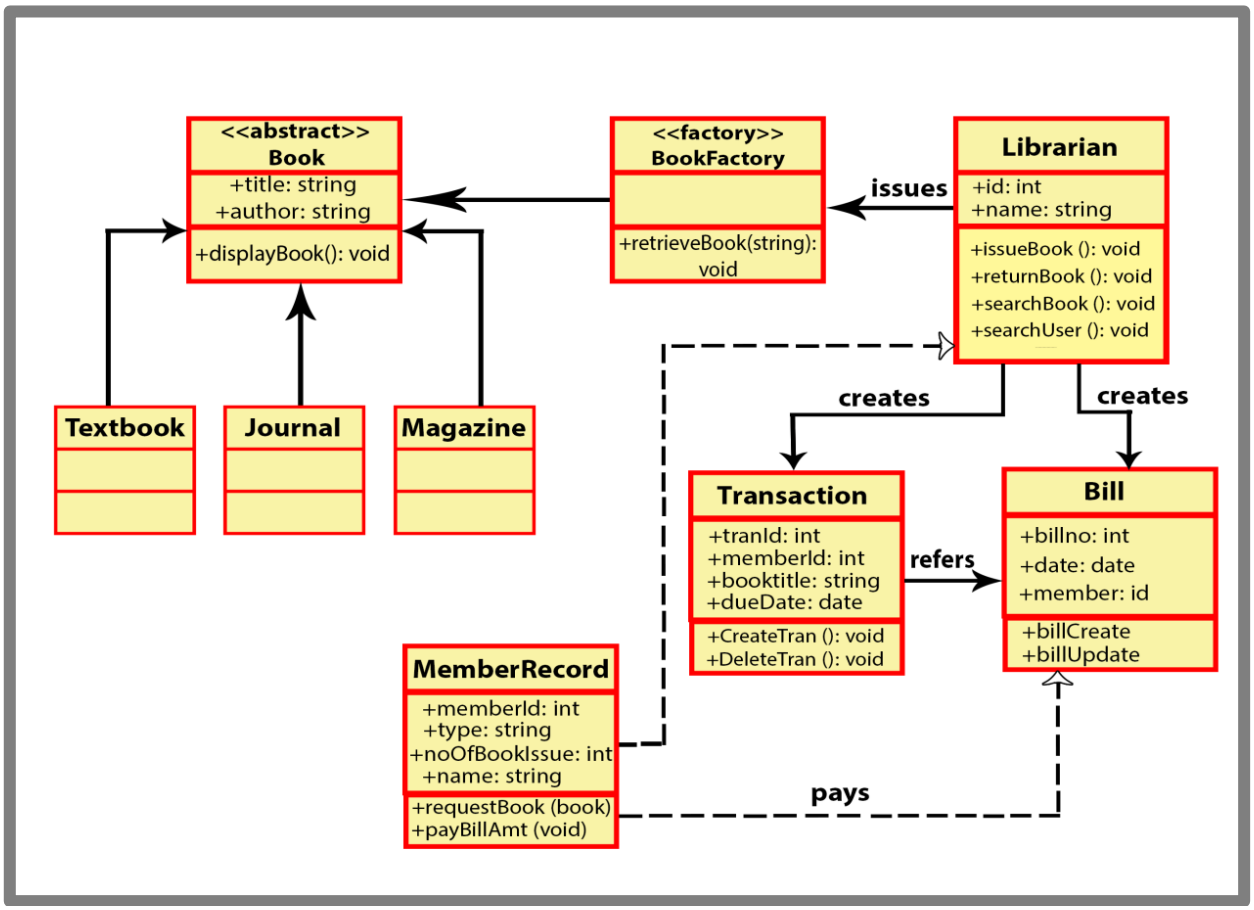

Fig.5 Modified class diagram with Factory Pattern

First, the LibraryFacade class is created which is related to all other classes. A function named requestBook() with String parameter and payBill() are declared in the LibraryFacade class. Therefore, whenever the MemberRecord class needs to request book, it would request from the LibraryFacade class first. LibraryFacade handles rest of complexity viz requesting the librarian, referring the transaction and paying the bill. The "Fig. 6" shown refined class diagram.

First, the MemberRecord class is modified to become an abstract class and a set and get method is written in the class for every instance declared. Then, other subclasses such as student and faculty class inherits from MemberRecord. It is used here to basically isolate different members based on their type (student, faculty). The "Fig. 7" shown refined class diagram.

The Factory Design Pattern encapsulate the object creation process and also prevents the code in the subclasses to be written everywhere in the different classes repeatedly. It makes object creation process reusable for the entire application. It hides the implementation detail of Book from librarian. Factory Design Pattern is one of the most frequently used design pattern across multiple technologies and framework. It also avoids errors introduced by redundancy. Using Factory Design Pattern allows one to return one factory object among a family of factory objects. Therefore, it is a detailed design pattern and more efficient as well as easily understood by anyone and mostly used in OOPS concepts for implementation.

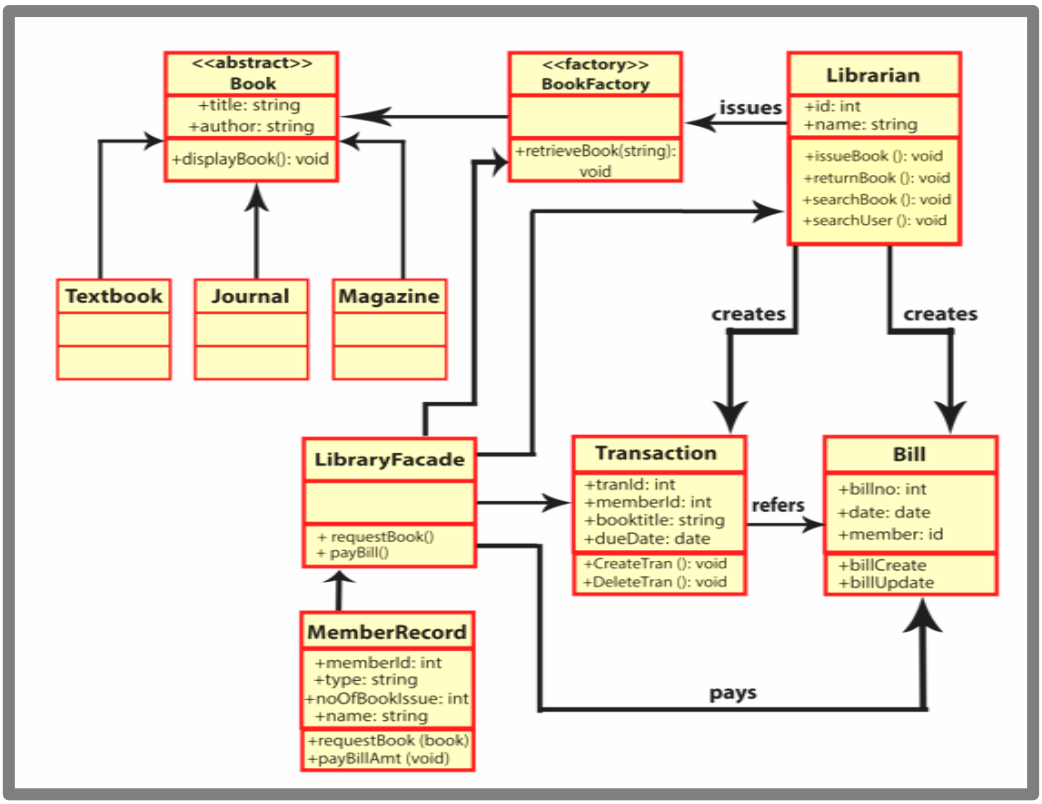

Fig.6 Modified class diagram with Facade Pattern

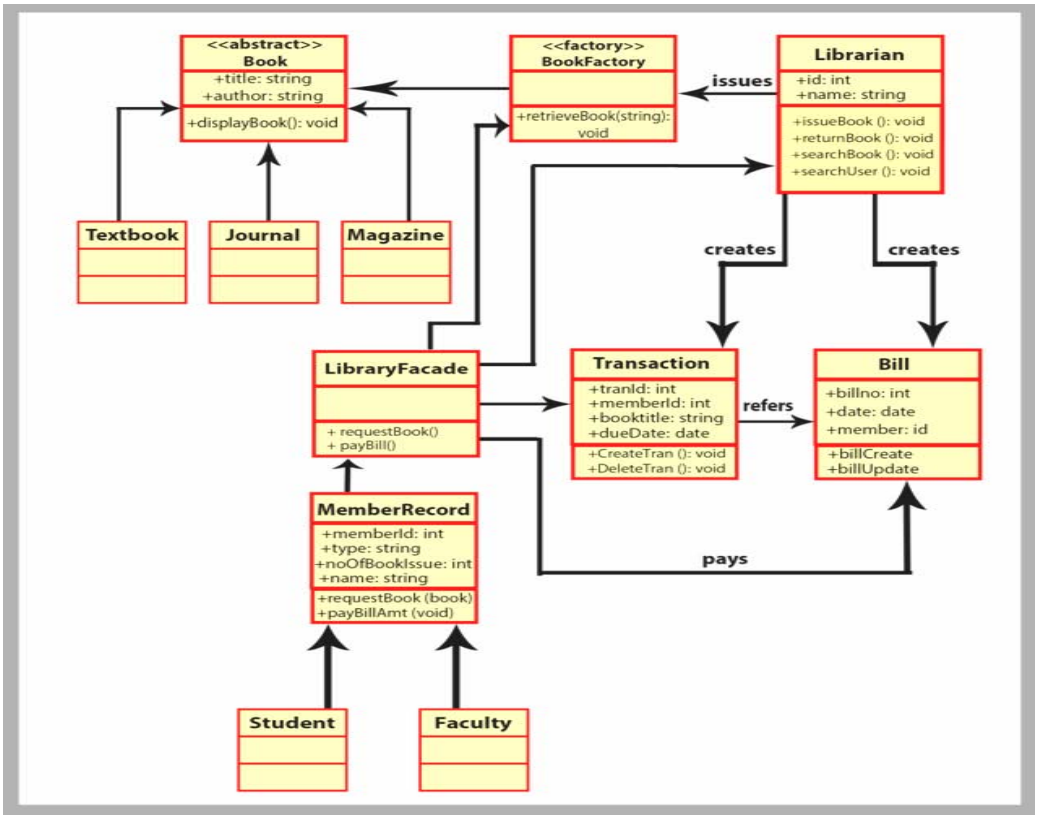

Fig.7 Modified class diagram with Template Pattern

V. CONCLUSION

In this paper, we have proposed design of library management system with and without design patterns. We have shown few problems in design proposed without using design patterns. This design is then improved using three design patterns: factory, facade and template design patterns. To improve the design firstly we added factory pattern to delegate responsibility of issuing book to BookFactory. Then, facade pattern is implemented to hide the complexities of entire system and provide interface to user, using which the user can access the system. Finally, template pattern is applied to isolate different Member Record, which have same property but different attributes.

VI. REFERENCES

[1] Pfleeger, S.L.: Software Engineering: Theory and Practice, 2nd edn. Prentice Hall, Englewood Cliffs (2001)

[2] Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns: elements of reusable object-oriented software. Addison Wesley, Reading (1994)

[3] http://www.dofactory.com . (2017). Available at: http://www.dofactory.com/net/design-patterns

[4] https://www.tutorialspoint.com. (2017). Available at: https://www.tutorialspoint.com/design_pattern/facade_pattern

[5] www.tutorialspoint.com. (2017). Design Pattern Factory Pattern. [online] Available at: https://www.tutorialspoint.com/design_pattern/factory_pattern.htm

[6] www.stackoverflow.com. (2017) Design Pattern Facade Pattern. [online] Available at: http://stackoverflow.com/questions/4798184/what-is-the-point-of-a-facade-in-java-ee