COE528 (W2025)

Book Store Project Lab Report

Group #39

Section #07

Jaswant Hemanth Kumar Rekha

Musheer Siddiqui

Mihir Patel

Dr. Olivia Das

Deadline: March 30th, 2025 (11:59 PM)

Introduction:

The objective of this coding project is to record the creation and implementation of a bookstore application using the Java Memory Model. This assignment is separated into 3 different phases to accurately and efficiently introduce a graphical user interface (GUI) using JavaFX, as well as a backend application utilizing object-oriented fundamentals. This lab report will provide a thorough understanding of how the bookstore application is implemented, with brief descriptions. Next, the report describes the importance and meaning of the use-case diagram that represents the functionality of the system. Finally, the report will illustrate the rationale behind the use of state design patterns to control various cases in the program, for example, handling different customer membership statuses.

Class Descriptions:

- **User**: This class is an abstract class containing common features and methods for the Customer and Owner classes. (Username & Password, Login & Logout, respectively)

- **Customer:** This class extends from User and exists to represent registered customers. The class contains functions for buying books together with point redemption capabilities. The design implements the State Pattern by transferring membership update operations to an individual state object.

- **Owner:** This class extends from the owner and represents a registered owner. The owner can add and remove books from the bookstore, as well as add and remove customers from the customer list. The class also includes the ability to comprehend (read and write) a customer file, which manages the current customer information in the bookstore.

- **BookStore:** This class helps in managing the inventory of available books. All methods for adding and removing books exist alongside book purchasing capabilities within this class. Also, file handling methods are in the class for saving the book list.
- **Book:** This class is to represent a book with features like a title and price
- **CustomerStatus (abstract):** The class outlines state-specific operations through abstract methods for customer membership promotions and demotions. The application implements the State Design Pattern through this abstract method.
- **Silver:** A concrete implementation of CustomerStatus. The state defines the behavior of Silver membership and contains the rules which will trigger a Silver customer's upgrade to Gold status upon reaching a certain number of points.
- **Gold:** The CustomerStatus class implements the Gold membership state through this particular concrete form. This class helps with demoting and keeping the customer at the right membership level based on what level the user is on.
- **GUI:** The application class, based on JavaFX, controls the whole user interface execution. Through the GUI class system, admins can conduct the login sequence while administering different owner and customer interfaces, which continuously conduct transactions from a single user interface screen.

Use Case Description:

The use case diagram is an important part of this project as it describes how a system functions from the actors' perspectives, all within the code. This process assists with defining what the program does, how it interacts with the user, and in what state the code functions. Use

cases are essential in the early stages of developing a project as they help to ensure the needs of the client are met. To understand the context of the bookstore application, the use-case description will summarize the behaviour of this application.

Name: BookStore Application

Actors:

- Customer
- Owner

Customer

Entry Conditions:

- An active account is linked to the customer.
- The customer must be logged into the bookstore application with the correct credentials.

Exit Conditions:

- The purchased book is removed from the booklist.
- The customer status and points are updated.

Flow of Events:

1. **Login:** Customer should enter the correct username and password for the system to validate the credentials and send the user to the customer start screen.
2. **Books Available:** The screen will show all the books available for purchase.
3. **Choice:** To buy a book, the customer may check one or more boxes from the available books table.

4. **Start Purchase:** Customer uses the “Buy” or “Redeem and Buy” button to validate if at least one or more boxes was checked. If true, progress to the next step.
5. **Transaction Confirmation:** The screen will show the charges due by the customer, their updated points and status.
6. **Completion:** Customer logs out of the account and is returned to the login screen.

Exceptions:

- At least one book must be selected from the available books table before proceeding to the ‘start purchase’ step.

Special Requirements:

- The entire process must be done in the single-window GUI.
- Points and membership status will be updated after every transaction.
- It is assumed there is one copy of each book.

Owner

Entry Conditions:

- The owner must be logged into the bookstore application with the correct credentials.

Exit Conditions:

- Available books and customer tables may be updated.

Flow of Events:

1. **Login:** The Owner should enter the correct username and password for the system to validate the credentials and send the user to the Owner's start screen.

2. **Start Screen Options:** The owner may choose to direct the customer to the information table or the available books table.
    a. **Customer Information Table:** The owner can add a customer (with a username and password) or use the checkboxes to delete customers. The back button takes the owner back to their start screen.
    b. **Book Availability Table:** The owner can add a book (with title and price), or use checkboxes to delete books. The back button takes the owner back to their start screen.
3. **Logout:** Customer logs out of the account and is returned to the login screen.

Exceptions:

- At least one customer must be selected to delete when using the delete button on the owner customer screen.
- At least one book must be selected to delete when using the delete button on the owner book screen.
- No identical book can be entered into the available book list.
- No identical customer can be entered into the available customer list.
- Both the price and title fields must be filled for each new book entry.
- Both the username and password fields must be filled for each new customer entry.

Special Requirements:

- The entire process must be done in the single-window GUI.
- Customer and book lists must be updated after every addition/removal.

Rationale Behind Using the State Design Pattern:

In general, the purpose of a state design pattern is to change the behaviour of a class based on a certain condition. In this project, the state design pattern is used to control a customer's membership status.

- **Encapsulation of State-Specific Behaviour:** This process will help with changing the membership status between Silver and Gold based on the total number of points a customer has. Although it is possible to update the status of a customer directly from the customer class, the use of a state design avoids redundant lines of code.
- **Improved Maintenance:** The pattern localizes state-dependent behaviours, providing an easier way to update the status transitions without impacting other parts of the design.
- **Scalability:** If new membership levels or rules are presented in the future, the new membership level will be easier to implement using the current system design.
- **Comprehension:** By separating the customer status from the rest of the program, the overall design becomes easier to understand, including testing and debugging.

Conclusion:

In summary, our JMM BookStore application successfully incorporates object-oriented design applications with a single-window JavaFX GUI. The project's design was manifested by UML use-case modeling, a class diagram, and the use of the State Design Pattern. The class breakdown and user case descriptions for each actor illustrate how the system meets its functional needs. Overall, the project illustrates a strong foundation and application of software development in Java and provides a good foundation for full-stack development.