Lab 4 Library Management System Report

Group members: Shondel Hayles, Simon Velez, Daniel Cuevas

Date: November 11, 2025

Course: 236 OO Systems A & D

The library management system shows how several GRASP principles work well in their overall design. This report explains how GRASP principles were applied in designing the library management system for this lab assignment.

1. Information Expert

Applied in: Book and Member classes

The Information Expert principle puts responsibility with the class that already holds the information needed. This principle was applied in the Book and Member classes because they each have the information needed to handle their own responsibilities.

For the Book class, it was made responsible for tracking whether it's available or borrowed. Since the Book object knows its own isAvailable status, it made sense to give it the borrowBook() and returnBook() methods. This way, the book manages its own state instead of having some other class change it directly.

Similarly, the Member class keeps track of which books a member has borrowed in the borrowedBooks ArrayList. Because the member has this information, it was given the responsibility to handle borrowing and returning books from the member's perspective. It can add books to its list when borrowing and remove them when returning.

2. Creator

Applied in: Library class

The Creator principle assigns the responsibility of creating instances to classes that contain, aggregate, or have the initializing data for those instances. This principle was used in the Library class because it's the most logical place to create Book and Member objects.

The Library class has the addBook() and registerMember() methods that create new instances. This makes sense because:

- The library contains the collections of books and members
- It receives the initial information needed to create them (like titles, authors, and member IDs)
- It's responsible for managing these objects throughout the program

Having the library create these objects keeps the design organized and centralized.

3. Controller

Applied in: Library and LibrarianController classes

The Controller principle assigns handling of system events to a class that is not part of the user interface.

The LibrarianController acts as the main controller for user operations like borrowing and returning books. When someone wants to borrow a book, the controller handles the request, finds the right member and book from the library, and then tells the member to borrow the book. This separates the coordination logic from the actual business logic.

The Library class also acts as a controller for managing the catalog and members. It handles operations like adding books, registering members, and searching for them. It coordinates these system-level tasks without getting into the details of how individual books or members work.

4. Low Coupling

Applied throughout the design

Low Coupling minimizes dependencies between classes to reduce the impact of changes. Throughout the design, low coupling was maintained by minimizing dependencies between classes.

The Member class only interacts with Book objects through their public methods like borrowBook() and getTitle(). It doesn't need to know about the Library or LibrarianController, which keeps things simple and reduces dependencies.

Defensive copying was also used in methods like getBorrowedBooks() and getCatalog() where new ArrayList copies are returned instead of the actual internal lists. This prevents other classes from accidentally modifying the internal state and helps keep coupling low.

The LibrarianController depends on Library to find books and members, but it doesn't need to know how the library stores or manages them internally, which also helps maintain low coupling.

Conclusion

Overall, the GRASP principles helped create a well-organized design. Each class has clear responsibilities: Book and Member manage their own data, Library creates and organizes everything, and LibrarianController handles user requests. By keeping coupling low and assigning responsibilities carefully, the system is easier to understand and modify if needed in the future.