CSDS 293
Software Craftsmanship
2024 Fall Semester

Programming Assignment 12

Due at the beginning of your discussion session on

Dec 2 – 6, 2024

Reading

In addition to the following topics, the quiz syllabus includes any material covered in the lectures:

- Section 19.6 in Code Complete
- Quick Reference Guide on Routine Names and Repeated Code in Canvas
- Items 28, 59, 60, and 62 in Effective Java

The following list is optional helpful reading:

- Sections 6.1, 6.2, 6.3, and 6.4 in Code

Grading Guidelines

Quality and thoroughness are key in this phase. We'll be looking closely at your code and branch coverage. While you can skip coverage for automatically generated methods and assertions, be aware that incomplete coverage will cost you points. Steer clear of these pitfalls that will automatically result in a C grade or lower:

- Routines with complexity greater than 4
- Substantially repeated code
- Improperly named routines

Remember that merely avoiding these issues doesn't guarantee high-quality craftsmanship.

Programming

Overview

Imagine you've been tasked with designing a state-of-the-art Library Management System. This system will be the backbone of modern libraries, helping librarians efficiently manage their resources and serve patrons better. Your challenge is to create an architecture that's both robust and flexible, capable of handling the following key operations:

- Book Management: Librarians should be able to easily add new books, remove outdated ones, update information, and search for books using various criteria. You should think about how to implement efficient search algorithms. Think about indexing and data structures that can optimize search operations.
- Patron Management: The system must handle patron registrations, information updates, and membership status changes seamlessly. Think about how you handle different types of patrons (e.g., borrowing privileges?)
- Lending Operations: This is the heart of the system. It should manage checkouts, returns, reservations, and keep track of due dates and late fees. Think about how you handle overdue books and calculate late fees. Consider designing a flexible fee structure that can be easily modified.
- Inventory Management: Librarians need to know what's on their shelves. Your system should track book copies and generate insightful reports on availability and popularity. Consider implementing the Observer pattern for real-time inventory updates.
- Notification System: To keep patrons engaged and informed, design a way to send reminders about due dates and notify them when reserved books become available. Think about how to make the notification system extensible for future notification methods. You can consider using the Strategy pattern.

Additional Considerations:

- Extensibility: Design your system with future expansions in mind. For example, how would you add support for e-books or audiobooks later?

- Concurrency: Multiple librarians might use the system simultaneously. How will you handle concurrent operations to prevent data inconsistencies?
- Data Persistence: While you don't need to implement a database in this phase, consider how your design will interact with a persistent storage system.
- Security: Think about how you'll handle user authentication and authorization. Not all users should have access to all operations.
- Integration: Consider how your system might integrate with other library services, such as an online catalog or an inter-library loan system.

Implementation

- Create your interfaces and classes as outlined in your design.
- Implement the methods, following the pseudo-code you developed earlier.
- Don't forget about error handling – it's crucial for a robust system.

General Considerations

At this stage, your code should have a reasonable number of comments to guide readers through your implementation. While we'll focus more on documentation in the next assignment, aim for a level of commenting similar to what was accepted in CSDS 132.

Discussion Guidelines

Come prepared to dive deep into algorithms and class design during the discussion. Be ready to explain and defend your architectural choices

Submissions

Submit design documents that describe your architectural decisions. Create a new git repository named library-management.git for your project. Make regular, small commits as you work – this helps us see your development process. Push your design document, implemented code, and test cases to this repository.