Library Management

**Library Management System Documentation**

**Overview**

The Library Management System is a desktop GUI application built in Java using design patterns to manage books, users, and transactions (borrow/return). It ensures a modular, reusable, and maintainable codebase while providing a user-friendly interface.

**Key Features**

1. **Add Books:** Add books with title, author, and type.
2. **Add Users:** Add users (Admin or Regular User).
3. **Log Events:** Log important actions (e.g., book/user addition) using a centralized logger.
4. **Persist Data:** Ensure books and users persist across sessions using file-based storage.

**Design Patterns Used**

**1. Singleton Pattern**

* **Logger Class:** Ensures only one instance of the logger is
* created and used throughout the application.
* **Why Singleton?** Prevents multiple instances, ensuring consistent logging across all actions.

**2. Factory Pattern**

* **BookFactory Class:** Creates different types of books (e.g., Software Engineering, Management, AI).
* **UserFactory Class:** Creates users (Admin or Regular User) based on input.
* **Why Factory?** Encapsulates object creation, promoting flexibility and scalability.

**3. MVC (Model-View-Controller)**

* Separates the application into:
  + **Model:** Handles data and logic (e.g., Book, User).
  + **View:** Manages the graphical user interface.
  + **Controller:** Coordinates between the model and the view.
* **Why MVC?** Enhances modularity and testability.

**Class Descriptions**

**1. Logger**

* **Purpose:** Centralized logging mechanism.
* **Key Methods:**
  + getInstance(): Returns the singleton instance.
  + log(String message): Logs messages to a file.
  + close(): Closes the log file during shutdown.
* **File:** log.txt

**2. BookFactory**

* **Purpose:** Creates books based on the type provided.
* **Key Methods:**
  + createBook(String type, String title, String author): Returns a Book object.

**3. UserFactory**

* **Purpose:** Creates users (Admin or Regular User).
* **Key Methods:**
  + createUser(String type, String username): Returns a User object.

**4. LibraryManagementSystem**

* **Purpose:** Main class managing the GUI and application logic.
* **Key Components:**
  + Buttons: Add Book, Add User.
  + Text Area: Displays list of books.
  + Input Fields: Title, Author, Type, Username.

**GUI Components**

**Add Book Panel**

* **Input Fields:**
  + Title (Text Field)
  + Author (Text Field)
  + Type (Dropdown)
* **Button:** Add Book
* **Actions:**
  + Validate input fields.
  + Use BookFactory to create a book.
  + Append the book details to the text area.
  + Log the action using Logger.

**Add User Panel**

* **Input Fields:**
  + Username (Text Field)
  + User Type (Dropdown: Admin, Regular User)
* **Button:** Add User
* **Actions:**
  + Validate input fields.
  + Use UserFactory to create a user.
  + Log the action using Logger.

**Data Persistence**

* **File-Based Storage:**
  + Books and users are stored in text files (e.g., books.txt, users.txt).
  + Upon startup, the application reads these files to load existing data.
* **Methods:**
  + writeDataToFile(String data): Writes data to a file.
  + loadDataFromFile(String filename): Reads data from a file and displays it in the GUI.

**Example Use Case**

1. **Adding a Book:**
   * User enters Title, Author, and selects Type.
   * Clicks **Add Book**.
   * The system:
     + Validates inputs.
     + Uses BookFactory to create a book.
     + Displays the book in the list.
     + Logs the action to log.txt.
2. **Adding a User:**
   * User enters Username and selects User Type.
   * Clicks **Add User**.
   * The system:
     + Validates inputs.
     + Uses UserFactory to create a user.
     + Logs the action to log.txt.
3. **Session Persistence:**
   * On application exit, books and users are saved to books.txt and users.txt.
   * On startup, the application loads these files and displays their contents.

**Future Enhancements**

1. **Database Integration:** Replace file-based storage with a database for better scalability.
2. **Observer Pattern:** Notify components of changes (e.g., new books/users).
3. **Transaction Management:** Implement borrowing/returning functionality.
4. **Enhanced Security:** Add user authentication and access control.

**Conclusion**

This Library Management System provides a foundation for managing library operations using Java and design patterns. Its modular and scalable design allows for future enhancements and ensures maintainability.