Homework

Part 1: Design Patterns

Part 1: Implementing Design Patterns

Choose two **design patterns** from the following list: *State, Command, Observer,* and apply them in a scenario based on the topic of your thesis.

Requirements:

- ✓ Clearly present your scenario and explain the role of each implemented pattern.
- ✓ The implementation must include fully functional and properly commented Java code.

Part 2: Automated Testing with JUnit

```java
class Book {
    private String isbn;
    private boolean borrowed;
    private String borrowedBy;

    public Book(String isbn) {
        this.isbn = isbn;
        this.borrowed = false;
    }

    public String getIsbn() { return isbn; }
    public boolean isBorrowed() { return borrowed; }
    public void setBorrowed(boolean borrowed) { this.borrowed = borrowed; }
    public String getBorrowedBy() { return borrowedBy; }
    public void setBorrowedBy(String borrowedBy) { this.borrowedBy = borrowedBy; }
}
```

```java
public class LibrarySystem {
    private Map<String, Book> books;

    public boolean borrowBook(String isbn, String userId) { /* implementation */ }
    public boolean returnBook(String isbn, String userId) { /* implementation */ }
    public boolean addBook(Book book) { /* implementation */ }
}
```

Requirements:

Write automated tests using JUnit to check the following conditions:

- ✓ A book cannot be borrowed if it is already checked out.
- ✓ A book can be returned only by the user who borrowed it.
- ✓ Adding a new book should reject duplicates based on ISBN.
- ✓ Use external files for test values (inputs and expected results).

Structure your tests so that they can be integrated into a Test Suite.