**1. Structure for Storing Book Information**

The structure for storing book information involves a **hierarchical tree**. Each category can have multiple subcategories and books associated with it. Books are leaf nodes in the hierarchy, while categories and subcategories act as intermediate nodes.

**2. Design Patterns to Define the System Structure**

The following design patterns are suitable:

1. **Composite Pattern**:  
   To represent the hierarchical structure where categories can contain subcategories or books. This allows treating categories and books uniformly.
2. **Adapter Pattern**:  
   To make the existing PrinterManager class compatible with the new system by creating an adapter class that translates the expected input format.
3. **Observer Pattern** (Optional):  
   If real-time updates or notifications are needed for changes in the book catalog.

**3. Class Diagram**

The class diagram represents the relationships between categories, books, authors, publishers, and printing functionality.

1. **Category Class**
   * name: string
   * subCategories: Category[]
   * books: Book[]
   * Methods:
     + addCategory(Category)
     + addBook(Book)
     + getBooks()
2. **Book Class**
   * title: string
   * year: int
   * isbn: string
   * description: string
   * author: Author[]
   * publisher: Publisher
3. **Author Class**
   * firstName: string
   * lastName: string
   * birthYear: int
4. **Publisher Class**
   * name: string
   * location: string
   * country: string
5. **CardPrinterAdapter Class** (Implements Adapter Pattern)
   * printCard(user: User, book: Book)
   * Converts data to match PrinterManager.printCard().
6. **PrinterManager Class** (Existing class)

**Hierarchical View of Classes**

* Category
  + Has many Book.
  + Has many Category as subCategories.
* Book
  + Has one Publisher.
  + Has many Author.
* CardPrinterAdapter
  + Uses PrinterManager.