**1. Singleton: Zarządzanie katalogiem książek**

**Klasa:** LibraryCatalog  
**Cel:** Zapewnienie, że istnieje tylko jeden katalog książek w systemie.  
**Zalety:**

* Centralizacja zarządzania książkami w jednym katalogu.
* Unikanie problemów z niezgodnymi lub wieloma instancjami tego samego katalogu.
* Wzorzec poprawia organizację systemu i zapobiega redundancji danych.

**2. Adapter: Umożliwienie wczytywania danych w różnych formatach**

**Klasy:** BookDataAdapter, JSONBookAdapter, CSVBookAdapter  
**Cel:** Konwersja danych o książkach z różnych formatów (JSON, CSV) na jednolitą strukturę używaną przez system.  
**Zalety:**

* Elastyczność w obsłudze wielu formatów danych.
* Ułatwia rozszerzanie systemu o nowe formaty bez zmieniania istniejącego kodu.
* Adapter upraszcza proces wczytywania książek do katalogu, co zwiększa wszechstronność systemu.

**3. Factory: Tworzenie użytkowników z różnymi uprawnieniami**

**Klasy:** UserFactory, User, Student, Teacher, Librarian  
**Cel:** Umożliwienie łatwego tworzenia obiektów użytkowników o różnych rolach w systemie.  
**Zalety:**

* Łatwość dodawania nowych typów użytkowników w przyszłości.
* Przejrzysta logika tworzenia użytkowników, co ułatwia zarządzanie uprawnieniami.
* Factory oddziela logikę tworzenia obiektów od ich używania, co poprawia czytelność kodu.

**4. Observer: Powiadamianie użytkowników o zmianach**

**Klasy:** Observer, UserObserver, LibraryNotifier  
**Cel:** Automatyczne powiadamianie subskrybentów (np. użytkowników) o istotnych zdarzeniach w systemie.  
**Zalety:**

* Wzorzec umożliwia dynamiczne dodawanie/usuwanie subskrybentów bez ingerencji w logikę powiadamiania.
* Ułatwia zarządzanie powiadomieniami w dużych systemach, poprawiając doświadczenie użytkownika.
* Powiadomienia są wywoływane automatycznie, co eliminuje potrzebę ręcznego informowania każdego użytkownika.

**5. Facade: Upraszczanie interakcji z systemem**

**Klasa:** LibraryInterface  
**Cel:** Zgrupowanie złożonych operacji zarządzania książkami i użytkownikami w prosty interfejs.  
**Zalety:**

* Użytkownicy systemu mogą korzystać z łatwego w użyciu API, bez potrzeby znajomości szczegółów implementacyjnych.
* Poprawia czytelność kodu, grupując powiązane funkcje (np. dodawanie książek, wypożyczanie, zwracanie).

**6. Iterator: Iteracja po książkach**

**Klasa:** BookIterator  
**Cel:** Zapewnienie wygodnego mechanizmu iteracji po książkach w katalogu.  
**Zalety:**

* Umożliwia przeglądanie książek w katalogu bez ujawniania szczegółów implementacji (np. listy).
* Iterator działa zgodnie z zasadami Pythona (\_\_iter\_\_ i \_\_next\_\_), co pozwala na użycie w pętlach for.
* Zwiększa czytelność i elastyczność przy przeglądaniu danych.

**Podsumowanie**

Użycie tych wzorców projektowych znacząco poprawia strukturę, elastyczność i użyteczność systemu zarządzania biblioteką:

* **Singleton** centralizuje dane.
* **Adapter** pozwala na integrację danych z różnych źródeł.
* **Factory** standaryzuje proces tworzenia użytkowników.
* **Observer** automatyzuje i organizuje powiadamianie użytkowników.
* **Facade** upraszcza interakcje z systemem.
* **Iterator** ułatwia przeglądanie danych.