**1. Singleton + Observer**

Klasa LibraryCatalog pełni podwójną rolę, implementując wzorce **Singleton** i **Observer**. Singleton zapewnia, że istnieje tylko jedna globalna instancja katalogu książek, co pozwala na spójne zarządzanie danymi w całym systemie. Wszystkie operacje na książkach, takie jak dodawanie czy wyszukiwanie, odbywają się na tej samej instancji. Wzorzec **Observer** pozwala subskrybować powiadomienia o dostępności książek. Użytkownicy mogą rejestrować się jako obserwatorzy, a system powiadamia ich, gdy dana książka stanie się dostępna. Takie połączenie wzorców zapewnia centralne zarządzanie danymi i dynamiczne informowanie użytkowników o zmianach w katalogu.

**2. Adapter**

Klasa DataAdapter umożliwia importowanie danych o książkach w różnych formatach, takich jak JSON, XML czy CSV, i konwertuje je na ujednolicony format. Dzięki temu system może łatwo integrować się z zewnętrznymi źródłami danych, niezależnie od ich formatu. Wzorzec Adapter upraszcza integrację, eliminując potrzebę ręcznego przekształcania danych w każdym przypadku, co zwiększa elastyczność i rozszerzalność systemu.

**3. Factory**

Klasa UserFactory upraszcza proces tworzenia różnych typów użytkowników systemu, takich jak Student, Teacher, Librarian czy Guest. Każdy typ użytkownika ma inne uprawnienia (np. limit wypożyczeń czy możliwość zarządzania katalogiem). Dzięki zastosowaniu wzorca **Factory**, tworzenie użytkowników jest dynamiczne i elastyczne, co pozwala łatwo dodawać nowe role bez zmiany istniejącego kodu.

**4. Facade**

Klasa LibraryInterface działa jako fasada, zapewniając prosty i spójny interfejs do interakcji z systemem bibliotecznym. Użytkownicy mogą dodawać książki, wyszukiwać je, rejestrować się w systemie, wypożyczać książki i zwracać, nie mając wiedzy o wewnętrznej strukturze systemu. Wzorzec **Facade** upraszcza dostęp do złożonego systemu, ukrywając szczegóły implementacyjne takich podsystemów jak LibraryCatalog czy UserManagement.

**5. Iterator**

Klasa BookIterator umożliwia iterację po książkach w katalogu. Zwraca dane w ustrukturyzowanej formie, co pozwala na łatwe przetwarzanie listy książek, np. w celu ich wyświetlenia. Dzięki wzorcowi **Iterator**, iteracja po kolekcji staje się czytelna i zgodna z Pythonowym protokołem iteratora, eliminując potrzebę ręcznego zarządzania indeksami.