Anton Bianca-Elena, M533

Documentație - Aplicarea principiilor SOLID

Am modificat proiectul inițial astfel:

1. **Single Responsibility Principle (SRP)**

* Problema identificată: Clasa LibrarySystem avea mai multe responsabilități, gestionând atât cărțile, membrii, cât și împrumuturile.
* Modificare efectuată: Am separat sarcinile în trei clase:

-BookManager - gestionează cărțile

-MemberManager - gestionează membrii

-BorrowManager -gestionează împrumuturile cărților

De asemenea am modificat clasa LibrarySystem pentru a se folosi de acestea.

1. **Open/Closed Principle (OCP)**

* Problema identificată: Nu puteam schimba tipul de carte/membru fără să trebuiască să modificăm LibrarySystem.
* Modificare efectuată:

- Am introdus interfețe (IBookManager.cs, IMemberManager.cs, IBorrowingManager.cs), pentru a putea extinde comportamentului fără a modificare codul existent.

-Am adăugat clasele NonfictionBook.cs și FictionBook.cs pentru extinderea comportamentului cărților fără a modifica implementarea existentă.

-Am adăugat clasele Admin.cs și NonAdmin.cs pentru extinderea comportamentului membrilor fără a modifica implementarea existentă.

-Am modificat LibrarySystem să utilizeze interfețe, astfel încât să nu fie nevoie de modificări la adăugarea unui nou tip de carte

-Clasele BookManager, MemberManger și BorrowManager extind interfețele menționate.

1. **Liskov Substitution Principle (LSP)**

* Problema identificată: Clasele care foloseau metodele din LibrarySystem nu puteau folosi obiecte derivate într-un mod în care să înlocuiască clasa de bază.
* Modificare efectuată:

-Am creat interfața IBook, iar clasele NonFictionBook și FictionBook o implementează.

- Funcțiile care operează pe IBook pot lucra cu orice tip de carte derivată.

- Putem extinde codul cu noi tipuri de cărți (ex. RomanceBook) fără a modifica codul existent.

1. **Interface Segregation Principle (ISP)**

* Problema identificată: ILibrarySystem avea toate metodele (AddBook, AddMember, BorrowBook, ReturnBook), astfel clasele care o implementau conțineau metode pe care nu le foloseau.
* Modificare efectuată: Am separat responsabilitățile din ILibrarySystem în mai multe interfețe:

-IBookManager.cs -pentru gestionarea cărților.

-IMemberManager.cs - pentru gestionarea membrilor.

-IBorrowingManager.cs - pentru gestionarea împrumuturilor de cărți.

1. **Dependency Inversion Principle (DIP)**

* Problema identificată: Inițial LibrarySystem depindea direct implementări concrete (Listele pentru cărți și membri).
* Modificare efectuată:

-Am implementat interfețele IBook și IMember pentru ca codul sa nu depindă de clase specifice pentru cărți și membrii.

-LibrarySystem nu mai depinde direct de liste, ci de interfețe (IBookManager, IMemberManager, IBorrowingManager). Astfel, putem schimba implementarea internă fără a afecta codul care folosește LibrarySystem.