Correction

1. Singleton Pattern

- Contexte d'utilisation : Peut être appliqué à la classe `Bibliothèque` si l'application gère une seule instance d'une bibliothèque en cours d'exécution. Cela garantit qu'il n'y a qu'une seule instance de la bibliothèque et fournit un point d'accès global à cette instance.

- Avantages : Contrôle global sur la bibliothèque. Utile si vous avez besoin d'une configuration partagée.

2. Observer Pattern

- Contexte d'utilisation : Peut être implémenté entre `Utilisateur` et `Livre`. Lorsqu'un livre est emprunté ou retourné, les utilisateurs intéressés peuvent être informés. Ce pattern est utile si vous envisagez de notifier automatiquement les utilisateurs lorsqu'un livre devient disponible.

- Avantages : Permet une communication simple entre objets sans couplage fort. Facilite la mise à jour automatique des états.

3. Composite Pattern

- Contexte d'utilisation : Peut être appliqué au `Catalogue`, surtout si vous souhaitez gérer des catégories ou sous-catégories de livres. Par exemple, un catalogue pourrait contenir des sous-catalogues par genre ou par section, chaque sous-catalogue ayant sa propre liste de livres.

- Avantages : Simplifie la gestion des collections hiérarchiques de livres. Facilite les opérations sur l'ensemble ou partie de la structure.

4. Factory Pattern

- Contexte d'utilisation : Peut être utilisé pour centraliser la création d'objets `Livre`. Si l'instanciation d'un livre nécessite une logique complexe (par exemple, vérification de l'ISBN, gestion des éditions), une Factory peut être utile.

- Avantages : Encapsulation de la logique de création d'objets. Permet la modification de la logique de création sans affecter les clients.