Violations identifiées :

```java
public class BookStore {
    public void addBook(String title, String type, double price) {
        if ("PHYSICAL".equals(type)) {
            System.out.println("Adding a physical book: " + title);
// Assume this logic adds a physical book to the database
        }
// No implementation for e-books or audiobooks, violating OCP
    }
    public void processOrder(String bookTitle, String userEmail) {
        System.out.println("Processing order for " + bookTitle);
// Assume this logic processes an order and notifies the user
        sendNotification(userEmail, "Your order for " + bookTitle + " has been processed.");
    }
    private void sendNotification(String email, String message) {
        System.out.println("Sending email to " + email + " with message: " + message);
// Assume this logic sends an email, creating a direct dependency, violating DIP
    }
}
```

SRP : BookStore gère trop de fonctionnalités différentes
OCP : Ne prends pas en compte les différents type de livre -> fermé à l'extension
DIP : Il ya une dépendence directe entre processOrder et sendNotification

```java
public interface UserActions {  2 usages
    void borrowBook(String bookTitle);  1 usage
    void returnBook(String bookTitle);  1 usage
    void reviewBook(String bookTitle, String review);  2 usages
    void addBookToStore(String title, String type, double price);  no usages
    void removeBookFromStore(String title);  no usages
}
```

```java
public class Customer implements UserActions {
```

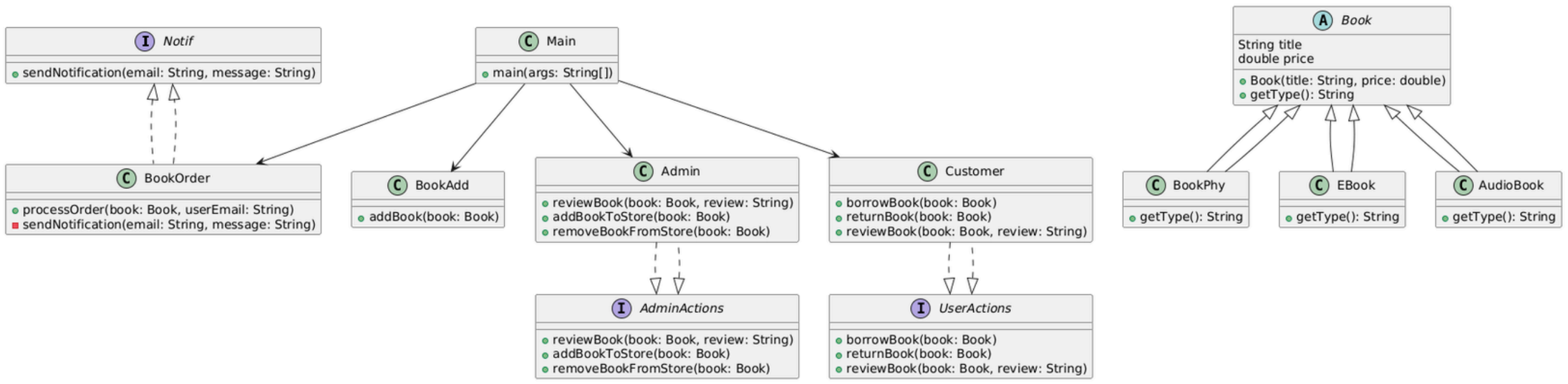
```java
public class Admin implements UserActions {
```

ISP : Customer et Admin implémentent la même interface user actions cependant seul l'admin a accès aux méthodes removeBookFromStore et addBookToStore

Solution apportées :

Principe	**Problème**	**Solution**
----------	----------	----------
SRP (Responsabilité unique)	`BookStore` gérait ajout de livres et commandes	Réfactorisation en `BookAdd` et `BookOrder`
OCP (Ouvert/Fermé)	`addBook()` ne supportait que les livres physiques	Utilisation d’une classe abstraite `Book` avec des sous-classes EBook, BookPhys & AudioBook
LSP (Substitution de Liskov)	Les sous-classes redéfinissaient mal `type`	Utilisation de `getType()` au lieu d’un attribut `type` pour une meilleure flexibilité
ISP (Ségrégation des interfaces)	`Admin & Customer` implémentait `UserActions` alors qu'ils n’avaient pas les mêmes fonctionalités	Séparation en `AdminActions` et `UserActions`
DIP (Inversion des dépendances)	`sendNotification()` était privé dans `BookOrder`	Création de l'interface `Notif` pour découpler

UML :



Améliorations et extensibilité :

Grâce à la refactorisation selon SOLID, l'application est devenue plus maintenable et extensible. Chaque classe a désormais une responsabilité unique, facilitant la compréhension et les modifications. L'ajout de nouveaux types de livres ou de notifications ne nécessite plus de modifier le code existant, rendant l'application évolutive. De plus, la séparation des interfaces et l'inversion des dépendances améliorent la réutilisabilité et la testabilité, garantissant un code plus clair, modulaire et évolutif.