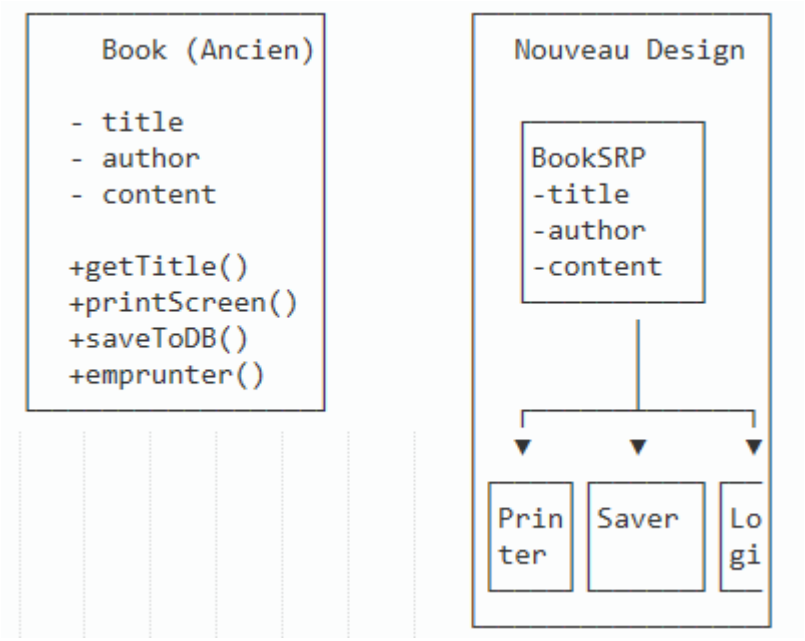
Problèmes des codes SOLID

1. SRP (Single Responsibility Principle)

Violations :

- Une classe fait trop de choses
- Changement de raison multiple
- Difficile à tester
- Réutilisation impossible

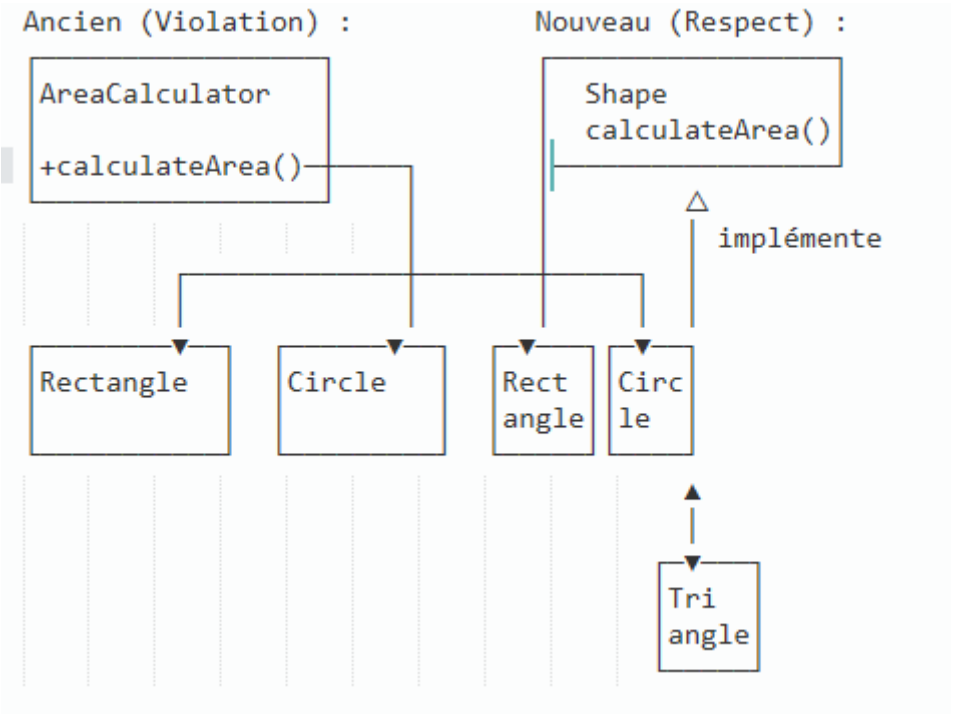
Diagramme SRP :



2. OCP (Open/Closed Principle)

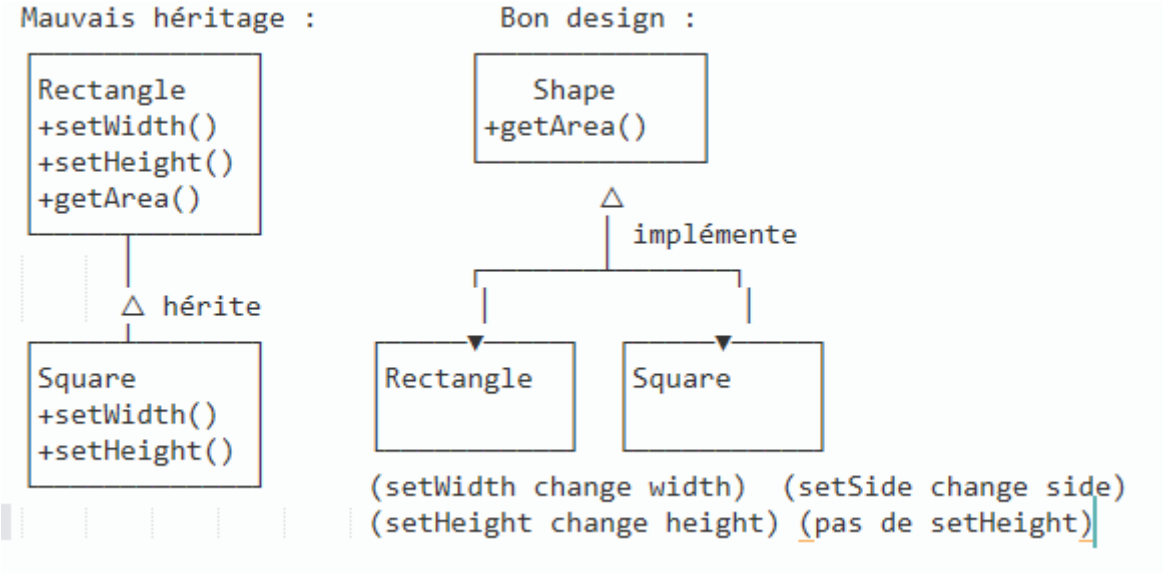
Violations:
Pas fermé à la modification
Pas ouvert à l'extension
Violation du polymorphisme
Couplage fort

Diagramme OCP :



3. LSP (Liskov Substitution Principle)

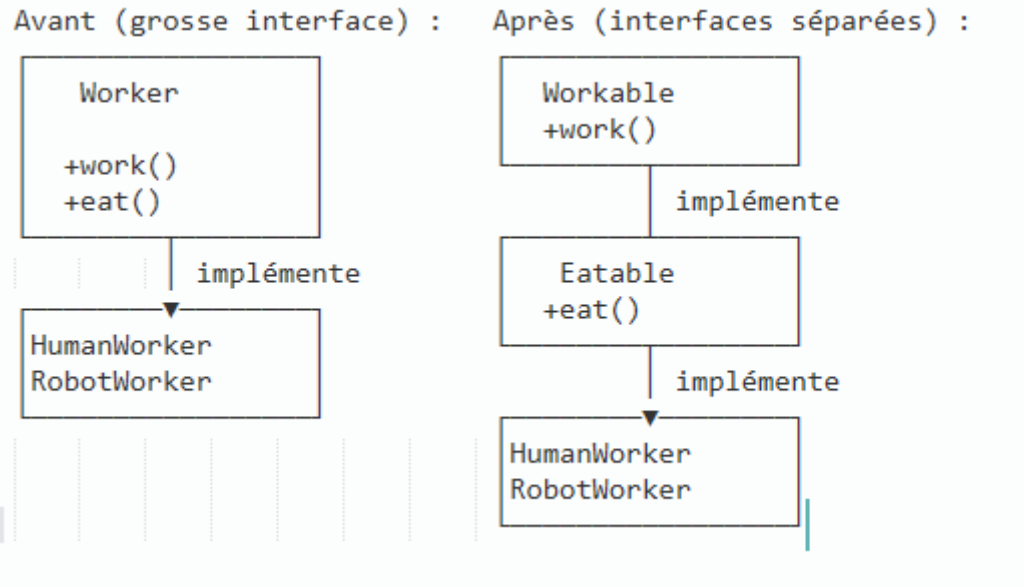
Violations :
Square ne peut pas remplacer Rectangle
Comportement différent pour les mêmes méthodes
Post-conditions violées
Préconditions renforcées
Diagramme LSP :



4. ISP (Interface Segregation Principle)

Violations :
Interface trop large

Clients forcés de dépendre de méthodes inutiles
Implémentation de méthodes "vides" ou lançant des exceptions
Couplage entre responsabilités non liées
Diagramme ISP :



5. DIP (Dependency Inversion Principle)

Violations :
Dépendance sur des classes concrètes
Impossible de changer d'implémentation
Difficile à tester (pas d'injection)
Violation du principe d'inversion
Diagramme DIP :

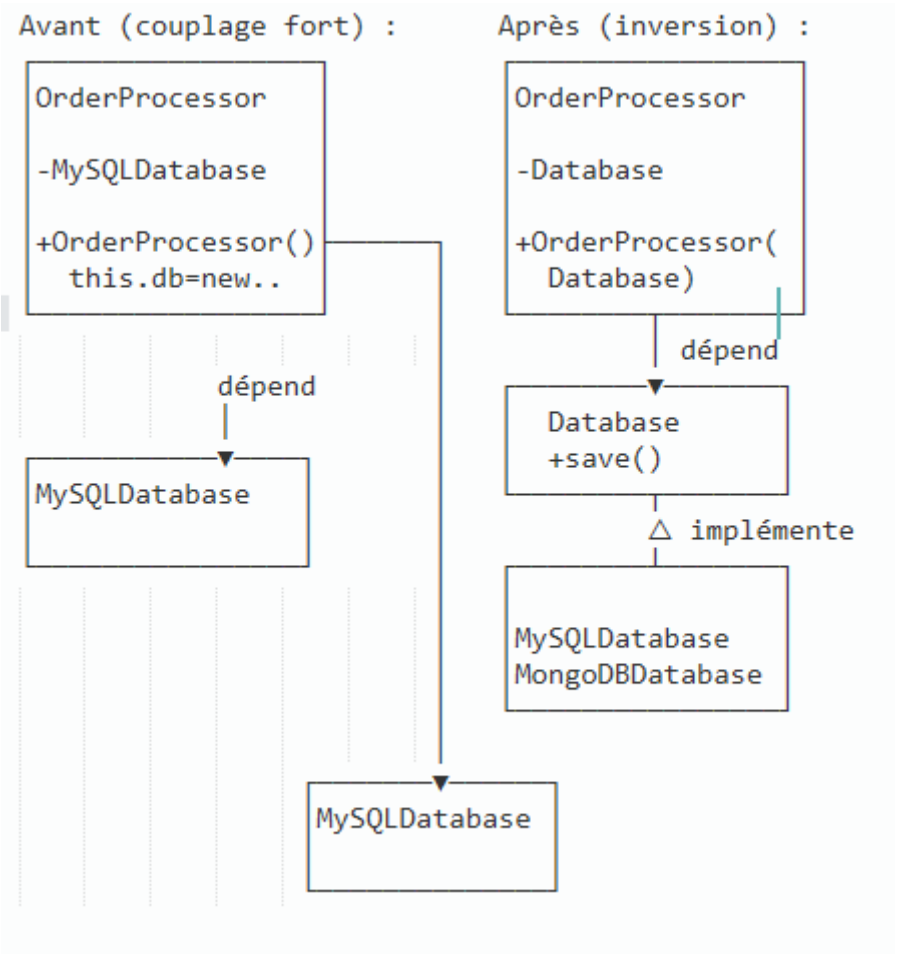
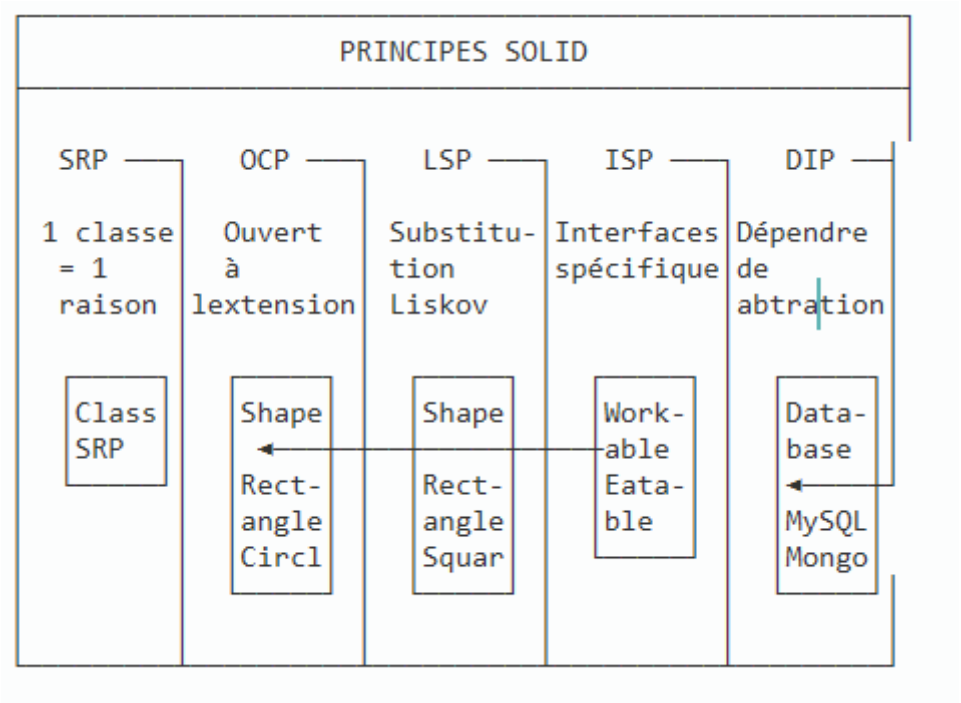


Diagramme global SOLID :



Résumé des problèmes :

Principe	Problème	Solution

SRP	Classe qui fait tout	Séparer en classes spécialisées
OCP	Modifier pour étendre	Utiliser des interfaces/abstraction
LSP	Héritage incorrect	Pas d'héritage inapproprié
ISP	Interface trop grosse	Plusieurs petites interfaces
DIP	Dépendance concrète	Dépendre d'abstractions

Bénéfices du respect SOLID :

- Maintenance facilitée
- Testabilité améliorée
- Réutilisabilité
- Évolutivité
- Moins de bugs
- Développement en équipe facilité

Les diagrammes montrent comment passer d'un design rigide et fragile à un design flexible et robuste grâce aux principes SOLID.