ШАБЛОНЫ ПРОЕКТИРОВАНИЯ

Основные понятия

Шаблон (паттерн) проектирования – это наиболее часто встречаемое решение какой либо проблемы при проектировании архитектуры программного обеспечения

Особенности:

- Описан в абстрактном виде, реализация может отличаться для разных языков программирования

Состав шаблона проектирования

1. проблема, решаемой паттерном
2. Структура классов для реализации решения
3. Пример на каком то конкретном ЯП
4. Особенности реализации в различных ситуациях
5. Связь с другими паттернами

Классификация паттернов

Порождающие
Описывают создание объектов без внесения в ПО лишних зависимостей

Структурные
Определяют различные способы построения связей между объектами ПО

Поведенческие
Определяют эффективные способы коммуникации между объектами

GOF book

Плюсы применения паттернов

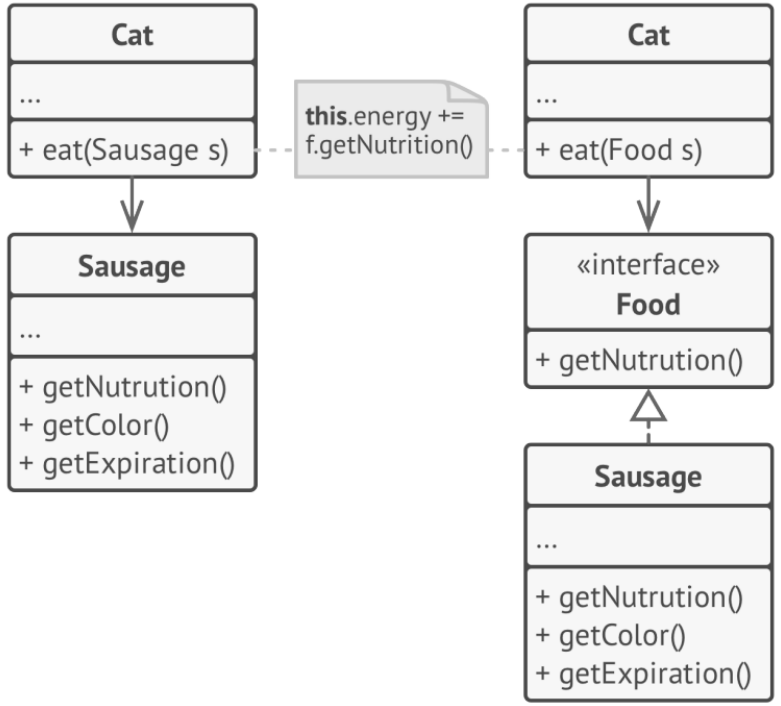
- Проверенные решения
- Стандартизация кода
- Общий словарь программистов

Основные принципы проектирования

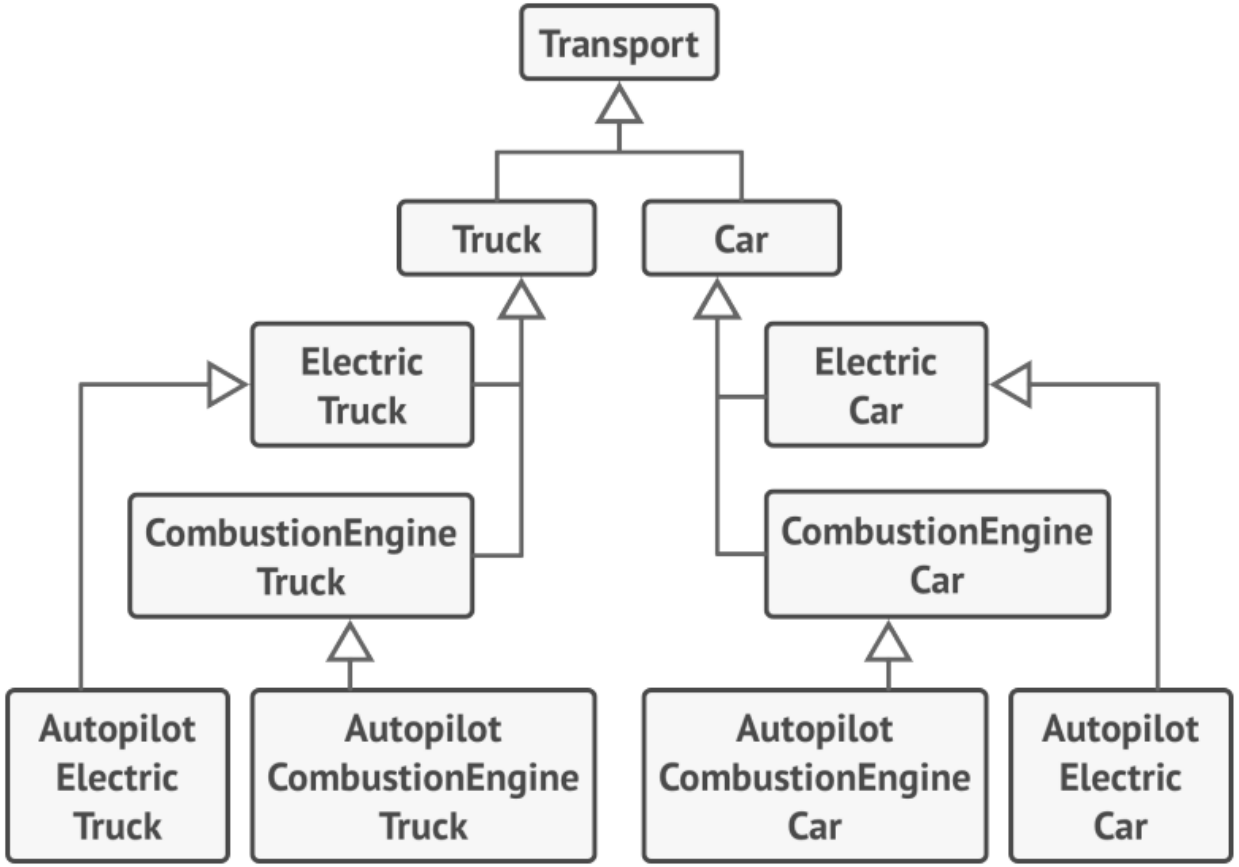
1 Инкапсуляция частей кода, которая подвержена изменениям
Определите аспекты программы, класса или метода, которые меняются чаще всего и отделите их того, что остаётся постоянным.

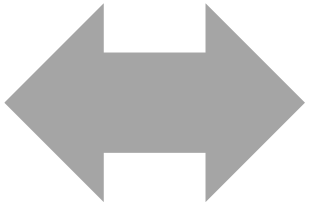
2 Программирование на уровне интерфейса
Программируйте на уровне интерфейса, а не на уровне реализации. Код должен зависеть от абстракций, а не конкретных классов.

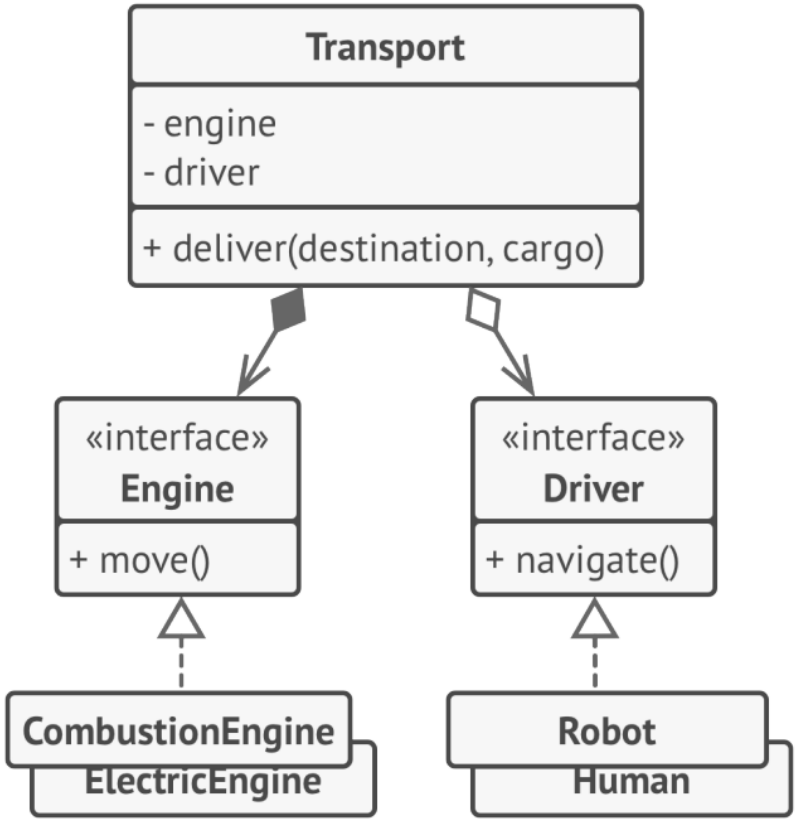
3. Композиция предпочтительней наследования
Проблемы наследования:
1. Невозможность класса потомка отказаться от интерфейсов или реализации родителя
1. Дополнительный контроль при переопределении
2. Наследование нарушает инкапсуляцию класса родителя
3. Подклассы тесно связаны с родителем
4. Быстро разрастается иерархия классов



Пример предпочтительности композиции и агрегации наследованию

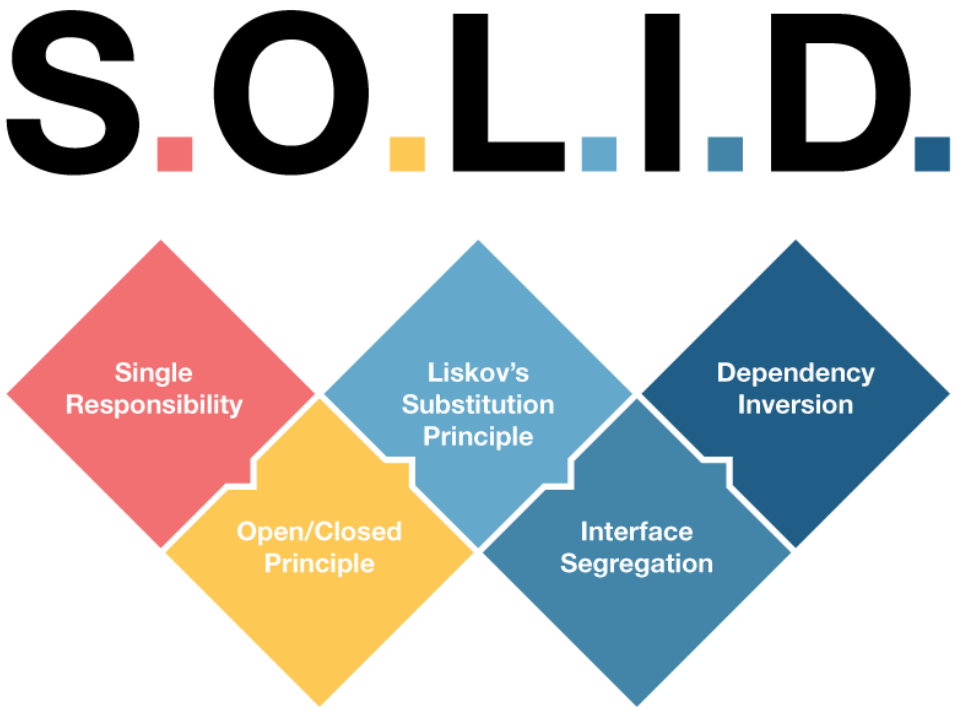






Принципы SOLID

Главная цель SOLID— повысить гибкость вашей архитектуры, уменьшить связанность между её компонентами и облегчить повторное использование кода.

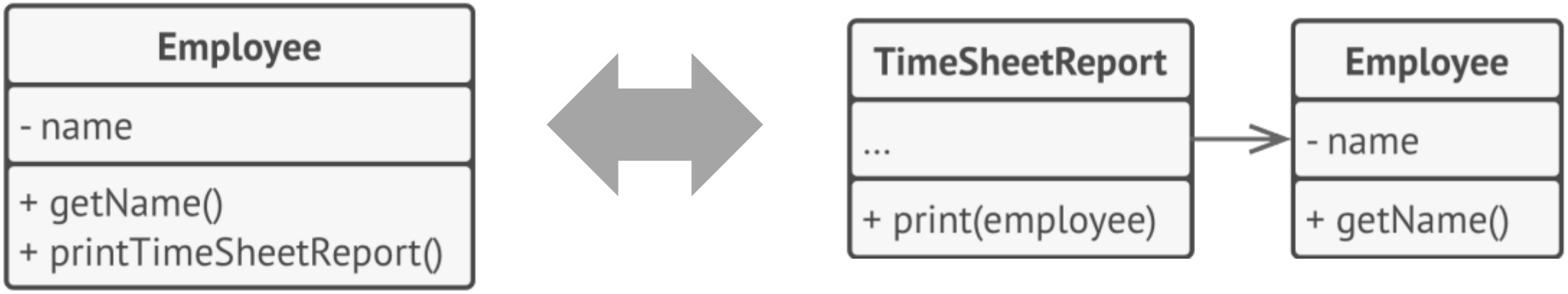


[Agile software development, principles, patterns and practices]

Single Responsibility Principle

S - Принцип единой ответственности

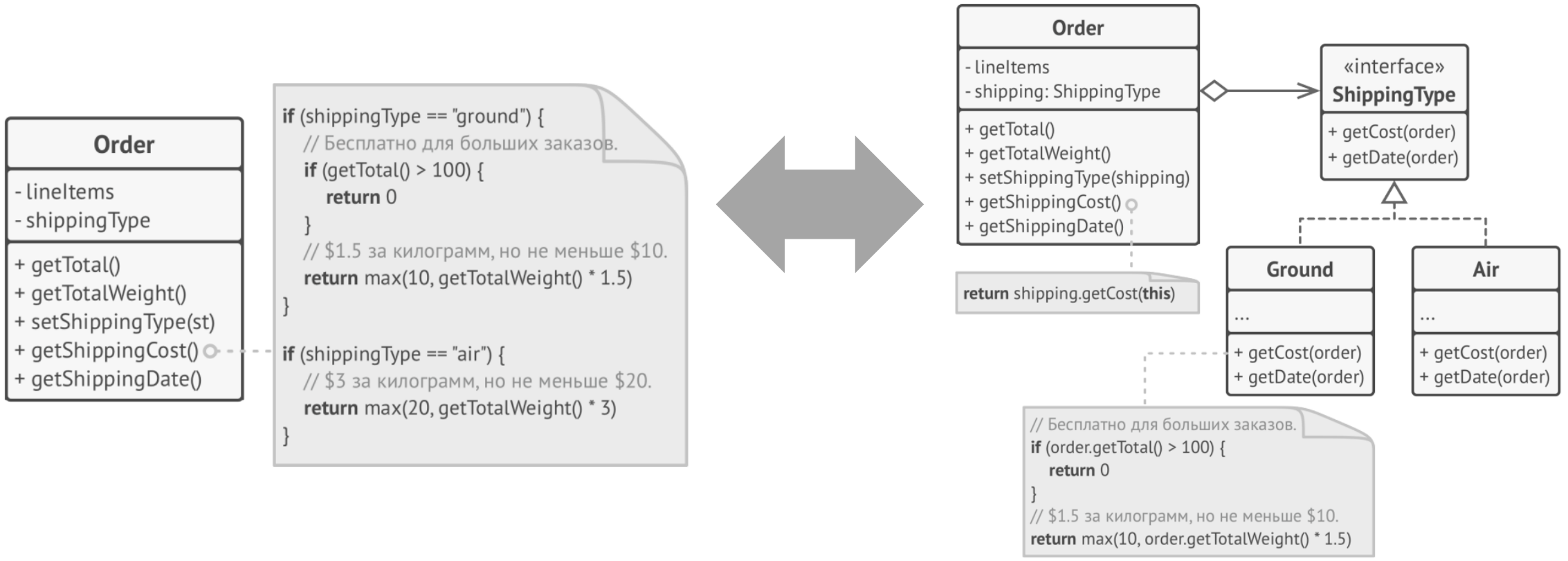
Класс должен отвечать только за одну часть функциональности программы. Эта функциональность должна быть полностью инкапсулирована в этот класс



Open/closed principle

O - Принцип открытости / закрытости

Расширяйте классы, но не изменяйте их первоначальный код.

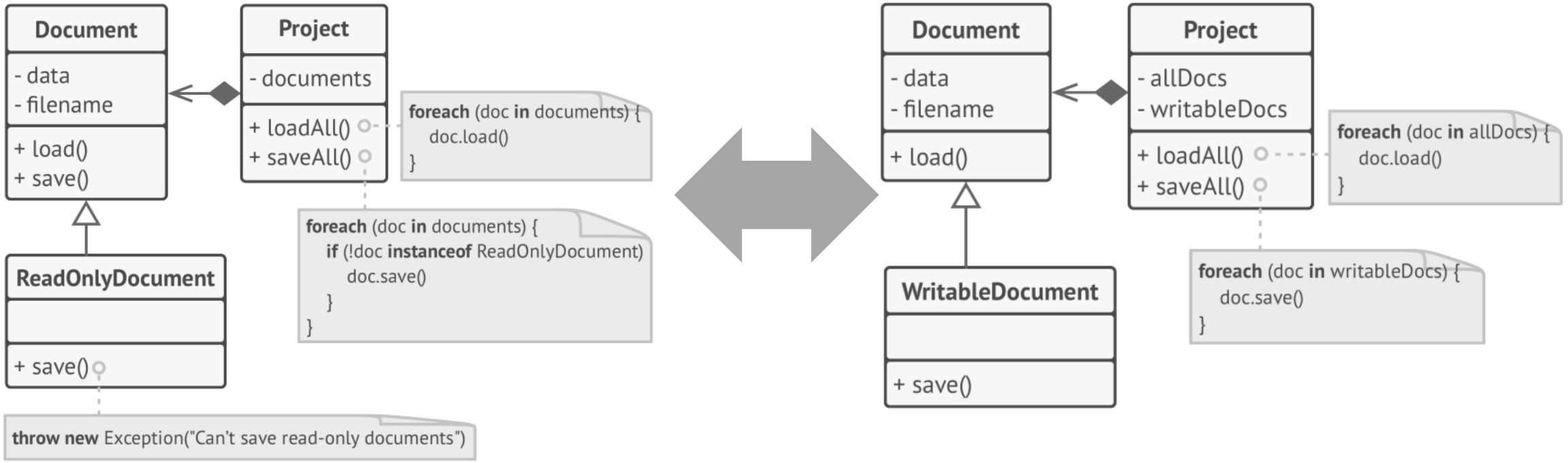


Liskov Substitution Principle

L

Принцип подстановки Лисков

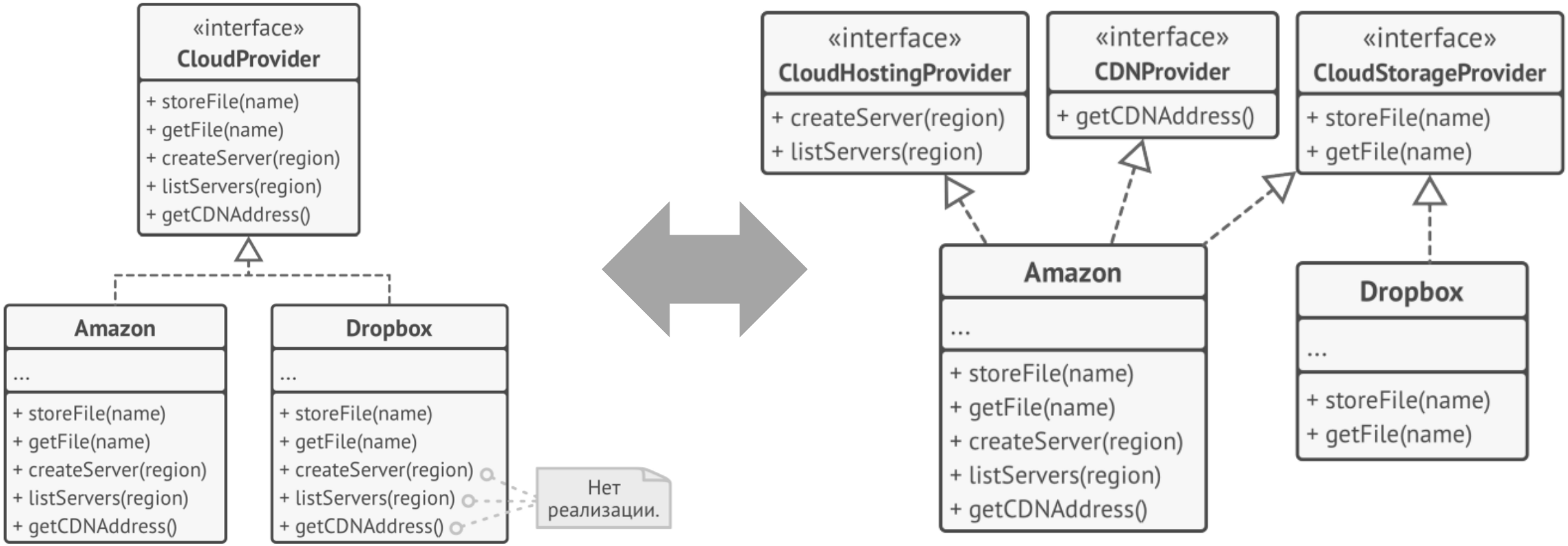
Подклассы должны дополнять, а не замещать поведение базового класса.



Interface Segregation Principle

Принцип разделения интерфейса

Клиенты не должны зависеть от методов, которые они не используют.



Dependency Inversion Principle

D

Принцип инверсии зависимостей

Классы верхних уровней не должны зависеть от классов нижних уровней. Оба должны зависеть от абстракций. Абстракции не должны зависеть от деталей. Детали должны зависеть от абстракций.