Структурные паттерны

Основные понятия

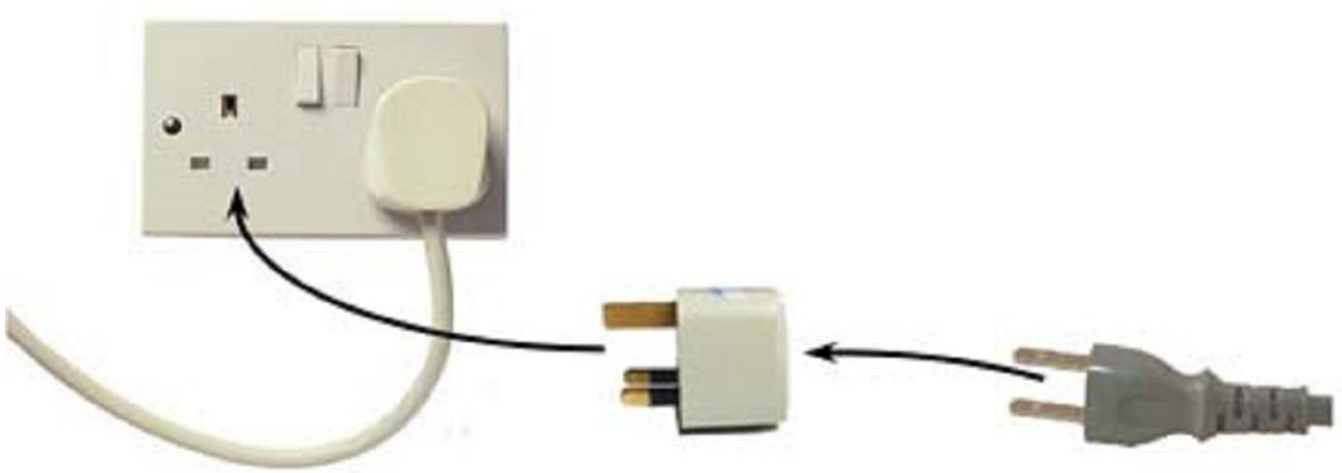
Структурные паттерны обеспечивают гибкую иерархию классов.

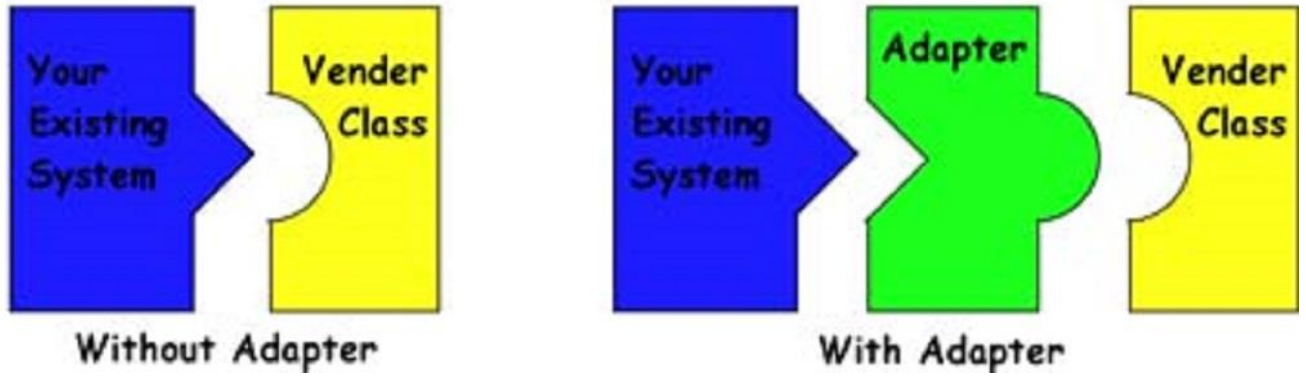
Основные представители:

- Адаптер
- Мост
- Декоратор
- Фасад
- Компоновщик
- Заместитель

Адаптер

Обеспечивает совместную работу объектов с несовместимыми интерфейсами

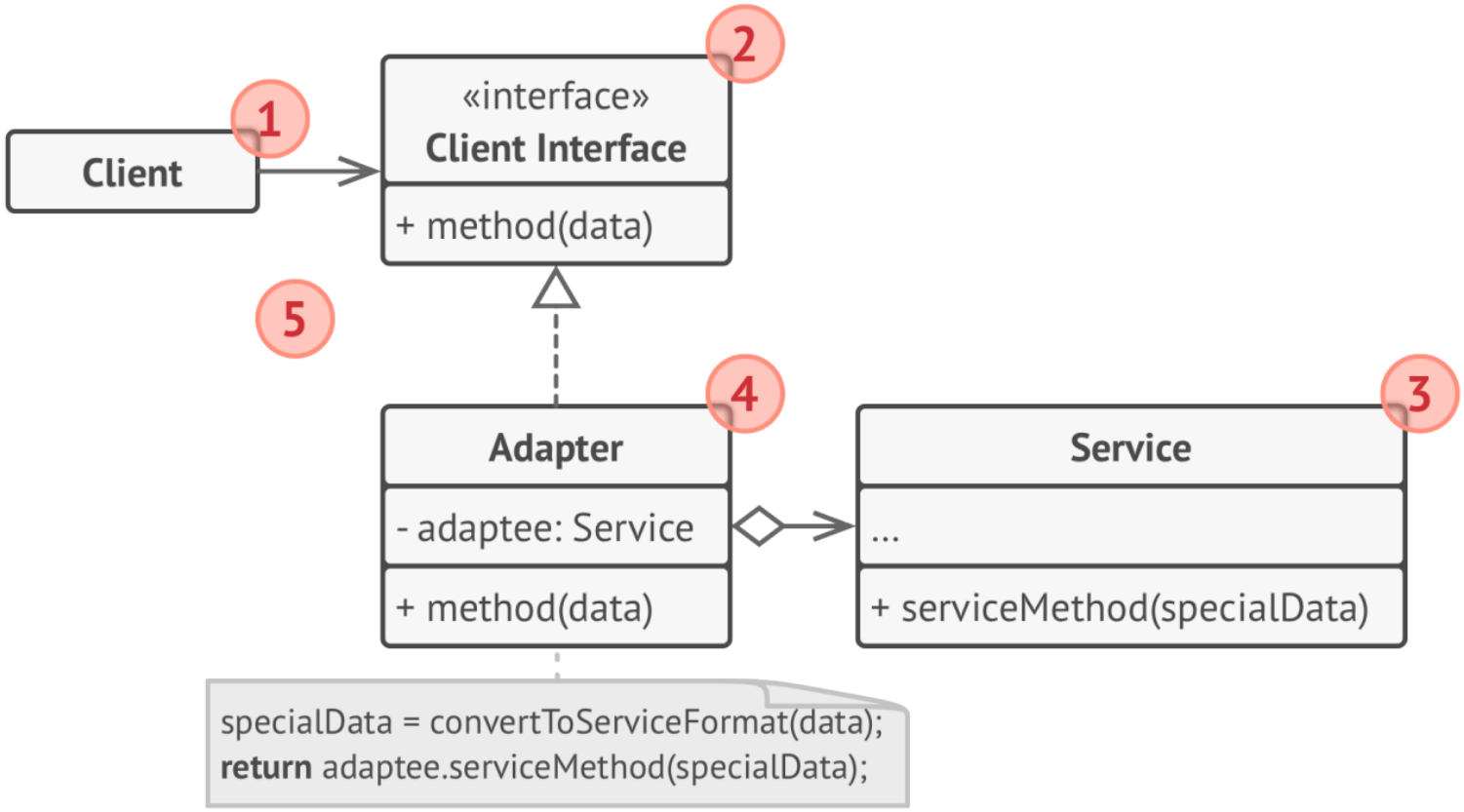




дополнительно

Адаптер объектов

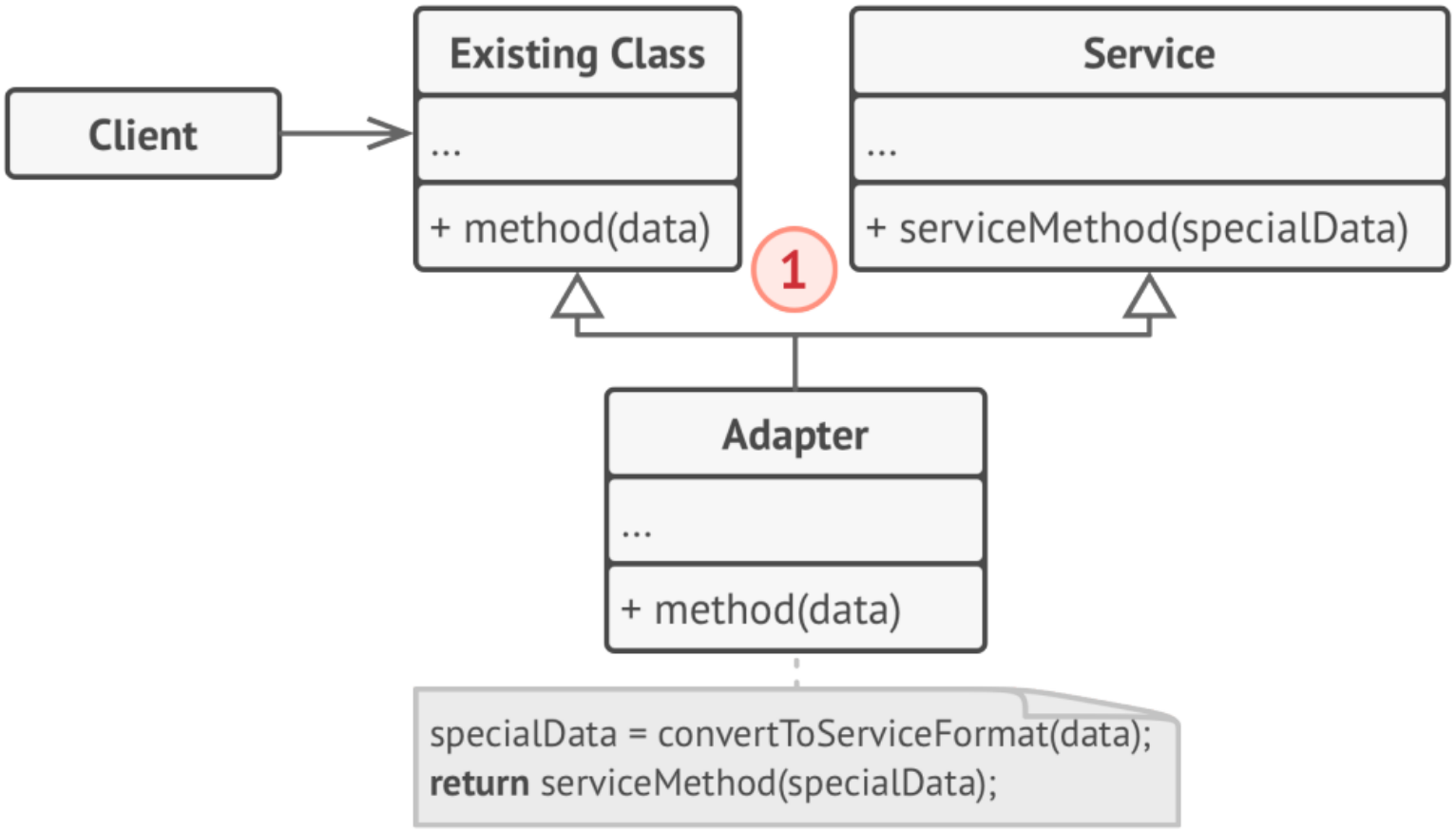
- Содержит ссылку на служебный объект



1 уже существующий класс
2. Клиентский интерфейс – способ работы клиента с другими объектами
3. Сервис – сторонний класс с нужной функциональностью, но неподдерживаемым интерфейсом
4. Адаптер — класс, преобразующий запросы от клиента в понятный формат для сервиса

Адаптер классов

Основывается на множественном наследовании.



Случаи применения

- При попытке использовать сторонние классы и библиотеки, с неподдерживаемым интерфейсом
- При необходимости использовать нескольких существующих подклассов без общей функциональности

Преимущества и недостатки

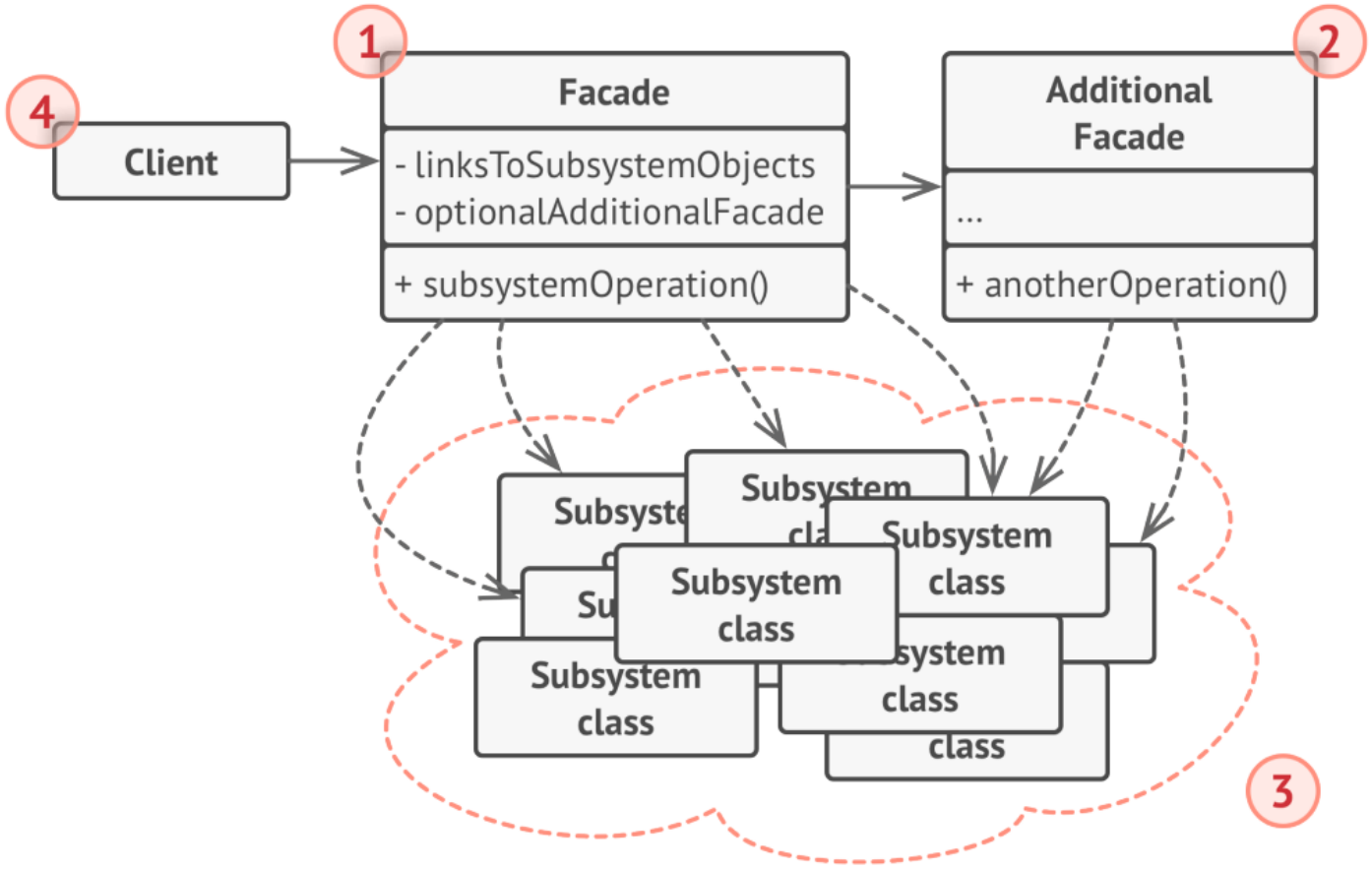
+ Скрывает особенности преобразования различных интерфейсов

- Усложняется код программы

Фасад

Предоставляет интерфейс к сложной системе классов, библиотеке или фреймворку

реализация



1. Класс Фасад – который дает доступ к определенной функциональности подсистемы

2. Дополнительный фасад – для разграничения доступа к разным типам функциональности

3. Подсистема – представляет сложную структуру классов, фреймворка, библиотеки и т..д.

4. Клиент – класс, реализующий работы с подсистемой через фасад

Случаи применения

Требуется упрощенный интерфейс при работе со сложной системой

Требуется разложить подсистему на отдельные функциональные блоки

Преимущества и недостатки

+ изоляция клиентов от компонентов системы
+ меньше зависимость между подсистемой и клиентами

- Может быть привязан ко всем классам программы