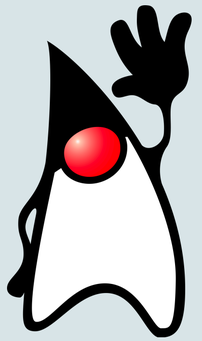
СБЕРБАНК

Корпоративный университет



Шаблоны. Поведенческие и структурные

Занятие №21

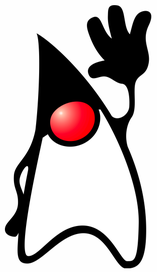
СБЕРБАНК
Корпоративный университет

Куценко Виталий

Маршрут

СБЕРБАНК
Корпоративный университет

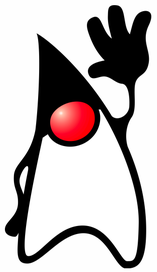
- Структурные шаблоны
- Поведенческие шаблоны



Договорённости

СБЕРБАНК
Корпоративный университет

- Активно участвуем. Не стесняйтесь задавать вопрос.
- Но off-topic обсуждаем в Telegram @sb_ku_java_2019_10
- Не стесняйтесь просто спрашивать в telegram.
-



Договорились?
Поехали!

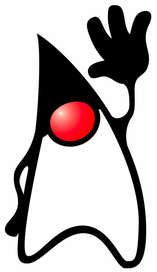
00

Вспоминаем зачем нам шаблоны

Что такое паттерн проектирования

СБЕРБАНК
Корпоративный университет

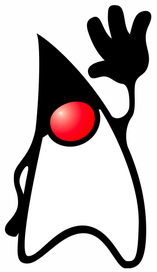
- Паттерн проектирования - типовой подход к решению типовых задач
- Формализованный опыт решения задач



Зачем нужны паттерны проектирования

СБЕРБАНК
Корпоративный университет

- Изучение опыта предыдущих поколений
- Упрощение процесса разработки
- Упрощение процесса сопровождения
- Упрощение коммуникации между разработчиками
- Прохождение «архитектурной» части интервью



Какие бывают виды паттернов

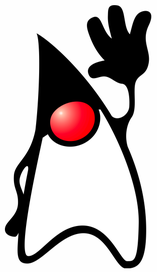
СБЕРБАНК
Корпоративный университет

- Виды паттернов GoF
  - Поведенческие
  - Структурные
  - Порождающие - смотрели раньше

- «Рядом» с классическими паттернами GoF находятся паттерны GRASP :
  общие шаблоны распределения ответственностей

- EIP, Шаблоны корпоративных приложении… это только только приоткрыли дверь в мир архитектуры

- А ещё бывают антипаттерны



01

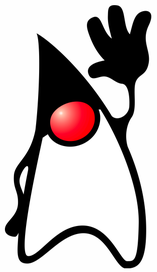
Поведенческие шаблоны

Список

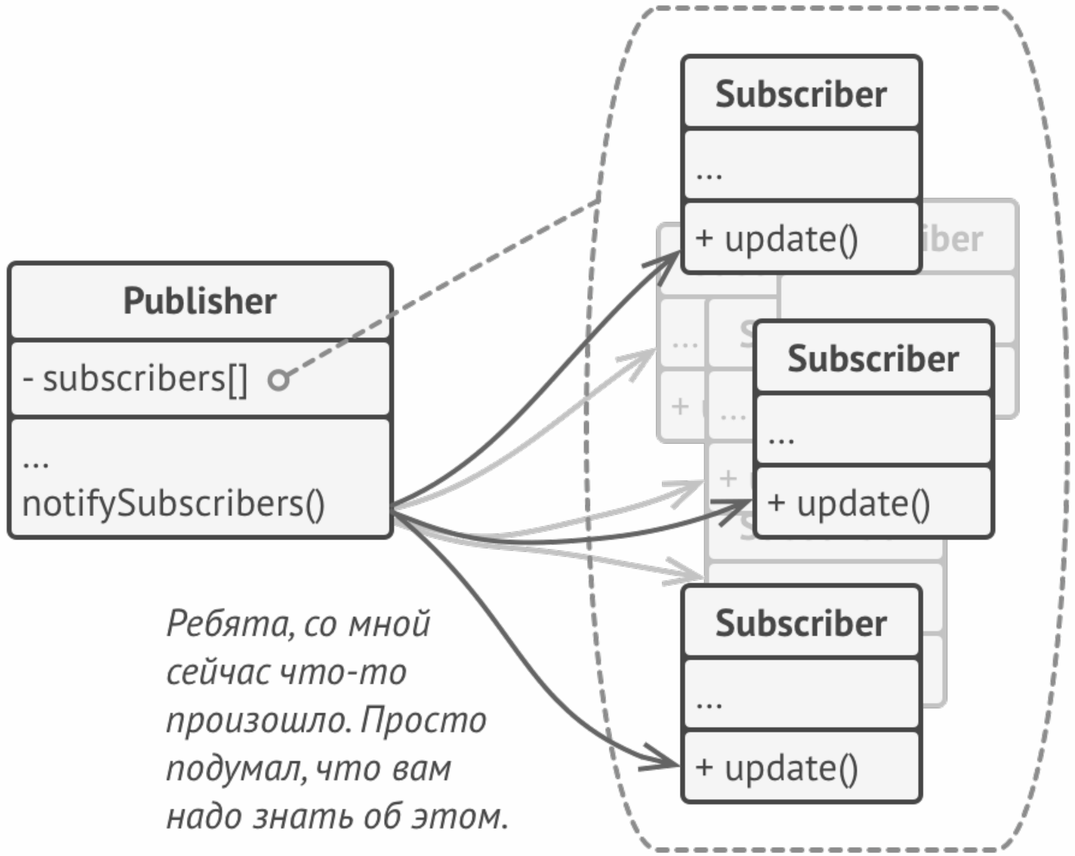
СБЕРБАНК
Корпоративный университет

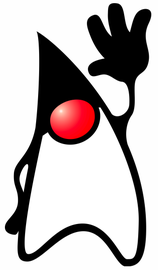
- Observer
- Command
- Chain of responsibility
- Memento
- State
- Strategy
- Visitor

https://refactoring.guru/ru/design-patterns/behavioral-patterns



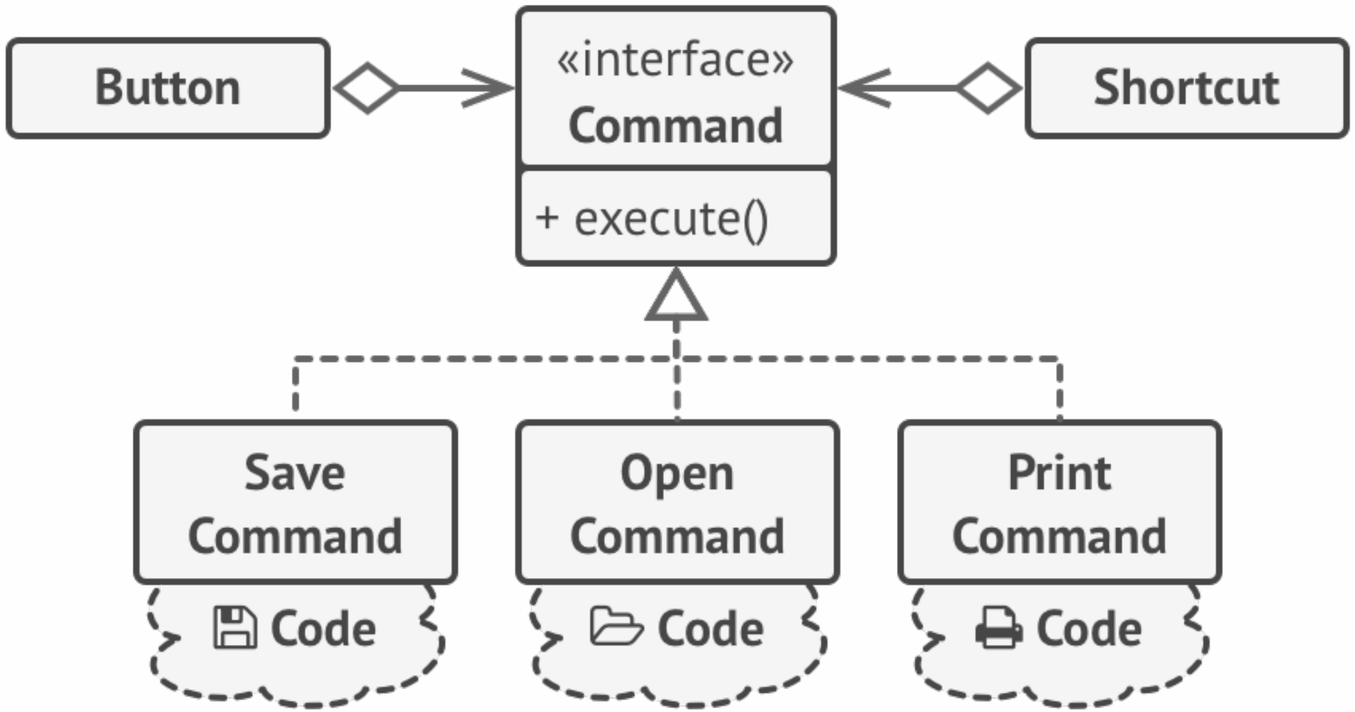
Типовая задача:
Взаимодействие «генератор событий - подписчики».
Реализация отношения «один-ко-многим»





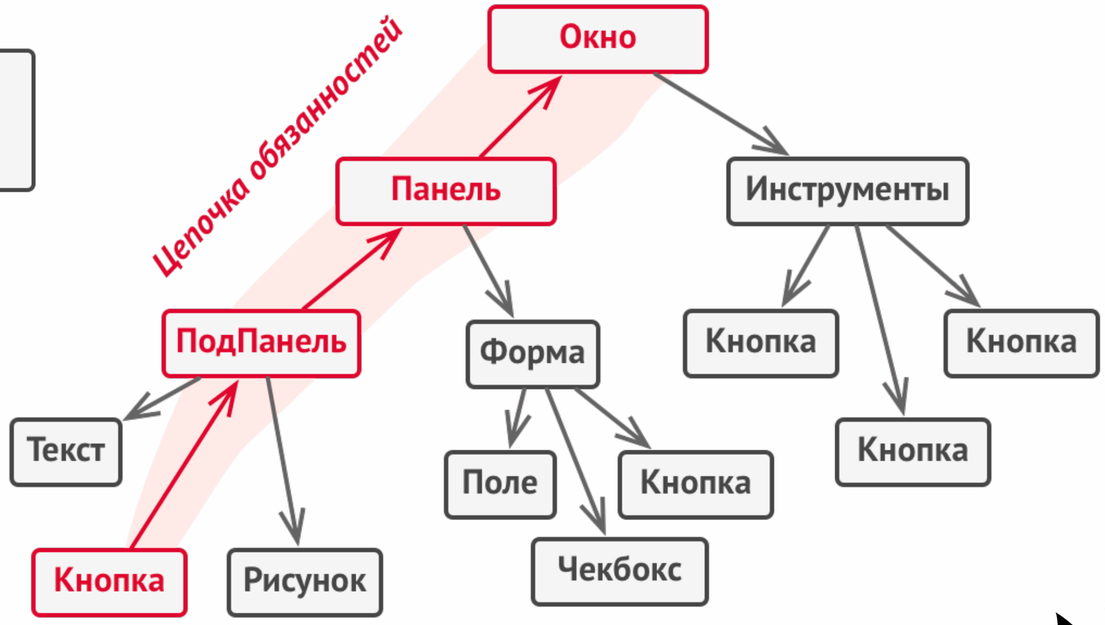
Command

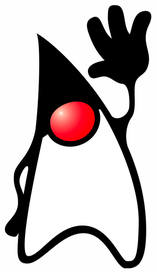
Типовая задача:
Логическое отделение объекта и обработчика этого объекта.
Типичный пример – очереди, обработчики сообщений.



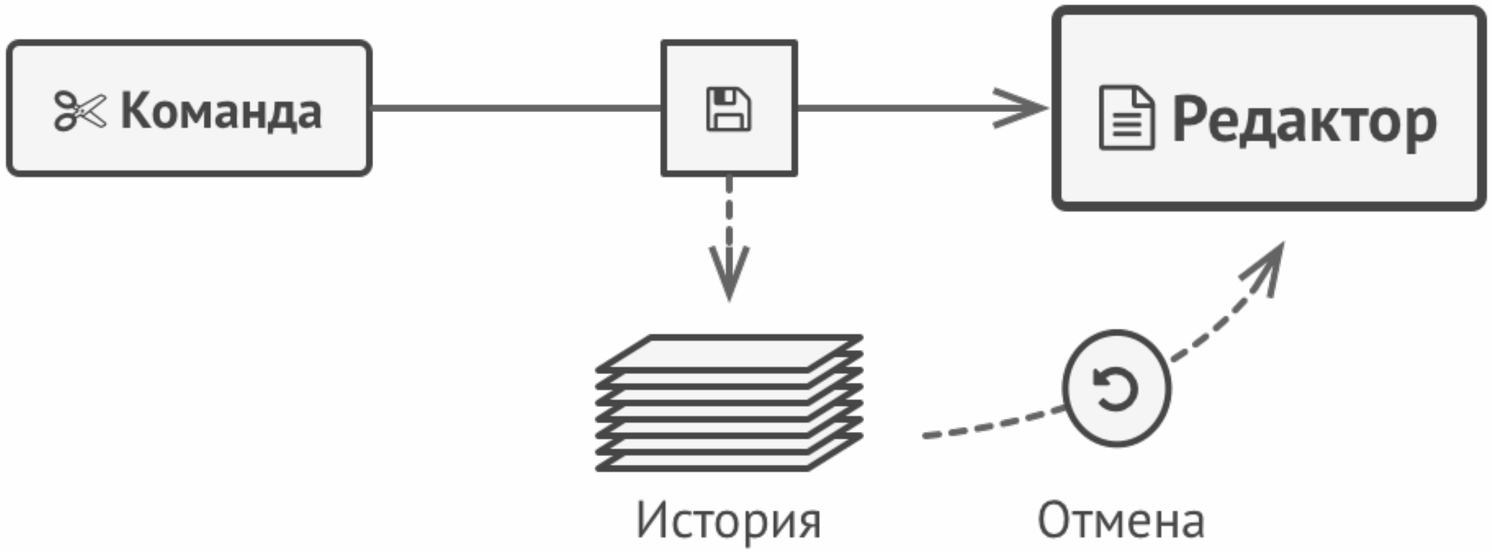
Chain of responsibility

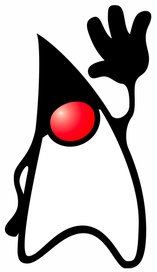
Типовая задача:
Последовательная обработка объекта цепочкой исполнителей.
Примеры: фильтры в сервлетах или «медкомиссия».



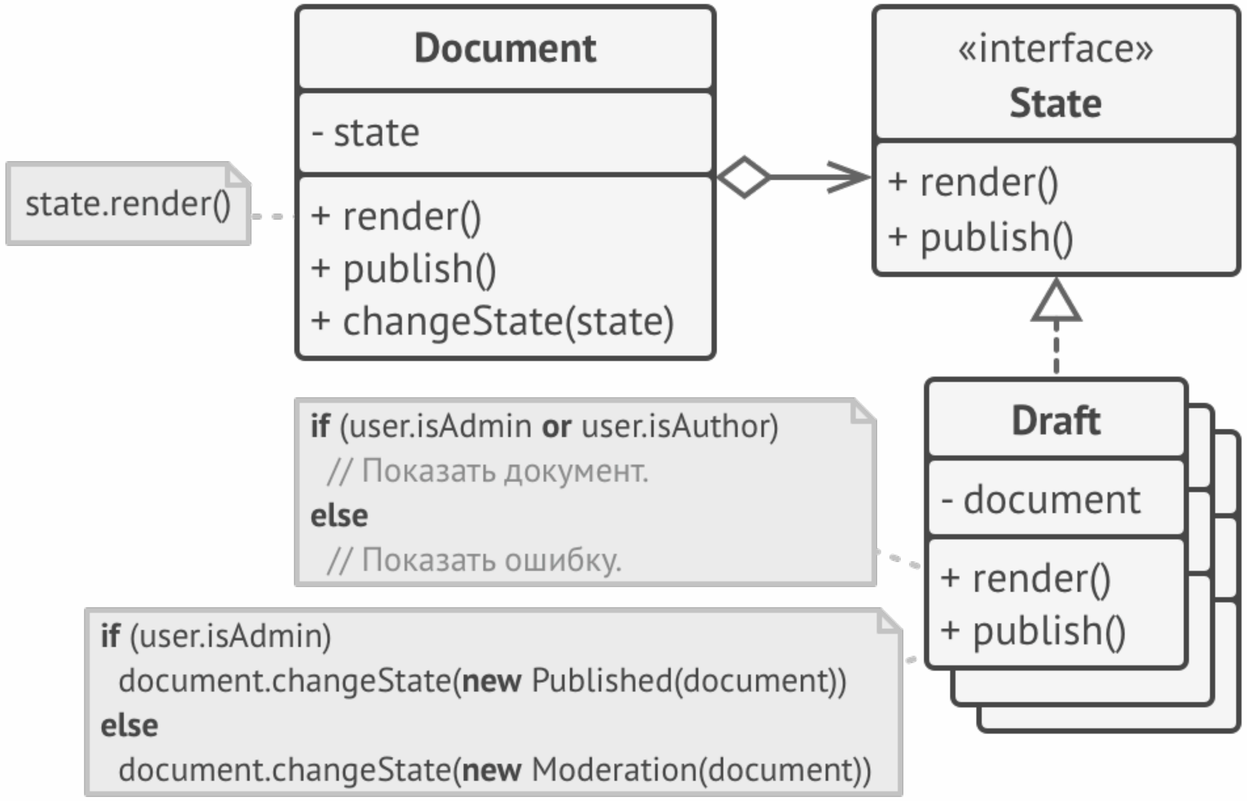


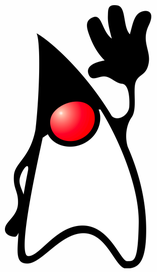
Типовая задача:
Реализация функционала «undo», «redo»
Пример: сохранение состояния системы и возврат к исходной точке.





Типовая задача:
Поведение объекта должно меняться в зависимости от его текущего состояния.
Пример: конечные автоматы.

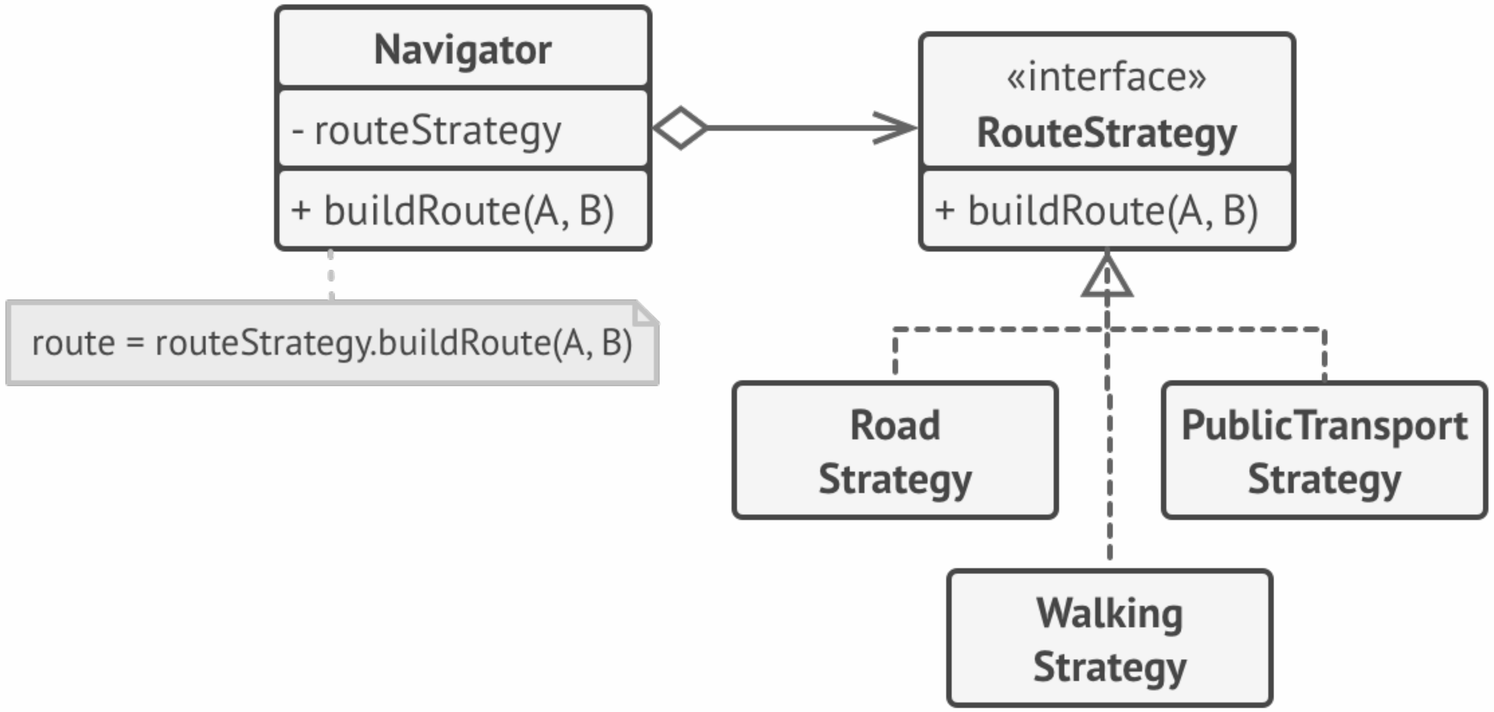


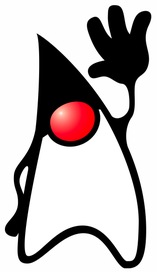


Strategy

СБЕРБАНК
Корпоративный университет

Типовая задача:
Обеспечение возможности изменять алгоритм обработки данных во время выполнения программы.
Пример (из sourcemaking): разные варианты как добраться до аэропорта.

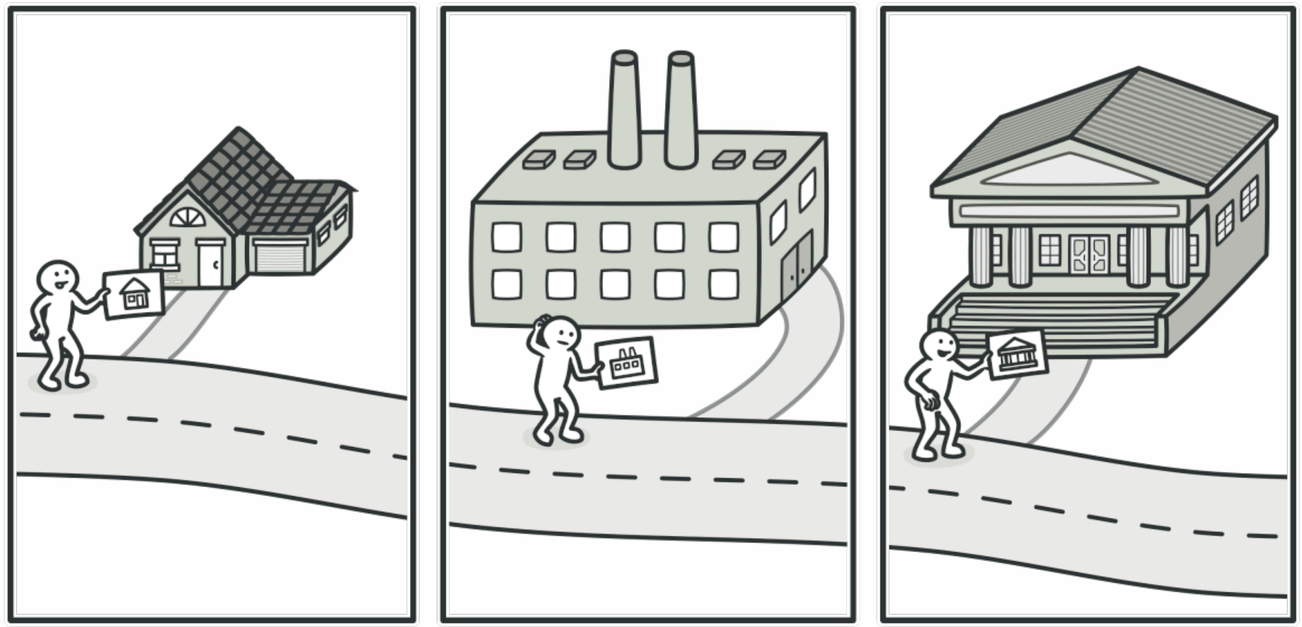


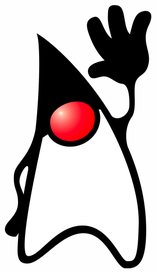


Visitor

СБЕРБАНК
Корпоративный университет

Типовая задача:
Надо выполнить операции над объектами – частями большого объекта. Причем логика операций отделена от частей.
Пример. Техобслуживание автомобиля. Надо выполнить сервисные действия над разными компонентами.





Ваши вопросы?

02

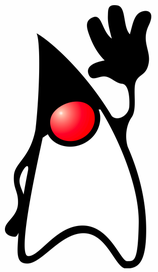
Структурные шаблоны

Список

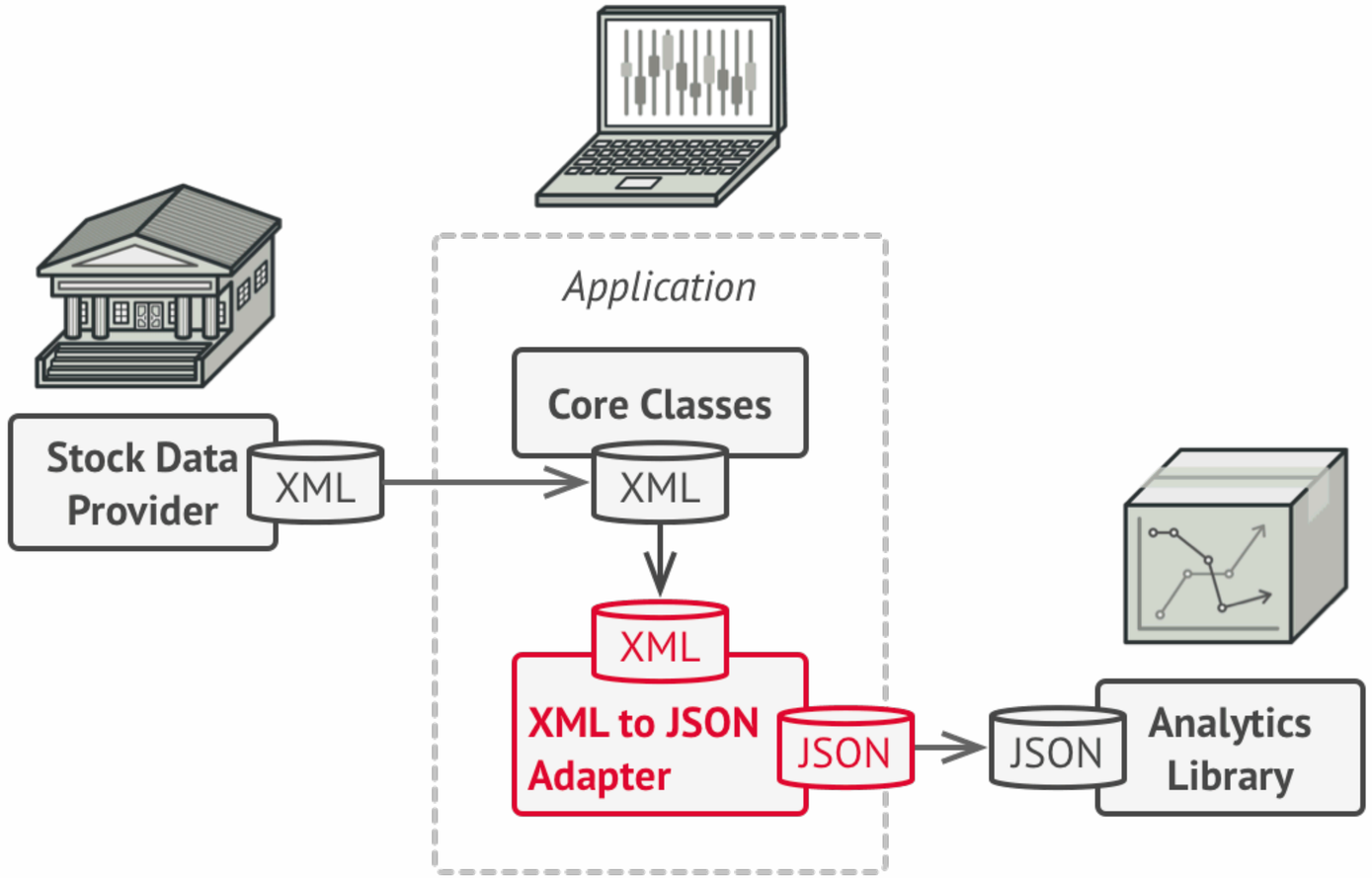
СБЕРБАНК
Корпоративный университет

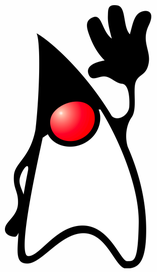
- Adapter
- Decorator
- Bridge
- Composite
- Facade
- Flyweight
- Proxy

https://refactoring.guru/ru/design-patterns/structural-patterns



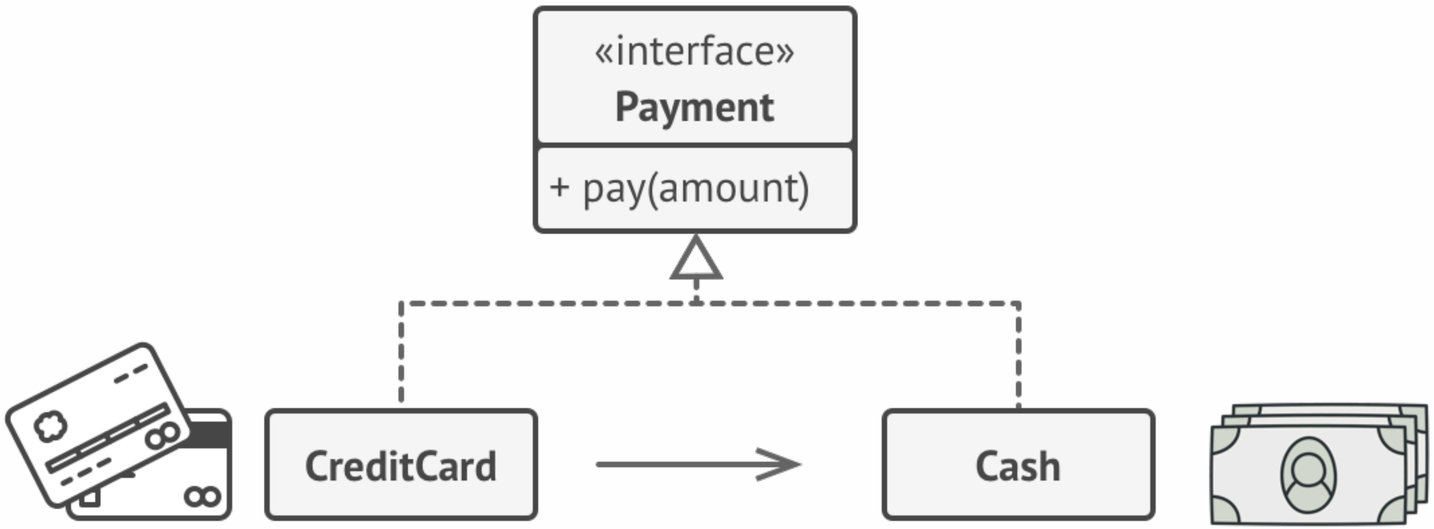
Типовая задача:
Преобразование интерфейса одного класса в интерфейс другого.

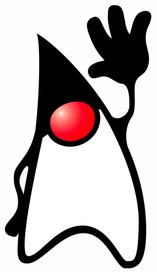




Proxy

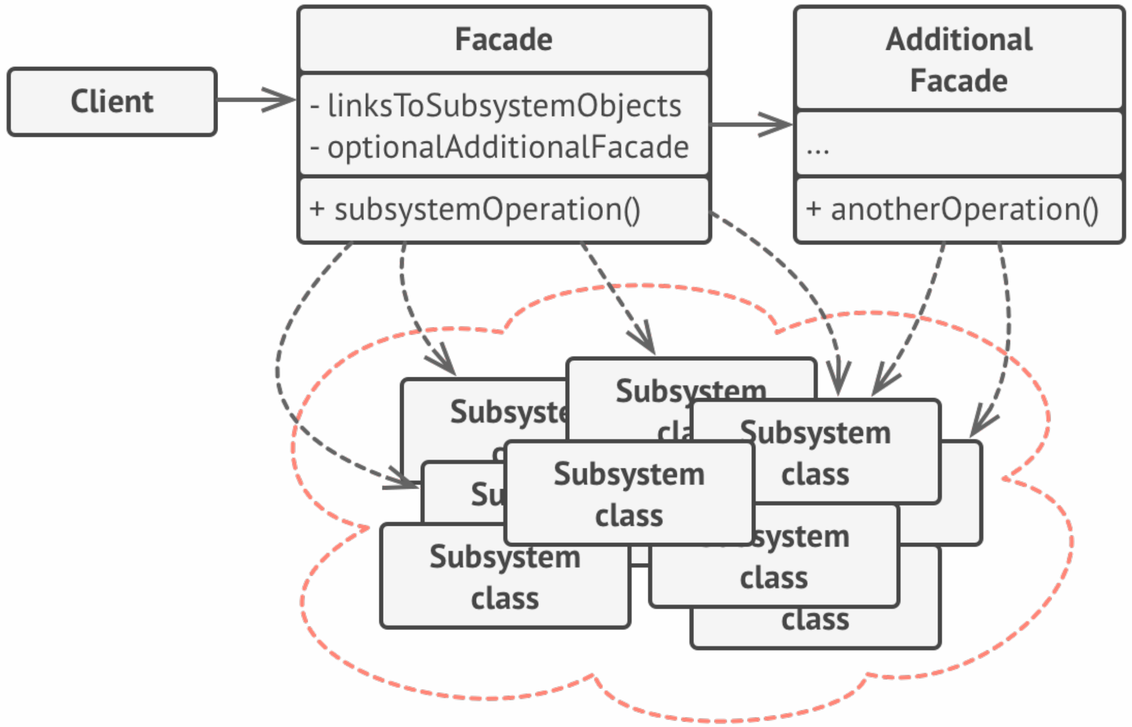
Типовая задача:
Применяется если надо основную функцию «обрамить» некими дополнительными действиями.
Примеры:
Логирование вызова функции.
«Ленивая» инициализация объектов.

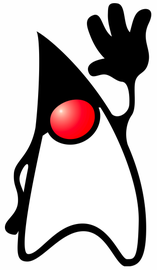




Facade

Типовая задача:
Надо предоставить общий (часто упрощенный) интерфейс к сложной системе или набору систем.
Пример: фасад логирования.

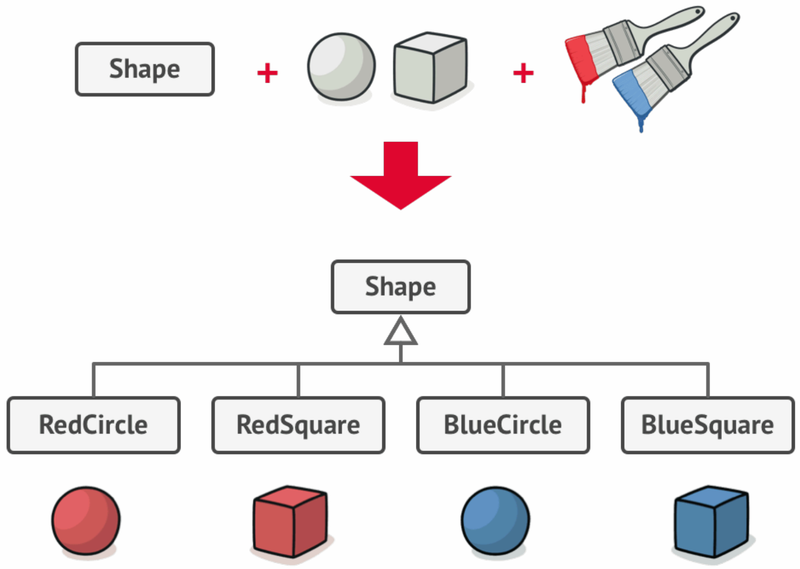


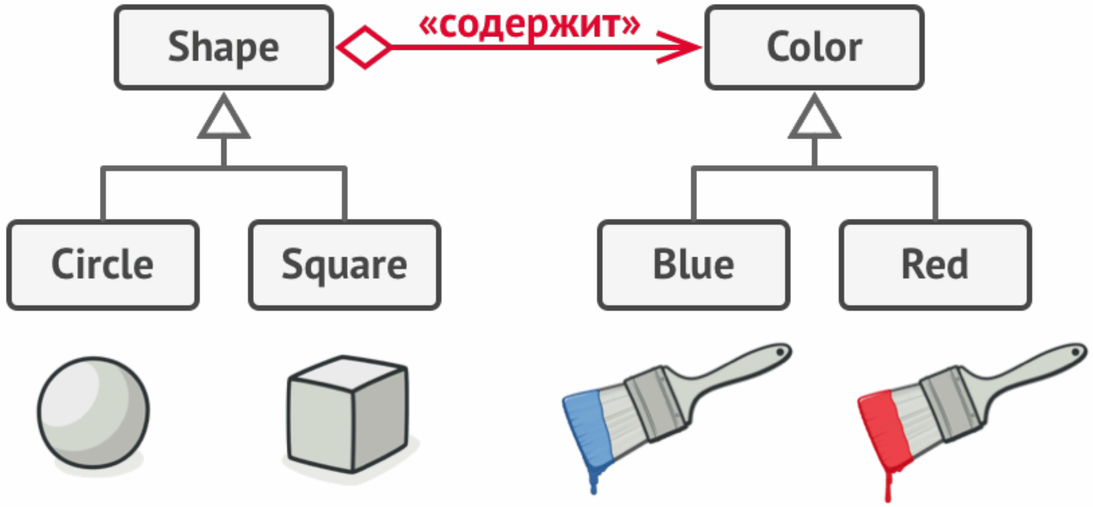


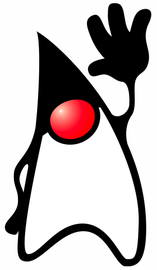
Bridge

СБЕРБАНК
Корпоративный университет

Типовая задача:
Применяется, когда надо иерархию интерфейсов отделить от иерархии имплементации.
Такое отделение требуется, если надо интерфейс и имплементацию менять независимо.
Применяется, если есть несколько плоскостей/критериев сущности.

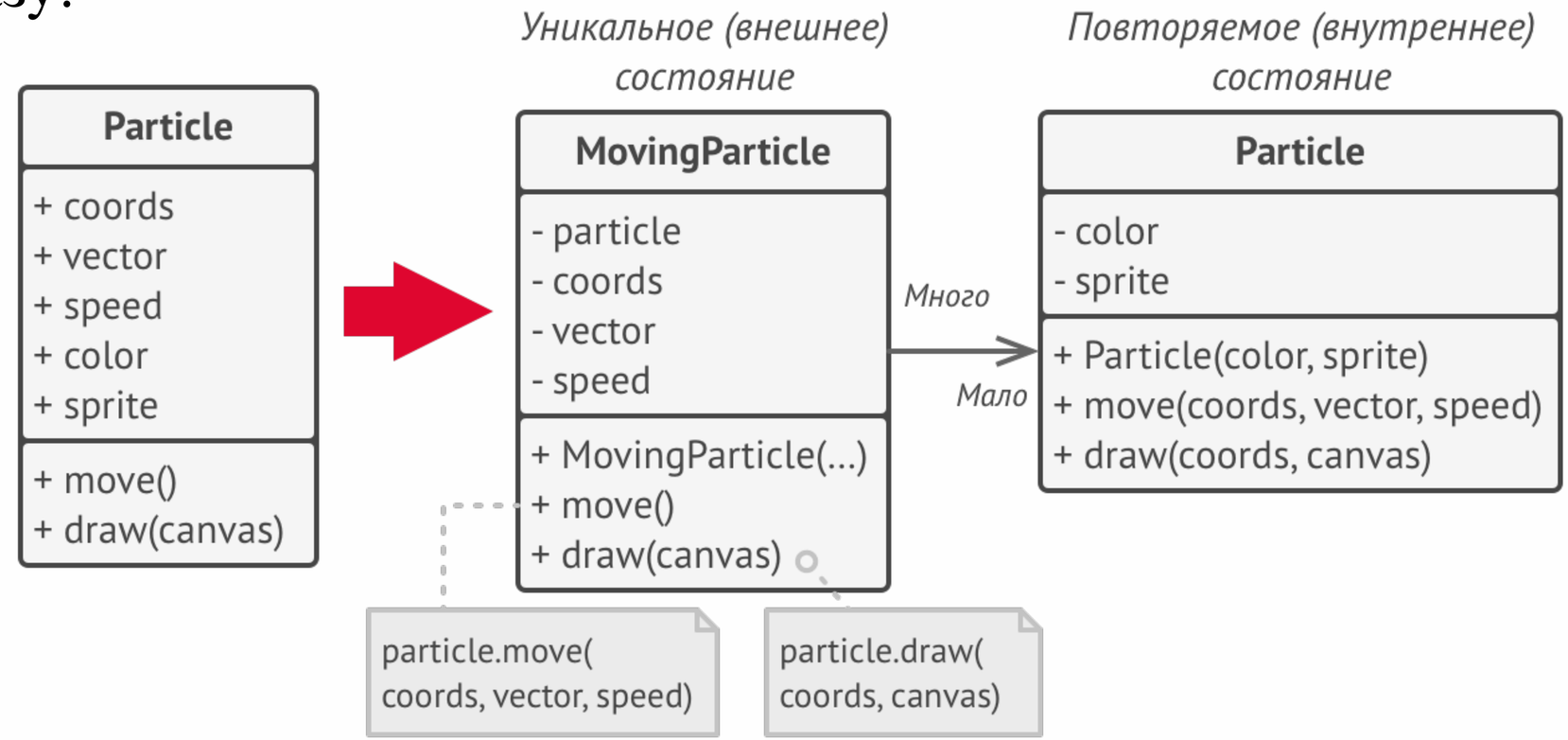






Flyweight

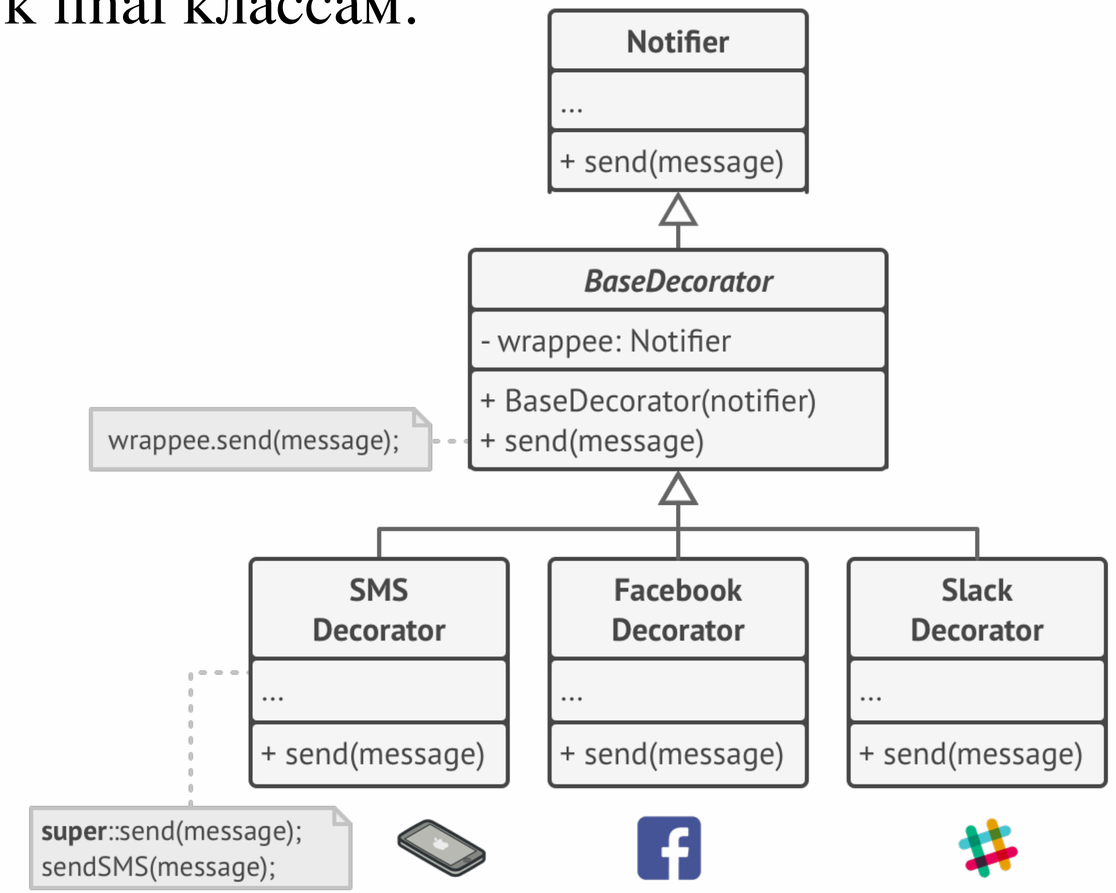
Типовая задача:
Есть группа тяжелых объектов. У всех объектов имеется одинаковая часть, причем это самая значительная часть объекта. Эту одинаковую часть можно вынести из всех объектов, а в самих объектах оставить только уникальные свойства.
Важно помнить, что общая часть не должна меняться. А если и меняться, то у всех объектов сразу.

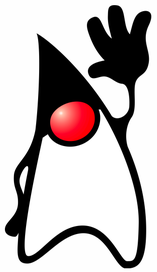


Decorator

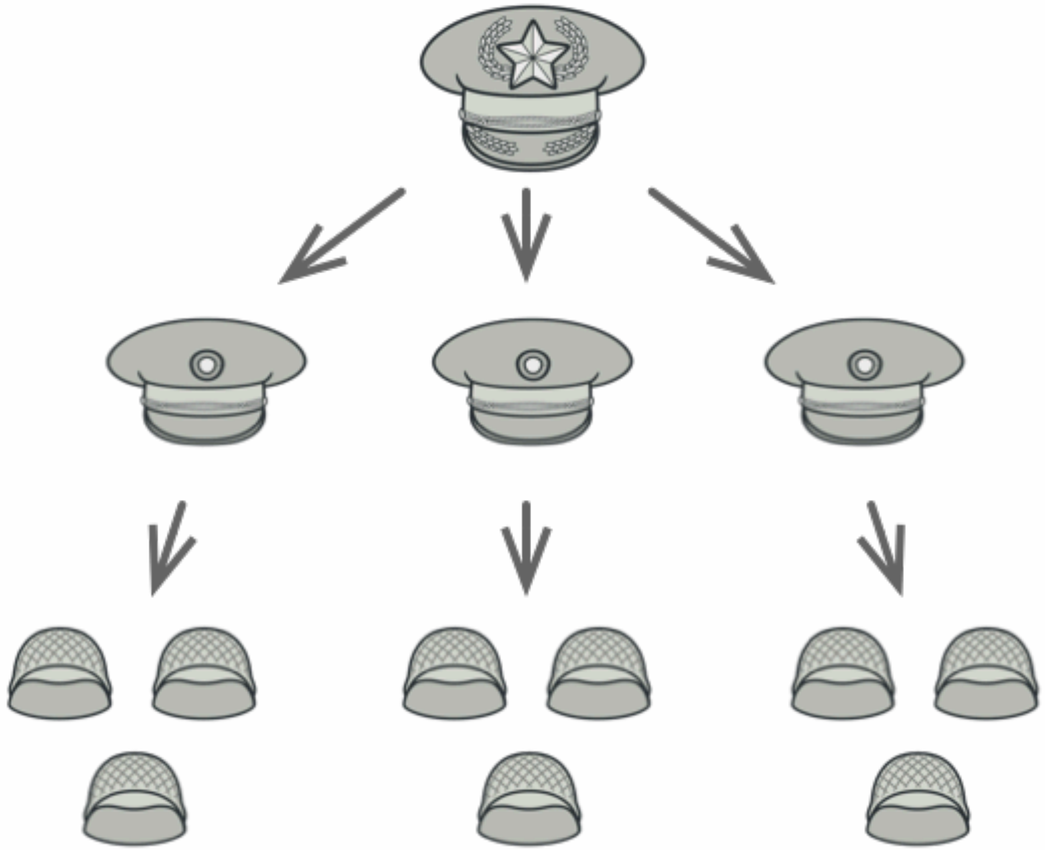
СБЕРБАНК
Корпоративный университет

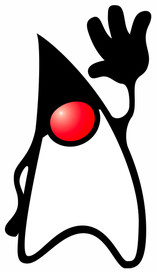
Типовая задача:
Динамическое добавление функционала во время выполнения программы с сохранением исходного интерфейса.
Особенно хорошо применять к final классам.





Типовая задача:
Позволяет объекты с общим интерфейсом объединить в группу и работать с этой группой как с одним объектом.





Ваши вопросы?

Спасибо за внимание!