Java Design Patterns

Command

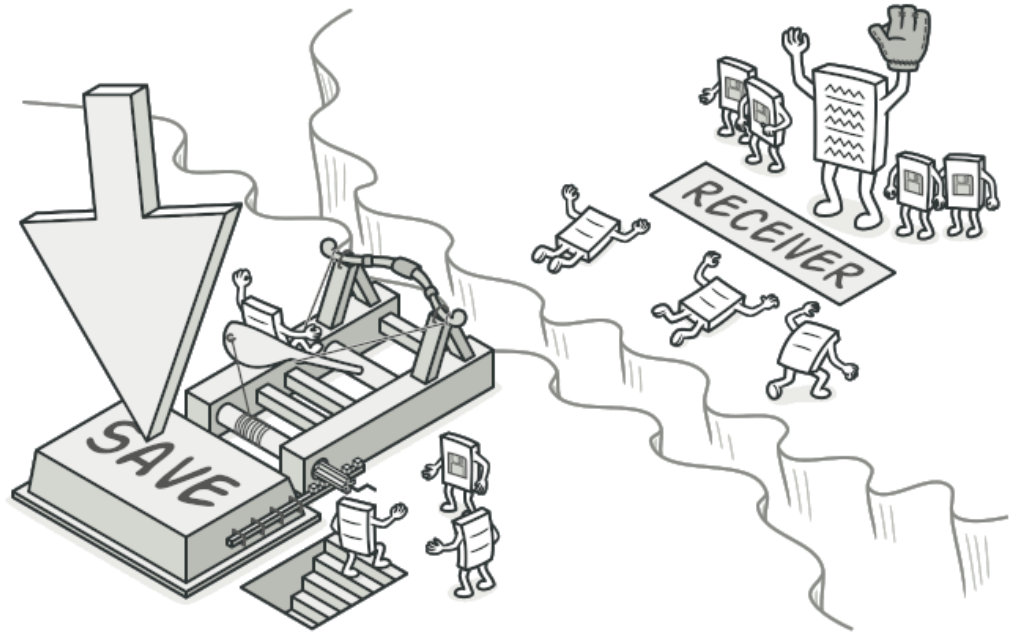
Тема

Command

Суть паттерна

Команда

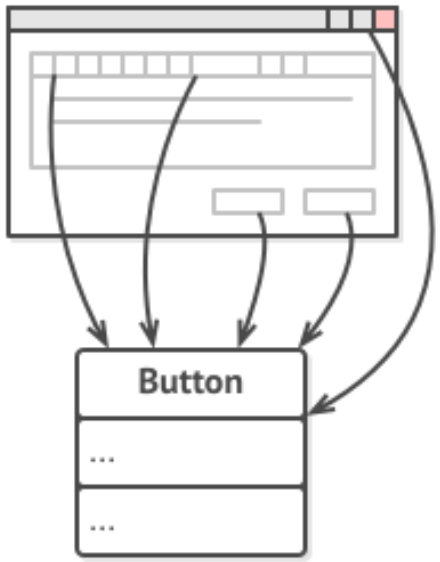
Команда — это поведенческий паттерн проектирования, который превращает запросы в объекты, позволяя передавать их как аргументы при вызове методов, ставить запросы в очередь, логировать их, а также поддерживать отмену операций.

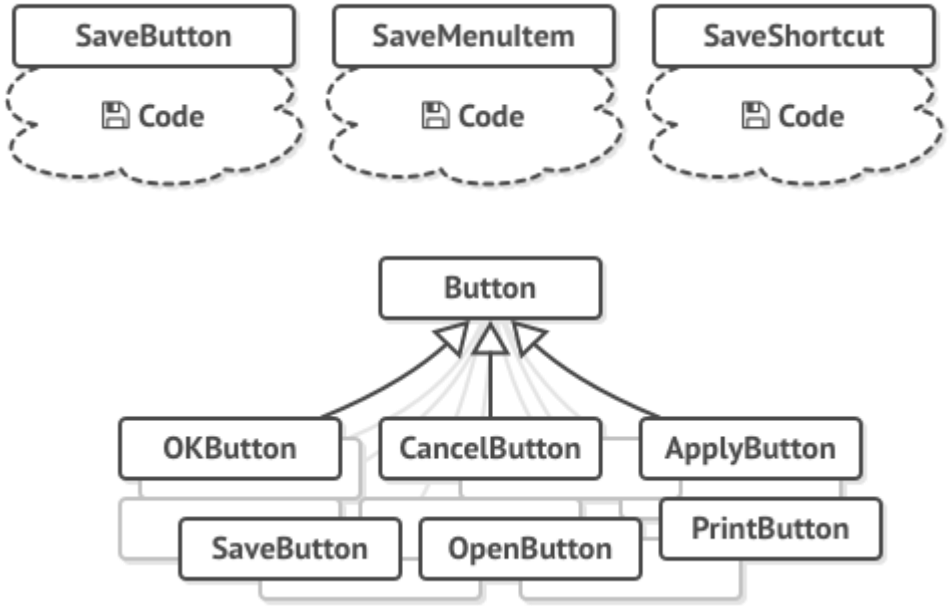


Проблема

Постановка задачи

Представьте, что вы работаете над программой текстового редактора. Дело как раз подошло к разработке панели управления. Вы создали класс красивых Кнопок и хотите использовать его для всех кнопок приложения начиная от панели управления, заканчивая простыми кнопками в диалогах.

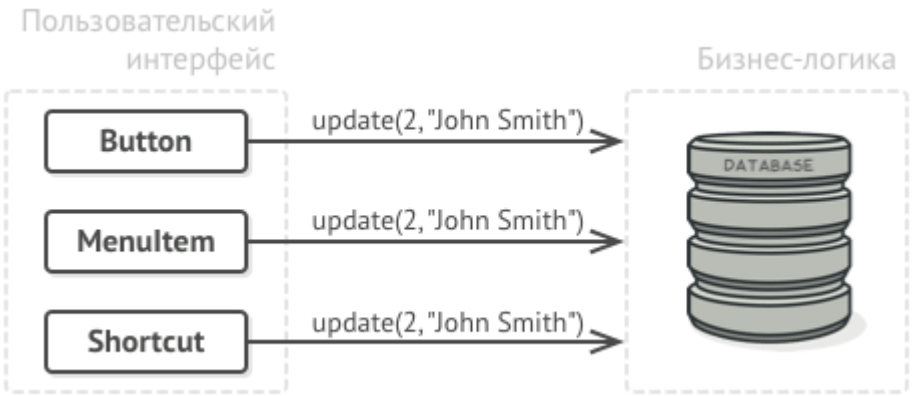


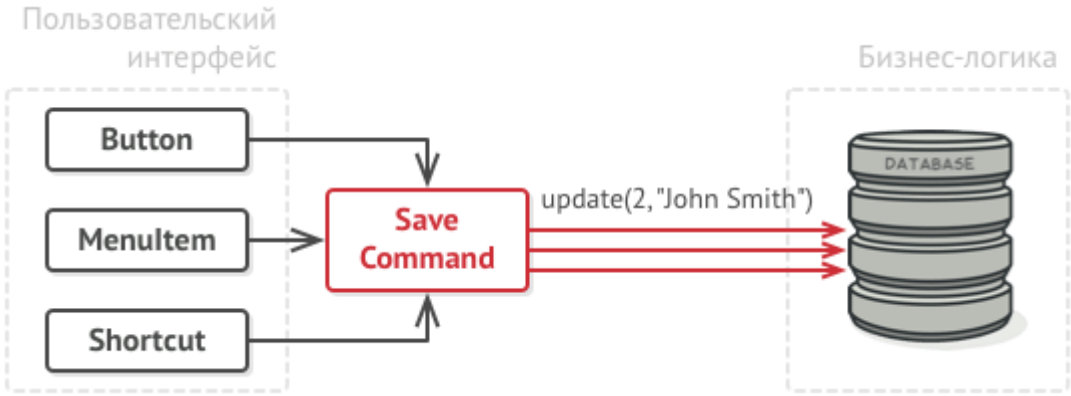


Решение

Решение задачи

Хорошие программы обычно структурированы в виде слоёв. Самый распространённый пример — слои интерфейса и бизнес-логики. Первый всего лишь рисует красивую картинку для пользователя. Но когда нужно сделать что-то важное, интерфейс «просит» слой бизнес-логики заняться этим.

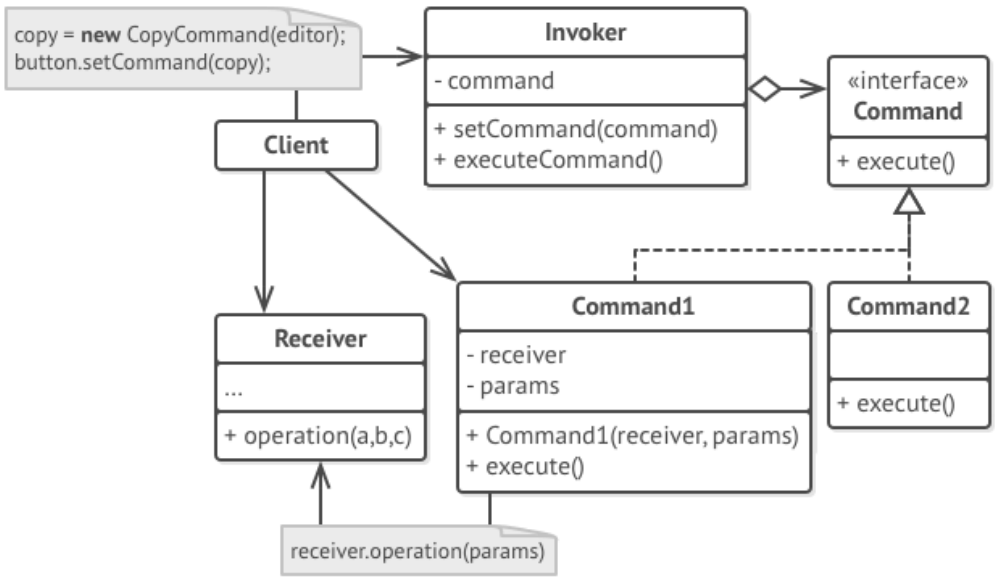




Структура

Структура паттерна

1. Отправитель хранит ссылку на объект команды и обращается к нему, когда нужно выполнить какое-то действие. Отправитель работает с командами только через их общий интерфейс. Он не знает, какую конкретно команду использует, так как получает готовый объект команды от клиента.
2. Команда описывает общий для всех конкретных команд интерфейс. Обычно, здесь описан всего один метод для запуска команды.
3. Конкретные команды реализуют различные запросы, следуя общему интерфейсу команд. Обычно, команда не делает всю работу самостоятельно, а лишь передаёт вызов получателю — определённому объекту бизнес-логики.
4. Получатель содержит бизнес-логику программы. В этой роли может выступать практически любой объект. Обычно, команды перенаправляют вызовы получателям. Но иногда, чтобы упростить программу, вы можете избавиться от получателей, слив их код в классы команд.
5. Клиент создаёт объекты конкретных команд, передавая в них все необходимые параметры, а иногда и ссылки на объекты получателей. После этого, клиент конфигурирует отправителей созданными командами.



Применимость

Применение паттерна

1. Когда вы хотите параметризовать объекты выполняемым действием.
2. Когда вы хотите ставить операции в очередь, выполнять их по расписанию или передавать по сети.
3. Когда вам нужна операция отмены.

Шаги реализации

Алгоритм реализации паттерна

1. Создайте общий интерфейс команд и определите в нём метод запуска.
2. Один за другим создайте классы конкретных команд. В каждом классе должно быть поле для хранения ссылки на один или несколько объектов-получателей, которым команда будет перенаправлять основную работу.
3. Добавьте в классы отправителей поля для хранения команд. Объект-отправитель должен принимать готовый объект команды извне через конструктор, либо через сеттер команды.
4. Измените основной код отправителей так, чтобы они делегировали выполнение действия команде.
5. Порядок инициализации объектов должен выглядеть так:
   - Создаём объекты получателей.
   - Создаём объекты команд, связав их с получателями.
   - Создаём объекты отправителей, связав их с командами.

Преимущества и недостатки

Плюсы и недостатки

Плюсы:

- Убирает прямую зависимость между объектами, вызывающими операции и объектами, которые их непосредственно выполняют.
- Позволяет реализовать простую отмену и повтор операций.
- Позволяет реализовать отложенный запуск команд.
- Позволяет собирать сложные команды из простых.
- Соблюдает *принцип открытости/закрытости*.

Минусы:

- Усложняет код программы за счёт дополнительных классов.

Отношение с другими паттернами

- Обработчики в Цепочке обязанностей могут быть выполнены в виде Команд. В этом случае множество разных операций может быть выполнено над одним и тем же контекстом, коим является запрос.
- Команду и Снимок можно использовать сообща для реализации отмены операций. В этом случае объекты команд будут отображать выполненные действие над объектом, снимки — хранить копию состояния этого объекта до того, как команда была выполнена.
- Если Команду нужно копировать перед вставкой в историю выполненных команд, вам может помочь Прототип.
- Посетитель это более мощный аналог Команды, которую можно выполнить сразу над объектами нескольких классов.

Информационный видеосервис для разработчиков программного обеспечения

ITVDN

IT VIDEO DEVELOPERS NETWORK

TestProvider PROMETRIC TEST CENTER CyberBionic systematics Microsoft Partner Silver Learning Windows Azure Cloud Partner EBA European Business Association