Java Design Patterns

Mediator

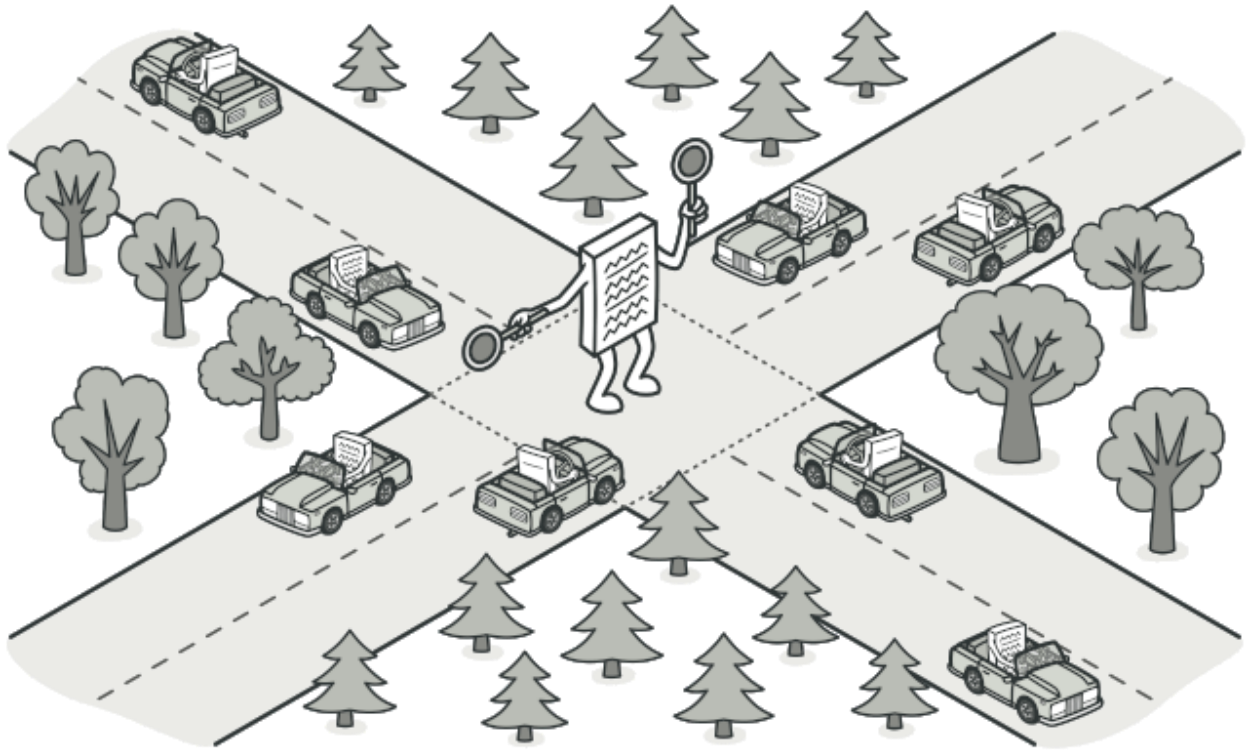
Тема

Mediator

Суть паттерна

Посредник

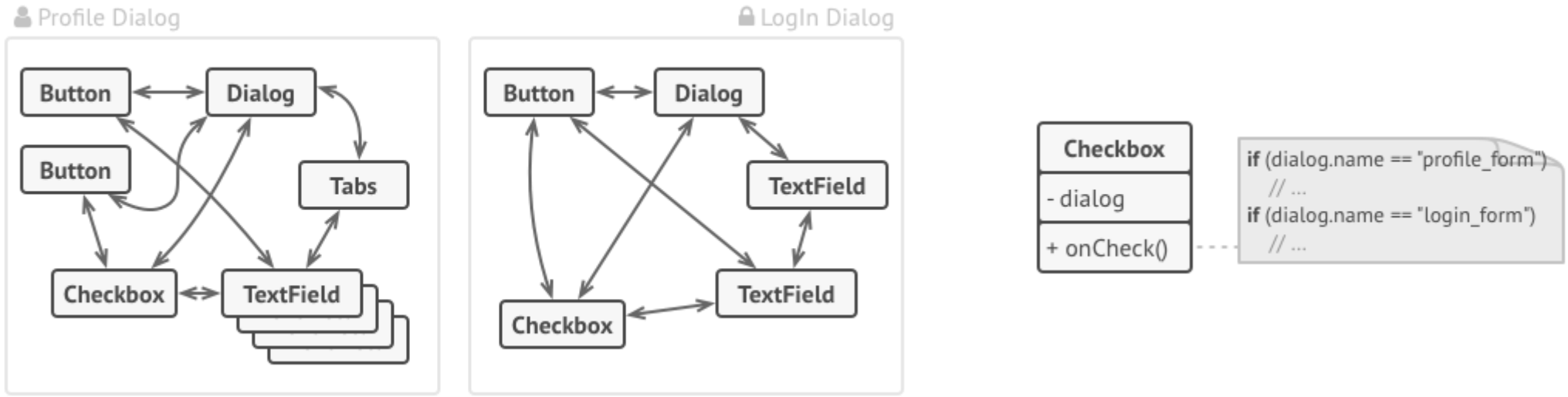
Посредник — это поведенческий паттерн проектирования, который позволяет уменьшить связанность множества классов между собой, благодаря перемещению этих связей в один класс-посредник.



Проблема

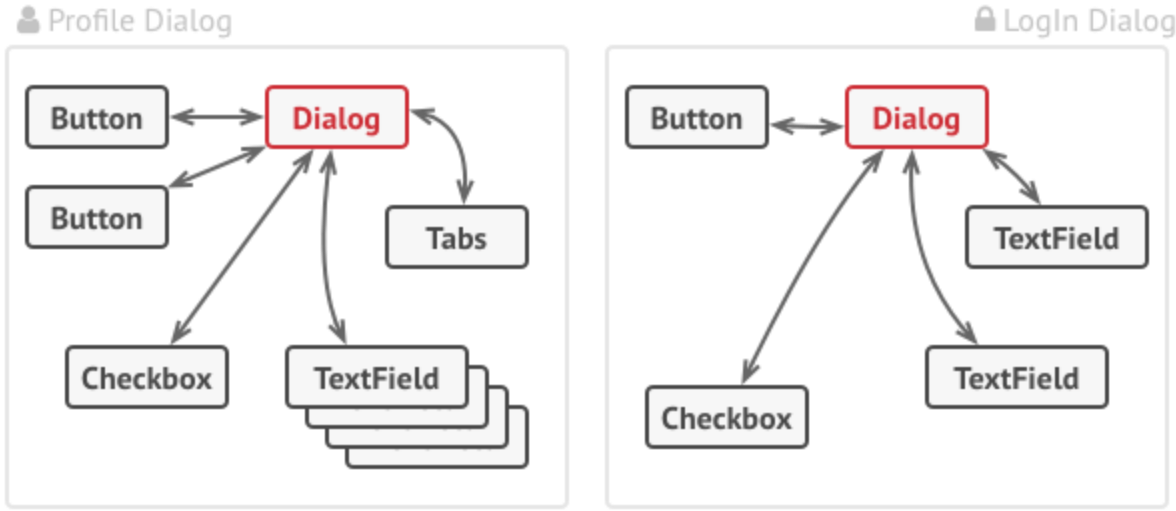
Постановка задачи

Предположим, что у вас есть диалог создания профиля пользователя. Он состоит из всевозможных элементов управления — текстовых полей, чекбоксов, кнопок.



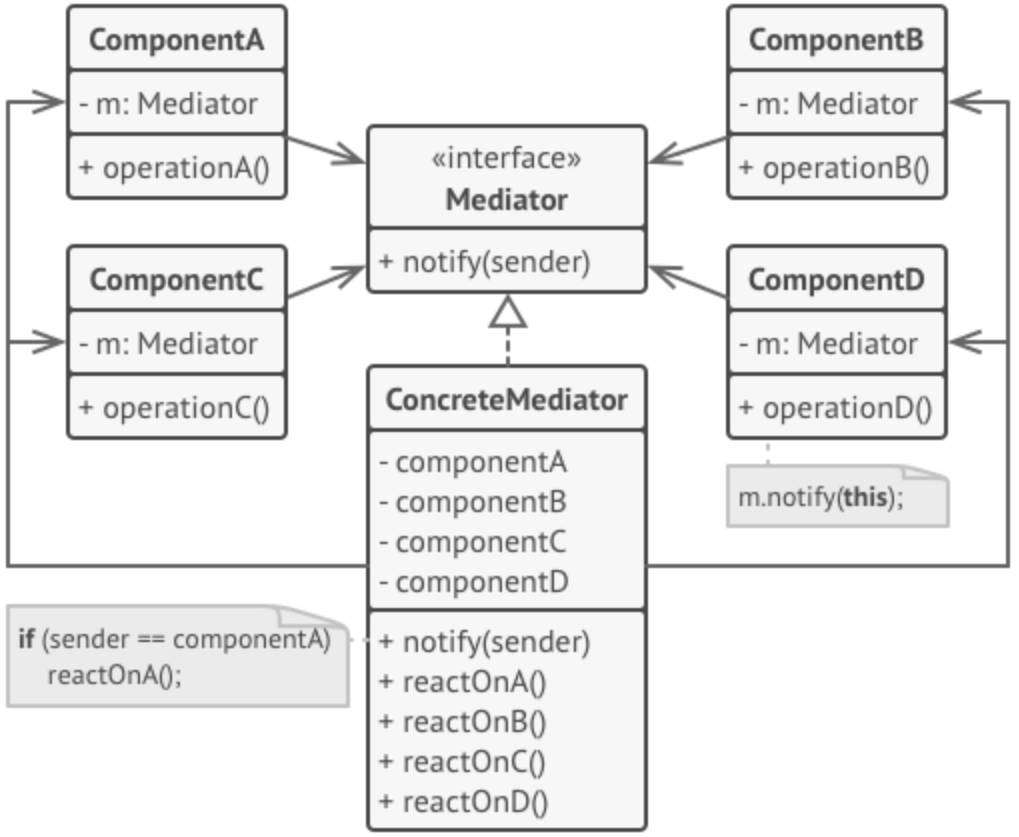
Решение задачи

Паттерн Посредник заставляет объекты общаться не напрямую друг с другом, а через отдельный объект-посредник, который знает, кому нужно перенаправить тот или иной запрос. Благодаря этому, компоненты системы будут зависеть только от посредника, а не от десятков других компонентов.



Структура паттерна

1. Компоненты — это разнородные объекты, содержащие бизнес-логику программы. Каждый компонент хранит ссылку на объект посредника, но работает с ним только через абстрактный интерфейс посредников.
2. Посредник определяет интерфейс для обмена информацией с компонентами.
3. Компоненты не должны общаться напрямую друг с другом. Если в компоненте происходит важное событие, влияющее на других, он должен оповестить своего посредника.
4. Конкретный посредник содержит код взаимодействия нескольких компонентов между собой. Этот объект создаёт и хранит ссылки на компоненты системы.



Применимость

Применение паттерна

1. Когда вам сложно менять некоторые классы из-за множества хаотичных связей с другими классами.
2. Когда вы не можете повторно использовать класс, поскольку он зависит от уймы других классов.
3. Когда вам приходится создавать множество подклассов компонентов, чтобы использовать одни и те же компоненты в разных контекстах.

Шаги реализации

Алгоритм реализации паттерна

1. Найдите группу тесно переплетённых классов, отвязав которые друг от друга, можно получить некоторую пользу. Например, чтобы повторно использовать их код в другой программе.
2. Создайте общий интерфейс *Посредников* и опишите в нём методы для взаимодействия с *Компонентами*. В простейшем случае достаточно одного метода для получения оповещений от компонентов.
3. Реализуйте этот интерфейс в классе *Конкретного посредника*. Поместите в него поля, которые будут содержать ссылки на все объекты компонентов.
4. Вы можете пойти дальше и переместить код создания компонентов в класс *Конкретного посредника*, превратив его в фабрику.
5. Компоненты тоже должны иметь ссылку на объект посредника. Связь между ними удобней всего установить, подавая посредника в параметры конструктора компонентов.
6. Измените код компонентов так, чтобы они вызывали метод оповещения посредника, а не методы других компонентов. С другой стороны, посредник должен вызывать методы нужного компонента, когда получает оповещение.

Преимущества и недостатки

Плюсы и недостатки

Плюсы:

- Устраняет зависимости между компонентами, позволяя повторно их использовать.
- Упрощает взаимодействие между компонентами.
- Централизует управление в одном месте.

Минусы:

- Посредник может сильно раздуться.

Отношение с другими паттернами

- Цепочка обязанностей, Команда, Посредник и Наблюдатель показывают различные способы работы отправителей запросов с их получателями:
- *Цепочка обязанностей* передаёт запрос последовательно через цепочку потенциальных получателей, ожидая, что какой-то из них обработает запрос.
- *Команда* устанавливает косвенную одностороннюю связь от отправителей к получателям.
- *Посредник* убирает прямую связь между отправителями и получателями, заставляя их общаться опосредованно, через себя.
- *Наблюдатель* передаёт запрос одновременно всем заинтересованным получателям, но позволяет им динамически подписывать или отписываться от таких оповещений.
- Посредник и Фасад похожи тем, что пытаются организовать работу множества существующих классов.
- Разница между Посредником и Наблюдателем не всегда очевидна. Чаще всего они выступают как конкуренты, но иногда могут работать вместе.

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