Пермский филиал федерального государственного автономного
образовательного учреждения высшего образования
«Национальный исследовательский университет
«Высшая школа экономики»

Факультет экономики, менеджмента и бизнес-информатики

Рязанов Иван Дмитриевич

ОРГАНИЗАЦИЯ ПАТТЕРНОВ ПРОЕКТИРОВАНИЯ. СТРУКТУРНЫЙ ПАТТЕРН «КОМПОНОВЩИК»

Лабораторная работа

студента образовательной программы «Программная инженерия»
по направлению подготовки *09.03.04 Программная инженерия*

Руководитель
к.т.н., доцент кафедры Информационных технологий в бизнесе НИУ ВШЭ-Пермь

А.В. Кычкин

Пермь, 2020 год

Оглавление

Глава 1. Паттерн «Компоновщик» **3**
Глава 2. Проектирование и реализация **7**
2.1 Проектирование 7
2.2 Реализация 7

Глава 1. Паттерн «Компоновщик»

Название и классификация паттерна. Компоновщик - паттерн, структурирующий объекты.

Назначение. Компонует объекты в древовидные структуры для представления иерархий часть-целое. Позволяет клиентам единообразно трактовать индивидуальные и составные объекты. Управление группами объектов может быть непростой задачей, особенно, если эти объекты содержат собственные объекты. Паттерн компоновщик описывает, как можно применить рекурсивную композицию таким образом, что клиенту не придется проводить различие между простыми и составными объектами.

Применимость. Использование паттерна Компоновщик целесообразно если:

1. Необходимо объединять группы схожих объектов и управлять ими.

2. Объекты могут быть как примитивными (элементарными), так и составными (сложными). Составной объект может включать в себя коллекции других объектов, образуя сложные древовидные структуры. Пример: директория файловой системы состоит из элементов, каждый их которых также может быть директорией.

3. Код клиента работает с примитивными и составными объектами единообразно.

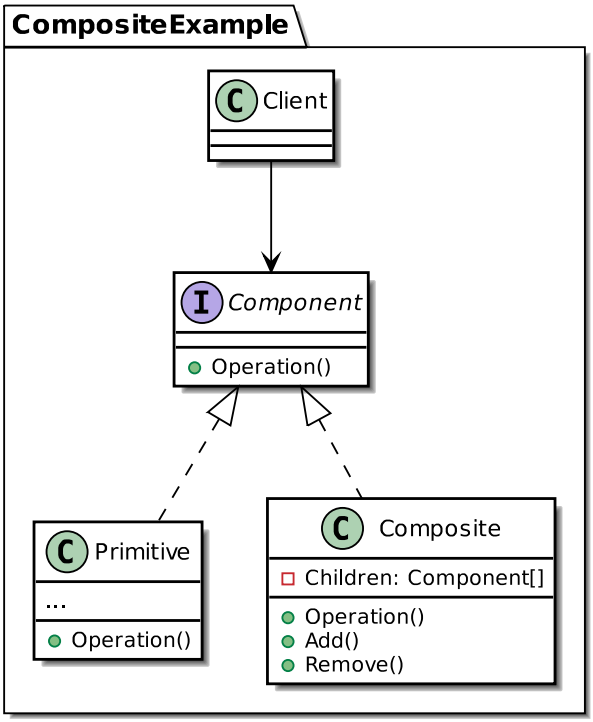

Рис. 1.1. Диаграмма классов паттерна «Компоновщик»

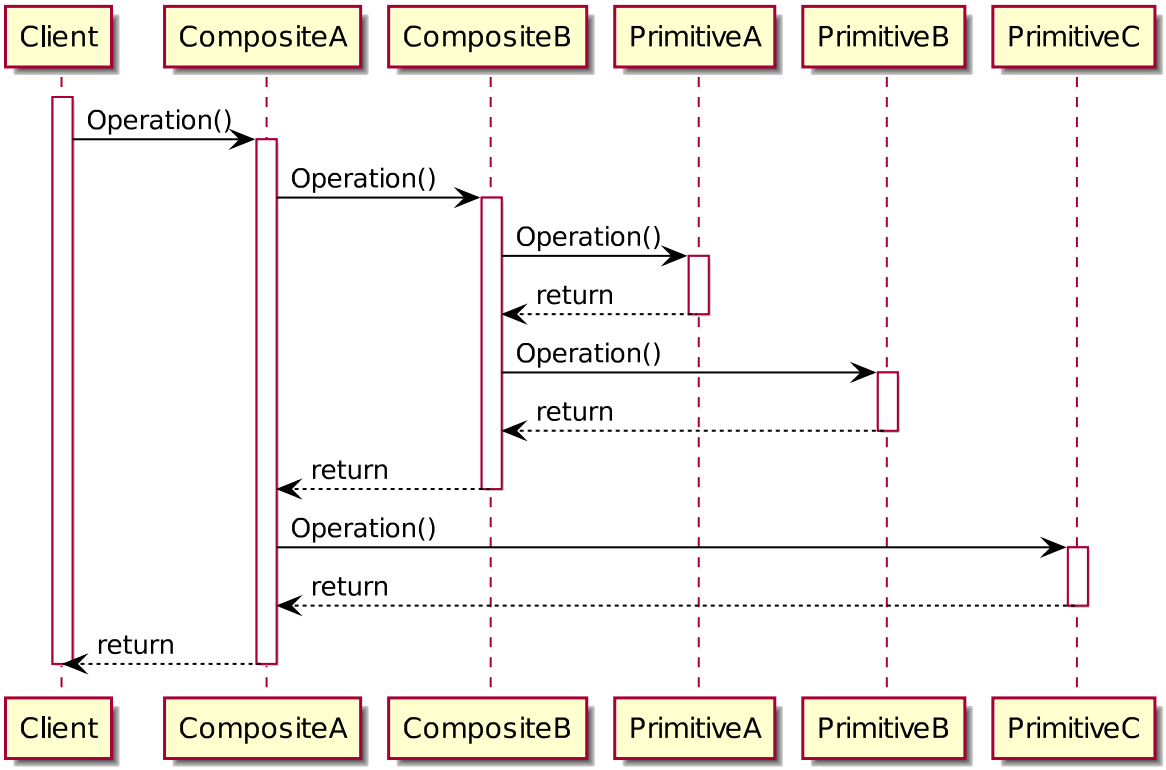

Рис. 1.2. Диаграмма последовательности паттерна «Компоновщик»

Участники

1. Component — компонент: - объявляет интерфейс для компонуемых объектов; предоставляет подходящую реализацию операций по умолчанию, общую для всех классов; объявляет интерфейс для доступа к потомкам и управления ими; определяет интерфейс для доступа к родителю компонента в рекурсивной структуре и при необходимости реализует его. Описанная возможность необязательна;

2. Primitive (PrimitiveA, PrimitiveB, PrimitiveC) — примитив: - представляет листовые узлы композиции и не имеет потомков; определяет поведение примитивных объектов в композиции;

3. Composite (CompositeA, CompositeB) — составной объект: определяет поведение компонентов, у которых есть потомки; хранит компоненты-потомки; реализует относящиеся к управлению потомками операции в интерфейсе класса Component;

4. Client — клиент: - манипулирует объектами композиции через интерфейс Component.

Отношения

Клиенты используют интерфейс класса Component для взаимодействия с объектами в составной структуре. Если получателем запроса является дочерний объект Primitive, то он и обрабатывает запрос. Когда же получателем является составной объект Composite, то обычно он перенаправляет запрос своим потомкам, возможно, выполняя некоторые дополнительные операции до или после перенаправления.

Плюсы и минусы

Плюсы:

1. В систему легко добавлять новые примитивные или составные объекты, так как паттерн Composite использует общий базовый класс Component;

2. Код клиента имеет простую структуру – примитивные и составные объекты обрабатываются одинаковым образом;

3. Паттерн Composite позволяет легко обойти все узлы древовидной структуры.

Минусы:

1. Неудобно осуществить запрет на добавление в составной объект Composite объектов определенных типов. Так, например, в состав Умного дома не могут входить станки с числовых программным управлением (ЧПУ).

Области применения

1. Приложение, которое требует базовой функциональности во всей своей иерархической структуре.
2. GUI приложений.

Глава 2. Проектирование и реализация

2.1. Проектирование

Для реализации был выбран 2 вариант:

«Описание сценариев Умного дома. Описание различных функций, получаемых путем комбинации датчиков, исполнительных механизмов, панелей оператора, мультимедиа систем, контроллеров управления, сетевых устройств.»

Перед началом работы построим диаграмму классов (см. рис. 2.1).

Участники

1. SmartHomeComponent — компонент умного дома: - содержит поле с названием компонета и статус (активен или неактивен). Имеет метод для активации/деактивации.

2. SmartTV, SmartLight, SmartCurtains, SmartFloor — примитивы, компоненты умного дома.

3. SmartGroup — объединенная группа компонентов умного дома.

4. SmartHomeScenario — сценарий умного дома - содержит поля для групп компонентов, которыми сценарий управляет.

5. Client — клиент: - запускает сценарии умного дома через объекты SmartHomeScenario.

2.2. Реализация

Реализация паттерна «Компоновщик» находится в git-репозитории по ссылке: github.com

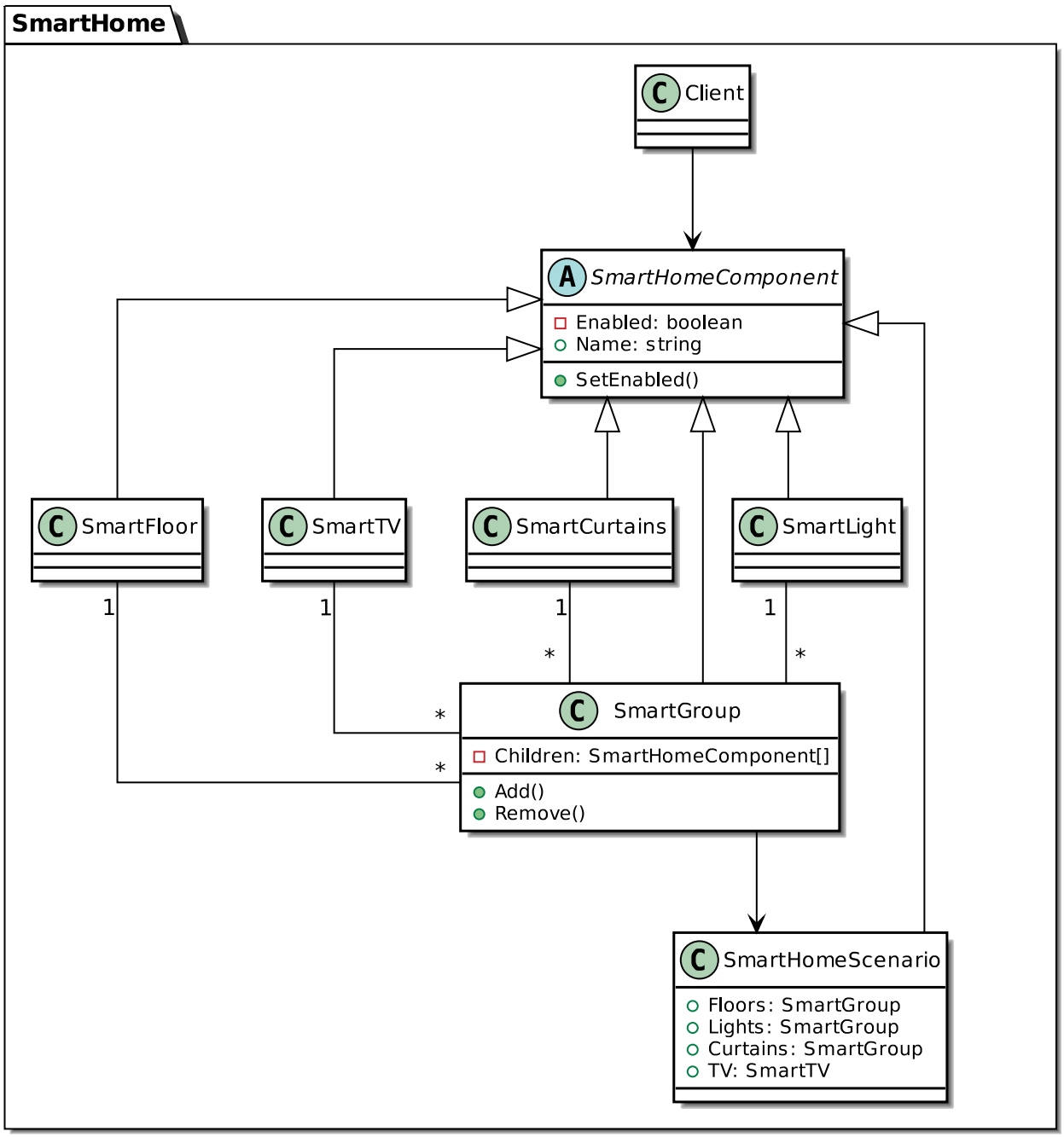

Рис. 2.1. Диаграмма классов