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. ПОВЕДЕНЧЕСКИЙ ПАТТЕРН «НАБЛЮДАТЕЛЬ»

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 «Наблюдатель»

Название и классификация паттерна

Наблюдатель – поведенческий паттерн, который реализует у класса механизм, позволяющий объекту этого класса получать оповещения об изменении состояния других объектов, и тем самым наблюдать за ними.

Назначение

1. Паттерн Observer определяет зависимость "один-ко-многим"между объектами так, что при изменении состояния одного объекта все зависящие от него объекты уведомляются и обновляются автоматически.

2. Паттерн Observer инкапсулирует главный (независимый) компонент в абстракцию Subject и изменяемые (зависимые) компоненты в иерархию Observer.

3. Паттерн Observer определяет часть "View"в модели Model-View-Controller (MVC).

Применимость

Наблюдатель следует использовать, когда:

1. существует, как минимум, один объект, рассылающий сообщения;

2. имеется не менее одного получателя сообщений, причём их количество и состав могут изменяться во время работы приложения;

3. у абстракции есть два аспекта, один из которых зависит от другого. Инкапсуляции этих аспектов в разные объекты позволяют изменять и повторно использовать их независимо;

4. при модификации одного объекта требуется изменить другие и заранее неизвестно, сколько именно объектов нужно изменить;

5. один объект должен оповещать других, не делая предположений об уведомляемых объектах. Другими словами, вы не хотите, чтобы объекты были тесно связаны между собой.

Данный шаблон часто применяют в ситуациях, в которых отправителя сообщений не интересует, что делают получатели с предоставленной им информацией. Паттерн Observer находит широкое применение в системах пользовательского интерфейса, в которых данные и их представления ("виды") отделены друг от друга. При изменении данных должны быть изменены все представления этих данных (например, в виде таблицы, графика и диаграммы).

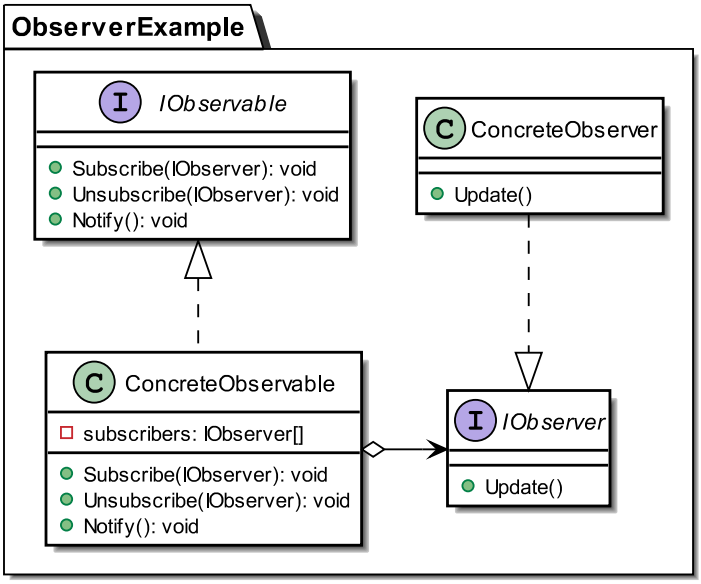

Рис. 1.1. Диаграмма классов паттерна «Наблюдатель»

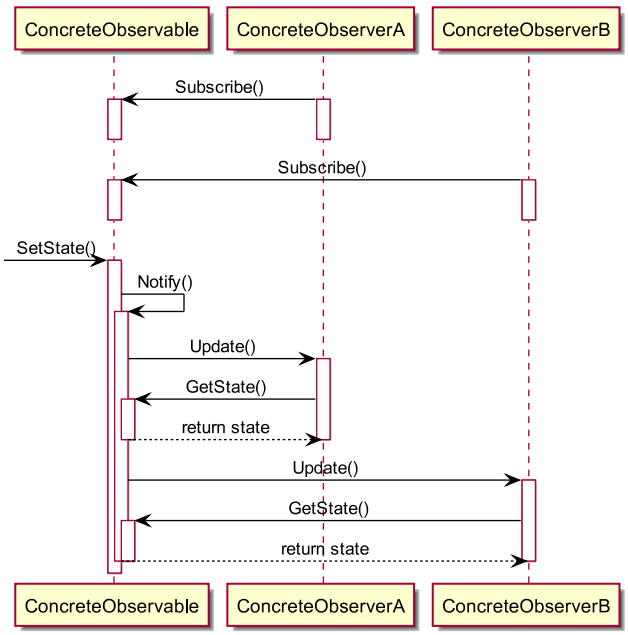

Рис. 1.2. Диаграмма последовательности паттерна «Наблюдатель»

Участники

1. `IObservable`: представляет наблюдаемый объект (издатель). Определяет три метода: `Subscribe()` (для добавления наблюдателя), `Unsubscribe()` (удаление наблюдателя) и `Notify()` (уведомление наблюдателей)
2. `ConcreteObservable`: конкретная реализация интерфейса `IObservable`. Определяет коллекцию объектов наблюдателей.
3. `IObserver`: представляет наблюдателя (подписчика), который подписывается на все уведомления наблюдаемого объекта. Определяет метод `Update()`, который вызывается наблюдаемым объектом для уведомления наблюдателя.
4. `ConcreteObserver`: конкретная реализация интерфейса `IObserver`.

Плюсы и минусы

Плюсы:

1. Издатели не зависят от конкретных классов подписчиков и наоборот.
2. Вы можете подписывать и отписывать получателей на лету.
3. Реализует принцип открытости/закрытости (SOLID).

Минусы:

1. Подписчики оповещаются в случайном порядке.

Области применения

1. Системы, обрабатывающие случайно приходящие события.
2. Графические интерфейсы.

Глава 2. Проектирование и реализация

2.1. Проектирование

Для разработки был выбран 1 вариант:

«Программа для учета успеваемости студентов».

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

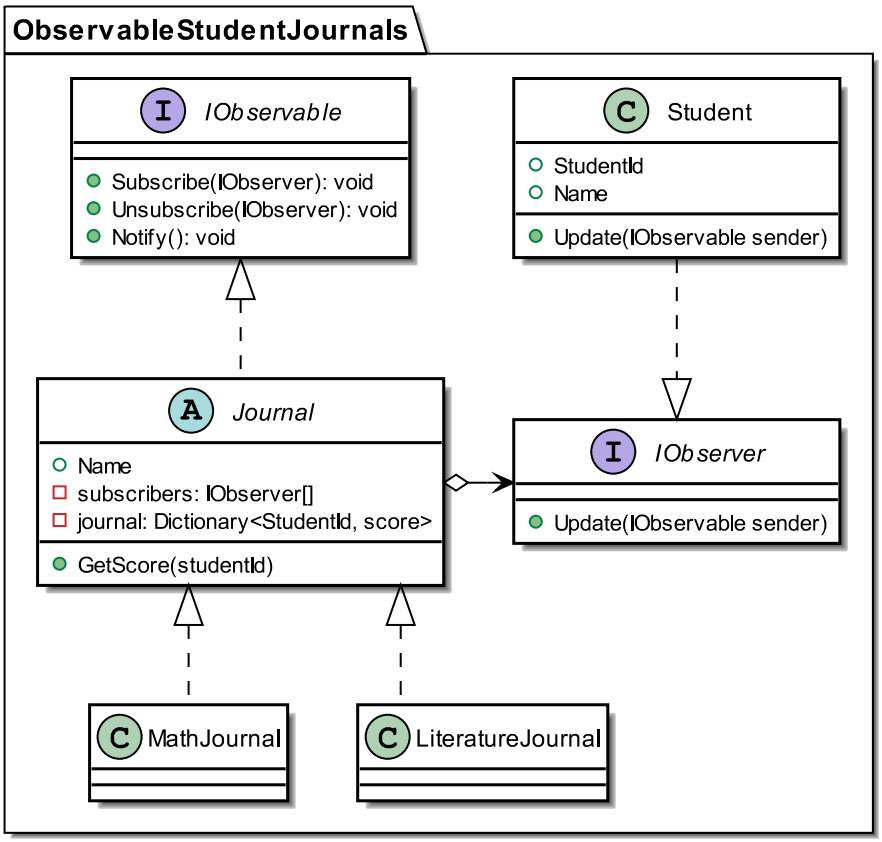

Рис. 2.1. Диаграмма классов

Участники

1. `IObservable`: представляет наблюдаемый объект (издатель).
2. `Journal`: конкретная реализация интерфейса `IObservable`. Абстрактный класс, определяющий список студентов, подписанных на изменения в журнале, сам журнал в виде словаря (ключ — ID студента, значение — оценка по предмету) и метод `GetScore(studentId)` для получения оценки по ID студента.

3. `MathJournal`, `LiteratureJournal`: журналы оценок по математике и литературе.

4. `IObserver`: представляет наблюдателя (подписчика), который подписывается на все уведомления наблюдаемого объекта.

5. `Student`: конкретная реализация интерфейса `IObserver`. Студент, отслеживающий оценки.

2.2. Реализация

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