УНИВЕРСИТЕТ ИТМО

ПИиКТ

Архитектура программных систем
Лабораторная работа №2

Выполнил:
Балтабаев Дамир Темиржанович

Группа: P33121

Преподаватель:
Перл Иван Андреевич

Санкт-Петербург
2022

Задание

Из списка шаблонов проектирования GoF и GRASP выбрать 3-4 шаблона и для каждого из них придумать 2-3 сценария, для решения которых могу применены выбранные шаблоны.

Сделать предположение о возможных ограничениях, к которым можем привести использование шаблона в каждом описанном случае. Обязательно выбрать шаблоны из обоих списков.

Выполнение

GoF:

- State
- Command
- Observer

GRASP:

- Protected Variations

State

State или же Состояние – GoF паттерн, позволяющий менять действия(поведение) объектам в зависимости от их текущего состояния. Для реализации создаются отдельные классы с реализацией поведения для каждого уникального состояния и вместо того, чтобы содержать в себе код всех состояний, объект будет содержать ссылки на объекты состояний.

Сценарии использования

1. Сервис обмена сообщениями

Допустим мы хотим создать собственный сервис для обмена сообщениями по типу WhatsApp, но при одном условии:

сообщения можно редактировать лишь при состоянии “черновик” или “отправлено”. То-есть при последнем состоянии (“просмотрено”) сообщение нельзя будет отредактировать.
Итого: При редактировании черновика мы можем просто вручную отредактировать еще не отправленное сообщение, при редактировании сообщения со статусом “отправлено” мы можем также вручную отредактировать сообщение в специально открытой форме, а при состоянии “просмотрено” метод попросту не будет делать ничего, либо же иметь не активное состояние. Тем самым, используя этот паттерн, мы избегаем огромного увесистого кода, где при добавлении нового состояния может поменяться логика других методов.

Ограничения и недостатки:
Для паттерна State наиболее характерным недостатком является неоправданное использование ресурсов. Иными словами: у нас могут быть 1-2 состояния, оба из которых могут редко меняться, либо не меняться совсем.

2. Выбор интересов при просмотре новостей на новостном сайте-агрегаторе

Отличный пример, демонстрирующий работу паттерна. Представим что мы хотим посмотреть новости на новостном сайте. И при регистрации мы указываем в полях “интерес” темы которые нам нравятся, к примеру “спорт” либо же “политика”. В зависимости от состояния этого поля (“интерес”), новостной агрегатор должен менять поведение метода отобразить_новости(). То-есть при выбранном интересе “спорт” он должен отфильтровать все новости и вывести спортивные новости, также и для политики, только с выводом новостей связанных с политикой.

Ограничения и недостатки:
Для предложенного сценария отсутствует минус, описанный мною для сценария №1, т.к количество интересов довольно обширно и часто сменяемое, так что выбранный паттерн отлично смотрится в подобном применении.

Command

Command паттерн – GoF паттерн, позволяющий при вызове метода передавать в аргументы запрос в виде объекта, ставить эти запросы в очередь и т.д. Шаблон предназначен для инкапсуляции в объекте всех данных, необходимый для выполнения заданного действия (команды).

Сценарии использования

3. Разработка небольшого приложения для работы с базой данных.

Предположим мы хотим сделать небольшую консольную игру, в которой главным объектом будет к примеру миньон. Мы хотим хранить миньонов в базе данных и обращаться к ним с помощью специального приложения и пула команд. Допустим мы хотим добавить миньона с таким-то именем и таким-то цветом кожи, получить сводку о всех миньонах, изменить параметры того или иного миньона и многое другое. Если просто создать некую абстрактную команду, а уже от нее создать подклассы вышеперечисленных команд, то может возникнуть проблема, что в некоторых командах может понадобиться функционал другой команды, допустим для команды изменения объекта понадобится код команды сохранения объекта, из-за этого код команды сохранения придется продублировать в код команды изменения объекта и так по нарастающей. Поэтому нам нужен паттерн Command. Он позволяет создавать класс каждому уникальному вызову с единственным методом, который будет осуществлять вызов. Эти классы и называют командами. В дальнейшем эти команда объединяются под общим интерфейсом с одним методом исполнения, благодаря чему реализуется работа с различными командами, не привязываясь к классам.

Ограничения и недостатки:
Представим ситуацию что у нас 200 уникальных команд, из-за того, что каждая уникальная команда “оберты вается” в отдельный класс у нас получится 200 классов. Я считаю что это является небольшой платой для достижения поставленной цели.

4. Создание игры в шахматы

Мы также можем использовать шаблон Command для написания шахмат с наличием следующих операций: сделать ход конем, ход пешкой, королем, сохранить игру, срубить и т.д. Для всех этих операций мы также используем паттерн Command по причинам описанным в 1-м сценарии.

Ограничения и недостатки:
Аналогичный недостаток, что и у первого сценария.

Observer

Observer паттерн или же наблюдатель – GoF паттерн, создающий механизм подписки, реализующий каскадное изменение объекта, то-есть все зависящие от него объекты оповещаются и следом автоматически обновляются

Сценарии использования

1. Выход нового видео на YouTube канале

Благодаря паттерну Observer можно легко отслеживать выпуск новых видео авторов, на которых вы подписаны. Представим вы подписаны на очень интересный канал с животными и вы каждый день заходите на канал в надежде на новое видео, но вместо пустой траты времени, вам может приходить уведомление на почту после выпуска нового видео, благодаря чему вы не тратите свое драгоценное время. Паттерн работает очень просто и эффективно: Издатель (в нашем случае канал автора) хранит список ссылок на почты подписчиков, причем этот список издатель ведет не самостоятельно, а лишь предоставляет методы для того, чтобы подписчики могли самостоятельно добавлять или же удалять себя из списка подписчиков этого канала.

Ограничения и недостатки:

Подписчики оповещаются в случайном порядке, из-за этого подписчики имеющие лишь 1 подписку(на этот канал) могут быть оповещены в самую последнюю очередь, в то время как подписчики имеющие сотни подписок на разные каналы – в первую. То-есть проблема в отсутствии приоритета.

2. Интерпретатор для БД

   Представим что мы пишем интерпретатор для БД, который при виде команды по типу ON DELETE ON UPDATE CASCADE у сущности, должен изменить значение у всех зависящих сущностей автоматически. Так вот здесь нам и пригодится паттерн Observer.

   Ограничения и недостатки:
   В случае многопоточности и огромного количества выборки может возникнуть следующая ситуация: допустим нам придется поменять 100000 значений. Представим, что первый поток хочет изменить все 100000 значений в сущностях на новое, а второй поток хочет прочитать уже новое значение из конкретной сущности, но из-за того, что выборка слишком большая, может сложиться так, что необходимое значение изменится самым последним и очевидно, что второй поток, работая параллельно, прочитает неизмененное значение. Проблема снова в случайном порядке оповещения.

Protected Variations

Protected Variations паттерн или же Устойчивость к изменениям – GRASP паттерн, который позволяет проектировать систему так, чтобы изменения одних объектов не влияло на другие. Принцип довольно простой, все что нам нужно, это определить “точки неустойчивости”, то-есть точки системы, которые наиболее часто будут подвергаться всяческим модификациям и обеспечить для них интерфейсы и в дальнейшем реализовав различные варианты

поведения. Данный паттерн позволяет эффективно и динамически изменять систему.

Сценарии использования

1. Система управления в игре

Представим, что есть некая игра с синхронизацией, позволяющей играть в эту игру и на ПК и на мобильном устройстве. Так вот, реализован весь функционал игры, а также реализовано взаимодействие пользователя с игрой, например на компьютере мы используем клавиши W,A,S,D и мышь для передвижения, а на мобильном устройстве это все реализуется при помощи сенсорного экрана. Вместо того, чтобы сильно переписывать код и делать систему непригодной к расширениям, мы может разделить реализацию передвижения, чтобы выбор того или иного способа передвижения не влиял на другие объекты. То-есть можно предоставить интерфейс взаимодействия функционала передвижения с конкретным типом устройства. Этот сценарий хорошо демонстрирует применение паттерна Protected Variations.

Ограничения и недостатки:

Такие “точки неустойчивости” требуют большого количества дополнительного функционала, а также немалое количество дополнительных расходов на поддержание системы.

2. JVM

Как известно, JVM предоставляет услуги кроссплатформенности, то-есть вне зависимости от операционной системы мы можем исполнить тот или иной код, ведь JVM – это по сути отдельный виртуальный процессор, предоставляющий функционал для работы с самыми известными операционными системами. В дальнейшем можно расширить функционал виртуальной машины, позволив ей интерпретировать код на новых операционных системах.

Ограничения и недостатки:

Ограничения и недостатки аналогичны первому сценарию, ведь действительно, приходится реализовывать большое количество доп. Функционала с немалым количеством расходов на поддержание самой системы.

Вывод

В ходе выполнения данной лабораторной работы я познакомился с понятиями GoF и GRASP паттернов, а также предоставил некоторые возможные сценарии выполнения и недостатки для самых известных шаблонов.