УНИВЕРСИТЕТ ИТМО

Факультет программной инженерии и компьютерной техники

Направление подготовки 09.03.04 Программная инженерия

Архитектура программных систем

Лабораторная работа №1

Студент:

Нуруллаев Д.Р.

P33121

Преподаватель

Перл Иван Андреевич

Санкт-Петербург, 2022 г.

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 шаблон: Builder(Строитель)

Сценарии использования:

1) Game-Design

   Представим ситуацию, нам нужно создать средневековую RPG игру с открытым миром. Естественно мир должен быть хорошо наполнен, чтобы он не казался пустым. Создавать вручную каждое строение(дома,замки и тд) будет слишком затратно и долго. Для таких случаев была придумана процедурная генерация, в которой нам понадобится шаблон билдер. Мы будем использовать разных билдеров для создания разных зданий.Все они строят одно и тоже – здания, но при этом используя разные материалы и разные дополнительные атрибуты у тех или иных домов.

2) Автоматизация завода по созданию машин

   Машины очень сложные объекты состоящие из других не менее сложных объектов. С помощью паттерна builder мы можем настроить пошаговую сборку машин и помимо всего создать множество билдеров под все типы машин.

Ограничения:

Проблема данного паттерна в том, что строитель который создает объект жестко связан с этим объектом, поэтому при внесении изменений в класс продукта скорее всего придется так же изменять и класс самого строителя.

GoF шаблон: Memento(Снимок)

Сценарии использования:

1) Редактор кода
   Представим ситуацию когда мы пишем какой-нибудь код и вдруг мы понимаем, что мы написали какую-то дичь мы благополучно удаляем этот кусок кода и через пару секунд до нас доходит, что все таки этот код был верным. Мы забыли что мы написали и теперь сидим грустные потому-что ничего не работает. Чтобы таких ситуаций не просходило существует паттерн memento, который помогает делать нам снимки состояния и мы всегда можем вернутся к предыдущему снимку. Теперь мы просто нажимаем ctrl+z чтобы отменить действие и предыдущий снимок заменяет своим состоянием текущее состояние.

2) Backup'ы системы
   Вторая ситуация в которой мы можем использовать наш паттерн – это Backups. Представляем такую ситуацию, решили мы покапаться значится в ядре линукса. И мы как не самый умный пользователь решили сделать это не в виртуалке, а на своей родной машине. И тут происходит ситуация, мы доковырялись и сломали пол функционала нашей операционной системы. Но хоть мы и не самый умный, но сделать backup перед тем как начать копаться в ядре додумались. Используем наш backup и возвращаем систему в норму. Mission complete.
   В данном случае так же применим паттерн memento для сохранения снимков нашей системы для того, чтобы мы могли восстановить систему.

Ограничения:

В данном паттерне нужно следить за тем, чтобы мы не переполнили память миллионами снимков, поэтому нужно освобожать память от старых снимков.

Так же есть проблема во многих языках программирования которые будут использовать данный паттерн. Проблема в том, что мы не сможем гарантировать то, что только исходный объект будет иметь доступ к состоянию снимка. Таких проблем не будет у языков имеющих вложеные классы(C#,Java,C++). Ведь в языках которые имеют механизм вложенных классов, класс снимков мы сможем хранить внутри класса с которого мы будем делать эти самые снимки.

GoF шаблон: Visitor(Посетитель)

Сценарии использования:

1) Карта города
   Представим ситуацию, у нас есть граф. Узлами этого графа являются сложные объекты. Для примера из жизни мы представим такую ситуацию. Мы берем карту города и хотим каждому виду бизнеса отправить финансирование, соответствующее именно данному типу бизнеса(кафе,столовая,ресторан,театр,кино,завод,ТЦ ...). Наш TeamLead запретил менять классы объектов добавляя в него методы отправки необходимого финансирования для каждого типа объекта объяснив это тем, что сегодня финансирование, завтра отправить им поссылку, послезавтра письмо, постоянно менять класс и поитогу сломать его он не позволит. В таком случае нам нужно придумать что-то, что сможет помочь пройтись по всему графу и отправить нужное финансирование каждому объекту.Здесь нам поможет паттерн visitor. Нам все таки придется добавить один метод в каждый класс, но при этом если нам придется выполнить что-то новое нам уже не придется переписывать класс, нам просто нужно будет создать новый класс посетителя который посетит каждый узел и выполнить нужное для него действие.
   Таким образом мы пройдемся по всему графу и для каждого объекта выполним свое действие.
2) Game-Design
   Опять же похожая ситуация. Берем такую игру как World Of Warcraft. Предствим ситуацию что Blizzard придумали какой-то новый суперский ивент. И теперь они хотят сделать так, чтобы во всех объектах мира игры (Столицы расс, Города,Деревни,Подземелья, Локациях без городов) появились NPC которые будут выдавать награды/задания в ходе этого ивента и при этом в зависимости от объекта будет разное количество NPC на объект а где-то они не появятся и вовсе. Они явно не будут менять код этих объектов которые уже больше десятка лет работают стабильно, в данном случае так же будет применим паттерн visitor, для того чтобы пройтись по всем объектам и в зависимости от их типа раскидать NPC по карте.

Ограничения:

Когда мы попытаемся добавить в нашу структуру, нам придется постоянно обновлять всю иерархию посетителя и его сыновей.

GRASP шаблон: Controller(Контроллер)

Сценарии использования:

1) Backend разработка с MVC
   Самое яркое применение шаблона контроллер мы прослеживаем в модели MVC. Контроллер отвечает за то, чтобы решать кому мы должны делегировать пришедшие нам запросы на выполнение.Использование контроллеров в модели MVC позволяет отделить бизнес-логику от представления,тем самым повышая возможность повторного использования кода, так же ограничивая нас от многопоточного кода.
2) Программирование клавиатуры
   Представим что нам надо запрограммировать клавиатуру, и при нажатии каждой клавиши у нас происходит определенное событие. Здесь так же присутствует контроллер. При нажатии на клавишу у нас идет запрос на контроллер который определяет какую клавишу мы нажали и запускает нужный нам обработчик событий. Здесь точно так же прослеживается модель MVC. Я нажимаю на клавиатуру(View), запрос приходит на контроллер(Controller), вызывается нужный метод для обработки этого собюытия(Model) и результат поступает на монитор(View).

Ограничения:

Контроллер может оказаться перегружен.

Вывод:

В ходе данной лабораторной работы я узнал о существовании шаблонов GRASP и повторил многие шаблоны GOF. Поработал над тем, что бы придумать практические применения всем выбранным шаблонам. Шаблоны очень помогают как оптимизировать, так и облегчить код для его читаемости.