Архитектура – это план как устроено ваше приложение (или будет устроено)

Паттерн – хорошо устоявшиеся решение

Фабрика – **порождающий** паттерн, метод или функция, которая создает нам объекты, в зависимости от параметров разные объекты. Функции-конструктор создают объект строго одного типа. (to-do: factory vs facade)

Facade – **структурный** паттерн, представление простого интерфейса для работы с множество различными операциями, которые другие классы не должны видеть.

Стратегия — **поведенческий** паттерн, позволяет менять выбранные алгоритм независимо от объектов, которые его используют. То есть инкапсулировать. (config менеджер поддерживающего различные форматы файлов. Json, cvs, ini. Аутентификации или авторизации – стратегии (google, facebook, apple, twitter) для оптимизации большого количества ветвей **if else**.

Более упрощенное **Шаблонный метод** (parseFile – абстрактный класс и реализация конкретного метода)

Наблюдатель — **поведенческий** паттерн, позволяет оповещать/наблюдать состояние других объектов. Имеет отношение одни ко многим. **Subject** и **Observer** (MutationObserver, IntersectionObserver, ResizeObserver). Похожие **Event Hub**, **Pub/Sub**

Выше декларативность – те это метрика, типа, яснее показывает, что код делает. Классы (какие методы приватные, какие публичные, инициализация объекта, тип). Иногда **new** раздражает и нам хочется от него избавится (приходится скобки ставить)

Паттерн представляет собой не какой-то конкретный код, а общую концепцию решения той или иной проблемы, которую нужно будет еще подстроить под нужды вашей программы.

**SOLID**

DDD

объект-сущность entity - в ограниченном контексте обладают собственной идентичностью и всегда сущ. только в корневом агрегате, те не могут сущ. независимо и не имеют жизненного цикла

объект-значение не обладают собственной сущностью и без затруднений могу быть изменены в любом экземпляре агрегата

событие всегда должно публиковаться от агрегата, если др. частью приложения (сервисом) то оно считается событием техническим, а не событием бизнес-домена оно логически вытекает из того факта, что только агрегат оповещает о произошедшем изменении состояния

Объекты-значения существуют в области видимости агрегата ограниченного

контекста. Объекты-значения не обладают собственной идентичностью и могут быть заменены в любом экземпляре агрегата.

Команда (command) в ограниченном контексте – это любая операция, которая изменяет состояние агрегата.

Запрос (Query) в ограниченном контексте – это любая операция, которая возвращает состояние агрегата.

операции модели предметной области в ограниченном контексте

выполняются связанные с состоянием агрегата

команды – изменяет состояние

запросы - читает состояние

события - оповещение об изменении состояния

События предметной области (домена) играют центральную роль в архитектуре микросервисов, поэтому весьма важно реализовать их надежно и корректно. Распределенная природа архитектуры микросервисов диктует использование событий через механизм хореографии (choreography mechanism) для

сопровождения состояния и сохранения логической целостности и согласованности транзакций между различными ограниченными контекстами приложения на основе микросервисов

Существует три типа сервисов модели предметной области (домена) для любого ограниченного контекста:

* входящие сервисы (Inbound Services), в которых реализуются корректно

определенные интерфейсы, позволяющие внешним потребителям взаимодействовать с моделью предметной области (домена); (api rest, cdi events)

* исходящие сервисы (Outbound Services), в которых реализуются все взаимодействия с внешними репозиториями и другими ограниченными контекстами;
* сервисы приложения (Application Services), действующие как внешний уровень, связывающий модель предметной области (домена) с входящими и исходящими сервисами.