## Что я узнал из книги “Приемы объектно-ориентированного программирования. Паттерны проектирования.”

\*\*\*

Инкапсуляция - это результат сокрытия поведения и реализации в объекте. При этом поведение невидимо и недоступно извне. Получение доступа и изменения (модификации) поведения объекта возможно только с помощью операций.

Данные, свойства, атрибуты, поля, состояние (состояние объявляется в полях, т.е. поля определяют поведение) объекта - это всё синонимы.

Поведение (поведение декларируется в методах, т.е. методы определяют поведение), операции, методы, функции, код - это тоже синонимы.

Делегирование - это процесс реализации, при котором запрос передается (делегируется) от одного объекта к другому, который становится уполномоченным для выполнения этого запроса.

Также я узнал о понятии “*программирование в соответствии с интерфейсом, а не реализации*”.

Программирование в соответствии с интерфейсом предполагает использование полиморфизма. Полиморфизм позволяет клиенту не беспокоиться:

* о типе, к которому принадлежат конкретные объекты.
* о классах, которые реализуют объекты. Достаточно знать только абстрактный класс (или другого класса), который реализует интерфейс.

Полиморфизм предоставляет гибкость в использовании кода. Точнее - предоставляет возможность повторного использования кода (реализации).

“Делегировать” - означает передать другому объекту.

Есть агрегирование и есть еще ???.

Также я узнал о еще одном понятии “*программирование с использованием композиции над наследованием*”. Типа композиция лучше наследования, но и тем и другим злоупотреблять не стоит.

Паттерны проектирования - абстракции, которые используются при решении определённых классов задач, позволяют при дальнейшем развитии программы упрощать процесс её обновления, указывают проектировщику на последствия выбранного решения и тд.

ВИКИ : **паттерн** в [разработке программного обеспечения](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8F) — повторимая [архитектурная конструкция](https://ru.wikipedia.org/wiki/%D0%90%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8F), представляющая собой решение проблемы [проектирования](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) в рамках некоторого часто возникающего [контекста](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BD%D1%82%D0%B5%D0%BA%D1%81%D1%82).

Каркас (фреймворк, приложение) - это тоже повторимая архитектурная конструкция, но она позволяет сделать для приложения готовую основу.

Каркас - это уже готовое тело программы, которое вызывает различный код. Такое тело можно использовать для разных приложений и все они будут похожими.

Паттерн проектирования - это более абстрактное понятие чем каркас. И паттерн может быть в различных вариациях в отличии от каркаса. То есть всё зависит от проблемы, которую нужно решать. Паттерн предполагает логическую обёртку кода.

Каркас - это более специальное понятие.

Пример:

* Библиотека сделана с использованием паттернов. В основной программе мы пишем тело и вызываем из него код, который ре-юзаем.
* Каркас - это готовое тело, под которое мы пишем код. То есть, здесь мы ре-юзаем каркас.

Паттерны помогают решить проблемы проектирования!

Паттерны упрощают дальнейшее обновление системы! (*проектирование в соответствии с будущими изменениями*)

Паттерны являются своеобразным мостом между КЛИЕНТОМ и ОБЪЕКТОМ - он скрывает ненужную информацию от первого!

Паттерны могут пригодится при написании универсального кода. Они могут устранять сильные зависимости приложений от программных и аппаратных платформ!

В общем, паттерны - это инструменты, которые предоставляют простое решение проблемы с упором на повторное использование (реюз) и дальнейшую модификацию системы.