# Инкапсуляция

Инкапсуляция применима не только к классам, но и к более крупным структурам: пакетам, модули, сабмодули, подсистемы, системы.

Существуют две трактовки, вторая не имеет смысла без первой.

1. В объект объединяются данные и методы, работающие с этими данными.
2. Сокрытие внутренней структуры объекта от внешнего мира.

Вторая трактовка требует конкретизации: все изменения состояния объекта производятся только методов самого объекта (объект не должен изменяться ничем, кроме его собственных методов).

В чём профит? Строгая регламентация работы с объектом позволяет ему не ломаться т. к. все изменения производит не васёк, а наши кристально чистые методы.

Например, у нас есть микроволновка, у неё есть сложная внутренняя структура – генератор микроволн, динамик дзинькающий, лампочка, крутящийся барабан и т. д. Если бы мы имели прямой доступ ко всему этому, то, скорее всего всё бы сломали. Поэтому у нас есть компактный интерфейс взаимодействия – выставить режим, выставить время, запустить. Это сильно упрощает взаимодействие с объектом, нам не нужно знать, как он работает, достаточно лишь понимания того, что на вход микроволновка получает холодную еду, а возвращает горячую. И такой подход позволяет избежать ломания нашего объекта. Поэтому инкапсуляция — это круто.

Необходимо скрывать всю внутреннюю структуру, которая никому не пригодиться.

# Наследование

Позволяет переиспользовать уже написанный код, раскрывает полиморфизм.

Класс наследника является классом предка. Наследник более узкий и конкретный объект. Тут работает отношение IS A (является). То есть наследник is a родитель (student is a person).

Важно понимать, когда нужно наследование, а когда делегирование.

Если у объектов действительно есть много общего – используем наследование.

Если нам просто нужны какие-то методы из класса, используем делегирование.

Наследование расширяет перед нами возможности полиморфизма (открывает так называемый «истинный» полиморфизм), во многом нужно именно для него.

# Полиморфизм

Самая главная часть ООП, причина по которой оно настолько мощное, причина по которой существует наследование и инкапсуляция.

На его основе строится большинство паттернов.

1. Ad-hoc – полиморфизм по запросу. Приведение данных, перегрузка методов.
2. Параметрический полиморфизм (истинный). Функция может принимать объекты разных классов. Невозможен без наследования