ООП

***Введение***

***Разработка современного программного обеспечения очень трудозатратный и трудоемкий процесс. ПО может содержать в себе тысячи файлов и в разы больше строчек кода. Что бы упростить процесс написания кода применяют различные методы программирования. Методы программирования - совокупность методов, применяемых при разработке***[***ПО***](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5)***и имеющих общий философский подход.***  ***Если проще, то это*** подход к проектированию программ; взгляд на саму структуру программы; стиль написания программ; способ мышления о программе.

Объектно-ориентированное программирование – это одна из таких моделей. Существуют, например, процедурная, структурная, аспектно-ориентированная, объектно-ориентированная; метапрограммирование и т.д.

Определение звучит так. Объектно-ориентированное программирование – это методология программирования, основанная на представлении программы в виде совокупности объектов, каждый из которых является экземпляром определенного класса, а классы образуют иерархию наследования.

*ООП как стиль написания программ подразумевает построение структуры, состоящей из взаимодействующих объектов. В рамках ООП нужно мыслить объектами.* Объекты расположены в иерархии, самостоятельны и как-то взаимодействуют. Программа состоит из модулей – блоков, которые решают какие-то задачи. Изменения в этих участках могут не отражаться на других участках.

Основные понятия

**Абстрагировать** – это выделить значимые характеристики объектов, и в ООП важно выделить их правильно.

**Класс** – это описание объектов, представителей класса: отличительные признаки, состояния, взаимодействия. Что это, из чего это состоит, что можно с этим делать и как именно? У классов и объектов есть атрибуты, данные (Data) и методы, то есть функции для управления данными.

**Методами** в ООП называют функции для работы с объектами (что и как с этим делать, как и с чем это взаимодействует). Методы могут быть доступны другим классам и управлять их действиями, а могут быть действительны только внутри класса.

**Иерархия**: модули классов; базовый класс (класс описывает свойства объектов, содержит множество объектов, подклассов), производный класс (наследует свойства родительского класса и тоже содержит объекты); отдельно взятый объект (данные/Data, описывающие свойства объекта, + методы, то есть функции для работы с объектом, алгоритмы управления, действия, которые выполняет объект).

**Совокупность** тех методов класса, которые доступны и другим классам, называются интерфейсом. Чтобы тренировать объектное мышление, можно проводить аналогии с работой физических объектов и систем (с приборной доской в самолете, с пультом управления, с домофоном или лифтом в многоквартирном доме). Методы могут быть публичными и инкапсулированными, то есть скрытыми. По аналогии: панель управления в машине – на виду (интерфейс класса), а детали под капотом работают скрыто (так называемый «черный ящик»).

**Инкапсуляция** — это свойство системы, позволяющее объединить данные и методы, работающие с ними, в классе и скрыть детали  
реализации от пользователя.

Наследование

**Наследование** — способ определения нового класса, когда новый класс наследует элементы (свойства и методы) существующего, модифицируя или расширяя их. Это способствует выражению специализации и генерализации.

Концепция **полиморфизма** – важная часть ООП. Все методы в языке Python изначально виртуальные. Это значит, что дочерние классы могут их переопределять и решать одну и ту же задачу разными путями.

* + Возможность легкой модификации (при грамотном анализе и проектировании)
  + Возможность простого отката при наличии версий
  + Более легкая расширяемость
  + «Более естественная» декомпозиция программного обеспечения, которая существенно облегчает его разработку.
  + Сокращение количества межмодульных вызовов и уменьшение объемов информации, передаваемой̆ между модулями.
  + Увеличивается показатель повторного использования кода.
* Требуется квалификация
* Значительно увеличивается время на анализ и проектирование систем
* Увеличение времени выполнения
* Размер кода увеличивается
* Неэффективно с точки зрения памяти
* Сложность распределения работ на начальном этап
* Себестоимость больше