Оглавление

[Основы ООП 2](#_Toc163940195)

[Атрибуты классов 2](#_Toc163940196)

[Биение по принципу того, что делают 2](#_Toc163940197)

[Биение по принципу того к чему относятся 2](#_Toc163940198)

[Основные принципы ООП 3](#_Toc163940199)

[Инкапсуляция 3](#_Toc163940200)

[Спецификаторы доступа 3](#_Toc163940201)

[Наследование 4](#_Toc163940202)

[Полиморфизм 4](#_Toc163940203)

[Сокрытие 4](#_Toc163940204)

# Основы ООП

**ООП** – подход к разработке, при котором предметная область приложения анализируется с последующем биением на объяты.

**Объект** –единица информации в памяти. Является сущностью, которая имеет свои параметры и поведение.

**Класс** – описывает модель объекта, его свойства и поведение. Является индукцией по сборки объектов.

**Экземпляр** – Конкретный объект каво-то класса. (Для краткости вместо «Объект, порожденный классом "Стул", говорят «экземпляр класса "Стул"».)

## Атрибуты классов

### Биение по принципу того, что делают

Обычно классы в ООП содержат в себе атрибуты. Атрибуты можно разбить по принципу того, что они делают. Атрибутами является следующее:

1. **Методы** – Функция в классе для воздействия с экземпляром класса. Действие, которое объект может выполнять над самим собой или другими объектами.
2. **Поля** – переменные для хранения какой-то информации, имеют определённый уровень доступа

### Биение по принципу того к чему относятся

Если разбить атрибуты по отношению к объектам то можно выделить следующие категории:

1. **Атрибуты класса** – такие атрибуты, которые созданы и инициализированы в классе и имеют определённое значение (имеются с самого начала и создаются и инициализируются не в методах а в классе). Данные атрибуты и их значение являются общими для всех экземпляров данного класса. Изменение данных атрибутов приводит и изменению и в других экземплярах данного класса.
2. **Атрибуты объекта/экземпляра** – такие атрибуты, которые принадлежат определённому конкретному экземпляру класса. При изменении данного атрибута, изменения происходят только для определенного экземпляра. Принадлежит не всему классу а конкретному объекту.

# Основные принципы ООП

Обычно выделяют следующие главные принципы в ООП:

1. **Инкапсуляция**.
2. **Наследование**.
3. **Полиморфизм**.
4. **Сокрытие**.

Данные принципы будут описаны далее по тексту

## Инкапсуляция

**Инкапсуляция** – Позволяет сделать класс единым целым. Данный принцип позволяет программисту работать только с тем, что есть в классе или подается в класс. Класс превращается в капсулу со всем необходимым для его функционирования. Работать из вне с классом можно только через разрешение методы и атрибуты.

### Спецификаторы доступа

Спецификаторы доступа – механизм позволяющий определять как будут доступны атрибуты

Виды спецификаторов доступа:

public – атрибуты которые доступны в:

1. не класса (то есть в любой точке кода)
2. классе (где определен атрибут)
3. классе, который наследовался от класса, где определен атрибут

Данный спецификатор нужен для того, чтобы создавать атрибуты, которые будут работать вне класса. В основном используется, чтобы управлять объектами классов.

protected – атрибуты которые доступны в:

1. классе (где определен атрибут)
2. классе, который наследовался от класса, где определен атрибут

Данный спецификатор нужен для того, чтобы создавать атрибуты, которые будут работать только внутри класса или в наследуемых от него классов. В основном используется для того, чтобы скрыть атрибут (с целью защиты от вмешательства), но оставить его видимым для наследуемых классов.

private - атрибуты которые доступны в:

1. классе (где определен атрибут)

Данный спецификатор нужен для того, чтобы создавать атрибуты, которые будут работать только внутри класса и только с этим классом. В основном используется для того, чтобы скрыть атрибут (с целью защиты от вмешательства).

## Наследование

**Наследование** – механизм позволяющий выстраивать иерархию среди классов. Предполагает определение базового (родительского) класса и класса приемника (произвольный класс / потомок).

В родительском классе определяют общие атрибуты что позволяет избавится от дублирования кода. Как правило родительский класс более абстрактный чем потомки.

В классе потомке определят только те атрибуты, которые необходимы для работы только этого класса. Как правило класс потомок более приближен к реальным целям чем его абстрактные родители.

Потомок наследует все атрибуты от класса родителя и может при этом обладать своими атрибутами

Родительский класс должен быть заменяем с любым классом потомком. Поведение при работе с дочерним и при работе с родительским класса не отличается (подстановка Барбары Лисков. Один из принципов SOLID).

## Полиморфизм

Полиморфизм – механизм позволяющий работать с объектами классов через определенные названия методов. Позволяет исполнять один и тот же код для данных разных классов без дополнительных условий. Можно реализовать через перегрузку методов.

Пример: Группа программистов пришит классы для геометрических фигур. Для того чтобы не занимать время на проверки к какому классу относится данный объект, программиста заранее договариваются как они назовут метод для действия, которое делает одно и тоже

class Circle:

    """Класс круга"""

    def draw(self):

        print("Рисуем круг")

class Square:

    """Класс квадрата"""

    def draw(self):

        print("Рисуем квадрат")

Метод draw является полиморфным (так как одинаково работает с классами Circle и Square)

## Сокрытие

Сокрытие позволяет спрятать от пользователя то, что знать ему не положено. Смысл заключается в том, чтоб пользователь мог работать только с тем, с чем ему разрешили работать.