Software
Engineering
School

Курсы программирования для
взрослых и детей

ООП: вспомнить все...

Чем мы сегодня займемся

- Вспомним прошлое
- Откроем для себя объекты заново
- Познакомимся с диаграммой классов

Но сначала викторина!

- Что такое ООП?
- Какие основные понятия есть в ООП?
- Что такое объект? А класс?
- Что объект хранит?
- Как объекты взаимодействует между собой?
- А как же все это реализуется в коде?
- А что такое инкапсуляция?...

Основа основ: что такое ООП?

Объектно-ориентированное проектирование – способ представления программы в виде объектов, взаимодействующих между собой, чтобы получить определенный результат.

Суть объектно-ориентированного программирования состоит в том, что все программы состоят из объектов, а каждый объект – определенная сущность со своими данными (поля, свойства) и набором доступных действий (методы, функции).

Что такое объект?

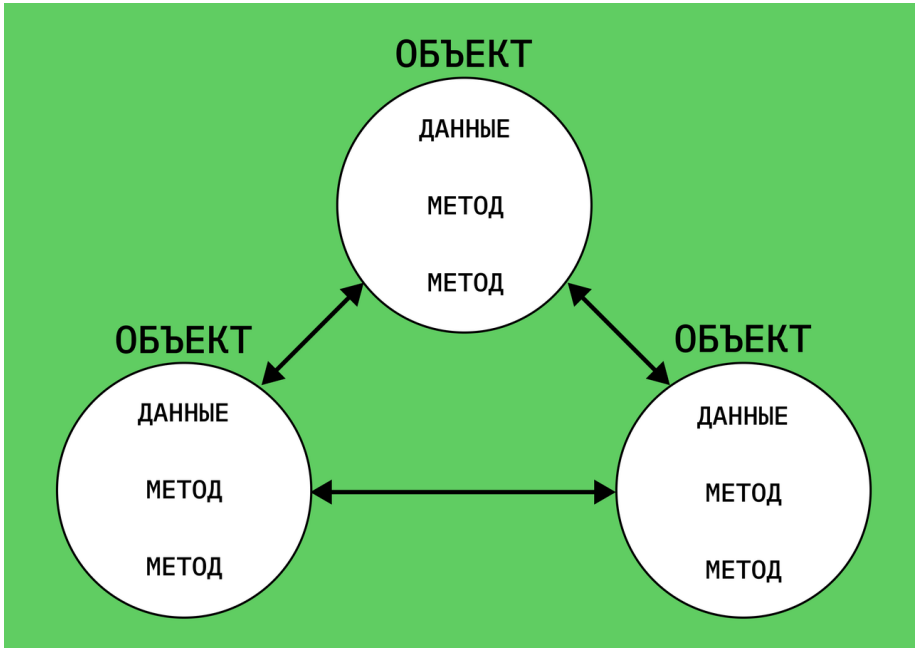
Объект – сущность, предмет обладающий свойствами и определенным поведением.

Свойства - характеристики, которые его описывают (реализуются при помощи полей и методов).

Поведение - набор действий, которые объект может осуществить (реализуется в виде методов).

Объекты в программах

Программа состоит из набора объектов, взаимодействующих друг с другом для получения какого-либо результата

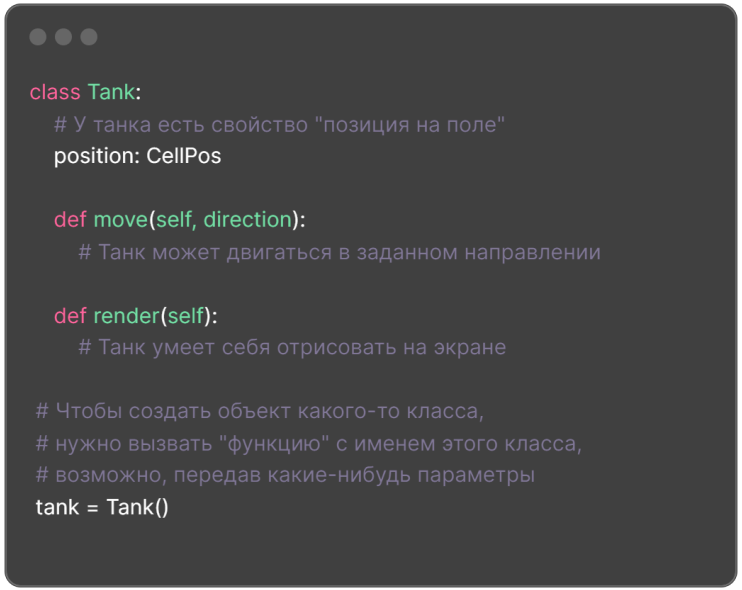


Что такое класс?

Объект нельзя создать просто так. Для этого нужно указать чертёж, по которому он будет сделан.
Класс - чертёж для объектов, который описывает их свойства и поведение.

Классы и объекты в Python

В коде свойства и поведение объектов мы реализуем при помощи полей и методов:



```
class Tank:
    # У танка есть свойство "позиция на поле"
    position: CellPos

    def move(self, direction):
        # Танк может двигаться в заданном направлении

    def render(self):
        # Танк умеет себя отрисовать на экране

# Чтобы создать объект какого-то класса,
# нужно вызвать "функцию" с именем этого класса,
# возможно, передав какие-нибудь параметры
tank = Tank()
```

Проектирование объектов

Прежде чем садится кодить, необходимо понять: а что у объекта должно быть, чтобы он мог помочь в решении задачи? Выделение существенных характеристик и поведения объектов называется абстракцией.
Проектируя объект, мы должны использовать только ту информацию о нем, которая помогает решать задачу!

Пример

Вам необходимо спроектировать систему учета занятий ученикам в школе. Было бы неплохо иметь объект «Ученик», только вот какие же характеристики и поведение у него должно быть?...

Пример

Смотря на реальный мир, мы видим, что у учеников очень много свойств – имя, фамилия, возраст, класс, рост, вес и так далее, но для того чтобы решить исходную задачу нам почти ничего из этого не нужно! Достаточно знать ФИО ученика и время его прихода в школу – т.е., всего 2 свойства. Мы воспользовались абстракцией и выделили только то, что нужно для решения задачи.

Вопросы

1. А зачем мне нужен класс? Можно же завести функцию `draw_tank(position)`, которая будет делать тоже самое...
2. А зачем мне нужен метод `move()`? Я же могу сам посчитать позицию, изменить её через `tank.position = new_position` и радоваться жизни...

Возможные ответы

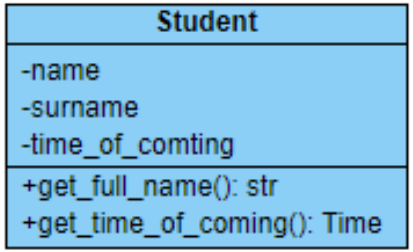
1. Хорошо, если нам потребуется только рисовать танк и двигать его, а когда мы добавим ему способность стрелять? Добавлять функцию `tank_shoot()`? А как понять какой танк стреляет – наш или вражеский? Класс позволяет прописать однотипным сущностям одни и те же свойства и поведение, не ограничиваясь всего лишь одним возможным объектом.
2. Вводя метод `move()` мы устраняем дублирование кода, да и к тому же, кто мешает нам присвоить `tank.position = None` и сломать всю программу?

UML

В мире культурных программистов писать код сразу не принято: чем больше программа, которую вы разрабатываете, тем сложнее удержать все детали в голове, как все устроено и как организовано. Поэтому, чтобы иметь представление о том, из каких объектов состоит ваша программа и как они взаимодействуют между собой, принято визуально «рисовать» программу – в этом нам, программистам, помогает язык UML (Unified Modeling Language – универсальный язык моделирования) и диаграмма классов!

Диаграмма классов

1. Прямоугольником мы обозначаем класс: сверху идет название класса, потом блок с полями класса и затем блок с методами класса
2. Знак «-» перед полями или методами класса обозначает, что они доступны только самому классу (а точнее – объекту!) и извне их нельзя менять (в случае полей) или вызывать (в случае методов)
3. Знак «+» обозначает, что метод или поле «публично» – любой другой объект, может вызывать этот метод



Размышляем

> Зачем понадобилось вводить разделение на публичные и приватные поля/методы?

> Если бы поле name у Student было бы публичным, кто угодно мог бы его поменять, и далеко не всегда задать допустимое значение, например, student.name = None не является допустимым именем. Методов это тоже касается: бывают настолько сложные алгоритмы, что в одном методе они не умещаются, и нужно выделять под-методы, которые в нормальных условиях не должны вызываться – в конце концов, зачем может понадобится вызвать только часть алгоритма, а не весь алгоритм? Такие методы должны быть приватными

Диаграмма класса Tank