Теоретический материал к занятию Основы ООП 2 занятие.

Предположим, нам необходимо создать класс транспорта, но видов транспорта может быть больше количество: легковые машины, грузовые, общественный транспорт и т.п. Эти виды имеют как общие, так и строго индивидуальные характеристики. Как пример грузовой автомобиль может иметь возможность выгружать перевозимый груз(самосвал), автобус же например, имеет большое количество посадочных мест и принадлежит не частному лицу, а предприятию и многое другое. Как же поступить в данной ситуации. Тут нам поможет возможность классов наследоваться друг от друга. Например мы можем создать общий класс транспорта(родительский), а потом создать дочерние классы. Дочерним классам будут доступны все поля и методы родительского класса.
Разберем пример

```
class Transport:
    def __init__(self, speed, color):
        self.speed = speed
        self.color = color

    def beep(self):
        print('beep')
```

Создали родительский класс и указали, что у транспорта есть скорость, цвет и возможность говорить бип
Теперь необходимо создать дочерние классы. Для указания, что класс является дочерним, необходимо при указании названия класса, в скобках указать класс от которого мы наследуемся.

```
class Car(Transport):
    def __init__(self, speed, color):
        super().__init__(speed, color)
```

В классе Car мы указали, что он наследуется от класса Transport, а также в методе инициализации указали вызов super(), в котором сослались на метод инициализации родительского класса, т.е простыми словами, мы сказали что нам необходимо вызвать метод init родительского класса. Теперь укажем для класса пару переменных и какой нибудь метод выведем в консоль.

```
class Transport:
    def __init__(self, speed, color):
        self.speed = speed
```

```
        self.color = color

    def beep(self):
        print('beep')

class Car(Transport):
    def __init__(self, speed, color, owner):
        super().__init__(speed, color)
        self.owner = owner

    def say_owner(self):
        print(f'Владелец {self.owner}')


car1 = Car(100, 'yellow', 'Василий')
print(car1.color)
print(car1.speed)
print(car1.owner)
car1.beep()
car1.say_owner()
```

Из данного примера мы видим, что объект класса Car имеет доступ не только к свойствам и методам своего класса, но и родительского.

Множественное наследование - когда класс наследуется более, чем от одного родительского класса. Также родительский класс в котором требуется указать больше данных требуется указать первым.

```
class Transport:
    def __init__(self, speed, color):
        self.speed = speed
        self.color = color

    def beep(self):
        print('beep')

class Car(Transport):
    def __init__(self, speed, color, owner):
        super().__init__(speed, color)
        self.owner = owner

    def say_owner(self):
        print(f'Владелец {self.owner}')


class SportCar(Car, Transport):
    pass


car1 = SportCar(100, 'yellow', 'Иван')
```

```
car1.beep()
car1.say_owner()
```

В данном примере мы создали класс,который наследуется от двух предыдущих классов. Несмотря на то что класс SportCar пустой, он наследует все методы и атрибуты от родительских классов.

Если же у нас в классе Car будет метод, с таким же названием как у другого родительского класса, то метод будет переопределен и вызовется метод из класса Car

```
class Transport:
  def __init__(self, speed, color):
    self.speed = speed
    self.color = color

  def beep(self):
    print('beep')

class Car(Transport):
  def __init__(self, speed, color, owner):
    super().__init__(speed, color)
    self.owner = owner

  def say_owner(self):
    print(f'Владелец {self.owner}')

  def beep(self):
    print('Hello')


class SportCar(Car, Transport):
  pass


car1 = SportCar(100, 'yellow', 'Иван')
car1.beep()
car1.say_owner()
```