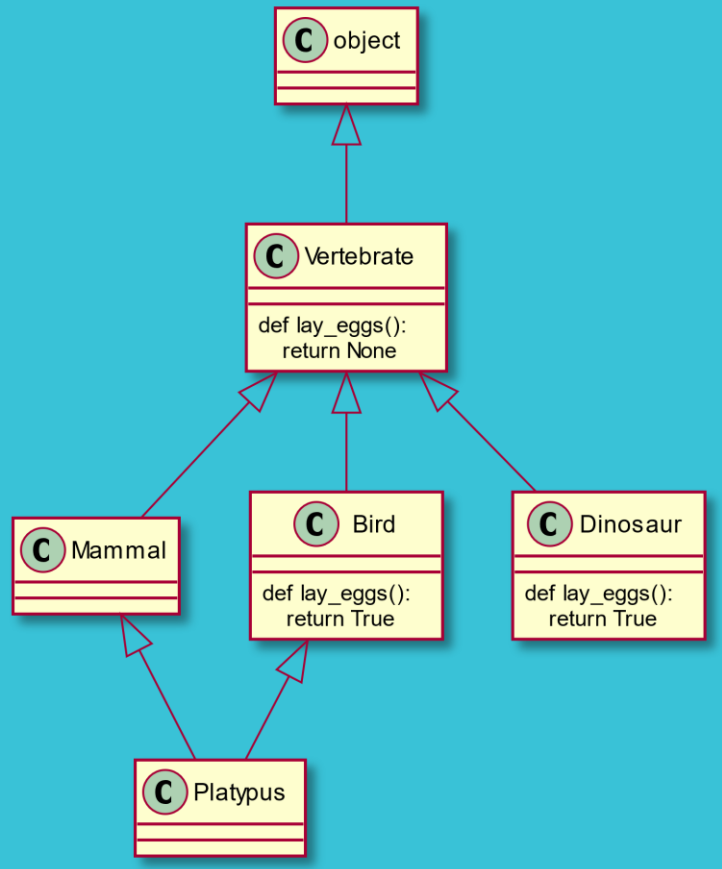
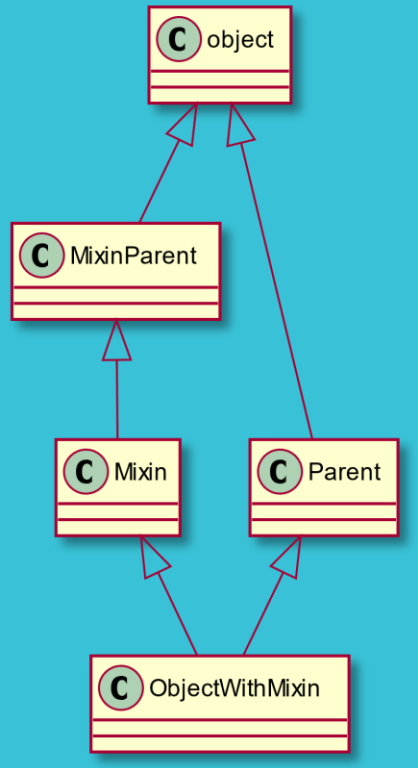
Ромбовидный Python

Загадка

```
class Vertebrate:
    def lay_eggs(self):
        return None

class Bird(Vertebrate):
    def lay_eggs(self):
        return True

class Mammal(Vertebrate):
    pass

class PlatypusMammalFirst(Mammal, Bird):
    pass

class PlatypusBirdFirst(Bird, Mammal):
    pass

print(PlatypusMammalFirst().lay_eggs())
print(PlatypusBirdFirst().lay_eggs())
```

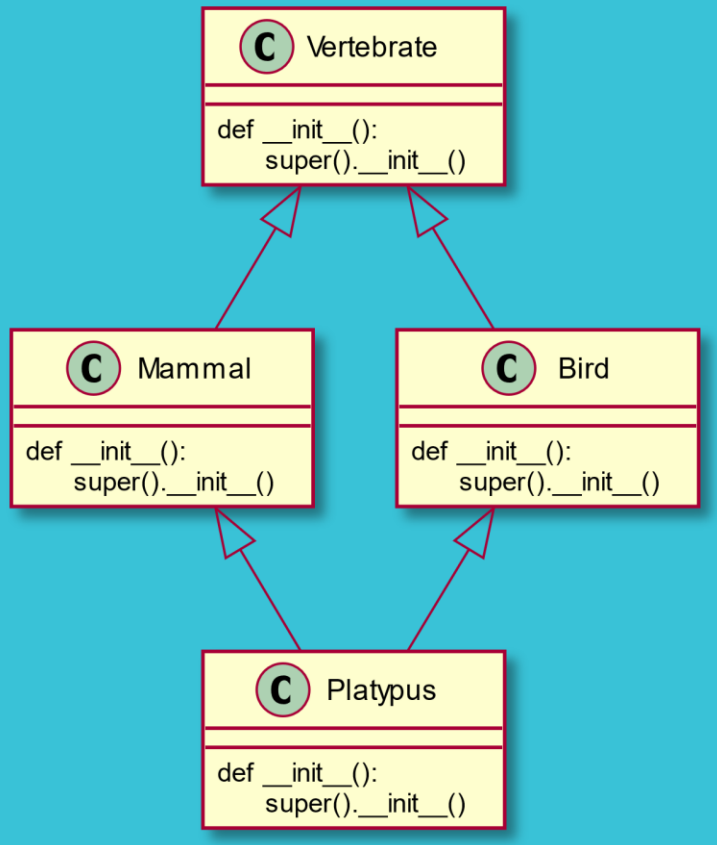
Diamond problem





Кооперативное наследование

```
class Vertebrate:
    def __init__(self):
        print('Vertebrate.__init__()')

class Bird(Vertebrate):
    def __init__(self):
        print('Bird.__init__()')
        super().__init__()

class Mammal(Vertebrate):
    def __init__(self):
        print('Mammal.__init__()')
        super().__init__()

class Platypus(Mammal, Bird):
    def __init__(self):
        print('Platypus.__init__()')
        super().__init__()

duckbill = Platypus()
```



Не столь кооперативное наследование

```
class Vertebrate:
    def __init__(self):
        print('Vertebrate.__init__()')

class Bird(Vertebrate):
    def __init__(self, beak_length):
        print('Bird.__init__()')
        super().__init__()

class Mammal(Vertebrate):
    def __init__(self, hair_length):
        print('Mammal.__init__()')
        super().__init__()

class Platypus(Mammal, Bird):
    def __init__(self):
        print('Platypus.__init__()')
        super().__init__(1)

duckbill = Platypus()
```

Super не совсем super

```
class Parent:
    def __getitem__(self, idx):
        return 0

class Child(Parent):
    def index_super(self, idx):
        return super()[idx]

kid = Child()
print(f'kid[0]: {kid[0]}')
print(f'kid.index_super(0): {kid.index_super(0)}')
```