Парадигмы ООП

Парадигма наследования

Парадигма наследования позволяет создавать сложные системы классов, избежать дублирования кода, упростить поддержку программ и многое другое.

При наследовании обычно говорят о классах-родителях и классах-потомках. У одного родительского класса может быте несколько классов-потомков, расширяющих функционал родительского класса. Если язык программирования поддерживает множественное наследование, то у одного класса-потомка может быть несколько родительских классов. Язык Python поддерживает множественное наследование. Поля родительского класса при наследовании переходят к классу-потомку. Кроме того, поля родительского класса могут переопределены у потомка.

```
class A:

    def some_function(self):
        print("First function")

    def other_function(self):
        print("Second function")


class B:

    def method_in_B(self):
        print("Third function")


class C(A):

    def other_function(self):
        print("Replaced function")


class D(B, C):

    pass


# Посмотрим все атрибуты класса, не являющиеся служебными
print("A:\t", list(filter(lambda x: "__" not in x, dir(A))))
print("B:\t", list(filter(lambda x: "__" not in x, dir(B))))
print("C(A):\t", list(filter(lambda x: "__" not in x, dir(C))))
print("D(B,C):\t", list(filter(lambda x: "__" not in x, dir(D))))
print()

# Посмотрим на реализацию функцй в D
```

```
d = D()
d.method_in_B()
d.some_function()
d.other_function()
print()
```

Парадигма инкапсуляции

Парадигма инкапсуляции предлагает объединять переменные и методы, относящиеся к одному объекту в единый компонент. По сути соблюдение парадигмы инкапсуляции и заключается в создании классов.

Парадигма полиморфизма

Парадигма полиморфизма позволяет вместо объекта базового типа использовать его потомка, при этом не указывая это явно.

```
class Parent:

    def some_method(self):
        print("This is Parent object")


class Child1(Parent):

    def some_method(self):
        print("This is Child1 object")


class Child2(Parent):

    def some_method(self):
        print("This is Child2 object")


def who_am_i(obj):
    obj.some_method()

p = Parent()
c1 = Child1()
c2 = Child2()

who_am_i(p)
who_am_i(c1)
who_am_i(c2)
print()
```