面向对象_08_多态

多态

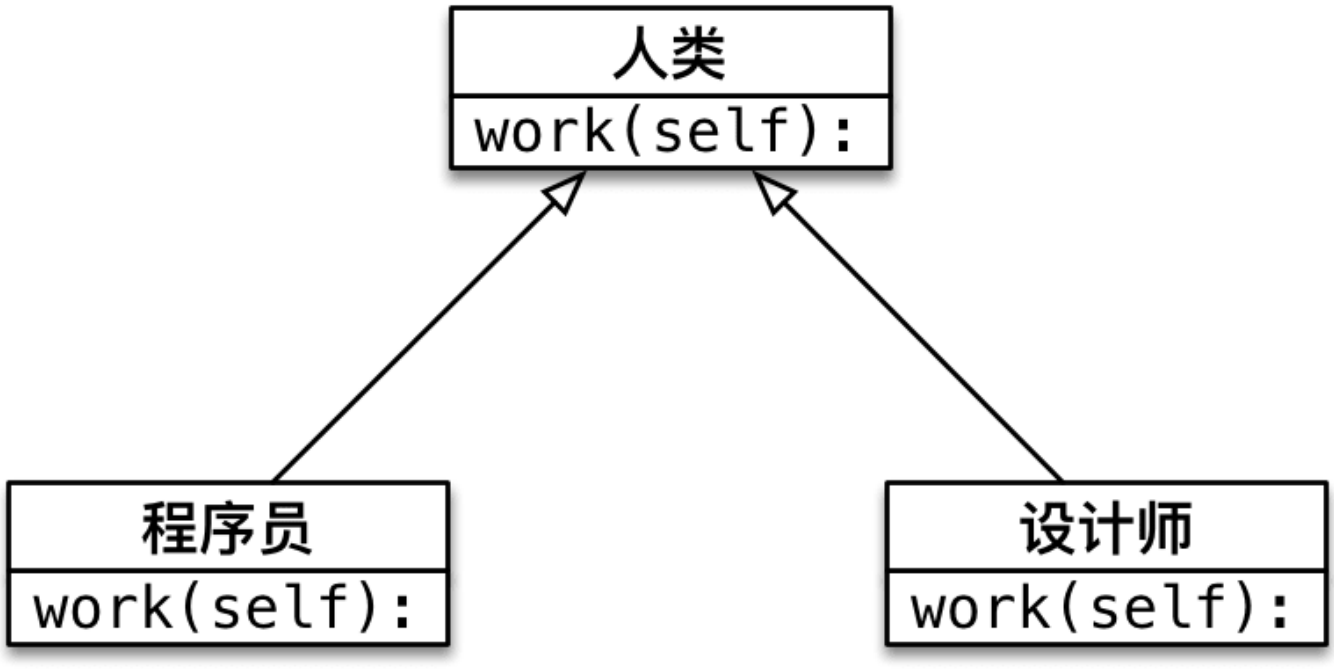
— 黑马程序员《Python 入门教程完整版》笔记

目标

- 多态

面向对象三大特性

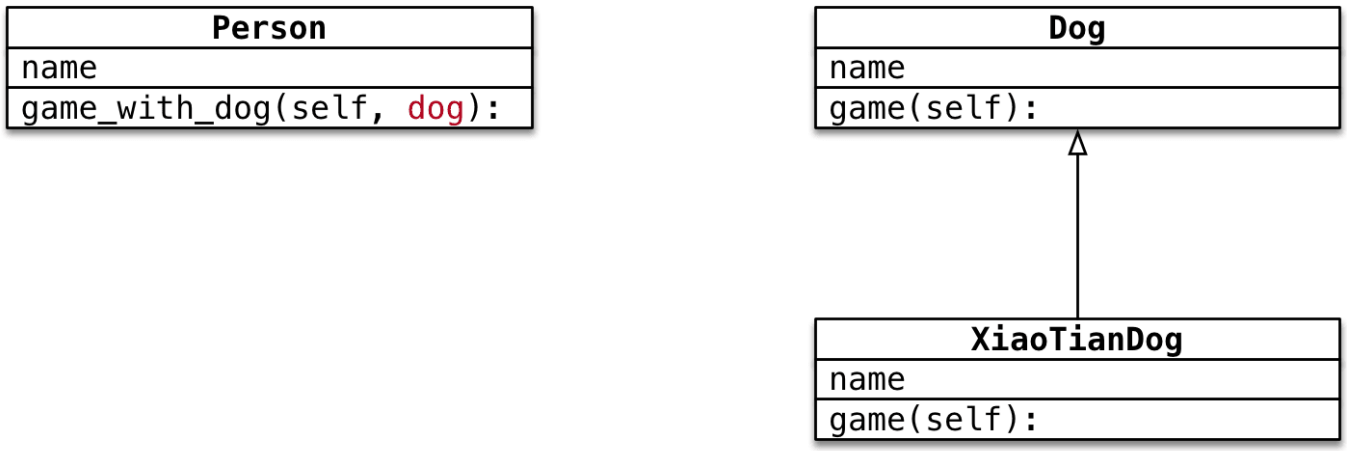
1. **封装** 根据 **职责** 将 **属性** 和 **方法** **封装** 到一个抽象的 **类** 中
    - 定义类的准则
2. **继承** **实现代码的重用**，相同的代码不需要重复的编写
    - 设计类的技巧
    - 子类针对自己特有的需求，编写特定的代码
3. **多态** 不同的 **子类对象** 调用相同的 **父类方法**，产生不同的执行结果
    - **多态** 可以 **增加代码的灵活度**
    - 以 **继承** 和 **重写父类方法** 为前提
    - 是调用方法的技巧，**不会影响到类的内部设计**



多态案例演练

需求

1. 在 `Dog` 类中封装方法 `game`
   - 普通狗只是简单的玩耍
2. 定义 `XiaoTianDog` 继承自 `Dog`，并且重写 `game` 方法
   - 哮天犬需要在天上玩耍
3. 定义 `Person` 类，并且封装一个 **和狗玩** 的方法
   - 在方法内部，直接让 **狗对象** 调用 `game` 方法



案例小结

- `Person` 类中只需要让 **狗对象** 调用 `game` 方法，而不关心具体是 **什么狗**
  - `game` 方法是在 `Dog` 父类中定义的
- 在程序执行时，传入不同的 **狗对象** 实参，就会产生不同的执行效果

> **多态** 更容易编写出出通用的代码，做出通用的编程，以适应需求的不断变化！

```python
class Dog(object):

    def __init__(self, name):
        self.name = name

    def game(self):
        print("%s 蹦蹦跳跳的玩耍..." % self.name)


class XiaoTianDog(Dog):

    def game(self):
        print("%s 飞到天上去玩耍..." % self.name)


class Person(object):
```

```
    def __init__(self, name):
        self.name = name

    def game_with_dog(self, dog):

        print("%s 和 %s 快乐的玩耍..." % (self.name, dog.name))

        # 让狗玩耍
        dog.game()


# 1. 创建一个狗对象
# wangcai = Dog("旺财")
wangcai = XiaoTianDog("飞天旺财")

# 2. 创建一个小明对象
xiaoming = Person("小明")

# 3. 让小明调用和狗玩的方法
xiaoming.game_with_dog(wangcai)
```