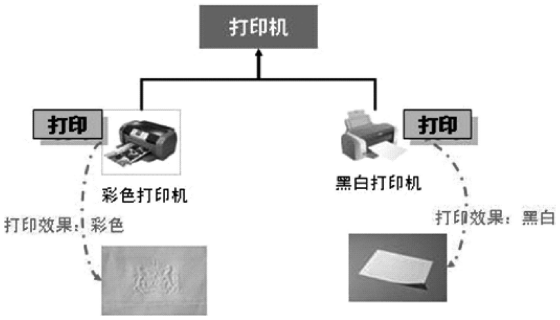
5. 多态

多态，顾名思义就是多种形态。多态(Polymorphism)是具有表现多种形态能力的特征。更专业化的说法：同一个实现接口，使用不同的实例而执行不同的操作。

为了便于理解，我们举一个身边的例子。



打印机可以看作父类，黑白打印机、彩色打印机是它的两个子类。父类打印机中的方法“打印”在每个子类中有不同的实现方式。例如，对黑白打印机执行打印操作后，打印效果是黑白的；而对彩色打印机执行打印操作后，打印效果是彩色的。很明显，子类分别对父类的“打印”方法进行了重写。

从这里也可以看出，多态性与继承、方法重写密切相关。

父类 Car，子类 OffRoad

```
#父类
class Car():
    def __init__(self,purpose = '运输'):
        self.purpose = purpose
    def show(self):
        print('这辆车的用途：',self.purpose)
#子类 OffRoad
class OffRoad(Car):
    def show(self):
        print('这辆车适合越野,最高时速 180')
```

子类 OffRoad 重写了父类的 show()方法。

```
suv = OffRoad()
suv.show()
```

输出结果为：

这辆车适合越野,最高时速 180

因为重写了父类的方法，子类在调用相同的方法时，就出现了不同的结果。这是多态的

一种最直接的体现。

再来看一个例子，体会一下多态。对每一个汽车子类，都重写 show()方法，通过销售公司售卖不同的车型而给出不同的 show()方法提示。

父类及子类分别如下：

```
#父类 Car
class Car():
    def __init__(self,purpose = '运输'):
        self.purpose = purpose
    def show(self):
        print('车的用途：',self.purpose)
#子类 OffRoad
class OffRoad(Car):
    def show(self):
        print('适合越野,最高时速 180，过山淌水能力强')
#子类 Bus
class Bus(Car):
    def show(self):
        print('公共汽车的载客量最大，安全平稳')
#子类 Home_Car
class Home_Car(Car):
    def show(self):
        print('家用轿车方便小巧，代步首选')
```

然后定义经销商类 CarShop，定义方法，参数为对象类型，根据输入不同的参数类型，输出的结果各不相同。

```
class CarShop():
    def sell(self,car):
        car.show()
cshop = CarShop()
cshop.sell(OffRoad())
cshop.sell(Bus())
cshop.sell(Home_Car())
```

输出结果为：

适合越野,最高时速 180，过山淌水能力强

公共汽车的载客量最大，安全平稳

家用轿车方便小巧，代步首选