# 继承与多态

### 继承

玩家：

#player.py

class Player:

name = ‘’

\_\_score = 0

def \_\_init\_\_(self, name):

self.name = name

self.\_\_score = 0

def addScore(self):

self.\_\_score += 1

def resortScore(self):

self.\_\_score = 0

def \_\_str\_\_(self):

return “name = ‘%s’, score = %s” % (self.name, self.\_\_score)

def \_\_repr\_\_(self):

return ‘Player(%s)’ % str(self)

人类玩家：

class Human(Player):

def \_\_repr\_\_(self):

return ‘Human(%s)’ % str(self)

\_\_repr\_\_相当于重新定义父类的同名函数。

计算机玩家：

class Computer(Player):

def \_\_repr\_\_(self):

return ‘Computer(%s)’ % str(self)

创建派生类对象：

#main.py

h = Human(‘Jerry’)

print(h)

h.addScore()

print(h)

### 多重继承

# -\*- coding: utf-8 -\*-

"""

Created on Thu Mar 29 09:30:38 2018

@author: flyintosky

"""

class People:

name = ''

age = 0

\_\_weight = 0

def \_\_init\_\_(self,n,a,w):

self.name = n

self.age = a

self.\_\_weight = w

def speak(self):

print("%s 说: 我 %d 岁。" %(self.name,self.age))

#单继承示例

class Student(People):

grade = ''

def \_\_init\_\_(self,n,a,w,g):

#调用父类的构函

People.\_\_init\_\_(self,n,a,w)

self.grade = g

#覆写父类的方法

def Speak(self):

print("%s 说: 我 %d 岁了，我在读 %d 年级"%(self.name,self.age,self.grade))

#另一个类，多重继承之前的准备

class Speaker():

topic = ''

name = ''

def \_\_init\_\_(self,n,t):

self.name = n

self.topic = t

def speak(self):

print("我叫 %s，我是一个演说家，我演讲的主题是 %s"%(self.name,self.topic))

#多重继承

class GoodStudent(Speaker, Student):

a =''

def \_\_init\_\_(self,n,a,w,g,t):

Student.\_\_init\_\_(self,n,a,w,g)

Speaker.\_\_init\_\_(self,n,t)

obj = GoodStudent("Tim",25,80,4,"Python")

obj.speak() #方法名同，默认调用的是在括号中排前地父类的方法

### 多态

人类玩家的移动棋子：

class Human(Player):

def \_\_repr\_\_(self):

return ‘Human(%s)’ % str(self)

def move(self):

while True:

try:

n = int(input(‘%s move(1-10):’ % self.name))

if 1 <= n <= 10:

return n

else:

print(‘Oops!’)

except:

print(‘Oops!’)

计算机玩家的移动棋子：

import random

class Computer(Player):

def \_\_repr\_\_(self):

return ‘Computer(%s)’ % str(self)

def move(self):

return random.randint(1, 10)

游戏函数：

def playGame(p1, p2):

p1.resetScore()

p2.resetScore()

m1 = p1.move()

m2 = p2.move()

print(“%s move: %s” %(p1.name, m1))

print(“%s move: %s” %(p2.name, m2))

if m1 == m2 – 1:

p1.addScore()

return p1, p2, ‘%s wins!’ % p1.name

elif m2 == m1 – 1:

p2.addScore()

return p1, p2, ‘%s wins!’ % p2.name

else:

return p1, p2, ‘no winner’

玩游戏主程序：

传入计算机和人对象：

c = Computer(‘robot’)

h = Human(‘lisa’)

print(playGame(c, h))

传入两个计算机对象：

c1 = Computer(‘robot1’)

c2 = Computer (‘robot2’)

print(playGame(c1, c2))

playGame函数传入不同的对象（可以传入人类和计算机对象，计算机和计算机对象，计算机和人类对象，人类和人类对象），里面的move执行相应对象的业务函数，体现了函数执行的多种形态，这就是多态。