# day2 一切都是对象

面向对象: Object Oriented,OO

1. 对象

英文 Object ，计算机业界现在已经习惯翻译为“对象”；口语化一些，中文的意思就是“事物”

比如学生，商品，订单

1. 属性  
   每个事物都有一些特点，人有身高，体重，在程序中我们称之为属性
2. 方法

每个事物都有一些行为，人要吃饭，睡觉，在程序中我们称之为方法。

1. 继承

学生是人，自然有身高，体重，自然要吃饭，睡觉。如果你把人看作一类事物，把学生看作一类事物；那么，人是父类型，学生是子类型。子类型从父类型自然得到属性、方法，我们称之为继承。

1. 多态

学生要考试，工人不要；工人拿工资，学生不拿（一般而言）。同一个父类，

不同的子类有不同的行为和状态，我们称之为多态。

人们编写程序，也就是在描述一类事物的特点（属性）、行为（方法）。有时候是模拟描述自然界中已有的一类事物，还有时候是创造地描述自然界中没有的一类事物。

1. 封装

当人们决定了代码世界中一类事物的属性、方法，在代码世界中，这类事物的属性、方法只有定义代码的人知道，其它的类是不知道的。这就是封装。

封装、继承、多态是面向对象编程的三个本质特征。

## 封装

### 变量的定义

参考wiki：<http://192.168.75.10/cn/index.php?cmd=read&page=%E6%8A%80%E6%9C%AF%2F%E5%BF%85%E9%A1%BB%E5%AD%A6%E7%9F%A5%E8%AF%86%E7%82%B9%2FRuby%2F%20Ruby%E7%9A%84%E7%B1%BB%E7%9A%84%E7%B1%BB%E5%AE%9E%E4%BE%8B%E5%8F%98%E9%87%8F%E3%80%81%E7%B1%BB%E5%AE%9E%E4%BE%8B%E6%96%B9%E6%B3%95%E5%92%8C%E7%B1%BB%E5%8F%98%E9%87%8F%E3%80%81%E7%B1%BB%E6%96%B9%E6%B3%95%E7%9A%84%E5%8C%BA%E5%88%AB&word=%E5%8F%98%E9%87%8F>

#### 局部变量

例如： foobar

若标识符首位是小写字母或“\_”，则该标识符就是局部变量或方法调用。在局部变量的作用域(类、模块、方法的定义部分)内，若对一个首位是小写字母的标识符进行首次赋值的话，也就意味着声明了一个属于该作用域的局部变量。若引用尚未被声明的标识符的话，就会被解释成一个无参数的方法调用。

局部变量的作用域起始于***声明处***，结束于该声明所在的块、方法定义、类／模块定义的结尾。随着块的消亡，局部变量也将寿终正寝(顶层局部变量则一直持续到程序终结)，但也有例外。若块已经变成过程对象的话，则局部变量将一直持续到该过程对象终结为止。若多个过程对象引用同一个作用域的话，局部变量将被这些对象所共享。

2.times {

p defined?(v) # (A)

v = 1 # 从(开始声明)起

p v # 到(块的终结)是 v 的作用域

}

#### 实例变量

例如：@foobar

以@开始的变量是实例变量，它属于特定的对象。可以在类或子类的方法中引用实例变量。若引用尚未被初始化的实例变量的话，其值为nil。

class Person

def initialize(name)

@name = name #实例变量

end

end

#### 类变量

例如：

class Foo

@@foo = 1

end

以@@开始的变量是类变量。在类的定义中定义类变量，可以在类的特殊方法、实例方法等处对类变量进行引用／赋值。

#### 全局变量

例如：$foobar

以$开始的变量是全局变量，可以在程序的任何地方加以引用(因此需要特别留意)。全局变量无需变量声明。引用尚未初始化的全局变量时，其值为 nil。

#### 常数

例如：FOOBAR = “test”

class Foo

FOO = “foo” # 定义Foo类的常数FOO (Foo::FOO)

end

以大写字母([A-Z])开始的标识符是常数.常数的定义(和初始化)由赋值过程完成.不能在方法中对常数进行定义.若对已定义的常数进行赋值的话,会出现警告信息.若引用未定义的常数的话,则会引发NameError异常.

可以在下列地方引用常数,如,定义常数的类/模块的定义句(也包括方法正文以及嵌套类/模块的定义句)中,继承该类的子类中,以及包含模块的类/模块中等等.在类定义之外(顶层)定义的常数属于Object.

若想在外部访问类或模块中的常数时,要使用"::"操作符.若想准确地访问Object类中的常数(顶层的常数)时,也需要也使用"::"操作符,但操作符左边为空.另外,不能使用该操作符对常数进行赋值.

#### 伪变量

除普通的变量之外，还有一种叫做伪变量的特殊变量。

self

当前方法的执行主体

nil

NilClass类的唯一实例

true

TrueClass类的唯一实例

false

FalseClass类的唯一实例。nil 和 false 表示“伪”。

\_\_FILE\_\_

当前源文件名

\_\_LINE\_\_

当前源文件中的行号

伪变量的值不可改变，若对伪变量赋值将引发语法错误。

### 方法的定义

关键字： def

def fun\_name(arg1, arg2…..)

#body

end

1. 使用关键字def，方法名必须以一个小写字母开始。可以是重定义的操作符，也可以是个名字。
2. 习惯约定：表示查询的方法通常以?结尾；会修改接受者对象的方法，以!结尾；赋值方法以=结尾。
3. 方法的返回值是执行的最后一个表达式的值，或return表达式的返回值。如果return不带参数，返回值为nil，如果带一个参数，则返回该参数的值，如果参数多于一个，则返回一个包含所有参数值的数组对象。

#### 实例方法的定义

#### 类方法的定义

<http://192.168.75.10/cn/index.php?cmd=read&page=Trash%2F%E6%96%B0%E5%91%98%E5%B7%A5wiki%2F2009%2F12%2F%E7%B1%BB%E6%96%B9%E6%B3%95%E4%B8%8E%E5%AE%9E%E4%BE%8B%E6%96%B9%E6%B3%95%E7%9A%84%E8%AE%BE%E5%AE%9A&word=%E7%B1%BB%E6%96%B9%E6%B3%95>

### 方法的访问控制

**public, private, protected**

public 方法可以被定义它的类和其子类访问，可以被类和子类的实例对象调用；

protected 方法可以被定义它的类和其子类访问，不能被对象直接调用，但是可以在类和子类中指定给实例对象；

private 方法可以被定义它的类和其子类访问，不能被实例对象调用。

initialize 方法除外，它永远是私有的

### 类的属性的读写控制

attr\_reader attr\_writer attr\_accessor

### 类的定义

class xxx # 类名首字母要大写

#### 封装

#### 继承

#### 多态

### Ruby的动态特性

### eval

### class\_eval

#### instance\_eval

### module\_eval

### instance\_variable\_set

(class\_variables)

### const\_set

### define\_method

### send

### remove\_method

### undef\_method

任务：

1、定义一个Person类，基本属性有年龄、性别、姓名，定义一个Men类和Women类继承自Person类。

2、给Men或Women类动态添加一个方法，设置自己的名字和年龄。

3、自由定义类常量、类变量、实例变量并使用。

4、面向对象的形式重写数组排序的方法