关于程序设计语言的 OO 特性调研报告*

马银萍 1601214515

摘　要: 面向对象程序设计已成为一种重要的程序设计范型，被广泛的接受和使用。本文主要是介绍几种典型的面向对象程序设计语言，包括 Ruby、Python、PHP 和 Java，了解面向对象程序设计的基本概念和性质，并且对比这些语言的设计思想和面向对象特性。

关键词: 面向对象程序设计语言；封装；继承；多态

1 面向对象程序设计语言的基础概念

面向对象程序设计语言有一些重要的组成元素和性质，包括：对象、类、封装、继承和多态等。

下面分别对各个条目进行解释：

面向对象（即 OO）：是一种新型的程序设计方法，或者是一种新的程序设计规范，其基本思想是使用对象、类、继承、封装、消息等基本概念来进行程序设计。

对象：系统中用来描述客观事实的一个实体，它是构成系统的一个基本单位。一个对象由一组属性和对这组属性进行操作的一组服务组成。

类：是具有相同属性和操作的一组对象的集合，它为属于该类的全部对象提供了统一的抽象描述，其内部包括属性和操作两个主要部分在程序设计中，类的作用是用来创建对象，对象是类的一个实例。

面向对象有三个基本特性：封装、继承和多态

封装：封装将数据以及加在这些数据上的操作组织在一起，成为有独立意义的构件。封装使对象能够集中而完整地描述并对应一个具体的事物，体现了事物的相对独立性，使对象外部不能随意存取对象的内部数据，避免了外部错误对它的“交叉感染”，对象的内部的修改对外部的影响很小，减少了修改引起的“波动效应”。

继承：继承是一种代码共享机制，子类拥有父类的数据属性和操作方法，对象的一个新类可以从现有的类中派生，这个过程称为类的继承。新类继承了原始类的特性，新类称为原始类的派生类（子类），而原始类称为新类的基类（父类）。派生类可以从它的基类那里继承方法和实例变量，并且派生类可以修改或增加新的方法使之更适合特殊的需求。

多态：指同一个命名可具有不同的语义。OO 方法中，常指在一般类中定义的属性或操作被特殊类继承之后，可以具有不同的数据类型或表现出不同的行为。多态性是指允许不同类的对象对同一消息作出响应。比如同样的加法，把两个时间加在一起和把两个整数加在一起肯定完全不同。又比如，同样的选择"编辑"、"粘贴"操作，在字处理程序和绘图程序中有不同的效果。多态性包括参数化多态性和运行时多态性。多态性语言具有灵活、抽象、行为共享、代码共享的优势，很好地解决了应用程序函数同名问题。

2 程序设计语言介绍

2.1 Ruby

Ruby 是一种面向对象、命令式、函数式、动态的通用编程语言。在 20 世纪由日本人松本行弘（Matz）设计并开发[1]。它是一门真正的面向对象语言。程序中每个东西都是对象，操作的结果本身也是对象。

2.1.1 Ruby 的特点

Ruby 的特点是完全的面向对象，它没有基础类型，任何东西都是对象，它和 Java 一样具有垃圾回收的存储管理机制，在使用对象的时候完全不用进行事先声明。

在 Ruby 中，所操作的一切都是对象，操作的结果也是对象。很多语言都说自己是面向对象的，但是他们往往对面向对象的解释都一样，大多是以自己特有的方式来解释什么是面向对象，而在实际情况中，这些面向对象语言又采用了很多非面向对象的做法。

以 Java 为例：如果你想取一个数字取绝对值，java 的做法是：

int num = Math.abs(-99)

也就是将一个数值传递给 Math 类的一个静态函数 abs 处理。因为在 java 中，数值是基本类型不是类。

而在 Ruby 中，任何事物都是对象，也就是说，数字–99 就是对象，取绝对值这样的操作应该属于数字本身，所以 Ruby 的做法就是：

c = -99.abs

2.1.2 Ruby 的面向对象特性

2.1.2.1 封装[2]

封装意味着一个对象的内部细节将会对外部隐藏起来，只有对象本身可以与他的内部数据进行操作。公共的方法（Public Mothod）是用来对对象的内部隐藏的结构进行操作的一种方法。

```
class Document
  attr_accessor :name
  def initialize(name)
    @name = name
  end
  def set_name(name)
    @name = name
  end
end
d = Document.new('name1')
d.set_name('name1')
```

进行封装的好处就是可以在不知道 Document 这个类的实现方法的情况下随意地改变想要创建的文件的名字，只需要 Document 这个类对它的内部数据进行更新。

2.1.2.2 继承

Ruby 的继承功能相当脆弱，尽管 Ruby 是一个面向对象语言，Ruby 内的许多规则，却使得子类有可能不小心就改写了父类的功能，在《The Ruby Programming Language》一书中，建议除非程序员对一个类别相当了解，否则尽可能不要使用继承。

```
#!/usr/bin/ruby
class Being
   def initialize
       puts "Being class created"
   end
end
class Human < Being
  def initialize
      super
```

```
      puts "Human class created"
   end
end
Being.new
Human.new
```

在这个程序中，一共有两个类，基类是 Being 类，派生类是是 Human 类，派生类继承了基类的基本属性和操作。

2.1.2.3 多态

Ruby 是动态语言，你可以在程序中修改先前定义过的类别。 也可以在某个类别的实例中定义该实例特有的方法，这叫做原型方法（prototype）。

```
class Document
  def initialize
  end
  def print
    raise NotImplementedError, 'You must implement the print method'
  end
end
class XmlDocument < Document
  def print
    p 'Print from XmlDocument'
  end
end
class HtmlDocument < Document
  def print
    p 'Print from HtmlDocument'
  end
end
XmDocument.new.print # Print from XmlDocument
HtmlDocument.new.print # Print from HtmlDocument
```

在这个例子中，类 XmlDocument 和类 HtmlDocument 都继承了 Document 这个类，然后都分别重写了 Document 这个类中 print 函数，，因此在调用 XmlDocument 类的 print 函数时输出的是 Print from XmlDocument，即运行重定义之后的 print 函数。这是 Ruby 多态的一个体现方面。

2.2 Python

2.2.1 Python 的设计思想

Python 是一种面向对象、解释型的计算机程序语言。它的基本设计思想就是一切皆对象。这使得 Python 语言有很强大的动态特点。python 的对象里都有一个__dict__属性，是字典类型，记录这个对象所有属性，包括方法和属性，调用时，解析器只要搜索一下__dict__就可以了，还可以通过赋值改变这个__dict__的内容来达到改变这个对象，比如添加字段，这样有点 Java 的反射机制，不过 python 更灵活。所谓的动态，主要

是指变量的动态，python 中所有变量都没有类型，其实可以看成是一个引用或指针，每一个变量只是指向一个对象，而且它可以随时指向另一个对象，包括函数对象，这种灵活得益于 python 里对象的设计。

Python 包括了一组功能完备的标准库，且语法简单，与其他大多数程序设计语言使用大括号不一样，它使用缩进来定义语句块。Python 具有垃圾回收机制，支持命令式程序设计、面向对象程序设计、函数式编程、面向侧面的程序设计、泛型编程多种编程范式[3]。

可扩充性可说是 Python 作为一种编程语言的特色。新的内置模块（module）可以用 C 或 C++写成。而我们也可为现成的模块加上 Python 的接口。Python 可以使用户避免过分的语法的羁绊而将精力主要集中到所要实现的程序任务上。

2.2.2 Python 的面向对象特性

2.2.2.1 对象[4]

Python 语言是由对象组成，对象时 Python 语言面向对象编程的一种基本构建块。

object_types.py

```
#!/usr/bin/python
# object_types.py
import sys
def function(): pass
print type(1)
print type("")
print type([])
print type({})
print type(())
print type(object)
print type(function)
print type(sys)
```

在这个例子中，所有的字符实体事实上都是对象，type()函数返回对象的类型。

整形、字符串型、列表类型、字典类型、函数和模块都是 Python 中的对象。

2.2.2.2 封装

```
class Encapsulation(object):
    def __init__(self, a, b, c):
        self.public = a
        self._protected = b
        self.__private = c
```

Python 也可以用 public、protect 和 private 对对象进行可视性的修饰。

2.2.2.3 继承

inheritance.py

```
#!/usr/bin/python
# inheritance.py
```

```
class Animal(object):
   def __init__(self):
      print "Animal created"
   def whoAmI(self):
      print "Animal"
   def eat(self):
      print "Eating"
class Dog(Animal):
   def __init__(self):
      Animal.__init__(self)
      print "Dog created"
   def whoAmI(self):
      print "Dog"
   def bark(self):
      print "Woof!"
d = Dog()
d.whoAmI()
d.eat()
d.bark()
```

在这个例子中，定义了两个类:Animal 类和 Dog 类，Animal 类是基类，Dog 类是派生类，派生了继承了基类的操作，比如在 eat()函数中所示，在 whoAmI()函数中，派生类修饰了基类中存在的行为。并且，派生类通过一个新的 bark()函数扩展了基类的操作。

2.2.2.4 多态

```
#!/usr/bin/python
# polymorphism.py
class Animal:
   def __init__(self, name=''):
      self.name = name
   def talk(self):
      pass
class Cat(Animal):
   def talk(self):
      print "Meow!"
class Dog(Animal):
   def talk(self):
      print "Woof!"
a = Animal()
a.talk()
c = Cat("Missy")
c.talk()
```

```
d = Dog("Rocky")
d.talk()
```

在这个例子中，一共有两个物种，分别是 dog 和 cat，他们都是 Animal 类，dog 和 cat 这两类都继承了 Animal 类，他们都有 talk()这个操作，然而他们通过重定义这个操作使得他们分别在调用这个函数的时候有不同的输出。

2.3 PHP

2.3.1 PHP 的设计思想

PHP 是一种被广泛应用的开放源代码的多用途脚本语言，它可嵌入到 HTML 中，尤其适合 web 开发[5]。其设计之初的目标就是服务器端的 web 开发，PHP 原本的简称为 Personal Home Page，是拉斯姆斯·勒多夫为了要维护个人网页，而用 C 语言开发的一些 CGI 工具程序集，来取代原先使用的 Perl 程序。然而自 v3.0，Zeev Suraski 和 Andi Gutmans 的重写，使得 PHP 真正有了生机。5.0 系列版本，加入了真正面向对象的功能。

PHP 没有清晰的设计思想，早期的 PHP 是受到 Perl 的启发，带有“out”参数的巨大的标准库函数是源于 C 语言的，其面向对象部分设计模式类似于 C++和 Java。

2.3.2 PHP 的面向对象特性

2.3.2.1 封装

```
<?php
class animal{
public $name  = 'cat';//动物的公共属性，内外部可以访问、赋值
public $sex   = 'male';
public $voice = 'miao';
public function shout(){//动物的方法，内外部可以访问
echo $this->name;//调用属性时，不加再加$
echo ' says ';
echo $this->voice;
echo '<br />';
}
}
$animal = new animal;
$animal->name = 'littleCat';//外部赋值
$animal->shout();//外部调用方法
echo $animal->sex;//外部访问属性
echo '<br />';
?>
```

在上述代码中，有动物这个类，它有一些属性 name，sex 等，并且定义了 shout 这个方法，这个类就将类中的属性和方法都封装起来了。封装把对象的全部属性和全部方法结合在一起，形成了一个不可分割的独立单位（即对象）。

2.3.2.2 继承

```
class dog extends animal{
public $name  = 'dog';//重写原有属性
public $style = 'traditional';//新增属性
}
$dog = new dog;
$dog->shout();
echo $dog->style;
echo '<br />';
```

dog 类继承了 animal 类，这样，dog 拥有 animal 的属性和方法，不用再写很长一段代码了。继承的类可以理解为被继承者的特殊化，因为它除了拥有父类的所有特性（属性和方法），还具备自己独有的个性（比如改写原有的，新增自己的）。

2.3.2.3 多态

PHP 是弱类型语言，多态这个特性表现得不是很明显，java 的多态体现的十分清晰，大体分两类：父类引用指向子类对象；接口引用指向实现接口的类对象。java 声明变量时都要给变量设定类型，所以存在什么父类引用和接口引用。而 php 则没有这点体现，php 声明变量不需要给变量设定类型，一个变量可以指向不同的数据类型。所以，php 不具有像 java 一样的多态。

```
abstract class animal{
    abstract function fun();
}
class cat extends animal{
    function fun(){
        echo "cat say miaomiao...";
    }
}
class dog extends animal{
    function fun(){
        echo "dog say wangwang...";
    }
}
function work($obj){
    if($obj instanceof animal){
        $obj -> fun();
    }else{
        echo "no function";
    }
}
work(new dog());
work(new cat());
```

上面通过一个关键字 instanceof 来判断，变量指向的对象是否是 animal 类的一个实例，下面 new cat()，new dog()都是 animal 子类的对象，而输出了“dog say wangwang...”和“cat say miaomiao...”，说明子类对象是父类的一个实例，从而达到了 java 多态的功能。

上边的类是抽象类，也表明了接口与实现接口的类对象同样可以适用。

至此，得出 PHP 虽然多态体现模糊，但还是具有多态特性的。

3 Ruby、Python、PHP 和 Java 的面向对象特性对比

Ruby、Python、PHP 和 Java 都是面向对象程序设计语言，他们彼此之间有相似之处，也有较大的差异，表 3.1 总结了 Ruby、Python 和 Java 这三种语言的面向对象的详细内容上的相同和不同之处。

表 3.1 Ruby、Python 和 Java 面向对象特征的差异[6]

Eiffel	Ruby	Python	Java
Object-Orientation	Pure	Hybrid	Hybrid
Static / Dynamic Typing	Dynamic	Dynamic	Static
Generic Classes	N/A	N/A	No
Inheritance	Single class, multiple "mixins"	Multiple	Single class, multiple interfaces
Feature Renaming	Yes	No	No
Method Overloading	No	No	Yes
Operator Overloading	Yes	Yes	No
Higher Order Functions	Blocks	Lambda Expressions	No
Lexical Closures	Yes (blocks)	Yes (since 2.1)	No
Garbage Collection	Mark and Sweep	Reference Counting	Mark and Sweep or Generational
Uniform Access	Yes	No	No
Class Variables / Methods	Yes	No	Yes
Reflection	Yes	Yes	Yes
Access Control	public, protected, private	Name Mangling	public, protected, "package", private
Design by Contract	Add-on	No	No
Multithreading	Yes	Yes	Yes
Regular Expressions	Built-in	Standard Library	Standard Library
Pointer Arithmetic	No	No	No
Language Integration	C,C++,Java	C, C++, Java	C, some C++
Built-In Security	Yes	No?	Yes
Capers Jones Language Level*	N/A	N/A	6

注：N/A 表示条目或该特征对于该语言是不适用的。

许多编程语言声称自己是面向对象的（Object-Oriented）。但当询问“面向对象”的确切定义时，不同的人的答案区别很大，下面列举几条大家公认的，面向对象语言所应具有的性质：

1）Encapsulation/Information Hiding （封装/信息隐藏）

2）Inheritance （继承）

3）Polymorphism/Dynamic Binding （多态/动态绑定）

4）All pre-defined types are Objects （所有预定义类型皆对象）

5）All operations performed by sending messages to Objects （所有操作都由向对象发送消息实现）

6）All user-defined types are Objects （所有用户定义的类型都是对象）

如果一门编程语言满足了所有这些性质，我们就可以认为这门语言是“纯粹的”面向对象语言。一门“混合型”语言可能支持这些性质中的某几条，但不是全部。特别的，许多语言支持前三条性质，但不支持后三条。

表 3.2 列出了几种常见的编程语言对这 6 条性质的满足情况对比。

表 3.2 Ruby、Python、PHP 和 Java 面向对象性质差异

Eiffel	Ruby	Python	PHP	Java
Encapsulation / Information Hiding	Yes	No	Yes	Yes
Inheritance	Yes	Yes	Yes	Yes
Polymorphism / Dynamic Binding	Yes	Yes	Yes	Yes
All pre-defined types are Objects	Yes	Yes	No	No
All operations are messages to Objects	Yes	No	No	No
All user-defined types are Objects	Yes	Yes	Yes	Yes

Ruby 都是纯的面向对象语言，支持上表列举的所有 6 条性质。Java 声称自己是纯粹的面向对象语言，但是由于其“基本”数据类型并不是对象，它不满足我们的第 4 条性质。它也不满足第 5 条性质，因为其基本的算术运算是内置的运算符，而不是对象的消息。

Python 经常被标榜为一种面向对象语言，但它对面向对象的支持似乎是被“贴上去的”。一些操作以方法的形式实现，而另一些则用的是全局函数。有一些人抱怨 Python 缺乏“private”或者“隐藏”属性，这违背了封装/信息隐藏的原则，而另一些人则认为“通过约定实现的私有”在不增加额外限制的同时，也提供了语言强制的封装带来的优势。在另一方面，Ruby 语言的发明一部分是由于 Python。Ruby 的设计者认为他想要创造一种“比 Perl 强大，比 Python 更面向对象”的语言。

PHP 的面向对象和 Java 相比有很多相似之处，可以说 Java 的面向对象编程纯度远远高于 PHP，他们的相同点是都不允许多重继承，不能继承父类的私有属性或者方法，但是允许多层继承。Java 中子类实例化先调用父类的构造方法，再调用自己的构造方法；但是 PHP5.3 是不会自动调用父类的构造方法，PHP 如果要调用父类的构造方法需要使用 parent::__construct()或者父类名称::__construct()，但是 PHP5.4 和 java 一样是会实例化父类的构造方法的。Java 中重载是可以实现的，但是 PHP 只能模拟重载。对于抽象类，这一块 Java 和 PHP 都是一样的，同样的抽象类里面可以没有抽象方法，但是如果一个类里面如有抽象方法那么这个类就一定是抽象类，还有就是一个类如果继承了抽象类就必须实现里面的所有抽象方法，最重要的一点就是抽象类里的抽象方法只允许声明不允许实现。

4 结论

本文主要介绍了面向对象程序语言的特点和基本的面向对象特性，主要对 Ruby、Python、PHP 和 Java 着四种语言的面向对象特性进行详细介绍，并比较他们的面向对象性质和这些语言的差异。如今面向对象编

程语言已经是程序设计的主流编程语言了，面对对象编程技术也越来越广泛地应用到各个特研究领域了。

References:

[1] https://zh.wikipedia.org/wiki/Ruby.

[2] https://devblast.com/b/ruby-inheritance-encapsulation-polymorphism

[3] https://zh.wikipedia.org/wiki/Python.

[4] http://zetcode.com/lang/python/oop/.

[5] http://www.cnblogs.com/atricfox/archive/2012/10/18/2729139.html.

[6] http://www.jvoegele.com/software/langcomp.html.