文章编号：2096-1472(2017)-08-01-03

基于Java与Python的面向对象编程的基本特征研究

韩宏峰，冯 石，罗羿隆
(北京邮电大学，北京 100876)

摘 要：Java与Python作为两种主流的不同类型的面向对象编程语言，有较深的研究价值。本文简述并比较了面向对象编程语言的继承与多态，Java因只支持单继承而与Python等系列语言不同，因此Java通过设计接口以间接实现多继承。另一方面，本文阐述了垃圾回收机制的意义、两种回收方法和主GC的触发条件，并与Python进行了对比，最后进行了Java内存的概况和结构分析。

关键词：面向对象编程；Java与Python；继承与多态；接口；垃圾回收机制
中图分类号：TP312 **文献标识码**：A

Research on the Basic Characteristics of Object-Oriented Programming Based on Java and Python

HAN Hongfeng,FENG Shi,LUO Yilong
(*Beijing University of Posts and Telecommunications*,*Beijing* 100876,*China*)

Abstract: Java and Python are two kinds of mainstream object oriented programming languages with comparatively higher research value.This paper sketches and compares the two major features of object oriented programming languages:inheritance and polymorphism.Different from Python and other programming languages,Java only supports single inheritance of classes,so Java indirectly implements multiple inheritance through the design interface.This paper also states the significance of the garbage collection mechanism,two kinds of recovery methods and the trigger condition of the main GC of Java,and then compares with Python.At last,the general situation and structure analysis of Java memory are carried out.

Keywords:object oriented programming;Java and Python;inheritance and polymorphism;interface;garbage collection mechanism

1 引言(Introduction)

程序设计的思想通常分为面向过程的编程和面向对象的编程[1]。面向过程的编程以函数为主，通常以线性步骤为特征，设计起来较为繁杂，需要具备扎实的基础，以C语言最为流行，是操作系统的设计语言；而在面向对象编程中，对象指类的实例，通过将对象作为程序的基本单元来提高软件的灵活性。其中面向对象的三个基本特征是：继承、封装和多态[2]。Java与Python作为两种主流的不同类型的面向对象编程语言，具有较深的研究价值。本文在论述时引入了静态语言和动态语言的概念来区分Java与Python的不同特征，在讨论多继承时通过接口的引入解决了Java单继承的局限性，间接实现了Python的多继承。另一方面，本文对Java的垃圾回收机制进行了详细解析并对比了Python。最后介绍了Java的内存管理模块。

2 多态与继承(Polymorphism and inheritance)

2.1 多态

多态是面向对象语言里一个常见的概念，指的是同名而内容不同的方法同时存在于一个程序中。在Java中，多态分为动态和静态[3]。动态指的是在程序运行时才可以通过虚函数或重写来动态地决定指针指向的对象。静态指的是在编译时系统能通过重载决定调用的函数名[4]。Java属于一种要求在编译时变量的数据类型必须确定的语言，即静态语言[5]。动态语言(Python)与静态语言(Java)相比，其不必检查变量的数据类型，只要方法存在并且参数正确便可直接调用。如下面代码所示，参数self可以是任何数据类型的实例，只要有getName()的方法即可。

```
class student(object):
    def getName(self):
        return'I am a student'
```

基金项目：中央高校基本科研业务费专项资金资助(supported by "the Fundamental Research Funds for the Central Universities").

2.2 继承

Java作为一种面向对象的语言，只支持单继承，即一个子类不能对应多个父类。好处在于单继承能使java的继承关系变得简单，程序变得更易于管理，而对多继承的需求可以通过接口实现。通过多重继承，一个子类就可以同时获得多个父类的所有功能。Python作为一种支持多继承的语言，子类如果没有写自己的初始化方法，会自动继承第一个父类的方法。为了更好地设计，通常进行一种称之为Mixin[6,7]的设计来减少多层次的复杂的继承关系。

3 Java的接口(Java interface)

3.1 接口的引入与形式

正如2.2节继承所述，Java不支持类的多重继承，这使得程序的结构更加简洁。但是和支持多重继承的其他语言相比，单继承有时并不能很好的表述比较烦琐的问题。在这种情况下，接口可以用于实现类似于多重继承的功能。接口是一种抽象的数据类型，也就是说它并不能实例化。抽象是接口的一大特点，因此在接口中并没有方法和变量，只有方法的定义与常量。声明接口和声明一个类十分相似。其中接口为一个抽象类，只有虚函数和静态数据被声明，相当于定义了一个程序之间的协议[8,9]。

3.2 接口的实现

接口由于自身只是一种抽象的数据类型，能避免多继承带来的许多矛盾。一个类可以通过实现多个接口来实现诸如Python的“多继承”功能。在具体使用中，接口的用法和类很相似，虽然不能直接对接口使用new操作符，但是可以使用接口作为类型名。

3.3 接口的多继承

如图1所示，一个接口允许同时继承(extends)多个接口。多继承使得多个接口规范能得到合并。程序的抽象结构层次便能产生于此。而由于接口中的方法只是抽象的原型，所以避免了重定义父类方法和重复继承的问题，在结构上相对简单[10]。

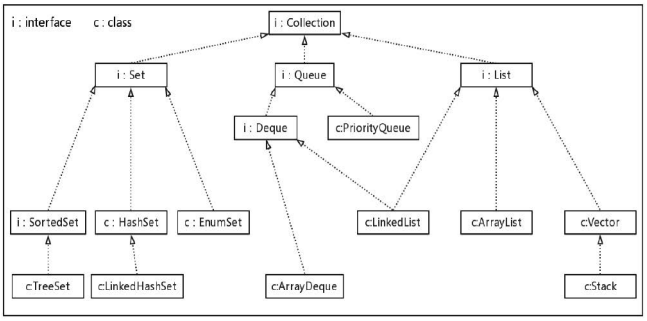

图1 接口的多继承

Fig.1 Multiple inheritance of interfaces

3.4 接口的应用

接口在实际应用中能够对类的特征进行一系列的描述。在Java中，接口也可以作为一个类使用，只是不能被new操作符直接声明。在运行程序时，调用的方法属于哪个类是在运行的时候被决定，使方法能被动态地调用。

4 垃圾回收机制(Garbage collection mechanism)

4.1 Java垃圾回收机制

4.1.1 垃圾回收的意义

Java的特色之一就是垃圾回收机制，这使得编程时令人头痛的内存管理问题得以解决，使得程序员的编程压力大大减小[11]。垃圾回收机制在一个变量或对象的生命周期结束后释放它所占用的资源。垃圾回收机制可以有效地利用空闲的内存，从而防止内存的浪费。由于现代计算机的性能不断提高，现在进行垃圾回收所需要的时间非常少，在这种情况下，就可以频繁的对程序进行垃圾回收。

4.1.2 垃圾回收方法

(1)finalize()方法

Java通过调用finalize()方法来回收垃圾，例如打开的URL、端口、文件等，但其只能由垃圾收集器被动调用。

(2)System.gc()方法

如果想要在程序中主动请求垃圾回收可以使用System.gc()方法。具体格式如下：

```
1. class TestGC
2. {
3.     public static void main(String[] args)
4.     {
5.         new TestGC();
6.         System.gc();
7.         System.runFinalization();
8.      }
9. }
```

(3)主GC

之前所说的GC对系统的影响都非常小，但是主GC可以对系统产生较大的影响，所以它只能在特定的情况下才被触发。触发主GC的条件有两个：

(1)在应用程序没有运行时，即应用线程都处于空闲状态时，GC会被调用。

(2)堆内存不足时，JVM会强制调用主GC来回收内存[12]。

4.2 Python垃圾回收机制

简洁作为Python的最重要特征，程序员同样无须关心对象的内存分配和释放等原理，取而代之的由Python解释器负责，是一种自动回收内存资源的技术。最简单的GC算法就是引用计数，一种效率不高的实现算法，意味着在每次内存对象被引用或引用被销毁的时候都必须修改引用计数，极大的影响性能。未来的Python解释器也许会采用更高效的方法来实现垃圾收集[13]。

5 Java内存管理(Java memory management)

5.1 Java内存概况

在Java中，内存是通过new分配给对象。例如：object obj=new object()而且在程序实际运行的过程中，每个对象所

拥有的内存也是动态的。上文提到的垃圾处理机制是JAVA的一个显著的特色机制，它能使系统资源得到充分的利用，同时也防止了由程序员错误操作引起的故障。垃圾回收机制有效地防止了内存的浪费，大大降低了编程的烦琐度，避免了内存分配不均导致的系统崩溃。

5.2 Java内存结构

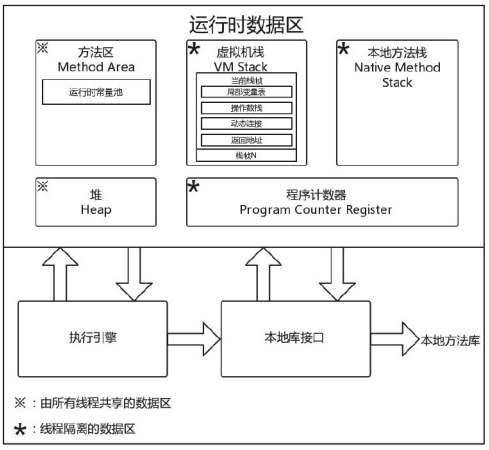

图2 内存结构

Fig.2 Memory structure

如图2所示，Java内存主要分为四个部分：堆、栈、方法区和程序计数器。其中方法区和堆是用于给进程分配空间，也是所有线程共享的。而栈和程序计数器是分配资源给每一个独立的线程。

(1)方法区

这是线程共享的区域，通常来说这个区域很少出现垃圾收集行为。若出现内存已满不足以进行新的内存分配时，程序将会抛出OutOfMemoryError异常。

(2)堆

堆用于给进程分配空间使用，是垃圾回收处理的主要区域。从内存分配上来说，Java堆可以分成多个线程独有的分配缓冲区，但是这些缓冲区存放的内容都是线程中的对象实例，之所以细分成独立的区域是为了更快的分配和回收。如果Java堆不能进行内存扩展，将会抛出OutOfMemoryError异常。

(3)程序计数器

程序计数器分配给每一个独立的线程，作为当前线程所运行的字节码的指示器。在线程执行方法时，程序计数器将指向运行代码的地址。

(4)栈

栈也是分配给每个线程的内存空间，它描述的是每个Java方法执行时调用内存的过程。栈可能抛出StackOverflowError和OutOfMemoryError两种异常[14]。

6 结论(Conclusion)

本文对基于面向对象编程的Java与Python进行了基本特征研究，在介绍OOP的三个基本特征继承、封装和多态时引入了静态语言和动态语言的概念，在讨论多继承时通过接口的引入解决了java单继承的局限性，实现了类的多继承。同时对Java的垃圾回收机制进行了详细解析，并与Python进行了对比。最后介绍了Java的内存管理模块。本文旨在让读者了解基本的面向对象编程的特征，熟悉两种代表性的语言Java和Python，为今后的编程打下良好的基础。

参考文献(References)

[1] 张丰,等.面向对象的地籍时空过程表达与数据更新模型研究[J].测绘学报,2010,39(03):303-309.

[2] Deshpande S.Collaboration of Object Oriented Programming and Software Development[J].International Research Journal of Engineering and Technology(IRJET),2016,03(09):524-527.

[3] 余双双,等.基于UML模型的多态性与Java接口代码信息一致性检测的方法[J].计算机应用与软件,2017,34(02):8-13;47.

[4] 黎海生.Java语言中的继承与多态[J].科技广场,2008(05):231-232.

[5] 钱宇虹.基于Java平台的多语言混合编程[J].软件工程师,2014(11):39-41.

[6] Burton E,Sekerinski E.An Object Model for Dynamic Mixins[D].Computer Languages,Systems & Structures,2017.

[7] Burton E,Sekerinski E.An object model for a dynamic mixin based language[C].Proceedings of the 31st Annual ACM Symposium on Applied Computing.ACM,2016:1986-1992.

[8] 杨晓霞,候锐锋.VisualC#.NET,DELPHI,JAVA与MATLAB接口技术的研究[J].云南大学学报(自然科学版),2008,30(S2):247-249.

[9] 姜慧霖,乔丽.浅析C++和Java的继承机制[J].开封大学学报,2005(03):85-87.

[10] Kramer S.A modularity bug in Java 8[J].arXiv preprint arXiv:1701.02189,2017.

[11] 张卫.综述java运行中垃圾回收机制[J].数字技术与应用,2017(02):231.

[12] 池炜成.Java垃圾收集的机制及调优[J].计算机应用研究,2004(03):144-148.

[13] 郭芬,刘明.Python垃圾收集器原理研究及应用[J].信息技术,2009(07):93-97.

[14] Dietrich J,Jezek K,Brada P.What Java developers know about compatibility,and why this matters[J].Empirical Software Engineering,2016,21(3):1371-1396.

作者简介：

韩宏峰(1996-)，男，本科生.研究领域：物联网工程，编程语言，人工智能，区块链.

冯　石(1994-)，男，本科生.研究领域：物联网工程.

罗羿隆(1996-)，男，本科生.研究领域：物联网工程.