Think In Python

序言(Preface)

本书的材料来自于Java的讨论会，关于本书我已考虑了好多年，与我研讨的对象是Larry O’Brien和 Bill Venners，Bill和我已经多次反复的讨论，并且我们我们的方案已经改变了很多次。在过去几年，我们通过研讨都学到更多的模式。

在创作本书的过程中我们得到了足够多的信息，这是因为我们个人都有自己的研讨会。有些研讨会被我们拒绝了，因为我俩之间的研讨会更有趣。(In the process we’ve both produced more than enough information for us each to have our own seminars, an urge that we’ve both strongly resisted because we have so much fun giving the seminar together.本段翻译我不知道是否正确)在美国的很多地方都召开这样的研讨会，比如在Prague（每年春天我们都会举行一个小型的研讨会，同时还有一些其他的研讨会也在这里举行。）偶尔我们也会举行本地的研讨会，但是对于我们两个组织者来说这是相当困难而且费用昂贵的。

我非常感谢这些年来参与我的研讨会的人，特别是Larry O’Brien和 Bill Venners。他俩对我的工作提出了很多建议和改进意见。我希望今后能够继续通过本书和研讨会得到各种各样的意见。

无论怎样我的写书是不会结束的。最初，书里这些资料用于我写的C++的书,然后是Java的书，然后终止于基于Java的书籍。但是最后，经过一番思考，我决定首先使用Python（从一开始我就知道Python是原型类语言）为基础创建我的设计模式类论文，然后把这些再用java重新写一本书。我有这方面的经验，首先使用一种强大的语言来写程序，然后把它翻译成另外一种语言，这样更容易让读者了解、明白你的意图(想法 idea)。

所以Think in Python这本书不是一本介绍Python语言的的书(市面上，已经有了很多介绍这个卓越语言的书。)这本书将来会翻译成使用Java为语言基础介绍设计模式的书。我感觉我的这个麻烦的主意更加令我兴奋，这比制作一个语言教程令我兴奋(虽然这可能是很多人期望得到的教程)。

引言（Introduction）

这是一本我为之努力了多年的关于设计模式的书，基本上从我开始读 GOF 写的《设计模式》（这本书是 1995 年，由 Gamma, Helm, Johnson & Vlissides, AddisonWesley 四人合写，通常称为他们为四人帮或简写成 GoF(*注 1））开始就有了这个想法。

在第一版 Think In C++里面有一章是讲设计模式的，这一章后来演变成 Think In C++的第二版的第二卷。而且，你有可能在第一版 Think In Java 中看到有一章也是讲设计模式的。有关设计模式这一章，我在 Think In Java 的第二版中把它删除了，被删除的原因一个是由于 Think In Java 这本书变得越来越大，更重要的原因是我已经计划写一本有关设计模式的书。这本书最终被完成了，就是这本 Think In Python.最后，请允许我使用 Python（这个便于表达这些更复杂的想法的语言）来把我思考的这些设计模式全部写出来。

这不是一个介绍性的书。我假设你会使用 Python，不管你是通过什么途径学习的这们语言，比如是 Mark Lutz 和 David Ascher 在 1999 年写的由 OReilly 公司出版的《Python 语言入门》，或者与这类似的书。

此外，我假定您不仅仅是掌握了 Python 的语法 。您应该已经很好的理解了什么是对象以及面向对象的各种特征，包括多态性等。(*注 2，译者加)

另一方面，通过这本书你会学到很多关于在不同情况下如何看面向对象的编程方法。如果你对面向对象之具有初步的了解，那么通过学习本书的过程，能够增加你的关于面向对象的知识。

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

注 1：四人帮(Gang of Four):这个挖苦性词语产生于中国的领导人毛泽东逝世以后。当时的四个人包括毛泽东的夫人(一个很有权力的女演员)，这是中国共产党丑化他们的一个词语

注 2：对象(object)是一件事、一个实体、一个名词，可以获得的东西，可以想象有自己的标识的任何东西。对象是类的实例化。一些对象是活的，一些对象不是。比如这辆汽车、这个人、这间房子、这张桌子、这株植物、这张支票、这件雨衣。概括来说就是：万物皆对象。
面向对象(ObjectOriented,OO)是当前计算机界关心的重点，它是 90 年代软件开发方法的主流。面向对象的概念和应用已超越了程序设计和软 件开发，扩展到很宽的范围。如数据库系统、交互式界面、应用结构、应用平台、分布式系统、网络管理结构、CAD 技术、人工

智能等领域。

面向对象的基本概念

(1)对象。

对象是人们要进行研究的任何事物，从最简单的整数到复杂的飞机等均可看作对象，它不仅能表示具体的事物，还能表示抽象的规则、计划或事件。

(2)对象的状态和行为。

对象具有状态，一个对象用数据值来描述它的状态。

对象还有操作，用于改变对象的状态，对象及其操作就是对象的行为。

对象实现了数据和操作的结合，使数据和操作封装于对象的统一体中

(3)类。

具有相同或相似性质的对象的抽象就是类。因此，对象的抽象是类，类的具体化就是对象，也可以说类的实例是对象。

类具有属性，它是对象的状态的抽象，用数据结构来描述类的属性。

类具有操作，它是对象的行为的抽象，用操作名和实现该操作的方法来描述。

(4)类的结构。

在客观世界中有若干类，这些类之间有一定的结构关系。通常有两种主要的结构关系，即一般--具体结构关系，整体--部分结构关系。

① 一般——具体结构称为分类结构，也可以说是“或”关系，或者是“is a”关系。

② 整体——部分结构称为组装结构，它们之间的关系是一种“与”关系，或者是“has a”关系。

(5)消息和方法。

对象之间进行通信的结构叫做消息。在对象的操作中，当一个消息发送给某个对象时，消息包含接收对象去执行某种操作的信息。发送一条消息至少要包括说明接受消息的对象名、发送给该对象的消息名（即对象名、方法名）。一般还要对参数加以说明，参数可以是认识该消息的对象所知道的变量名，或者是所有对象都知道的 全局变量名。

二、面向对象的特征

(1)对象唯一性。

每个对象都有自身唯一的标识，通过这种标识，可找到相应的对象。在对象的整个生命期中，它的标识都不改变，不同的对象不能有相同的标识。

(2)分类性。

分类性是指将具有一致的数据结构(属性)和行为(操作)的对象抽象成类。一个类就是这样一种抽象，它反映了与应用有关的重要性质，而忽略其他一些无关内容。任何类的划分都是主观的，但必须与具体的应用有关。

(3)继承性。

继承性是子类自动共享父类数据结构和方法的机制，这是类之间的一种关系。在定义和实现一个类的时候，可以在一个已经存在的类的基础之上来进行，把这个已经存在的类所定义的内容作为自己的内容，并加入若干新的内容。

继承性是面向对象程序设计语言不同于其它语言的最重要的特点，是其他语言所没有的。

在类层次中，子类只继承一个父类的数据结构和方法，则称为单重继承。

在类层次中，子类继承了多个父类的数据结构和方法，则称为多重继承。

在软件开发中，类的继承性使所建立的软件具有开放性、可扩充性，这是信息组织与分类的行之有效的方法，它简化了对象、类的创建工作量，增加了代码的可重性。

采用继承性，提供了类的规范的等级结构。通过类的继承关系，使公共的特性能够共享，提高了软件的重用性。
(4)多态性(多形性)
多态性使指相同的操作或函数、过程可作用于多种类型的对象上并获得不同的结果。不同的对象，收到同一消息可以产生不同的结果，这种现象称为多态性。
多态性允许每个对象以适合自身的方式去响应共同的消息。
多态性增强了软件的灵活性和重用性。
三、面向对象的要素
(1)抽象。
抽象是指强调实体的本质、内在的属性。在系统开发中，抽象指的是在决定如何实现对象之前的对象的意义和行为。使用抽象可以尽可能避免过早考虑一些细节。
类实现了对象的数据（即状态）和行为的抽象。
(2)封装性（信息隐藏）。
封装性是保证软件部件具有优良的模块性的基础。
面向对象的类是封装良好的模块，类定义将其说明（用户可见的外部接口）与实现（用户不可见的内部实现）显式地分开，其内部实现按其具体定义的作用域提供保护。
对象是封装的最基本单位。封装防止了程序相互依赖性而带来的变动影响。面向对象的封装比传统语言的封装更为清晰、更为有力。
(3)共享性
面向对象技术在不同级别上促进了共享
同一类中的共享。同一类中的对象有着相同数据结构。这些对象之间是结构、行为特征的共享关系。
在同一应用中共享。在同一应用的类层次结构中，存在继承关系的各相似子类中，存在数据结构和行为的继承，使各相似子类共享共同的结构和行为。使用继承来实现代码的共享，这也是面向对象的主要优点之一。
在不同应用中共享。面向对象不仅允许在同一应用中共享信息，而且为未来目标的可重用设计准备了条件。通过类库这种机制和结构来实现不同应用中的信息共享。
4.强调对象结构而不是程序结构
四、面向对象的开发方法
目前，面向对象开发方法的研究已日趋成熟，国际上已有不少面向对象产品出现。面向对象开发方法有 Coad 方法、Booch 方法和 OMT 方法等。
1.Booch 方法
Booch 最先描述了面向对象的软件开发方法的基础问题，指出面向对象开发是一种根本不同于传统的功能分解的设计方法。面向对象的软件分解更接近人对客观事务的理解，而功能分解只通过问题空间的转换来获得。
2.Coad 方法
Coad 方法是 1989 年 Coad 和 Yourdon 提出的面向对象开发方法。该方法的主要优点是通过多年来大系统开发的经验与面向对象概念的有机结合，在对象、结构、属性和操作的认定方面，提出了一套系统的原则。该方法完成了从需求角度进一步进行类和类层次结构的认定。尽管 Coad 方法没有引入类和类层次结构的术语，但事实上已经在分类结构、属性、操作、消息关联等概念中体现了类和类层次结构的特征。
3.OMT 方法

OMT 方法是 1991 年由 James Rumbaugh 等 5 人提出来的，其经典著作为“面向对象的建模与设计”。

该方法是一种新兴的面向对象的开发方法，开发工作的基础是对真实世界的对象建模，然后围绕这些对象使用分析模型来进行独立于语言的设计，面向对象的建模和 设计促进了对需求的理解，有利于开发得更清晰、更容易维护的软件系统。该方法为大多数应用领域的软件开发提供了一种实际的、高效的保证，努力寻求一种问题 求解的实际方法。

4.UML(Unified Modeling Language)语言

软件工程领域在 1995 年～1997 年取得了前所未有的进展，其成果超过软件工程领域过去 15 年的成就总和，其中最重要的成果之一就是统一建模语言 (UML)的出现。UML 将是面向对象技术领域内占主导地位的标准建模语言。

UML 不仅统一了 Booch 方法、OMT 方法、OOSE 方法的表示方法，而且对其作了进一步的发展，最终统一为大众接受的标准建模语言。UML 是一种定义良好、易于表达、功能强大且普遍适用的建模语言。它融入了软件工程领域的新思想、新方法和新技术。它的作用域不限于支持面向对象的分析与设计，还支持从需 求分析开始的软件开发全过程。