DVD 超值多媒体光盘

✓ 这是一本看得懂的书
✓ 这是一本学得会的书
✓ 这是一本用得着的书

Java面向对象B/S后台开发精粹

王剑南 等编著

清华大学出版社

Java面向对象B/S后台开发精粹

王剑南 等编著

清华大学出版社
北京

内 容 简 介

本书介绍了通过 Java 语言进行 B/S 服务器端后台开发的知识，全书主要内容包括 Java 面向对象编程基础，着重介绍了 Java 编程规范，Java 面向对象编程三大重要基础——类、接口和抽象类，I/O 文件读取等基础知识；书中还介绍了数据库开发与数据库连接的知识。书中最后介绍了 Java 网络编程知识，包括 Servlet，JSP 和开发架构等内容。本书凝聚了作者长达 20 年的开发和教学培训经验，语言丰富幽默，讲解突出重点，是面向程序员实训的典范之作。

本书适合高校软件工程专业本科和研究生学习使用，也适合在职软件工程师工作参考。

图书在版编目（CIP）数据

Java 面向对象 B/S 后台开发精粹 / 王剑南等编著. —北京：清华大学出版社，2013.4
ISBN 978-7-302-28227-3

Ⅰ. ①J…　Ⅱ. ①王…　Ⅲ. ①JAVA 语言-程序设计　Ⅳ. ①TP312

中国版本图书馆 CIP 数据核字（2012）第 038849 号

责任编辑：夏兆彦
封面设计：柳晓春
责任校对：徐俊伟
责任印制：杨　艳

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　邮　　编：100084
社 总 机：010-62770175　　邮　　购：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质 量 反 馈：010-62772015，zhiliang@tup.tsinghua.edu.cn

印 装 者：北京鑫海金澳胶印有限公司
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　　张**：14.25　　**字　　数**：338 千字
附光盘 1 张
版　　次：2013 年 4 月第 1 版　　**印　　次**：2013 年 4 月第1 次印刷
印　　数：1～4000
定　　价：39.00 元

产品编号：042861-01

序

紫光软件是一个国内存在了很久的IT企业，而且是这些企业中，软件开发所占经营份额比较多的企业。紫光软件开发过很多国内政府、军队以及上市公司的大型系统，虽然不能说是业内的龙头，但是作为软件开发的龙头之一还是没什么争议的。

软件开发的核心是人员，可以说人员直接决定了软件的成败。于是紫光软件同其他的软件开发公司一样极为重视开发人员的培养。作为紫光软件的总裁，我发现了一个很有趣的现象——既无论多好的大学，无论学习什么专业，只要是应届生都无法快速地投入到真正的开发工作中，最快的应届生如果想要对最基本的项目有所贡献都需要3个月以上的时间，而绝大多数人是半年。而软件行业另一个有趣的现象是新人差不多半年就会跳槽。于是新进人员的培养和保有就成为软件企业非常沉重的负担。这也是为什么大多数的软件企业招聘时强调开发人员具有工作经验的原因。而这种有工作经验人员的成本将远远高于应届毕业生。

但是因为项目的周期、成本等原因，过多的保有程序员将会使公司的经营承担很大压力。为此紫光软件成立了自己的培训中心，希望以此能快速地获得可用的程序开发人员，为此这个团队已经付出了4年的努力。

这本书的出版意味着，这几年的教学研究终于划上了一个比较完美的句号。这本书所讲述的并非是软件开发中最尖端的技术、也不是最新的开发模式，而是将软件开发以一种新入门的菜鸟能够看得懂、学得会的方式展现在大家面前。这本书的内容非常简单，甚至我认为高中生都能读懂，但是将软件这种高科技技术以一种简单到极点的方式展现，本身就说明了它的不简单。从这本书在培训中试用的成果来看，读者只要学习并达到本书中的要求，几乎没有任何一个软件开发企业会对他的技术水平有所抱怨。顺便说一句，紫光软件培训中心的学员不但为紫光软件提供了很多合格的软件开发人员，同时也为数十家软件公司输送了大量的合格程序员，很多早期毕业的学员都已经走上了项目经理的工作岗位。

最后说一下本书的作者，这个胖子已经在紫光软件工作了近十个年头，参与、领导了很多大型项目的开发工作，做过项目经理、售前工程师、咨询师、系统分析师甚至架构师等技术职位，在紫光软件这样的大集体中也算是一个难得的多面手。如果说开发的技术水平也许他不是紫光中最优秀的一个，但是他绝对是所有紫光软件技术开发人员中最能交流的一个，整天都会说个不停，也许正是因为他爱好交流，才能将一般人认为非常复杂的技术，讲解得如此简单。

最后祝本书的读者能够快速的成为软件开发大军中的一员，实现大家的人生理想。

王依群

紫光股份专务副总裁

紫光软件集团董事长兼总裁

前　言

自从开始打算写 Java 教材开始，一本书的名字就不断出现在我脑海，这本书可以说是所有 Java 程序员的圣经——Thinking in Java，中文也叫《Java 编程思想》。

如果让我选择世界上对我影响最大的一本技术书籍的话，我肯定会说就是大神 Bruce Eckel 所写的 Thinking in Java，对于这本书我认为从学习面向对象到掌握 Java 编程原理来说已经足够了，基本可以说是一书在手天下我有。

那么，我能够写一本比圣经还好的书吗？这点上我可以负责任地说："不能"，那写一本比已有的书更差的书到底有什么意义呢？我想了很久，最后找到了以下几个借口。

（1）Thinking in Java 的阅读者其实是程序员，而且是水平不低的程序员，其目的是将 C 或 C++程序员引入 Java 编程，所以 Thinking in Java 对于初学者来说还是有点深了。

（2）Thinking in Java 针对的是 J2SE 编程，而我们希望通过教材让大家掌握的是 J2EE，Thinking in Java 没有这部分内容。

（3）Thinking in Java 是一本圣经，也就是大而全的一本秘籍，但是对于快速掌握 Java 编程来说，其重点不够突出，实战性不是很强。这不是说这本书有缺陷，而是写这本书的目的就是如此。

为此，我只好冒着被同行痛扁的风险开始写一个能够快速了解面向对象、掌握 J2SE 和 J2EE 编程的书籍。

最后希望学习完这本书大家都去读一下 Thinking in Java，但是建议读其中文第一个版本和英文版，后面翻译的版本动辄 800～900 页，增加了诸多更新的技术和理念，但我倒觉得冲淡了其作为理念经典的光芒，很是有点本末倒置（不过也能理解，作者大神也是人，再版如果不出点新咚咚，也卖不出个好价钱，可以理解、可以理解）。还有就是后续版本还有我国台湾人翻译的版本，不是说歧视台湾人，只不过他们的表达方式和对英文的翻译实在是与大陆差别太大（他们把面向对象翻译成面对物件，这个我实在是觉得很无语），看着太过辛苦。

目　录

第 1 章　面向对象初探

面向对象在英文中称作 Object Oriented，从其概念提出的那一刻起对计算机软件界就产生了巨大的影响。从最初的面向对象语言——small talk（公认的面向对象的鼻祖语言，一种美国军方为了航天器所开发的语言）到现在几乎每种高级编程语言都有其面向对象语言的变种，C 对应 C++、PASCAL 对应 Delphi、Basic 对应 VB 等等。现今的编程行业中可以说程序员绝大多数是使用面向对象语言开发的（当然我认为使用面向过程的程序员都是高手，如果能使用低级语言，比如汇编更是老神仙一级的人物，可惜高手和神仙都是很少的）。

Java 作为一种非常有代表性的面向对象编程语言，我们在讲解 Java 之前就必须对面向对象及其相关的基本概念有一定的了解。

1.1　什么是对象

“对象是什么”，这个在无数的面向对象方面的图书中，有无数的定义。在此我并不想重复一些大神对对象的定义，这样不但有赚取稿费的嫌疑，同时也浪费大家的时间。为此我就说些我个人的观点，也许有些问题，也并不一定十分的严密，但我想对于读者来说应该更容易理解。

所谓对象（Object）这个词不但是个程序名词，在现实中也具有其含义，在十几年前的中国还着重代表过男女朋友。所以我希望大家对于对象的理解不要仅仅局限在程序中，因为对于面向对象而言，其设计方式应该与现实相对应。“对象”在我国台湾翻译成为“物件”。无论是对象还是物件，其实都是一个东西。对于整个世界来说，万物皆对象，其完全符合唯物论的思想。大到宇宙，小到原子每一个东西都是一个对象。从这上面讲，大家应该理解的就是，面向对象其实就是构建一个由程序所组成的世界。可以说面向对象程序的编程就是对象的创建过程，而面向对象程序的运行，就是对象间的相互作用。

一般而言讲到面向对象，就要讲到面向过程，这两种思想是两种完全不同的学术流派，在几年前还有激烈的争吵。一方面面向对象认为面向过程的程序太过复杂、不容易维护；而另一方面面向过程认为面向对象的程序速度太慢、占用硬件资源过多。这两派的争吵就像真的不共戴天一样，对此我当时就很疑惑，面向对象就没过程吗？那 if…else 算什么？面向过程就没对象吗？那函数怎么说？

最后，我不是从计算机方面想明白了这个问题，反而是从哲学角度想清楚了这个问题，面向对象和面向过程的关系与唯物和唯心的关系一样，并非是唯物不承认精神，也并非唯心不承认存在，只是谁优先的问题，说白了也就是出发点问题。面向对象和面向过程也一样，首先对象和过程在任何程序中都会存在，这两个东西不存在本质上有对象没过程，有过程没对象的关系；其次面向对象和面向过程只不过是在编程设计时谁更优先的问题，就

好像唯物和唯心中存在和精神谁优先的问题一样。

我想我需要再次强调一下，这些观点不是主流观点，更多的算是作者的主观臆测，所以也不必讨论这些观点的出处。本理论纯属原创，如有雷同必属山寨。对于面向对象我一直不认为它是一种技术，我认为面向对象和面向过程就是编程设计中的思路问题。当然有些人会说，面向对象语言有明确定义，比如继承、比如多态等等（一共有四个特点，作为不希望 Copy 赚取稿费的我来说，我就不把它罗列出来了，何况其前面两条我认为面向过程也有，不具有鲜明的特点，大家有兴趣可以去网上阅读），但那是面向对象语言的定义，只能说明这种语言支持面向对象的编程方式，并不代表使用面向对象语言就是面向对象的编程，其逆定理并不成立（李白好酒，但酒鬼多了，可不都是李白）。

所以我认为面向对象编程和面向过程编程其本质就是在程序设计上的差别，我认为一个使用面向对象语言（比如 Java）编程，但所有功能都在一个类中的一个对象中实现的程序，那就实在算不上是面向对象编程（这里指复杂程序，Hello World 不算，当然如果你能把 Hello World 都写成面向对象，我只能说你好生无聊）。

讲了半天我想我希望表达的意思已经很清楚了，在此作个总结。

（1）万物皆对象，面向对象的核心就是通过创建对象、使用对象，来完成我们希望的工作。

（2）面向对象和面向过程并非完全否定对方的存在，只是谁更优先的问题。

（3）面向对象编程不仅仅是技术而更多是一种设计思想。

讲了一通定义，也不知道大家能不能明白，不如举一个现实的例子，这样更有助于让大家理解对象和过程在设计上的区别。

现在假定有个客人到访，而我们希望杀一只鸡作为晚餐，我们看看面向对象和面向过程思路上的差别，如图 1-1 所示。

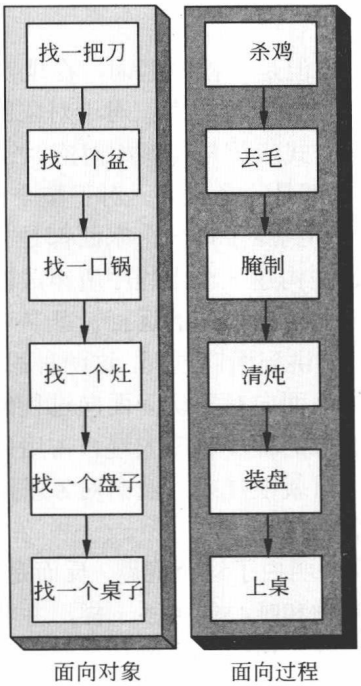

图 1-1

从上图我们可以看出，对于晚饭吃鸡这件事面向对象的思路是找到相应的工具也就是对象，而面向过程正好相反，是要找到具体的流程。

从这个例子可以发现面向对象最大的好处，就是复用（程序对应不同的功能要求而重复使用），比如刀不但能杀鸡，还能切菜，但面向过程中杀鸡就是杀鸡，很难将其中任何一个过程进行复用，除非过程完全一样，这就难了，所以复用对象远比复用过程容易和方便，这也就是为什么面向对象这种速度慢、资源占用多的编程方式会越来越流行的原因。

1.2　方法与属性

解释完了对象，现在要说明对象是怎么使用的，用计算机的行话来说叫调用。任何一个对象，都有且只有两个叫什么好呢，叫特征吧，也许定义不是很准确，大家理解就行了，其中一个叫属性，另一个叫方法，这个无论是在程序中还是现实中都是适用的。比如一个人，其身高、体重、性别都是这个人的属性，而这个人会什么——会下棋、会打球、会睡觉等等就是这个人所具有的方法。

属性没什么可再详细说的，所谓属性就是对这个对象的描述值，比如身高、体重，这个很简单也很好理解。方法就比较麻烦，所谓方法就是一个对象所具有的功能，比如一个人会写书法，那么写书法就是这个人的一个方法，而写书法需要有书写的内容，这个内容是写书法必需的条件，那么这个内容也被称为句柄（C++称为参数），写书法除了要写的内容，还需要笔墨纸砚等等这些东西，这也就是句柄，所以句柄可以有很多个，而且每个句柄的类型都可以不同。书法写完了不是就完了，总要留下一些写满字的纸吧，这个纸就叫做墨宝，在程序中称为返回值，就是运行方法的结果。那么要怎样定义会书法的人呢？

从属性来说，人的属性很多，但是和书法相关的不多，所以我也只选取了 3 个，大家发现这 3 个属性中，生日和性别是不能改变的（当然，要练葵花宝典或变性除外），而国籍是可以改变的，所以属性就有可修改属性和不可修改属性的差别，这个概念在编程中会经常用到。

从方法上来说，因为我们只需要会书法，所以吃喝拉撒什么的就都不写了，写书法是一个方法，其句柄为内容、笔、墨、纸、砚，而其写完的书法我们可以叫墨宝，也就是说这个方法是一个方法名为：**书法**；句柄为：**内容、笔、墨、纸、砚**；返回值为：**墨宝**的方法，见图 1-2。

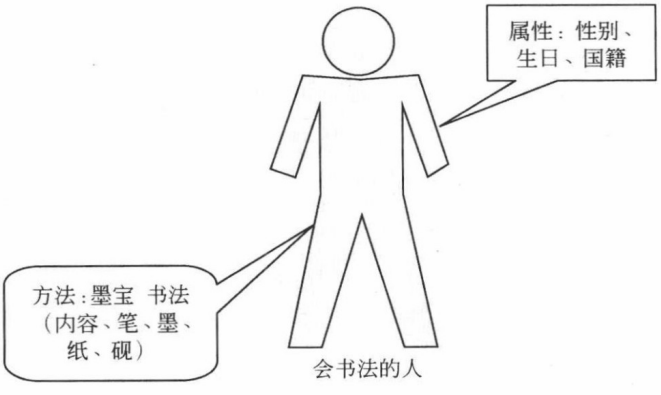

图 1-2

最后再次总结，对于一个对象来说，属性是对其特征的描述，属性有可变属性和不可变属性两种；方法是这个对象能提供的功能，方法由方法名、句柄和返回值构成。这里强调一下，方法的方法名是必需的，但句柄和返回值却未必一定都要有。比如发呆，方法名无疑就是发呆了，但发呆不需要任何条件，而发呆也不会产生任何的结果输入，不过 Java 要求返回值必须写，所以如果没有返回要告诉对方，返回值为空。

1.3 实例化

实例化的意思就是把一个对象指定一个实体，就是把一个虚幻的对象明确成一个具体的事物。听上去很拗口，如果想要理解这个问题，我还想讲一回哲学。大家可能总觉得，你好好讲你的计算机编程，讲哲学干什么？其实我觉得哲学对于编程尤其是面向对象编程是非常重要的，因为哲学是对现实世界的宏观抽象，而面向对象是现实世界在程序的投影。如果我没记错，大仲马先生在其名著《基督山伯爵》中这样描述哲学——哲学是一切科学的综合。不管对错，我认为哲学是我们对世界认知的一门科学，这对我们理解面向对象这种与现实世界紧密联系的编程思想是非常有帮助的。

废话说多了，现在继续讨论面向对象。我觉得如果要定义对象实例化就要理解哲学中一个很重要的概念——白马非马（如果这个记不住的话，自己去查书或者找政治老师给你讲讲），鉴于我不是哲学老师，我也不想向这方面发展，所以这个概念我就不引申了，白马非马的核心我认为是说，白马是概念上的东西，是一个具有相同特征的马的集合体，而非特指任何一匹具体的马，我们可以理解一下白马也就是我们所说的对象，这个对象是一个描述、一个分类、一个集合，但唯独不是一匹具体的马，也就是说白马只存在概念而非现实存在。那么我们想要一匹能骑能吃的白马怎么办？那我们就要将白马这个对象进行实例化——这一匹白马或那一匹白马。

1.4 引用与继承

面向对象有一个非常、非常重要的特征就是继承了，面向对象所谓的继承和遗传学上的继承还真比较接近（和继承财产不大一样，这个要记住），但是更加机械，缺少了变异这个环节。其概念为：被继承的主体称为父，继承者称为子。子将继承父的所有属性和方法（行为），子可以自己定义新的方法和属性，或对继承的方法和属性进行重新定义。

继承存在于对象（抽象）之间的关系，而非实例（实体）之间的关系。以白马为例，白马就是继承了马这个对象，但其颜色的属性被限定为白色。

很多开始编程的人都喜欢疯狂使用继承，他们认为继承可以极大地减少代码，事实上也确实是。原来相同的代码都需要用拷贝粘贴，现在继承一下就行。但我认为这个不是继承的本质，或者说这么用继承，除了使程序更难看懂、运行速度更慢以外，没有什么好处。

我认为继承的优点并不在于少写几行代码，而在于继承的子对象，在父对象改变时会跟着改变，假定上帝是程序员，人类（无论黑人、白人还是黄种人）都是两只眼睛（正常的情况下）。现在上帝需要人们长三只眼，OK，上帝如果像拉双眼皮一样，一个一个拉，估计他老人家得干个 1 千年都未必干得完。于是上帝修改了人这个父对象，所有人都是三只眼，一瞬间所有继承了人这个对象的对象以及其实例，眼睛都变成了三只。

为了可以使用已有了的功能，我们更愿意使用的形式是引用，也叫调用，就是把对象在自己写的程序内部实例化后使用，这种方法不但效率更高，而且也更容易使别人阅读。

下面是对引用和继承的简单解释。

引用

只使用目标对象的一部分功能，其目标对象的改变除非被使用那部分功能也改变，才会引起引用对象的改变，所以引用是一种简单、有效，并且比较安全的使用其他对象的方式。

继承

在没有重新定义的情况下，子对象和父对象的功能和属性完全相同，如果希望改变子对象中父对象的功能，则需要对该功能进行重新定义，使用继承需要有几个条件：① 继承对象间应该有逻辑上的关系（比如让青蛙继承老鹰，就为了获得他们都能呼吸的方法，那就是神经病）；② 继承对象间相似度尽量高，不要说继承了父对象的方法其中 80%的方法需要重定义，那样意义就很小；③ 如果继承对象中有完全相同的方法或属性，那么要确定这些属性是否存在联动关系，比如让青蛙继承蛤蟆，如果蛤蟆变了三条腿（那个古时候叫金蟾好不好），那青蛙是否也变三条腿（那个如今叫污染变异）。

1.5　UML 介绍

前面讲了很多关于面向对象的定义和名次，那么软件行业对于面向对象是如何设计的呢？

说实话如果是 20 年前，大家都是拿个破纸画来画去，也没有一个固定的方式和语言，近些年随着面向对象的飞速发展，推出了一个现在全世界面向对象开发者都使用的语言——UML（Unified Modeling Language），中文译作统一建模语言。UML 是一种图形化语言，说是语言实际上是定义了一堆在面向对象设计和开发中所需要的图标，使我们的设计文档能够被别人看懂。

UML 语言在本书中并不是重点，所以我也不想对其作太多的介绍，尤其是这个语言的出身什么的，我觉得大家在这个时候完全没必要知道。总之 UML 语言是一门针对面向对象设计（OOA）和开发（OOD）的一门以图形化为主的建模语言；其功能涵盖需求、设计、开发和部署，可以说是软件开发的全部流程都有涉及；UML 有 7 种图的类型，而在本书中为了使大家更好地理解类和类之间的关系，我们就介绍 UML 其中的一种图——类图（Class Diagram），UML 为这种类对象类型起了一个特别的名字：“分类器”。通常你可以把分类器当做类，但在技术上，分类器是更为普遍的术语。

类的 UML 表示是一个长方形，垂直地分为三个区，如图 1-3 所示：

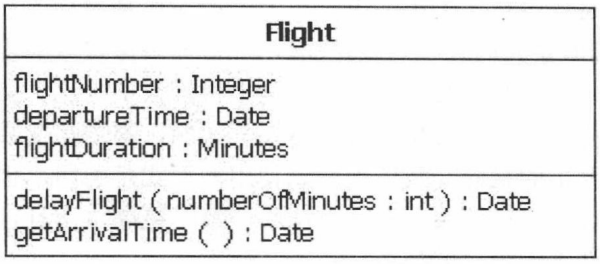

图 1-3 类图示例

- 顶部区域显示类的名字。
- 中间的区域列出类的属性。
- 底部的区域列出类的方法。

如图 1-4，这个类的名字为 Flight，这个类有三个属性分别为：

- **航班号（flightNumber）** 整型。
- **起飞时间（departureTime）** 日期。
- **飞行时间（flightDuration）** 分钟。

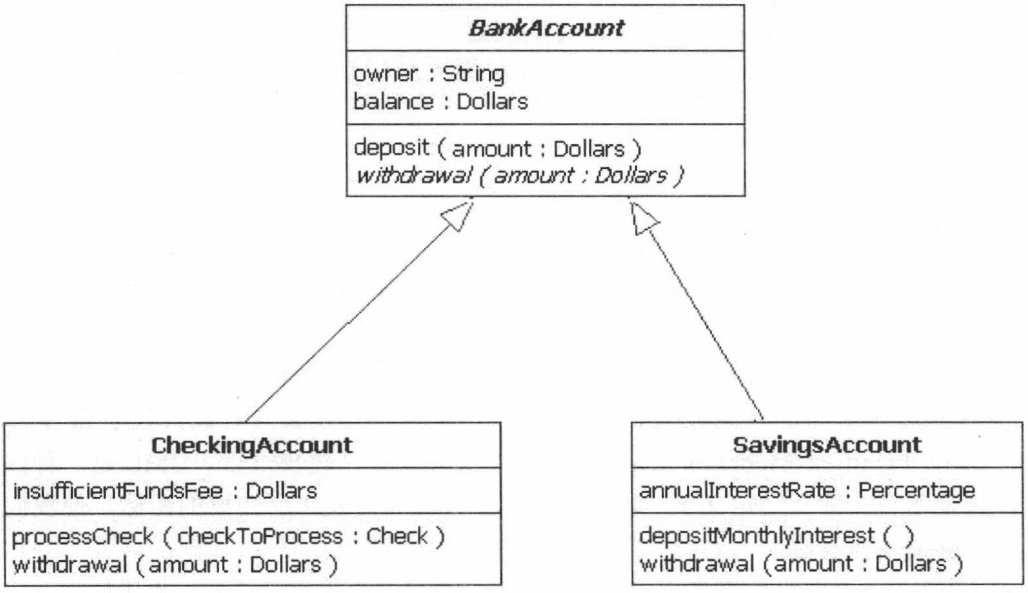

图 1-4 继承图例

这个类有两个方法，分别为：

- 航班延误（delayFlight），有一个句柄延迟时间（numberOfMinutes ）整型，返回值为时间类型（Date）。
- 获取到达时间（getArrivalTime），没有句柄只有返回值，其返回值为时间类型（Date）。

类和类之间的关系有两种：调用和继承。对于继承又包含继承实体类和继承接口，调用更标准的 UML 称谓叫关联，关联有 5 种。说实话我不打算在这里讲述太多关于 UML 的知识，毕竟本书讲述的是 Java 语言而非 OO 设计和 UML 语言图例，所以我只说明继承和关联的基本形态，而不想过多说明这些在设计中的含义，或者说我只是让读者在后续阅读中能了解我画的图，而非让读者精通 UML 这门语言。

首先说明继承，在面向对象的设计中继承是一个非常重要的概念，其指的是一个类（子类）通过继承这种方式拥有另外一个类（超类）的所有功能，并增加它自己的新功能或改变旧功能的一些特点（具体的继承概念我希望在本书后面说明）。

UML 语言中是从子类拉出一条闭合的三角形箭头的实线指向超类来表示继承关系。以图 1-4 银行账户的类型为例：CheckingAccount 和 SavingsAccount 类是从 BankAccount 类继承而来。CheckingAccount 和 SavingsAccount 类拥有 BankAccount 类的所有属性和方法的同时还各自拥有自己特殊的属性和方法。

在一个单向关联中，两个类是相关的，但是只有一个类知道这种联系的存在，单向关联表示为一条带有指向调用目标类的单向箭头。如图 1-5 所示 OverdrawnAccountsReport 类调用 BankAccount 类，而 BankAccount 类则对 OverdrawnAccountsReport 类一无所知。

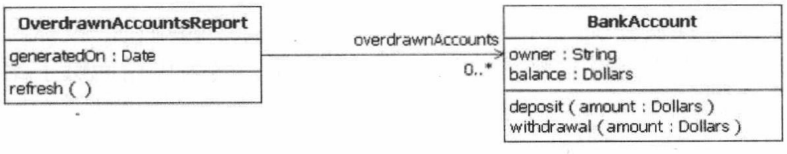

图 1-5　单向关联实例

第 2 章　Java 绪论

终于进到正题了，大家庆祝一下吧。

Java 语言是现今世界上最流行的一种编程语言，其活跃在各种各样的编程领域和几乎所有的软硬件平台上，可以说 Java 语言是世界上用途最广泛的语言，从前台页面到后台程序；从 ERP 企业信息管理到电子游戏；从手机到大型机，Java 语言可以说无所不在。在讲解 Java 语言语法之前，先让我们领略一下 Java 所创造的传奇。

2.1　Java 传说

Java 自 1995 诞生，至今已经 17 年历史。据 Java 语言的创始人 James Gosling 大神回忆，最初这个为 TV 机顶盒所设计的语言在 Sun 内部一直称为 Green 项目（搞笑的是这个项目为了美国在线机顶盒开发的语言，并且在投标中输掉了该项目的投标，实在是太有喜感了，就好像波音 747 也是在预警飞机项目中输掉投标，转为民用而赚了比预警机项目更多的钱一样）。因为输掉了投标但是也不能就此烂在手里，那就需要重新包装，于是就需要重新设计一个名字。开始的时候，Gosling 注意到自己办公室外一棵茂密的橡树 Oak，这是一种在硅谷很常见的树。所以他将这个新语言命名为 Oak。但 Oak 是另外一个注册公司的名字，所以这个名字不可能再用了。于是在命名征集会上，大家提出了很多名字。最后按大家的评选次序，将十几个名字排列成表，上报给商标律师。排在第一位的是 Silk（丝绸）。尽管大家都喜欢这个名字，但遭到 James Gosling（Sun 当时的总裁，Boss 必杀技——否决权^_^）的坚决反对。排在第二和第三的都没有通过律师这一关。只有排在第四位的名字，得到了所有人的认可和律师的通过，这个名字就是 Java。

Java 的名字的来源是印度尼西亚爪哇岛的英文名称，因盛产咖啡而闻名， Java 也作为咖啡的代名词，而咖啡是世界程序员最喜欢的三大饮料之一，另外两种嘛，就是可口可乐和百事可乐了，很显然这两种可乐有版权保护，用是犯法滴。顺便说一句，我就很喜欢可口可乐。Java 咖啡（现在是咖啡的一个品种，和蓝山一样又指产地，也指品种）是一种非常苦的咖啡，如果大家有机会一定要尝一下。

为此 Sun 为 Java 设计的标识也正是一杯正冒着热气的咖啡（不过缩小的时候真的很抽象，曾经一个朋友指着我的屏幕说：“哟，拉面！” 囧）。

Java 分为三个体系。

- **Java2SE**　Java2 Platform Standard Edition，Java 平台标准版，主要用于单机软件和网页展现。
- **Java2EE**　Java 2 Platform Enterprise Edition，Java 平台企业版，主要用于服务器端程序和网络程序的开发，但我更觉得 J2EE 像个大杂烩，凡是 Sun 开发了功能不知道放在哪里的，一律 J2EE 的干活。