水木书荟

软件是这样“炼”成的

——Java学习全演练

王朔韬　编著

- 独特的场景式案例描述
- 全纪实的记录手法
- 融合理论与实践
- 引领学习Java开发

清華大學出版社

软件是这样“炼”成的

——Java学习全演练

王朔韬　编著

清華大學出版社
北　京

内 容 简 介

本书的定位是针对 Java 学习中有案例式参考资料需求的读者。全书选用易于理解的“学籍管理软件”作为案例，讲解 Java 的核心知识点及应用场景。从文字组织到书的结构设计，既不是以理论为主调的“学院派”编写方法，也不是以应用介绍为主调的“应用派”编写格调，而采用了情景对话、场景在线的方式，解决读者在学习 Java 过程中所感觉到的知识点的零散性和应用场景不清楚的状况。将 Java 知识点体系化和系统化，以案例驱动的编写方法实现了“边学边用，边用边做”的 Java 学习思路。

本书适合于从事软件开发的管理人员、系统分析师、架构师和程序员阅读，也可以作为大学计算机相关专业学习 Java 课程的项目实训教材。

图书在版编目(CIP)数据

软件是这样“炼”成的：Java 学习全演练/王朔韬编著. —北京：清华大学出版社，2018
ISBN 978-7-302-47929-1

Ⅰ. ①软…　Ⅱ. ①王…　Ⅲ. ①JAVA 语言　Ⅳ. ①TP312.8

中国版本图书馆 CIP 数据核字(2017)第 193552 号

责任编辑：黄　芝　王冰飞
封面设计：谜底书装
责任校对：梁　毅
责任印制：李红英

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
课件下载：http://www.tup.com.cn，010-62795954
印 装 者：清华大学印刷厂
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　　张**：24.25　　**字　　数**：591 千字
版　　次：2018 年 12 月第 1 版　　**印　　次**：2018 年 12 月第 1 次印刷
印　　数：1～1500
定　　价：79.00 元

产品编号：073359-01

前言 foreword

计算机软件工程课程是一门应用型很强的专业学科，能够很好地应用起来才是软件开发语言的重要学习目标。特别是计算机语言类课程，本来就没有什么深奥的理论基础，我们按照既定的语法规则进行编程即可。从理论的角度来说，我们似乎没有什么可发挥的了。

编者应邀承担高校计算机相关专业的Java语言授课工作，在教学中发现学生们最大的疑问点是，Java语言到底是用来干什么的？Java语言在实践项目中是如何应用的？这些知识点的应用场景是什么？Java知识点之间的关系是怎样的？算法是什么东西？为什么要有这些知识点？等等。如果这些疑问无法解决，学生们学习Java语言的兴趣自然而然地就会逐渐地减少。

“边学边用”是学习Java语言最好的办法。目前Java语言教科书非常多，但是不少作者将Java语言中的每个知识点零散地推给学生们，教材中的例子自然只是为知识点服务的，导致整本书中的例子没有连贯性，甚至同一章节中的例子都是分散的。容易导致学生只知其一不知其二，没有形成系统的知识结构，更不会把这些实践案例有效地结合起来。

本书是编者编写的“软件是这样‘炼’成的”系列书的四本之一，其最大的特点是将学院派和应用派的两大著书思想有效地结合起来，既没有空洞的理论，也没有泛泛而谈的应用，而是将理论与实践融合起来，给读者以新的感受和收获。在文字组织上，采取场景再现、情景对话等方式，将Java语言中的每个关键知识点应用到“学籍管理软件”这个案例中，这种“边学边用”的学习过程，使读者能够系统和完整地了解Java的知识结构并应用之。

历经3年之久，编者总算艰难完成了“软件是这样‘炼’成的”系列书的编写，尤其前3本书的出版得到了广大读者的殷切关注和大力支持，并且提出了许多宝贵意见，在此真心地表示感谢。希望各位继续提出宝贵的意见。

在本书的编写过程中，编者得到了家人和朋友的大力支持。在此，我要感谢我的妻子商莉和我女儿王贝思在本书的编写过程中给予的最大帮助和支持。

由于作者水平有限，书中难免有疏漏和不足之处，恳求各位专家和广大读者提出宝贵的意见。如果读者在阅读过程中遇到问题，可以直接和笔者联系（QQ：307050843）。

编　者

2018年3月

目录 Contents

第1章 何为面向对象

1.1 面向对象的基础知识

面向对象(Object-Oriented ,简称 OO)方法作为一种独具优越性的新方法正引起全世界越来越广泛的关注和高度的重视,被誉为"研究高技术的好方法",更是当前计算机界关心的重点。

面向对象的实质是主张从客观世界固有的事物出发来构造系统,提倡采用人类在现实生活中常用的思维方法来认识、理解和描述客观事物,强调最终建立的系统能够映射问题域,也就是说,系统中的对象以及对象之间的关系能够如实地反映问题域中固有事物及其关系。

OO 方法起源于面向对象的编程语言(简称 OOPL)。20 世纪 50 年代后期,在用 FORTRAN 语言编写大型程序时,常出现变量名在程序不同部分发生冲突的问题。鉴于此,ALGOL 语言的设计者在 ALGOL 60 中采用以"Begin…End"为标识的程序块,使块内变量名是局部的,以避免它们与程序中块外的同名变量相冲突。这是编程语言中首次提供封装(保护)的尝试。此后程序块结构广泛地应用于高级语言如 Pascal 、Ada、C 之中。

20 世纪 60 年代中后期,Simula 语言在 ALGOL 基础上研制开发,它将 ALGOL 的块结构概念向前发展一步,提出对象的概念,并使用类,同时也支持类继承。20 世纪 70 年代,Smalltalk 语言诞生,它以 Simula 的类为核心概念,很多内容借鉴于 Lisp 语言。由 Xerox 公司经过对 Smautalk 72、76 持续不断的研究和改进之后,于 1980 年推出商品化的产品,它在系统设计中强调对象概念的统一,引入对象、对象类、方法、实例等概念和术语,采用动态联编和单继承机制。

从 20 世纪 80 年代起,人们在原来的基础上提出的有关信息隐蔽和抽象数据类型等概念,以及由 Modula 2、Ada 和 Smalltalk 等语言所奠定的基础,再加上客观需求的推动,进行了大量的理论研究和实践探索,不同类型的面向对象语言逐步地发展和建立并完整起来。

20 世纪 80 年代以来,人们将面向对象的基本概念和运行机制运用到其他领域,获得了一系列相应领域的面向对象的技术。面向对象方法已被广泛地应用于程序设计语言、形式定义、设计方法学、操作系统、分布式系统、人工智能、实时系统、数据库、人机接口、计算机体系结构以及并发工程、综合集成工程等,在许多领域的应用都得到了很大的发展。

1986 年在美国举行了首届"面向对象编程、系统、语言和应用(OOPSLA'86)"国际会议,使面向对象受到世人瞩目,其后每年都举行一次,这进一步标志 OO 方法的研究已普及全世界。

面向对象的核心概念包括领域、类及类型、消息和服务、接口、封装、抽象、继承、组件图等。以下将分别展开论述。

1.2 面向对象的核心概念

在面向对象的软件开发过程中,开发者的主要任务就是先建立模拟问题领域的对象模型,然后通过程序代码来实现。关于对象、属性、状态、行为、方法、实现等概念将在本章中学习。

1.2.1 对象

1. 万物皆为对象

不管处于什么样的环境，不管我们处理什么事件人们必须面对许多对象。我们在学习过程中，书本、电脑、我们的同学和我们的老师都是对象；如果我们在踢足球，足球、场地和球门都是对象；如果我们正在吃饭，饭碗、筷子和餐桌都是对象；对象可能是非常小的，例如，分子；对象也可以非常大，例如，宇宙、银河系、亚洲、中国、北京市、北京科技大学等。

2. 对象的属性

对象的属性用于描述对象的状态、特征以及组成部分。

如果对一位学生进行描述的话，可能会涉及的内容包括姓名？性别是男还是女？年龄是少年还是青年？身高是高个子还是矮个子？体重是超重还是标准？血型是A型还是B型？等等。这些描述全是用于描述一位学生的特征的。

有用于描述人的状态的，包括周岁、男或女、30还是40等。

3. 对象的行为

对象的行为也就是对象能够完成的功能，每个对象都会有自己的行为，行为用于改变对象自身的状态，或者向其他对象发送消息，有时候一个行为会同时包含这两者。现在只讨论改变自身状态的行为，关于向其他对象发送消息的行为将在后面讨论。我们以学生为例来研究对象的行为。

每个对象都存在大量的行为。有些行为是可以看到的。学生在整个生活中有学习文化课、考试、吃饭等必要的社会活动。

另外，还有一些行为是作为学生所必需却不被注意的。这些行为包括上课、完成作业、参加考试、完成实验，这些行为是日常的我们却不以为然了，并且这些行为是体现其学生身份的关键所在，也是关系着能否毕业的关键所在。

4. 对象的标识

系统中的每个对象都有自己的标识，这个标识对于系统中的每个对象是唯一的。标识有些是大家熟悉的，例如，专业；有些标识是不熟悉的，例如，学号。但是学号对于学生来说肯定熟悉。因为属于系统的不同的用户，所以关注的对象也不一样。

作为一个程序员，要完成这样一个系统，系统中所有用户关注的信息我们都需要关注。标识也是一个属性。通常标识也是用于描述对象的，这和前面的属性是相同的。标识本身可能有意义，例如，学生的专业名称；其标识本身也可能没有意义，仅仅作为标识，但是，学号仅用于标识一个学生。

5. 对象之间的关系

对象之间的关系包括：

① 整体与部分的关系，例如，学生和性别的关系，整体属性是学生，部分属性是性别。

② 关联关系，例如，学生洪杨俊是计算机专业的学生。

1）整体与部分的关系

具有这种关系的两个对象之间有比较强的依赖关系，就像某位学生必须有性别属性，如果没有性别这个属性，这个学生就不能被安排住宿，也就是说只要是一个学生，就一定有属于他的性别。

这种关系的对象一旦创建完成之后，都是作为一个整体来使用，通常情况下不会单独考虑组成部分，如果要考虑组成部分也是先考虑整体。例如，要改变这个学生的性别，通常学校会调查是什么原因。但是不管怎样，即使直接说性别，也会有一个前提，就是某人的性别。这种关系一旦建立，通常不再改变。

2）关联关系

关联关系的两个对象之间通常没有依赖关系。就像一个学生在学习过程中就会不停地上学、毕业，这样他/她所上的学校也在不停地变化。学生大学上了北京科技大学，学生和北京科技大学之间的关联关系就建立起来了。关联关系建立之后，学生就按照北京科技大学的相关规定来学习和生活。学生毕业了，学生和北京科技大学这种关联关系就解除了，北京科技大学的相关制度不再会对这位学生产生影响。另外，不像整体与部分的关系，一旦创建基本不再变化，关联关系可以根据需要随时创建，随时解除。

3）关系中的量

一个系有3个专业，一个专业有三个班，一个班有30个学生，这里的数字就是关系中的量。不管是整体与部分的关系，还是关联关系都存在着量。根据关系中的量可以把关系分为一对一、一对多、多对一和多对多4种。

1.2.2 类

具有相同属性（数据元素）和行为（功能）的对象的抽象就是类。例如，学生毕业登记表、学生奖惩记录表。因此，类是经过对象抽象而得到的。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。

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，即一般和整体。一般指具体结构，称为分类结构，也可以说是“或”关系，或者是“is a”关系；整体指部分结构，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。

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收消息的对象名、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。类中操作的实现过程叫做方法，一个方法有方法名、参数、方法体。

关于类的详细内容将在后续章节中讲述。

1.2.3 继承

继承是类与类之间的关系，分为父类和子类。在父类和子类之间同时存在着继承和扩展关系。子类继承了父类的属性和方法，同时，子类还可以扩展出新的属性和方法，并且还可以覆盖父类中方法的实现方式。

覆盖是指子类中重新实现父类中的方法。

注意：子类只能继承父类的部分属性和方法。父类中private修饰的属性和方法，对子类是透明的。

继承与扩展同时提高了子系统的可重用性和可扩展性。

继承与扩展导致面向对象的软件开发领域中架构类软件系统的发展。关于类间继承关系将在后续章节中详细说明。

1.2.4 接口

Java接口是一系列方法的声明，是一些方法特征的集合，一个接口只有方法的特征说明，没有方法的实现，接口功能是为其他类提供服务。现实世界中，接口也是实体。在面向对象中，接口是一个抽象的概念，是指系统对外提供的所有服务的定义。

接口描述系统能够提供那些服务，但是不包含服务的实现细节。

对象中所有向使用者公开的方法声明构成了对象的接口。

接口是提高系统松耦合的有力手段。接口还可提高子系统的可扩展性。

(1) 概念性接口，即对外提供的所有服务，对象中表现为 public 类型的方法的声明。

(2) 接口类型指用 interface 关键字定义的实实在在的接口，它用于明确地描述系统对外提供的所有服务，更清晰地把系统的实现细节与接口分离。

1.2.5 封装与透明

1. 封装

封装是指隐藏对象的属性和实现细节，仅对外公开服务。

其优点如下。

① 便于正确使用、方便理解，防止使用者错误修改系统的属性和方法。

② 有助于建立子系统之间的松耦合关系，提高独立性。

③ 提高软件的可重用性，每个系统都是一个相对独立的整体。

④ 降低了构建大型企业系统的风险，即使系统不成功，个别的子系统还是有价值的。

良好的系统会封装所有的实现细节，把它的服务与实现清晰地隔离开来，系统之间只通过接口进行通信。

封装的访问类别如下。

(1) public：最高级别，对外公开。

(2) protected：类本身或者子类公开，即保护级别。

(3) 默认：只对同一个包中的类公开，包级别。

(4) private：不对外公开，只能在对象内部访问，级别最低。

到底哪些对象的属性和方法应该公开，哪些应该隐蔽呢？有两大原则：①把尽可能多的方法和属性藏起来，对外提供简洁的接口。②系统的封装程度越高，那么对象独立性就越高，使用也更方便。

将类中的属性尽量地封装起来，把所有的属性藏起来，有以下几个优点。

① 更符合真实世界中外因通过内因起作用的客观规律。

② 能够灵活地控制属性的读和修改访问级别。

③ 防止使用者错误地修改属性。

④ 有助于对象封装实现细节。

2. 透明

与封装具有相同含义的一个概念就是透明，对象实现细节，也就意味着对使用者是透明的。

1.2.6 多态

多态是指多个方法只能有一个名称，但可以有许多形态，也就是程序中可以定义多个同名的方法，用“一个接口，多个方法”来描述。可以通过方法的参数和类型引用。例如，人们喜欢说一句话“想死人了”，这句话本身的应用场景不一样其含义就不一样了。如果这句话用于亲人之间，表达的是一种思念之情；如果用于两个敌对仇人之间，就是表达对对方的恐吓。多态其实就是方法名相同，参数不同。以上不同的参数就是敌对仇人和亲人两个不同的参数。

在 Java 语言中，多态性体现在两个方面，即由方法重载实现的静态多态性(编译时多态)和方法重写实现的动态多态性(运行时多态)。

1. 编译时多态

在编译阶段，具体调用哪个被重载的方法，编译器会根据参数的不同来静态确定调用相应的方法。

2. 运行时多态

由于子类继承了父类所有的属性(私有的除外)，所以子类对象可以作为父类对象使用。程序中凡是使用父类对象的地方，都可以用子类对象来代替。一个对象可以通过引用子类的实例来调用子类的方法。

1.2.7 组合

组合是一种用多个简单子系统来组装出复杂系统的有效手段。

对于一个组合系统，用 UML 语言描述。组合系统与它的子系统之间为聚集关系，子系统之间则存在关联关系或依赖关系。

1.2.8 绑定

绑定指的是将一个过程调用与相应代码链接起来的行为。动态绑定是指与给定的过程调用相关联的代码只有在运行期才可知的一种绑定，它是多态实现的具体形式。

1.2.9 消息

对象之间需要相互沟通，沟通的途径就是对象之间收发信息。消息内容包括接收消息的对象的标识，需要调用的函数的标识，以及必要的信息。消息传递的概念使得对现实世界的描述更容易。

1.3 类间关系

类间关系大致分为关联、依赖、继承、实现、组合和聚合，下面将通过“学籍管理软件”的实例来逐一说明。

1.3.1 关联

关联表示两个或多个类间存在的一种语义关系。这种关系可以是单向的，也可以是双向的，它使得类之间可以知道彼此的属性和方法。当人们在系统建模时，特定的类会彼此关联。这里将首先详细讨论常用的类间关联关系。

1. 双向关联

关联是两个类间的联接。如果关联是双向的，那就意味着两个类彼此知道它们间的联系，除非我们限定一些其他类型的关联。如获奖情况(AwordsSituation)的例子，获奖情况和优秀毕业生之间存在一种双向关联的关系。根据获奖情况自身的方法，可以判断一名学生是否为优秀毕业生。同样的，优秀毕业生本身拥有获奖情况这一属性(当然知道自己所获奖项)。图 1-1 显示出在获奖情况(AwardsSituation)类和优秀毕业生(OutstandingGraduates)类之间的一个标准类型的关联。

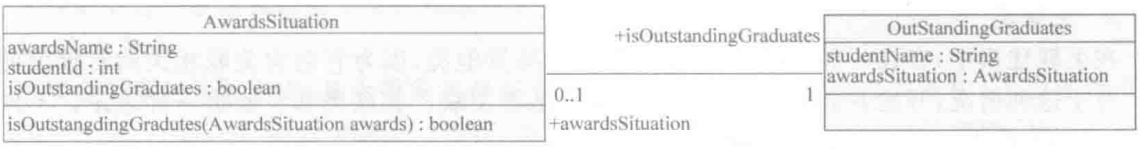

图 1-1 类间双向关联例图

一个双向关联用两个类间的实线表示。在线的任一端，可放置一个角色名和多重值。图 1-1 显示 AwardsSituation 与一个特定的 OutstandingGraduates 相关联，而且 AwardsSituation 类知道这个关联。因为角色名以 AwardsSituation 类表示，所以 AwardsSituation 承担关联中的“awardsSituation”角色。紧接于 AwardsSituation 类后面的多重值描述 0..1 表示：当一个 AwardsSituation 实体存在时，可以有一个或没有 OutstandingGraduates 与之关联（也就是，OutstandingGraduates 可能还没有被分配）。图 1-1 也显示 AwardsSituation 知道它与 OutstandingGraduates 类的关联。在这个关联中，OutstangdingGraduates 承担“isOutstandingGraduates”角色；图 1-1 告诉大家，OutstandingGraduates 实体可以不与 AwardsSituation 关联（例如，他还未毕业）。

由于对那些在关联尾部可能出现的多重值描述感到疑惑，表 1-1 列举一些多重值及其含义的例子。

表 1-1　多重值和它们的表示

表　示	含　义
0..1	0 个或 1 个
1	只能 1 个
0..*	0 个或多个
*	0 个或多个
1..*	1 个或多个
3	只能 3 个
0..5	0 到 5 个
5..15	5 到 15 个

2. 单向关联

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。图 1-2 显示单向关联的奖学金的一个实例。Scholarship 是根据分数确定奖学金等级的，而分数却不知道奖学金。因此，这是一种单向关联。

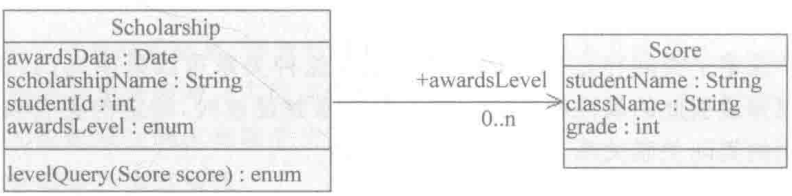

图 1-2　单向关联实例

一个单向的关联，表示为一条带有指向已知类的开放箭头（不关闭的箭头或三角形，用于标志继承）的实线，单向关联包括一个角色名和一个多重值描述，但是与标准的双向关联不同的是，单向关联只包含已知类的角色名和多重值描述。在图 1-2 的例子中，Scholarship 知道 Score 类，而且知道 Score 类扮演“awardsLevel”的角色。然而，和标准关联不同，Score 类并不知道它与 Scholarship 相关联。

3. 关联类

在关联建模中，存在一些情况，即某个类需要包括其他类，因为它包含关联相关的有价值的信息。对于这种情况，开发者会使用关联类来绑定其基本关联。关联类和一般类一样表示。不同的是，主类和关联类之间用一条相交的点线连接。图 1-3 显示一个就业推荐实例的关联类。根据获奖情况判断是否为优秀毕业生，如果是，则作为优秀毕业生推荐工作。

在图 1-3 显示的类图中，在 AwardsSituation 类和 OutstandingGraduates 类之间的关联，产生了

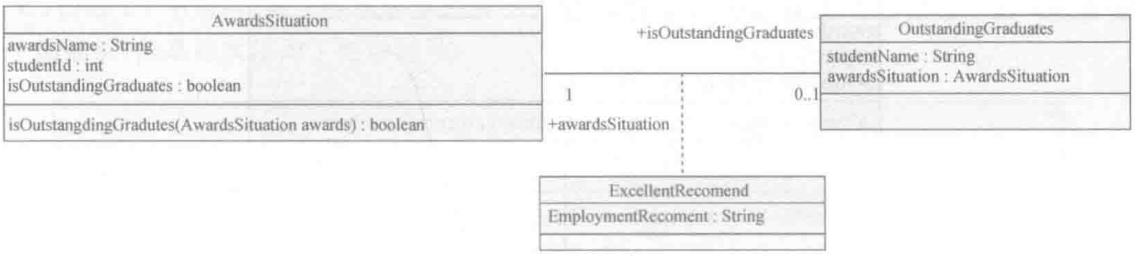

图 1-3 增加关联类 ExcellentRecomend

称为 ExcellentRecomend 的关联类。这意味当 AwardsSituation 类的一个实例关联到 OutstandingGraduates 类的一个实例时，将会产生 ExcellentRecomend 类的一个实例。

4. 反射关联

现在已经讨论了所有的关联类型。就如读者可能注意到的，我们的所有例子已经显示了两个不同类之间的关系。然而，类也可以使用反射关联与它本身相关联。起先，这可能没有意义，但是请记住，类是抽象的。图 1-4 显示一个 RollStudent 类如何通过 checkIfColonel 角色与它本身相关。当一个类关联到它本身时，这并不意味着类的实例与它本身相关，而是类的一个实例与类的另一个实例相关。校级三好学生也是三好学生的一种，三好学生本身具有 isColonelRollStudent 方法来判断自身是否为三好学生：通过自身与校级三好学生的对比。因此，这是一种反射关联，即便两名三好学生不是同一对象，但从属于同一类。我们这里讨论的是类的范畴。

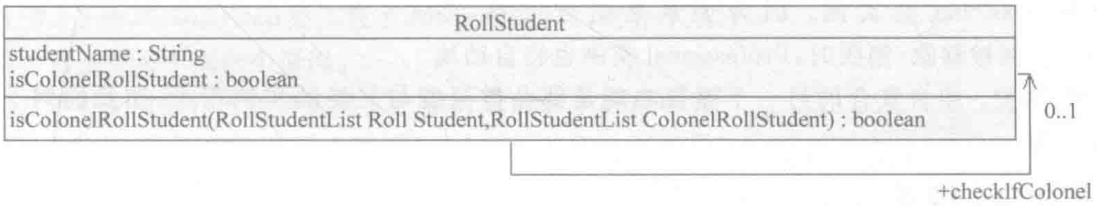

图 1-4 反射关联实例

图 1-4 关系说明一个 RollStudent 实例可能是另外一个 RollStudent 实例的上一级。然而，因为“checkIfColonel”的关系角色有 0..1 的描述；一个三好学生可能不是校级三好学生。

1.3.2 聚合

聚合是一种特殊的关联关系，是“强”关联类型，是整体到部分的关系。在基本的聚合关系中，部分类的生命周期依赖于整体类的生命周期，而部分类的生命周期独立于整体类的生命周期。

举例来说，我们可以想象，获奖情况是一个整体实体，而证书是整个获奖情况的一部分。证书可以独立存在于获奖列表。在这个实例中，CertificateAwards 类实例清楚地独立于 AwardsSituation 类实例而存在。然而有些情况下，部分类的生命周期并不独立于整体类的生命周期，这称为组合，是更“强”的聚合。举例来说，考虑学院与专业的关系。学院和专业都建模成类，在学院存在之前，专业不能独立存在。这里 Professional 类的实例依赖于 College 类的实例而存在。

让我们更进一步地探讨基本聚合和组合聚合。

1. 基本聚合

有聚合关系的关联指出，某个类是另外某个类的一部分。在一个聚合关系中，子类实例可以比父类存在更长的时间。为了表现一个聚合关系，需画一条从父类到部分类的实线，并在父类的关联末端画一个未填充棱形。图 1-5 是奖励和证书的聚合关系的例子。

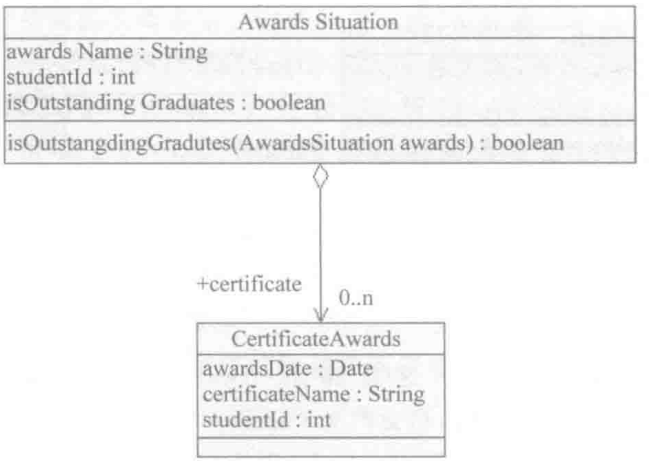

图 1-5 基本聚合关系的例子

2. 组合聚合

组合聚合关系是聚合关系的另一种形式，但是子类实例的生命周期依赖于父类实例的生命周期。图 1-6 中显示 College 类和 Professional 类之间的组合关系，注意组合关系如关系一样绘制，不过这次菱形是被填充的。

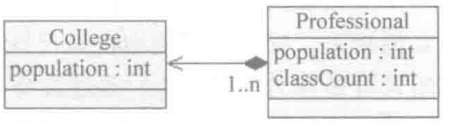

图 1-6 组合关系的例子

在图 1-6 的关系建模中，一个 College 类实例至少总有一个 Professional 类实例。因为关系是组合关系，当 College 实例被移除/销毁时，Professional 实例也将自动地被移除/销毁。组合聚合的另一个重要功能是部分类只能与父类的实例相关（如我们例子中的 College 类）。

1.3.3 继承

在面向对象的设计中，一个非常重要的概念——继承，指的是一个类（子类）继承另外的一个类（父类）的同一功能，并增加它自己的新功能（一个非技术性的比喻，譬如作者本人继承了其父亲的音乐能力，但是在其家中他是唯一一个玩电吉他的人）。为了在一个类图上建模继承，从子类（要继承行为的类）拉出一条闭合的，单向箭头（或三角形）的实线指向超类。考虑三好学生的类型：图 1-7 显示 ColonelRollStudent 和 ProvincialRollStudent 类如何从 RollStudent 类继承而来。校级和省级三好学生继承了父类的属性 studentName(public)，并扩展自己的属性 extraAwards。

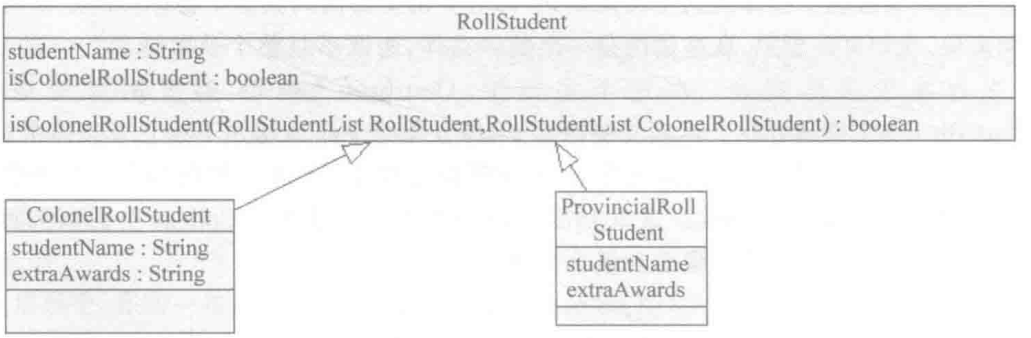

图 1-7 继承箭头说明

在图 1-7 中，继承关系由每个父类的单独的线画出，这是在 IBM Rational Rose 和 IBM Rational XDE 中使用的方法。然而，有一种称为树标记的备选方法可以画出继承关系。当存在两个或更多