3

分析业务模型——类图（Class Diagram）

类图（Class Diagram）可能是用得最多的一种 UML 图。类图的基本语法并不复杂，你可能学习最多两三天就可以掌握，然而要真正做到活用类图则可能需要几年的时间。类图是锻炼面向对象分析（OOA，Object-Oriented Analysis）和面向对象设计（OOD，Object-Oriented Design）思想的重要工具，是业务结构建模的重要工具。本章将会有大量的实战练习，你的 OOA 思想将会接受极大的考验和提升。

3.1 面向过程与面向对象

本小节的内容涉及到编程方面的知识，如果你有相关经验，请认真阅读本小节，本小节的目的是澄清开发人员的一些面向过程和面向对象的理解误区。如果你没有编程经验或者对此不感兴趣，可忽略本小节直接阅读下一小节，忽略本小节并不影响你对后文的理解。

20 世纪 90 年代初，我读高中的时候首次接触电脑，并且学习了第一门编程语言 Basic。当时不知道什么是面向过程，也不知道何为面向对象，只知道不断地学习 Basic 语言的算法，感受编程的乐趣。当时学习的 Basic 语言，现在看来是很老土的面向过程的语言。

后来学习了 C 语言，不久后朋友告诉我应该学习 C++，我问：C 和 C++有什么不同？于是朋友告诉我：C 是面向过程的语言，C++是面向对象的语言；C++比 C 最不同的地方就是 C++有类（Class），而 C 没有。

这就是我对面向过程和面向对象的第一印象，后来又学习了一些面向对象的知识，似乎将很多

东西变成类，类里面有特性和操作，就是面向对象。然而工作后发现完全不是那么回事，面向对象真的是只可意会难以言传啊。下面说说我对面向过程和面向对象的理解，希望对你有帮助。

很多年前的程序只有一行行的代码，后来出现代码难以组织、不好阅读、重复代码多等问题。于是“发明”了方法，将一段代码放到方法里面，实现一定的功能，供别的地方调用。方法的“发明”是编程史上的一大进步，其实方法就是一定程度上的封装，只要调用者给出符合要求的输入，方法就会返回合适的输出，调用者完全不用理会方法的具体实现，而方法里面又可以调用方法。随着后来的发展，出现了结构化编程，将编程的艺术更推进一步。

无论是方法还是结构化编程，都是我们提高编程技术以更好地解决复杂的、高难度的问题的一种手段而已。但后来发现问题越来越复杂，结构化编程开始招架不住了，于是有人提出面向对象编程。面向对象编码是一种基于类的编程方法，每一个类有特定的作用，类中有属性和方法，一条条语句只存在于属性或方法中。用面向对象的思路来求解问题，就是要设计出能解决问题的一个或多个类，通过类之间的相互操作和协作来解决问题。类是对代码的进一步封装，比方法对代码的封装要进一大步，类的出现要求编程的思想更进一步。

对于面向过程和面向对象编程存在这样的一些误区：

1．面向对象比面向过程更高级，无需注重结构化编程和编程基本功。

前面提到的编码发展史，简单说就是以下几个阶段：

- 一行行的代码
- 用方法组织起来的代码
- 结构化代码
- 面向对象的代码（用类来组织的代码）

看上去似乎后面的可以取代前面的，特别是到了面向对象编程阶段，似乎人人都可以喊自己是面向对象的，真正能写出好代码的人并不多。其实编码基本功相当重要，结构化编程也相当重要，如果这些基础不行，面向对象只能喊喊而已。我在以前公司招聘程序员，编程基本功是必考的。

2．面向对象编程就是将代码放进一个个类中而已。

我最开始对面向对象编程的看法基本上就是这样，后来用 VB 编程还是未能真正体会面向对象编程，直到后来使用真正面向对象的语言 C#以及学习了 UML 和设计模式，才开始真正体会到什么是面向对象编程。如何设计、提炼、规划类，是很讲技巧和功力的事情，面向对象一点都不容易！

3．将业务概念直接转变为类，赋予合适的属性和操作，就可以解决问题。

需求阶段的建模与设计阶段的建模是很不一样的，需求建模是对业务和需求的提炼，优秀的需求建模是设计建模的良好开始，但优秀的设计建模还需要考虑更多的设计上的事情，并不是简单地将业务模型直接转化为设计模型就可以解决问题的。

本书不会具体介绍如何面向对象地编程，而是如何面向对象地进行需求分析，我们将会借鉴面向对象编程的思想用于需求分析工作中。有开发经验的人士从事需求分析工作时，受面向过程和面向对象编程的思维习惯影响，容易处于“技术实现”的角度来分析问题。这需要一个转变过程，强烈建议你先忘掉自己的开发经历。本书接下来的内容，将会通过一个又一个的具体案例和练习，让

你体会面向对象分析需求的方法。当完成这个转变时，你会发现编程思想和分析需求思想有共同之处但又不太一样，你在编程时形成的严谨、全面、深入的分析方法会让你在需求分析工作中受益匪浅。

3.2 类图的基础知识

3.2.1 类图有什么用

某项目客户提供的原始需求资料中，有下面的一段话，请你仔细阅读，看看能不能将你搞晕？

“本项目是在一期的基础上增加对电缆、通信工程的管理和施工详细数据的记录和统计，使整个系统更好地管理各工程项目从中标开始到竣工验收的全部过程的资料和分析施工过程的数据。本系统将一条或一个标段的架空电力线路工程定为一个单位工程，即系统中的一个工程项目；每个单位工程分为若干个分部工程；每个分部工程分为若干个分项工程；每个分项工程中又分为若干相同单元工程。”

这段话中带下划线的文字，可能是本系统的一些关键业务概念。

如果你还没有晕的话，请回答下面的问题：

（1）你能用一句话描述这个系统是做什么的吗？

（2）这段话有什么业务概念？每个业务概念是什么意思？

（3）这些业务概念之间是怎样的关系？

上面那段文字充斥了大量的术语、概念（带下划线的字），如果你不是专业人士，恐怕难以读懂上述文字。项目初期我们往往对业务一无所知，我们最急迫需要解决的问题就是理清楚这些业务概念以及它们的关系。

每个软件系统都会涉及到很多人、业务概念和物品等，这些东西之间可能会有很多关系，发生很多事情。类图能帮助我们识别出这些人、业务概念、物品和事情等，并理清它们的关系。

3.2.2 什么是类

你大概了解了类图的用途了吧？我们暂时不去深究那段让人头晕的业务描述，我们先看看什么是类。

需求中提到的各种业务概念、人物等，经过抽象后都可以视之为类。为了更好地体验什么是类，请看下面这个练习。

练习：如果对本书的读者进行分类，你会如何分类呢？

强烈建议你先思考写下答案后再继续往下看。

- 男人、女人

人无非就是男人和女人两种，所以本书的读者不是男人就是女人。这样分类合适吗？

男人和女人在看这本书的时候，会有什么差异吗？将书的读者分为男人和女人，有什么好处？

如果不分为男人和女人，分为老人与年轻人，这样合适吗？

- 学生、在职人员

学生和在职人士读本书的时候应该是有所差异的，毕竟两者的基础不太一样。如果你是本书的作者，你觉得本书的目标读者是谁呢？编写本书时，你会更照顾学生还是在职人士呢？我们对读者进行分类，并不是为了分类而分类，而是希望通过对读者这个群体进行分析，写出一本内容更精彩、销量更好的书。

将某类东西归纳在一起，可以称为一个类。类有很多种提炼角度，我们需要根据系统的目标、业务的场景等，选取合适的角度对事物进行归纳。

3.2.3 什么是类图

只有一个类的类图，可能就是最简单的类图了，请看图 3.1。

图 3.1 只有一个类的类图

一个类就是一个矩形的方框，最上面是类的名字，中间是属性（Attribute），最下面是操作（Operation）。表示一个类时，可只显示类名，也可以只显示类名和属性，或者是类名和操作。

我们看看这个属性：+属性 1:int。

前面的“+”号表示这个属性是 public 类型的，实际上在需求分析时不需要管属性是 public 还是 private，全部画成 public 就可以了。

冒号后面的 int，表示属性的类型是 int 型（整数型），往往在需求分析初始阶段，可不必标识属性的类型。

至于操作，用类图进行业务建模时，一般不需要标识出来。

一个类图通常不止有一个类，有多个类时，我们还需要表达出类之间的关系，后面我们将介绍类之间的关系。

3.2.4 如何识别类

用类图获取需求的大致步骤如下：

1. 识别出类。
2. 识别出类的主要属性。
3. 描绘出类之间的关系。

4．对各类进行分析、抽象、整理。

我们通过下面这个练习来体验一下步骤 1、2。

练习：你需要做一个培训管理系统，请你用类图识别出课室中有什么人？这些人有什么关键属性？

强烈建议你先独立完成再继续阅读下文。

课室中有以下两类人，如图 3.2 所示。

图 3.2　学生与讲师 1

说明：该图是类图的简单画法，只表达了类名。

这两个类有这样的关键属性，如图 3.3 所示。

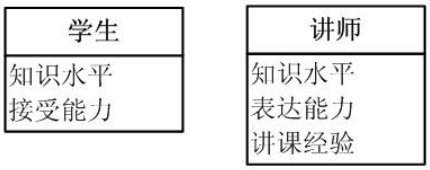


图 3.3　学生与讲师 2

说明：上面的类图同时表达了类名和类的属性。属性没有标记 public 还是 private，也没有标记属性的类型。业务建模时类图的属性可以全部看成是公开的，也不必标记属性的类型。

这个练习的场景是：你需要做一个培训管理系统，所以你识别出类以及他们的属性的时候，务必从这个角度出发。如果你得到的类是男人和女人，那就可能没有什么意义了。

如果你识别出来的属性是身高、体重，这些属性无论是属于学生还是老师，对于培训管理系统来说，可能是没有什么价值的。思考你识别出来的类的属性，能帮助你判断这个类是否合适。每一个类应该具备能表征它核心特点的关键属性，而一般的无特别意义的属性，可不必标记进去。

类图的基本语法是很简单的，但要体会什么是类、准确识别出类就不是那么简单了。实际工作中，我们需要将需求调研中了解到的所有业务对象、人物等列出来，画出他们的关系，反复推敲，才能逐步得到合适的业务模型。下面我们将开始学习类之间的关系。

3.3　类之间的关系

表达类之间的关系时，类只需要画出名字就可以了，属性和方法可以省略显示。

3.3.1　“直线”关系

A、B 两个类，它们之间有关系，但又不能确定是怎样的关系，我们可以这样画，如图 3.4 所示。

图 3.4　“直线”关系

这个“直线”关系其实就是关联（Association）关系，“关联”是 UML 中文术语的标准说法，但为了能让大家更容易理解和记忆，我会使用一些“老土”的说法。

做软件需求分析时，如果觉得两个业务概念之间有联系，但暂时不能确定具体是怎样的，那么就先画一条线将两者连起来再说。随着你对业务的理解，这条线条会进一步具体化，可以为这条线添加更多的元素。

图 3.5 中 C、D 两个类用一条直线相连，但在直线两端各有一个数字 1，表示一个 C 对应一个 D。

图 3.5　一对一关系

图 3.6 表示一个 E 对应 0 到多个 F，*号的意思就是表示 0 到多个。

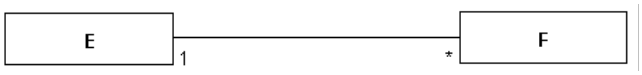


图 3.6　一对多关系

图 3.7 表示一个 G 对应 0 到 3 个 M，“0..3”表示 0 到 3 个，“1..4”表示 1 到 4 个，“x..y”表示 x 到 y 个（x，y 表示任意自然数，而且 x < y），注意有两个点（“..”）而不是一个点（“.”）。

图 3.7　一对零到三个关系

图 3.8 表示 I 和 J 之间有关系，在这个关系中 I 的身份是上司，J 的身份是下属。我们可以在线条的两端标记两者分别是处在怎样的角色。

图 3.8　角色关系

你可能会留意到，为什么“上司”、“下属”前面有一个“+”号？“+”号表示这个角色的类型是 public，“-”号表示 private，这些符号在软件设计时才需要用到，我们做软件需求分析时，不需要理会这些符号，全部画成“+”号就可以了。

这条直线如果变成带箭头的，又是表示怎样的意思呢？请看图 3.9。

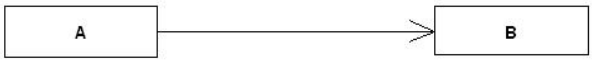


图 3.9 “导航”关系

这个图表示由 A 可找到 B，箭头表示方向，由 A 可“导航”到 B。

写代码时，如果 A 类有一个成员变量保存的是类 B 的引用，也就是说由类 A 可以找到类 B，那么可以画成图 3.9 的样子。这是从软件设计的角度来解释这个箭头的意义，如果是软件需求分析，这个箭头是怎样的意思呢？下面是一个实例，如图 3.10 所示。

图 3.10 请假单与请假者的关系

请假单上会列明是谁请的假，所以我们由请假单可以找到请假者。进行业务分析时，往往会发现由业务概念 A 可找到 B，这时可以使用带箭头的线条。

直线关系是最常见的关系，最简单的直线关系就是两个类之间画条线就可以了。我们也可以进一步细化这条直线关系：在这条直线的两端，可以标记上数字和名称，数字表示是几对几的关系，名称则表示在这个关系中，直线两端的两个类分别是怎样的一个角色，而这条直线也可以变成带箭头的直线。直线、几对几的关系、角色、箭头，以上内容可以搭配使用，只要能准确反映出业务关系就可以了。

直线关系只是一种老土的说法，UML 中文术语标准是关联（Association）关系。另外要说明的是：有时候因为类太多，为了排版方便，可以将“直线”画成“折线”，如图 3.11 所示。

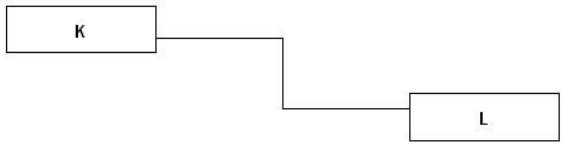


图 3.11 “折线”关系

3.3.2 “包含”关系

一个部门有多个员工，用类图可以这样表示，如图 3.12 所示。

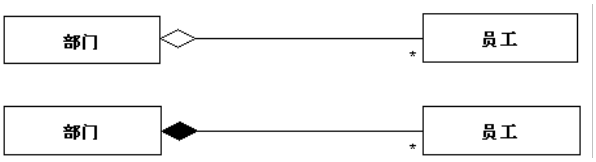


图 3.12 “包含”关系

这里有两种表示法，一种是空心菱形，一种是实心菱形。两种菱形表示包含的强烈程度不同，

空心菱形是“弱”包含，实心菱形则是“强”包含。可以这样记忆：空心菱形是空心的，显得虚弱一点，这是“弱”包含；实心菱形是实心的，显得更加强壮，这是“强”包含。

“弱”包含表示如果部门没有了，员工也可以继续存在；“强”包含表示如果部门没有了，员工也不再存在。这两者的另外一个重要区别是：如果是“弱”包含关系，儿子可以有多个父亲（当然只有一个父亲也是可以的）；如果是“强”包含关系，则儿子只能有一个父亲。

做软件需求分析时，我往往会将所有的包含关系画成“弱包含”，如果后面发现某些关系可以表示为“强包含”时，我才转为实心菱形。

请留意包含的方向，谁包含谁，刚学习的朋友很容易把方向画反了。

员工这边的“*”号表示零到多名的意思，如果是“1..100” 则表示 1 到 100 名；而部门这边没有具体的数字，则表示是“1”，则一名员工只能属于一个部门。如果一名员工同属于多个部门，那应该怎样画呢？如图 3.13 所示。

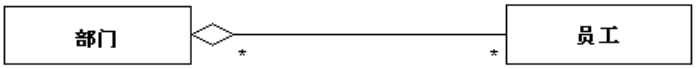


图 3.13　部门与员工的多对多关系

图 3.13 部门这边的“*”表示一名员工可同属于多个部门。请注意，在“强包含”关系中，一名员工只能属于一个部门。

“弱包含”、“强包含”的说法只是一种方便大家记忆和理解的老土说法而已，空心菱形的 UML 中文术语标准说法是聚合（Aggregation），实心菱形是组合（Composition）。以前看 UML 资料遇到聚合和组合两个词都会让我头晕一番，因为那些解释说得太复杂了，就算是现在我遇到这两个词也需要稍微停顿一下来想一想。刚学习包含关系的朋友，建议你只需要记住“弱包含”“强包含”的说法就可以了。

3.3.3　“继承”关系

我以前的公司有一个每日培训的制度，由公司内部员工做讲师，分享知识和经验。员工可以做学生，也可以上台做讲师，图 3.14 是学生和讲师的类图。

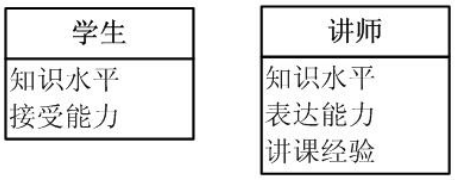


图 3.14　学生和讲师

请思考，学生和讲师有什么共性呢？

学生和讲师不都是员工吗，凡是员工都有这样的属性了，如图 3.15 所示。

图 3.15　员工

说明：此图只列了员工的三个属性，仅作示意。

员工、学生、讲师可以表示为如图 3.16 所示的关系。

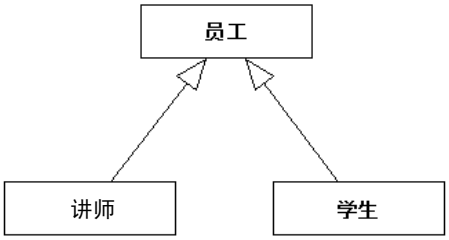


图 3.16　员工、学生、老师关系

学生、讲师都“继承”了员工，他们具备员工的属性，同时也有自己特有的属性。另外一种说法是：学生、讲师是员工的一种。

“继承”的基本画法如图 3.17 所示。

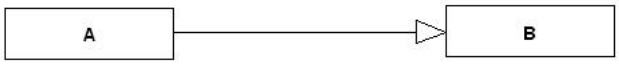


图 3.17　“继承”关系

这表示 A 继承了 B，A 具备 B 的特点，同时也有自己特有的特点，注意不要搞错继承方向。

“继承”同样是一种老土的说法，UML 中文术语标准是泛化（Generalization），该图可这样读：A 泛化为 B。泛化这个词比较难理解，你可以理解为抽象、提炼等。

在实际的软件需求分析工作中，我们往往有两种认识事物的角度，我们以员工、学生、讲师的关系为例子来说明。

角度一：在培训现场，我们看到的是学生和讲师，后来你发现，原来讲师是内部员工来的！于是你可以从学生和讲师这两个类出发，发现学生和讲师其实都是员工！

角度二：作为这个公司的领导，希望公司形成一种学习和进步的风气，促进公司的进步，于是领导希望员工之间能互相分享知识和经验。从这个角度看来，领导先想到的是员工，然后再进一步发现员工可以当学生也可以当讲师。

在泛化关系中，以图 3.17 为例，我们有可能先发现 A，然后导出 B，这时可以说由 A 泛化为

B；也有可能是先发现 B，然后导出 A，这时可以说 A 继承 B。泛化（继承）是我们进行业务提炼的重要手段，后面我们将有更多的具体例子和练习。

3.3.4 “依赖”关系

如果一个烟鬼嗜烟如命，没有烟不能生活，用类图可以这样表示，如图 3.18 所示。

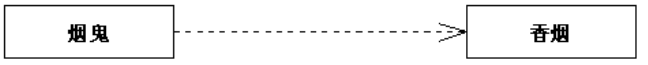


图 3.18 烟鬼与香烟的关系

这个虚线箭头就是依赖（Dependency）关系，这里虚线箭头与导航关系的实线箭头很相似，注意不要搞混了，两者表示的意思是完全不一样的。

如果说类 A 依赖于类 B，类图表示为如图 3.19 所示。

图 3.19 依赖关系

所谓的依赖关系，依赖程度是相对而言的，不一定是 A 没有 B 就不能“生存”了。在具体的业务逻辑中，对于某个事情，A 需要 B 来协助才能完成，这样也是一种依赖。

这个小节内容非常多，你可能有点消化不良了。上面介绍的内容其实在需求分析工作中是经常需要用到的，而其中最常用的是直线关系。

下面开始将会通过一个个的练习来帮助你理解和巩固这些知识，强烈建议你看完题目后先独立思考完成，然后再继续看参考答案。

3.4 演练类之间的关系

练习 1、2、3 是简单的小练习，而练习 4 的难度会有所增加。这些练习不仅仅是让你巩固上小节学习的知识，中间还会穿插一些前面还没有介绍的基础知识，而且会让你体验什么是面向对象分析，领悟用类图分析需求的要诀。你准备好接受挑战没有？

3.4.1 练习 1：你和你另外一半的关系

你结婚了吗？如果你已婚，那么请你用类图描绘你和你的另外一半的关系。

如果你是单身的，你有男朋友或女朋友吗？有的话，请你用类图画出你们两人的关系。

如果你还没有另外一半，而你又已经到了适合恋爱的年龄，那请你虚拟一位你的意中人，用类图画出你和你的虚拟意中人的关系。

如果你还没有到恋爱或结婚的年龄，那么你不需要完成这个练习，直接看后面的参考答案。

如果你是已婚人士，那么你们的关系应该如图 3.20 所示。

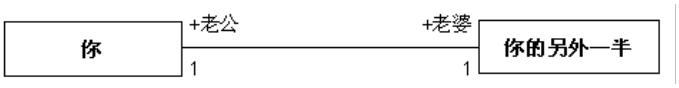


图 3.20　你和你的另外一半关系 1

如果你是男生，你在这个关系中的角色就是老公，如果你是女生，你就是老婆。一个老公只能对应一个老婆，你应该不会画出 1 对多吧？

这个图也可以画成这样，如图 3.21 所示。

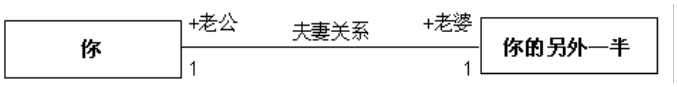


图 3.21　你和你的另外一半关系 2

直线上面的“夫妻关系”表示这个关系的名称，你可以为关联关系命名，但这不是必需的，在需求分析工作中也很少有这种需要。

如果你未婚，但你同时有多个男朋友或者女朋友，那么你们的关系可以这样表示，如图 3.22 所示。

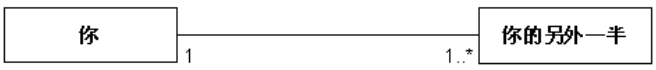


图 3.22　你和你的另外一半关系 3

“1..*”表示 1 到多个，不要因为你能 1 对多个男朋友（或女朋友）就很开心，这是一种很不好的关系，强烈建议你将 1 对多的关系变为 1 对 1，而且说不定有朝一日你会被别人 1 对多。

如果你还没有另外一半，你可以画成这样，如图 3.23 所示。

图 3.23　你和你的另外一半关系 3

你的另外一半是作为“虚拟情人”存在的。

如果你很爱你的另外一半，你依赖于你的另外一半，没有她（他）你简直不能活，她（他）是你的生存必需品，你可以画成这样，如图 3.24 所示。

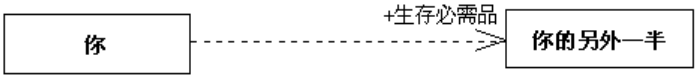


图 3.24　你和你的另外一半关系 4

你可以跟你的另外一半画画这个图，跟她（他）解释一下是什么意思，你的另外一半一定开心死了。

用类图表达你和你的另外一半的关系，并没有固定的标准答案，你画出来的可能跟上述的参考

答案不一样，只要你的逻辑正确，这个图就是合适的。

下面介绍读图检查法，能帮助你检查类图画得是否合适。

你可以分别从左到右、从右到左来读图，看看有没有不合理的地方。以图 3.22 为例，从左到右读：1 个你对应 1 个到多个你的另外一半。从右到左读：1 个你的另外一半对应 1 个你，而不要读成：多个你的另外一半对应 1 个你。注意由“多”的一边往另外一边读时，仍然是 1 个什么对应多少个什么，无论你从哪边开始读起，都是以“1 个……”开头。

3.4.2 练习 2：公司与雇员的关系

前面学习了部门与员工的关系，公司与雇员是怎样的关系呢？请用类图画出来。

图 3.25 表示公司“包含”多名员工，而公司这边也有一个“*”号，这表示一名雇员可受雇于多家公司。事实上很多公司是禁止员工同时受雇于另外一家公司或者是兼职的，这样公司这边就不能画“*”号。

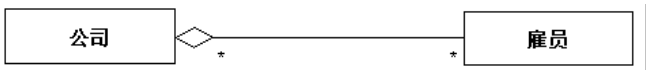


图 3.25 公司与雇员的关系

这里的包含是弱包含，能不能画成强包含呢？公司如果不存在了，雇员还存在吗？一个公司没有了，这个公司应该就不会有任何雇员，但不代表原来的雇员都消失了，他们还是存在的。这个问题就比较纠结了，到底是弱包含还是强包含，每个人的标准可能不一样，我不建议在弱包含还是强包含上过于纠结，我做需求分析时绝大部分情况只会用弱包含，强包含只会在很明显的情况下才用。

3.4.3 练习 3：香蕉、苹果、梨子的关系

你吃过香蕉、苹果和梨子吗？这三个东西有怎样的关系？请用类图画出来。

你可能觉得这个练习有点“无厘头”，这三种水果能有怎样的关系？它们无非都是可以吃的！

图 3.26 表示香蕉、苹果、梨子都是水果的一种，这就是这三者的关系。用专业一点的说法就是：香蕉、苹果、梨子泛化为水果。和前面提到的老师、学生泛化为员工不一样，员工是确实存在的，而水果只是一种泛称，没有一样东西的名字直接叫水果的，我们见到的水果都是具体的一种水果。泛化以后的类，有可能是一种经过“抽象”后的东西，这个东西是看不到摸不着的，是我们脑袋里面提炼出来的一种概念。

香蕉、苹果、梨子泛化为水果，水果可以再泛化为食物，食物又可以进一步泛化。有没有必要不断泛化呢？泛化到怎样的程度才是合适的呢？一般来说，如果有 A、B、C 等两个或者以上的业务概念，我们发现它们有一些共同的特征，则可以考虑将它们泛化为另外一个东西，这样能帮助我们发现事物的本质；但如果只有一个 A 时，就没有必要对 A 再进行泛化，例如：香蕉、苹果、梨子已经泛化为水果了，而水果则没有必要泛化为食物。当然这只是一般准则，具体要泛化到怎样的层次要看具体的业务分析需要，要靠你自己来把握。

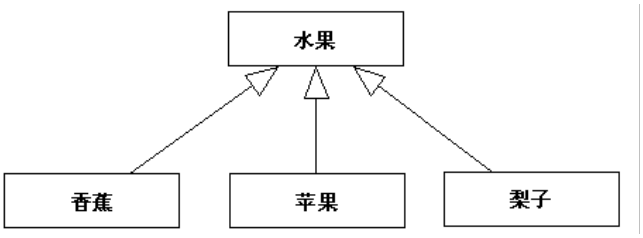


图 3.26　香蕉、苹果、梨子的关系

3.4.4　练习 4：公司的组织架构

这个练习开始有点复杂了，请你用类图描述你所在公司的组织架构。如果你们公司比较庞大，你不是很了解整个公司的组织架构，那么请你选择你熟悉的部分用类图来描述它的组织架构。如果你是学生，那么请你描述你所在大学、学院或学系的组织架构。

我们可以用组织架构图来描绘组织架构，为什么要用类图来表达呢？组织架构图画起来很方便，用类图反而觉得有点别扭。用类图来表达组织架构，是不是应该有更大的好处呢？请你带着这些问题来完成这个练习。

某公司只是一个中小型的公司，该公司由一个一个的部门组成，用类图表达其组织架构可能是这样的，如图 3.27 所示。

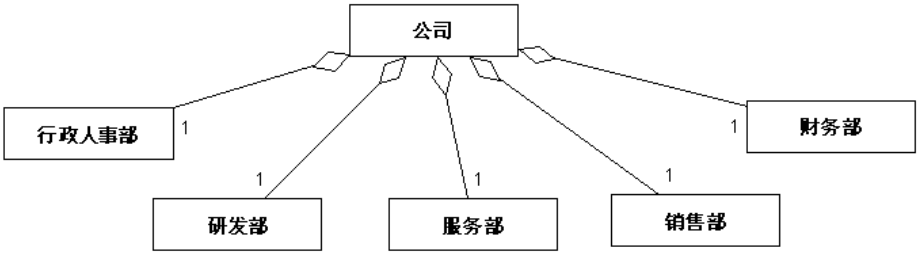


图 3.27　公司的组织架构 1

该公司有一个行政人事部、一个研发部、一个服务部、一个销售部、一个财务部。这个图中如果公司有多少个部门，就多画一个包含就搞定了，这样画似乎一点都显示不出类图的优势。

图 3.28 这种画法又如何呢？

注意图中抽象部门这四个字是斜体字，这表明这个类是抽象类（Abstract Class），抽象类表示这个类是提炼出来的一种概念，不是具体存在的，具体存在的是继承抽象部门的各个具体的部门。

前面提到的香蕉、苹果、梨子泛化为水果，水果其实也是一种抽象的概念，前面那个图的水果可以画成抽象类。

这个组织架构图已经一定程度地揭示了公司组织架构的本质，一个公司无非就是由一个个部门组成的，只是每个公司具体的部门可能不一样而已。这样的表达效果，用普通的组织架构图是表达不出来的，而类图就可以发挥抽象和提炼的优势。

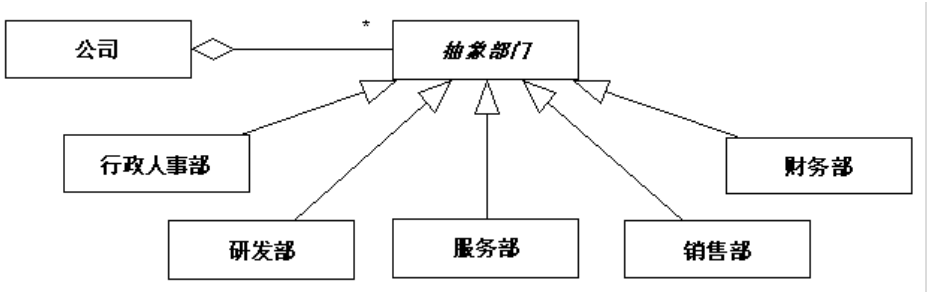


图 3.28　公司的组织架构 2

图 3.29 这个图将更进一步揭示公司组织架构的本质。

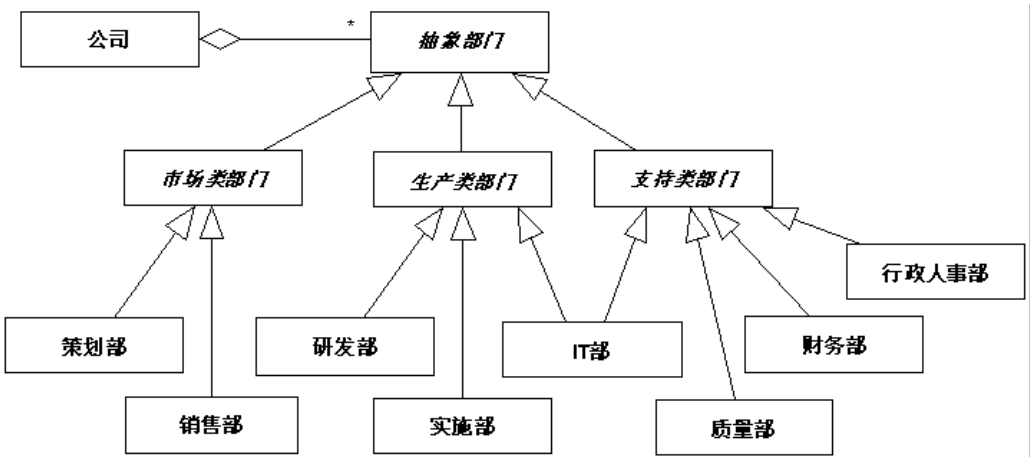


图 3.29　公司的组织架构 3

公司由一个个的部门组成，但要构成一个完整的公司，这些部门一般分为三类：

（1）市场类部门：负责公司形象推广、产品营销方面的部门。

（2）生产类部门：直接生产公司产品的部门。

（3）支持类部门：不直接生产公司产品，但是支持产品生产或支撑公司运作必不可少的部门。

在这个图中，市场类部门有策划部、销售部，生产类部门有研发部、实施部、IT 部，支持类部门有 IT 部、质量部、财务部、行政人事部，其中 IT 部既是生产类部门，也是支持类部门。

下面对其中一些具体部门进行解释。

实施部是负责将软件系统安装到客户现场，保障系统上线运行的部门。

IT 部主要负责两方面的职能，一方面要保障公司内部的办公软硬件环境，另一方面承接一些外部的网络工程，直接为公司赚钱。第一方面的工作是属于支持类方面的工作，而第二方面的工作则是生产类的工作。

质量部负责测试及过程保障的工作，这个部门是支援研发部和实施部工作的，故也属于支持类的部门。

将部门分为市场类、生产类和支持类，只是其中一种的抽象方法，每个人可能会有不同的标准，遇到不同情况可能会有不同的抽象办法。以上这个仅是一个例子，你千万不要将其当成一个固定的标准。

总体来说，上述三个用类图表示的公司组织架构，所针对的公司都不是大型的公司，大型的公司可能会有分公司、子公司、事业部等不同的划分办法，组织架构异常复杂，想用类图准确地表达出来并且能揭示其本质相当不容易。希望通过上述三个例子，能让你初步体会用类图提炼业务的优势。

上面四个练习，基本覆盖了你在前面小节学习到的类之间关系的知识。在我的经验看来，直线关系（关联关系）、包含关系是最常用的，泛化关系（继承关系）用得也比较多，而依赖关系用得不是很多。而从使用的难度来说，泛化关系（继承关系）是最考验人的了，很考验你发掘事物本质的能力。

类图是不是很有意思呢？下面小节将会更加有意思，但同时难度也会进一步增大，喜欢挑战的你一定不要退缩！

3.5 类的“递归”关系与“三角”关系

这个小节是类图的进阶知识，有一点难度，但这些知识在需求分析工作中非常实用。

3.5.1 “递归”关系

我在以前公司面试过的人数可能有数百人，如果面试者说他懂类图，我 100%会问这个问题：Windows 操作系统中有文件夹与文件，请你用类图表达出文件夹与文件的关系。

经过前面的学习，你已经具备了能画出这个图的基本知识了，请你不要看后面的参考答案，先自己尝试完成这个题目。

很多面试者很快就会画出这样的图，如图 3.30 所示。

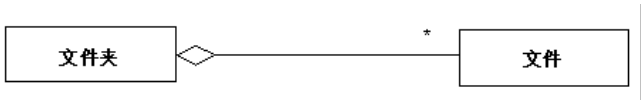


图 3.30 文件夹与文件关系 1

我会接着问这些问题：

1．文件夹里面也可以有文件夹啊，这个怎样表示出来？

2．文件夹里面的文件夹的里面也可以有文件夹啊，咋办？

很多面试者傻眼了，只有很少数人可以画出来。

其实画不出来也不用灰心，我刚学习类图时，也被这个题目一下子难倒了，后来看到参考答案后恍然大悟，同时对类图产生一种莫名的敬仰！

如图 3.31 所示，文件夹里面有文件夹，里面的文件夹里面可能有文件夹，这可能是无穷无尽

的“递归”啊！而这个包含关系可以自己指向自己，可以“自包含”，这个无穷“递归”的问题就解决了，实在太完美了！

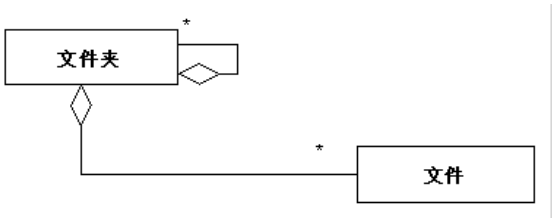


图 3.31　文件夹与文件关系 2

无论是弱包含还是强包含，都可以“自包含”。除了“自包含”可以形成“递归”，其实直线关系同样是可以指向自己的，这个叫“自关联”，这样也形成了“递归”关系，请看图 3.32。

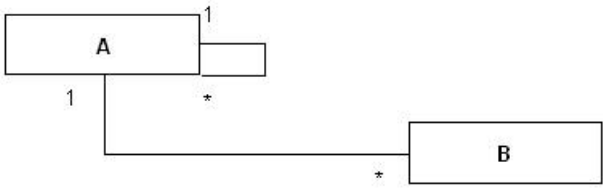


图 3.32　自关联

这种“递归”结构，一旦展开就会形成一棵树型的结构。需求分析时如果发现树型的业务结构，你可以考虑使用“自包含”或者“自关联”来分析。

其实“自包含”、“自关联”的说法是不严谨的，只是方便记忆和理解，实际上具体的一个文件夹是不会包含自己的。这里我们需要进一步理解类图中的每一个类所代表的意义，一个类并不是指具体的一个业务对象，一个类泛指属于这个类的任意一个业务对象。这里的解释可能还不够清楚，你可以暂且放下，在对象图的小节中我们再具体说这个问题。

3.5.2　“三角”关系

前面有个练习，要求你画出公司与雇员的关系，现在要求你分别列出公司和雇员至少 3 个关键属性。

待你列出关键属性后，请你思考这些问题：

1．薪金是雇员的关键属性吗？合同期、职位呢？

2．公司与雇员，这两者的关系在法律上是如何确立的？

你一定会想到，公司与雇员要签署劳动合同，而劳动合同上会有薪金、合同期、职位这些重要的内容，那么薪金、合同期、职位还算是雇员的属性吗？公司、雇员、劳动合同这三者是怎样的关系？

图 3.33 中在表示公司与雇员的关系的直线上，拉出一条虚线，虚线另外一端连接劳动合同类，这样的类叫做关联类（Association Class），关联类是对两个类的关系的进一步约束。

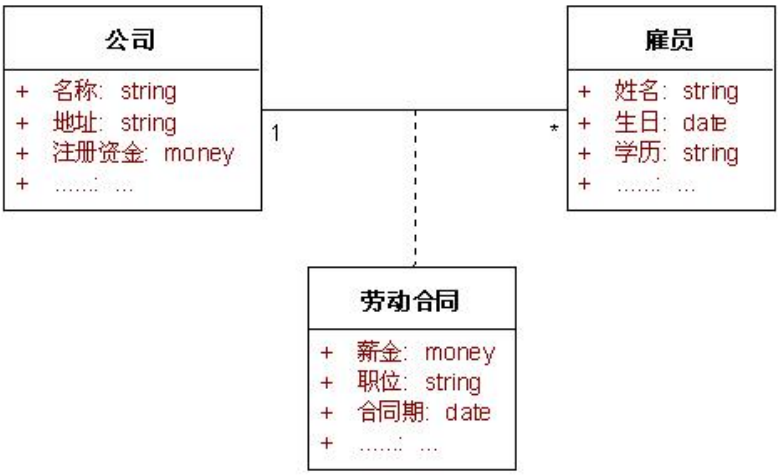


图 3.33　公司、雇员、劳动合同的关系 1

最开始你可能会认为薪金、职位、合同期这些似乎应该是雇员的属性，但现在你应该认识到，这三个关键内容应该体现在公司与雇员的关系上，体现在劳动合同上。再进一步思考，雇员的薪金、职位、合同期是会变化的，你可能会跟同一个公司签署多份劳动合同，也可能是签署一份劳动合同后又有多次合同变更。

要识别出能表征两个类关系的关联类，难度是有点高，有这样的一些实践建议供你参考：

1．如果觉得两个类有关系，则先拉上一条直线再说。

2．如果觉得两个类有关系，但怎样画都觉得这个关系不太合适，那么可以思考是不是漏了一个关联类。

3．分别列出这两个类的关键属性，思考这些属性的属性值是不是由该类本身就可以确定了。例如：如果我们最开始将薪金作为员工的属性，那么你可以思考薪金的具体数字，是不是员工自己本身可以确定的？你会发现薪金其实是由公司和员工商定后确定的，并不是员工自己本身可以决定。

4．通过对属性的思考，可能会发现这个属性应该是属于另外一个类的，思考这个类是不是能表征原来两个类关系的关联类。

关联类这样复杂的东西，客户是不太可能直接告诉你的，你需要在需求分析中发现和提炼出关联类，这对需求的理解以及项目后期的设计工作将会有很大的帮助。

将薪金、职位、合同期这些信息直接当成是雇员的属性也不是不可以的，这跟我们做系统的目标很有关系。如果我们只是做很简单的员工信息管理，可能就没有必要将合同提炼出来。如果我们要做一个人事管理系统，甚至要产品化，这样就需要我们将业务模型分析得更加透彻。

回到前面的“3.4.4　练习 4：公司的组织架构”，如果我们要做一个通用的公司管理系统，我

们希望能尽量适应不同的公司情况，那么例子中对公司组织架构的提炼是很必要的，如果我们能看清楚公司部门架构的本质，那么就能尽量多地适应不同的情况。

我的实践建议是：在需求分析阶段应尽量对业务分析得透彻一点，这样后期工作将会更加主动。业务需求模型最终变成设计模型时，我们可以把握设计的“度”，可以做出弹性很高的设计，也可以做出一个比较“老土”的设计。

公司、雇员、劳动合同的关系，其实还可以画成，如图 3.34 所示。

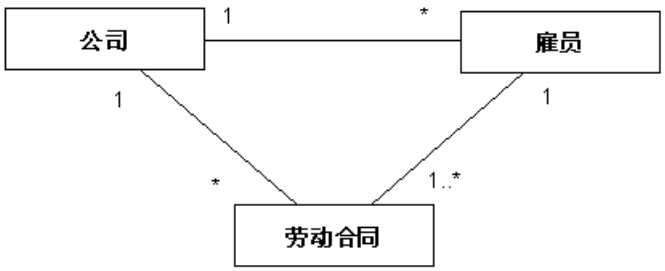


图 3.34　公司、雇员、劳动合同的关系 2

这个图可能最体现它们的“三角”关系了，关联类也可以表达成这样的方式。但我在实际工作还是以关联类的方式来表达，我觉得关联类的表达方式更加贴切和专业一点。

在具体的需求分析工作中，如果你发现三个类形成了类似该图的“三角”关系，你可以思考其中一个类是不是可能是关联类，但要注意并不是凡是出现了“三角”关系就一定会有关联类。

怎么样？本节的难度已经更上一层楼了！

类图的最大魅力在于帮助你发掘和提炼业务模型，其他的非 UML 图可能是做不到的。当然真正要做好发掘和提炼，还是需要你的深厚功力了！

下小节将要完成一个综合练习，以应用你所学习到的全部类图知识。

3.6　考试管理系统——类图综合训练

做这综合练习有以下几个目的：

1．巩固所学到的类图知识。

2．演练用类图分析需求的基本步骤。

3．学习一些提炼类的新知识。

本练习我们将会演练类图分析需求的基本步骤：

1．识别出类。

2．识别出类的主要属性。

3．描绘出类之间的关系。

4．对各类进行分析、抽象、整理。

本综合训练的题目如下：

某学校打算做一套考试管理系统，当前情况如下：

1．讲师会讲很多门课，大部分的课程需要安排一次考试，有些就不需要。

2．考试题目由讲师出题。

3．学生需要参加很多考试，每门考试都有成绩。

请你思考：

1．考试是一个类吗？如果是，考试这个类代表怎样的意思？

2．分析出与考试直接相关的类都有哪些？

3．考试类与其他类是怎样的关系？

本系统围绕考试开展，我们首先要确定考试是怎样的一个类，考试类代表考试试卷吗？还是代表考试这个事情？这是考试管理系统，不是考试试卷管理系统（当然试卷也需要在本系统中管理），需要对考试这个事情进行管理，所以考试类代表的是考试这个事情。

将需求分析中遇到的人、物、概念识别为类，这是比较容易做到的，而对于事情，例如考试，我们就不一定能将其识别为类。因为普遍认为，类代表的是一些静态东西，而事情是动态的，不适合用类来表示。这并不是绝对的，由系统的目标出发，有时候我们需要将某些事情、动作等动态内容识别为类。当我们做某某管理系统，而某某是指某个事情时，其实最终系统是通过管理该事情的记录来实现对该事情的管理。例如：考试管理系统，其实最终系统管理的是考试记录；请假管理系统，其实系统最终管理的是请假记录。为了让这些事情能被管理，将这些事情识别为类是很必要的。

考试类的意义基本确定了，它的属性有考试时间、地点等内容，现在要思考与考试直接相关的类有哪一些呢？课程、试卷、讲师、成绩、学生这些合适吗？

请你先列出与考试直接相关的类，并尝试画出它们与考试的关系，然后再继续往下看。

请注意只需要找出与考试直接相关的类就可以了，不需要找间接相关的，另外只需要画出其他类与考试的关系就可以了，至于其他类之间的关系暂不用考虑。

说明：图 3.35 并没有画出课程、讲师、学生之间的关系，此图重点表达的是其他类与考试类的关系。

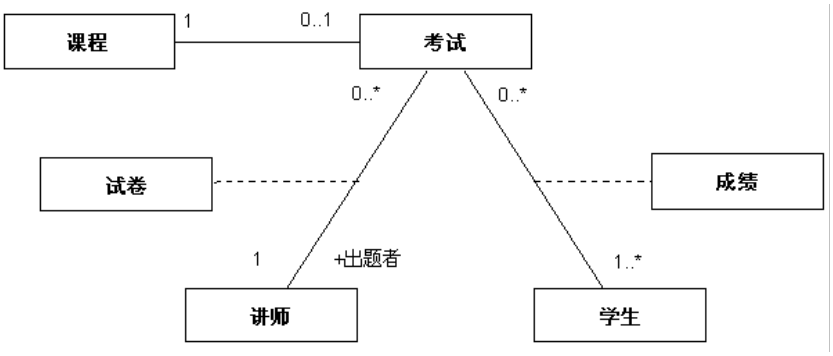


图 3.35　考试类与其他类的关系

此图表达了这样的情况：

1．每个课程要么安排一次考试，要么没有考试，而每个考试只对应一门课程。

2．一名讲师作为出题者对应零到多次考试，而每一次考试必对应一位出题的老师。

3．一名学生需要参加零到多次的考试，而每次考试有一到多个学生参加。

至于成绩和试卷，要重点说明一下。

作为一名学生，他参加一门考试就会得到一个成绩，他参加多门考试就得到多门成绩，于是就可以计算出该名学生的平均分、最高分、分数排名等，这些内容可以列为学生的属性。

作为考试来说，一次考试有很多学生参加，这样这门考试就会产生很多个成绩，根据这些成绩，我们可以算出这次考试的平均分、最高分、优秀率、合格率等，注意平均分、最高分这些也可以叫做这次考试的成绩，这些内容可以列为考试的属性。

学生的分数排名、考试的优秀率这些东西如果列为属性，这些属性可以成为“导出属性”，意思就是通过其他基础数据算出来的属性。需求分析时，我们要重点识别基础属性，基础属性是指“原生”的属性，不根据其他东西计算出来，而是直接得到的。

图 3.35 中所定义的成绩类是指一名学生参加一次考试所得到的成绩，这个成绩是原生的，通过这个成绩，系统可以从考试的角度或者从学生的角度导出很多其他统计数据。

理解了成绩这个类，应该就比较容易理解试卷这个类了。一次考试对应一名出题老师，老师为这次考试设计了一份题目，这份题目就是试卷。

上述只是参考答案，这个题目并没有标准答案，怎样识别出类以及画出类之间的关系，从不同的角度就会有不同的结果，关键还是要从系统的目标出发，做出合适的分析。如果你的答案与参考答案很不一致，不代表你画得不好，只要你能有条理地解释这些类和它们的关系，就是合适的！

通过这个综合训练的过程（而不是结果），总结以下几点实践建议供你参考：

1．从系统的目标出发来思考问题。

2．用类图分析需求的基本步骤：识别类、识别属性、画出关系、整理和提炼，只是大致的参考步骤，并不是绝对的。

3．识别类的关键属性，能让我们思考类是否合适，尝试画出类的关系也会让我们再次思考这些类是否合适。

4．多读图，“从左到右”和“从右到左”两个方向来读两个类的关系，能帮助你发现更多问题。

5．只需要表达出类的直接关系就可以了。例如：A 和 B 有关系，B 和 C 有关系，这样其实 A 和 C 也是有关系的，它们有间接关系，间接关系不需要直接画出来，只需要画出所有的直接关系，我们可以通过类图的关系网络看到类之间的间接关系。

6．不要试图用一个类图表达所有的内容。考试系统这个题目其实已经简化不少了，实际系统的类图可能有几十个甚至上百个类，要规划好用多个类图来表达不同的内容，每个类图有不同的表达重点。

7．注意识别“原生”的内容，并且根据这些“原生”内容能导出什么“导出内容”，不要将“导出内容”当成“原生内容”。

8．识别关联类是难点也是关键点，分离出关联类会让我们更加看清楚事物的本质。

9．没有所谓绝对正确的答案，关键是要有自己的合理分析，逐步求精，持续优化你的想法。

10．多练习、多讨论，逐步增强你的面向对象分析素养。

3.7 关于对象图

写过代码的朋友比较容易理解什么是对象，类（class）的实例（instance）就是对象（object）。第 1 章曾讲解过对象图，我们再来复习一下。

这是 Person 类，如图 3.36 所示。

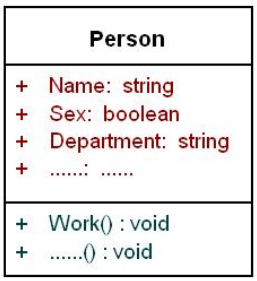


图 3.36 Person 类

下面这句代码将 Person 类实例化为 person 对象：

```
Person person = new Person();
```

用对象图表示，如图 3.37 所示。

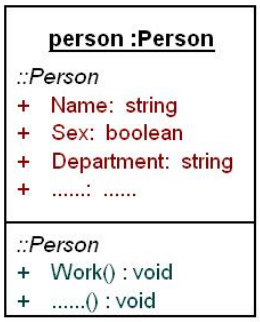


图 3.37 person 对象

一个公司包含多个员工，用类图这样表示，如图 3.38 所示。

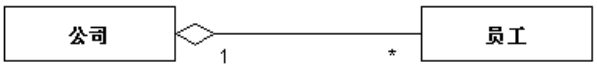


图 3.38　公司和员工的关系

此图的公司和员工并没有指具体是哪个公司或者哪个员工，如果某公司 A 有甲、乙、丙三位员工，用对象图则可以表示为如图 3.39 所示。

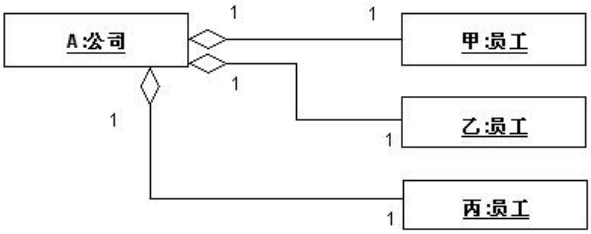


图 3.39　公司 A 与员工甲、乙、丙的关系

“A:公司”表示对象 A 是公司这个类的实例；如果是“:公司”，则表示这是公司这个类的实例，但没有给出这个实例的具体名称。

类是某一类东西的抽象或者叫统称，而对象则是具体的一个东西，A 公司如果有 1000 名员工，那么就需要画 1000 个“包含”才能表示出 A 公司与所有员工的关系。对象与对象之间如果有关系，那肯定是一对一的关系，因为两者都是具体的东西，不可能存在第二个。比方说张三和李四是好朋友，他们的关系就是一对一的好朋友关系，因为不可能再有另外一个张三或者李四，但如果我们将张三和李四抽象为人时，一个人可以与很多人交朋友，这样就可以建立多对多的朋友关系。

需求分析时，其实我们接触到的是一个个具体的东西，如：见到一个个具体的人，接触到一份份具体的业务数据等，这些具体的东西其实就是对象。而我们分析需求不能就事论事，我们需要将这些对象提炼为类，这样的分析才更具有代表性。软件系统并不是用来解决具体某次事件中的一个问题，而是希望能解决某一类问题。

在我的工作经历看来，需求分析工作中很少需要用到对象图。我基本不会使用对象图，而直接使用类图，在少数需要使用对象图时，我甚至会直接用类图代替，这样做也并没有不妥，而且容易理解和解决问题。

前面有一个练习，让你用类图画出你和你的另外一半的关系，其实准确地说你和你的另外一半已经分别是很具体的一个人了，应该用对象图来表达，但我觉得将其“混淆”为类图也没有什么不妥，而且更简单易懂。

前面“类的递归关系”小节提到“自包含”、“自引用”的问题，“自”的意思并不是指对象自己本身，而是指其他的属于同一个类的实例。

对象图就简单介绍到这里，如果你对类图还不是很熟悉的话，建议对象图了解到这样的程度就可以了。

3.8 小结与练习

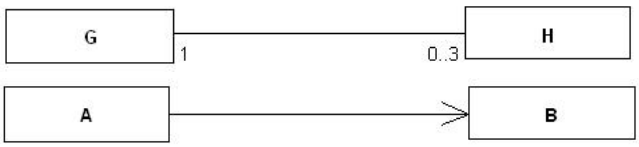
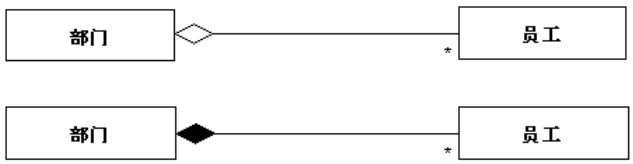
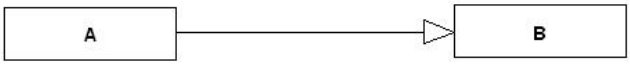
3.8.1 小结

类图是最常用的 UML 图，是用来训练你 OOA（面向对象分析）思想的最好武器。类图的语法不算很难，要看懂类图难度不大，但要用好类图就相当不容易了。

本章一开始，专门对开发人员进行了“洗脑”，端正你对面向过程和面向对象的认识。如果你不是开发人员，那么这个“洗脑”就可以免了。

接下来你学习了一大堆类图的基本语法，并做了很多练习，你还记得表 3.1 列出来的内容吗？

表 3.1 类图基本语法

通俗叫法	学术名称	图例
直线关系	关联（Association）	
包含关系	聚合（Aggregation） 组合（Composition）	
继承关系	泛化（Generalization）	
依赖关系	依赖（Dependency）	

还学习了类图的“递归”关系与“三角”关系，如图 3.40、图 3.41 所示。

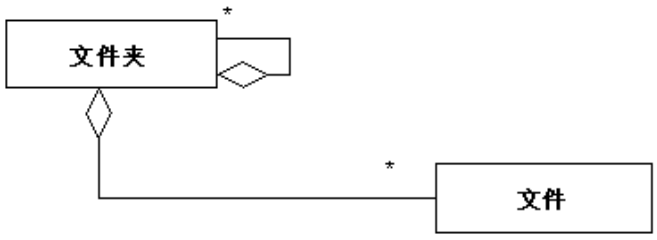


图 3.40 “递归”关系示例

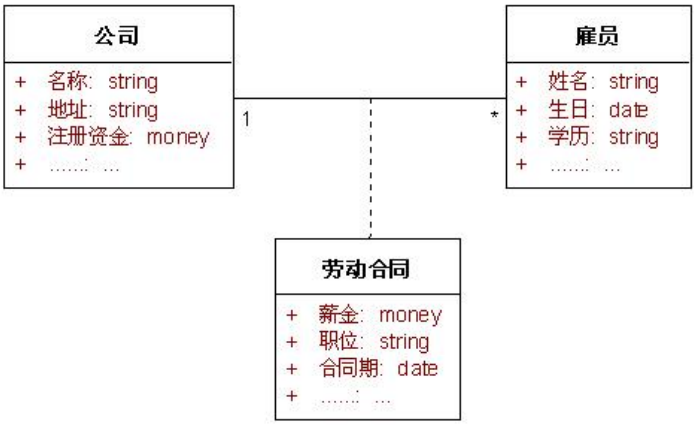


图 3.41 “三角”关系示例

一个个练习除了让你巩固学到的类图知识，更重要的是通过具体的实例让你体会用类图分析问题的思路和方法。

类图分析需求的基本步骤：

1．识别出类。

2．识别出类的主要属性。

3．描绘出类之间的关系。

4．对各类进行分析、抽象、整理。

类实例化后就是对象，表达这些对象及对象关系的图，就是对象图。需求分析中很少需要使用对象图。

多思考、多练习、多讨论、多总结，不断锻炼和提升你的面向对象分析能力吧！

3.8.2 练习

1．一辆小车有 4 个轮子，请用类图表示出来。

2．一辆货车也有 4 个轮子，但货车的前轮和后轮不太一样，用类图如何表示？

3．请用类图表示项目组的人员组成。提示：请思考项目组包含怎样的角色？项目组架构是树型架构还是网络架构？

4．你要设计一个论坛，请用类图表达出分区、版块、子版块、帖子等论坛常见元素的关系。

5．请在你做过或者正在做的项目中挑选一个，用类图来分析该项目的需求或者部分需求。