D. 应用程序不调用框架，就像好莱坞禁止演员给制作方打电话推销自己一样

解析

很遗憾，据笔者所知，从来没听说过有哪个好莱坞女演员同时也是一位能提出 MVC 框架的杰出程序员。

在好莱坞，电影制作者对演员说“Don't call us, we will call you”（不要给我们打电话，我们会给你打电话），这就是好莱坞原则。好莱坞原则用来形容所有的控制流程都由框架决定，应用程序的处理则使用多态，根据需要进行调用。

本章重点

本章将介绍 OOP 带来的两种可重用技术。

一种是软件本身的重用，即准备通用性高的软件构件群进行重用，诸如类库、框架和组件等都属于这种技术。另一种是思想或技术窍门的重用，即对软件开发或维护时频繁出现的固定手法进行命名，形成模式，以供更多人重用。如今，在软件开发的各个领域提出了很多模式，本章将介绍其中最具代表性的设计模式。

我们在第 3 章到第 5 章中介绍过，OOP 的目的是提高软件的可维护性和可重用性，它拥有类、多态和继承等结构。本章介绍的内容就是有效利用这些结构，实现实际的大规模重用。

6.1 OOP 的优秀结构能够促进重用

在使用 OOP 开发应用程序的情况下，并不是每次都从零做起，通常都是使用已经存在的可重用构件群，如源代码或运行形式的模块。这些可重用构件群称为类库、框架或组件等。

另外，框架、组件等词语有时会作为潮词使用，以强调面向对象的优点。这种潮词的定义本身就有些模糊，随着时代的进步，其含义也会发生变化，甚至会因过时而不再使用。不过，从某种意义上说，语言也是有生命力的，所以其定义会随时间变化，这也是没有办法的事情，现在也很难再重新定义整个业界的术语了。因此，本章是在接受这种术语定义模糊的前提下，来介绍这些可重用构建群的。

另外，本章还将介绍 OOP 带来的思想的重用。所谓思想的重用，是指对各种技术窍门和手法进行命名，实现模式化。如今，这种模式遍布于设计、编程、需求定义和开发流程等各个领域，其中最广为人知的就是设计模式。

为了使优秀的设计思想能够重用，通过对其命名并实现文档化，就形

成了设计模式。与前面介绍的可重用构件群不同，设计模式并不可以直接用来编程，但通过熟练运用设计模式，可以提高应用程序的灵活性和可重用性。另外，掌握设计模式对理解并熟练运用类库和框架结构也非常有用。因此，对于使用 OOP 进行应用程序设计和编程的人来说，这是必备的知识。

在介绍各个内容之前，我们来看一下可重用构件群和设计模式的发展历程（图 6-1）。

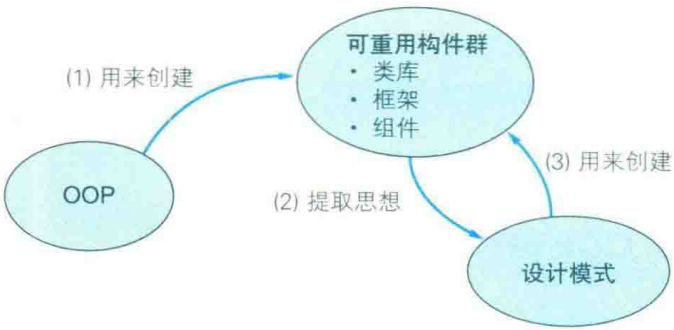

图 6-1　可重用构件群和设计模式的发展历程

从历史上来说，二者是按照图 6-1 中的 (1)(2)(3) 的顺序发展的。也就是说，首先，利用 OOP 创建可重用构件群。然后，提取可重用构件群中共同的设计思想，形成设计模式。最后，为了创建可重用构件群，会利用设计模式。

这里，图 6-1 中的 (2) 和 (3) 是循环的，表示两者会相互促进。也就是说，将创建可重用构件群时的技术窍门提取为新的设计模式，并用于接下来的可重用构件群的开发。

OOP 的优秀结构不仅促进了编程语言的进化，还为运用 OOP 的可重用构件群的发展，以及运用 OOP 的思想的重用提供了基础。

6.2 类库是 OOP 的软件构件群

我们先来介绍一下类库。

类库（library class）中的“类”就是指 OOP 结构中的类。字典中“library”的解释是“图书馆”“藏书”，所以这里可以理解为“储存了很多内容的东西”，而类库则“集合了很多具有通用功能的类”。

这种通用的类库过去就存在，称为“函数库”或者“公共子程序库”等。由于 OOP 中描述软件的单位是“类”，所以也就称为“类库”。

不过，类库并不只是简单地将描述软件的单位由子程序改为类。通过以 OOP 结构为前提，应用程序中的使用方法也有了很大进步。类库和函数库的最大不同就在于使用方法的进步。

在传统编程语言中，可重用的构件只有子程序。因此，应用程序中只是调用子程序库进行使用（图 6-2）。

图 6-2 只是调用函数库

而 OOP 中拥有类、多态和继承等结构，因此，并不只是简单地从应用程序进行调用，还可以执行下述操作（图 6-3）。

① 从库中的类创建实例，汇总方法和变量定义进行使用（利用类）。
② 可以将库调用的逻辑替换为应用程序固有的处理（利用多态）。
③ 向库中的类添加方法和变量定义，来创建新类（利用继承）。

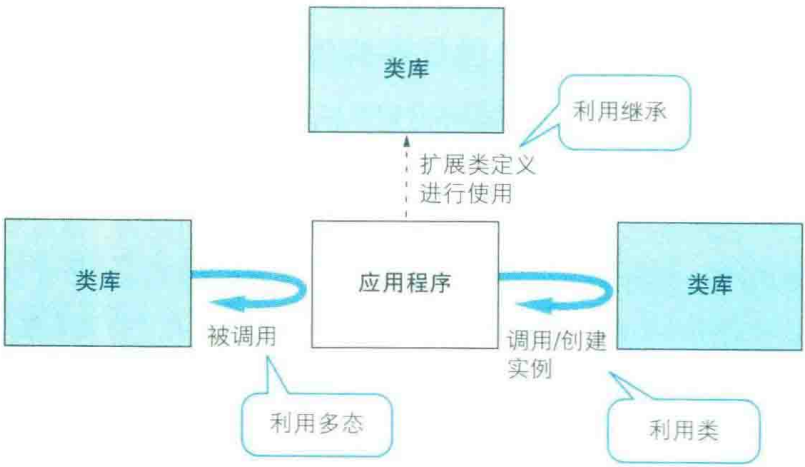

图 6-3　类库利用 OOP 的结构

通过以 OOP 结构为前提，与之前相比，可重用的功能范围得到了大幅扩展。

6.3　标准类库是语言规范的一部分

类库分为编程语言附带的标准类库和编程语言另行提供的产品（包含免费的）。一般来说，前者提供不依赖于特定应用程序的通用功能，而后者则提供面向特定用途的功能。

实际上，Java 和 .NET 中就附带了包含大量类的库，提供了字符串操作、算术计算、日期计算、文件访问、GUI、数据库处理和通信等多种功能。像这样，OOP 中通常都是将必需的功能作为类库来提供，这样既可以确保语言规范的兼容性，又能提供各种功能。因此，OOP 中附带的类库并不是语言的附属品，而是可以看作语言规范的一部分。实际上，查看 Java 之前的版本也会发现，语言规范并没有被修改很多，而附属的类库则得到了大幅扩展。

因此，在使用 OOP 进行编程的情况下，最重要的是熟练使用类库。

6.4 将 Object 类作为祖先类的继承结构

一般来说，语言的标准类库都是继承结构，继承结构的最顶端存在一个唯一的类。大家知道这个类的名称吗？

它就是 Object。

在 Java 中，名为 Object 的类是所有类的超类，如图 6-4 所示。同样，在 Smalltalk 和 .NET 中，最顶端的类的名称也是 Object[①]。Java 的 Object 类中的主要方法如表 6-1 所示。这些方法是所有类中都默认定义的方法，进行的都是最基本的处理。

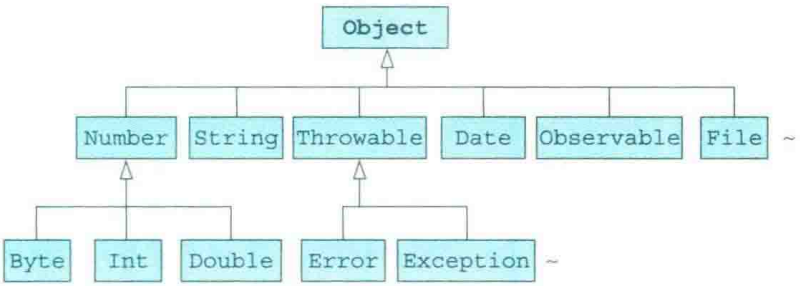

图 6-4 将 Object 类作为祖先类的继承结构

表 6-1 Java 的 Object 类持有的主要方法

方 法 名	说 明
clone()	创建对象的副本
equals()	判断是否与参数中指定的对象相等
hashcode()	获取对象的散列值
toString()	获取对象的字符串表示
wait()	暂停正在运行的线程

① 在 Ruby 1.8 之前，Object 类也位于继承结构的最顶端，但 Ruby 1.9 中又在 Object 类的上端增加了 BasicObject 类。在 Objective-C 中，最顶端的类是 NSObject 类。

笔者认为，有人将面向对象扩展到哲学和认识论领域，其中一个原因就是存在这种结构。根据该结构，包含自己编写的类在内，所有的类都是 Object 类的子类。之所以将最顶端的类命名为 Object（物体），可能也是因为只能这样叫吧。不过，这种“Object 支配所有”的结构会给初次接触的人留下“面向对象是使用程序来表示万物的技术”的印象。

6.5　框架存在各种含义

接下来介绍框架。

框架（framework）一词在英文中的含义是“结构”“骨架”，除了可被用于软件领域，有时还表示用于解决商业课题的工具和思考方法。

在软件开发领域，框架的定义有些模糊，但大致可以分为两种情况：作为“总括性的应用程序基础”这种比较笼统的含义使用；指代针对特定目的编写的可重用构件群。

前者主要用来表示开发和运行环境、商业产品的概念，比如指代使用网络技术的应用程序基础的“Web 应用程序框架”、微软公司提供的开发和运行环境“.NET 框架”等。将应用程序的整体结构分为 Model、View 和 Controller 三部分进行设计的概念“MVC 框架”也可以说是其中一种[①]。

后者是指使用特定编程语言编写的具体的软件构件群。现在用很多语言编写的框架都是开源的。在 Java EE[②] 中，主要提供 Web 应用程序的画面控制功能的 Struts、轻量级容器 Spring、用于数据库控制的 iBATIS 和 Hibernate 等都已成为事实标准。另外，以日本组织为中心推进开发的 Seasar2、Ruby 的 Rails 也很有名。

---

① MVC 框架最初在 Smalltalk 环境中被提出来时，是指使用 Observer 设计模式，在 Model、View 和 Controller 之间传递信息。但是，自从该词被用来表现 Java EE 的应用程序结构，就只是简单地表示将应用程序分为三个任务进行设计。

② 这是 Java Enterprise Edition 的缩写，是指面向企业系统的 Java 开发和运行环境的规范。

6.6 框架是应用程序的半成品

接下来，我们基于后一种定义——使用特定编程语言编写的具有特定目的的可重用构件群，来介绍一下框架的相关内容。

基于该定义，框架和类库都指可重用的软件构件群，一般会根据目的和使用方法区分使用二者。通常在称为类库的情况下，只是指利用 OOP 结构创建的可重用构件，并不限制其目的和使用方法。但在称为框架的情况下，则并不只是指利用 OOP 创建的类库，还指用于特定目的的应用程序的半成品。另外，从应用程序中的使用方法上来说，并不是像传统的函数库那样简单地进行调用，而是从框架来调用应用程序。也就是说，在框架端预先提供基本的控制流程，在应用程序中嵌入个别的处理（图 6-5）。

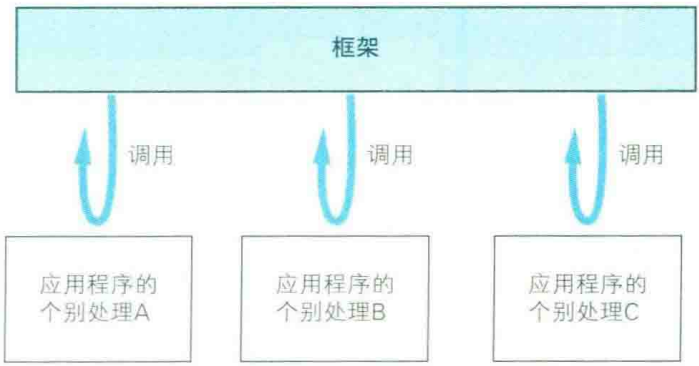

图 6-5 框架的概念

在这种框架中，多态和继承具有非常重要的作用。基本的处理由框架端提供，应用程序特有的处理则利用多态进行调用。另外，关于应用程序特有的处理，我们还会利用继承结构预先设置默认的功能。

好莱坞原则一词可以用来表示这种框架结构的特征。在好莱坞，电影制作者对演员说“Don't call us, we will call you”（不要给我们打电话，我们会给你打电话）。这显示出了制作方对演员的傲慢态度。这句话的巧妙之处在于，表示打电话的“call”可以引申为表示方法调用的“call”。也就是

说，这里借用好莱坞的话，诙谐地表示了“所有控制流程都由框架端决定，应用程序的处理则使用多态在需要时进行调用”这一结构。这对母语不是英语的人来说可能有些难以理解，但这话非常有趣，我们至少记住“好莱坞原则”是一个表示框架结构的词语吧（图 6-6）。

如果这种框架完全适用，那么我们就可以轻松地创建复杂的应用程序。基本的使用方法就是继承框架提供的默认类（或者接口），并编写一些方法。曾经备受关注的 Java Applet 等就是典型的例子，仅通过继承 Applet 类，并编写 init、start 等一些方法，就可以轻松地创建出动态的 Web 页面。正是得益于具有多态和继承结构的 OOP，这种做法才成为可能。

图 6-6　好莱坞原则

6.7　世界上可重用的软件构件群

前面介绍的类库和框架等可重用构件群的质量一般都比较高，源代码也是公开的，因此在全世界被广泛使用。尤其是现在，随着互联网的普及，再加上免费公开源代码的潮流，我们能够以非常小的成本轻松享受到可重用的好处。作为实现这种大规模、大范围的重用的基础，第 4 章介绍的避免重名的包结构，以及第 5 章介绍的消除平台差异的虚拟机，都起到了非常重要的作用。

6.8 独立性较高的构件：组件

下面我们来介绍一下组件。

组件（component）的英文具有“成分”“元件”之意。在软件开发领域，我们使用“元件”这层意思，而实际的定义则稍有不同。

前面介绍过，类库和框架经常被作为同义词使用，但“组件”这个术语的语感与它们略有差别。

组件的一般定义如下所述。

- 粒度比 OOP 的类大
- 提供的形式是二进制形式，而不是源代码形式
- 提供时包含组件的定义信息
- 功能的独立性高，即使不了解内部的详细内容，也可以使用

说起来还是微软公司开发的 Visual Basic 的控件将“组件”这一术语渗透到业界的。具体来说，就是提供控制 GUI 的独立性较高的构件，支持在开发环境中进行属性设置和拖曳操作，通过组合这些构件，从而轻松地创建出画面。该技术使用方便，并且出现于 Windows 真正普及时期，时间点也非常好，因此得以迅速普及。

Java 中为了使独立性较高的构件得以流通也进行了各种努力，但主要集中于 EJB（Enterprise JavaBeans）组件技术。不过，由于该技术运行性能的开销太大，并且业务应用程序的处理本质上依赖于数据结构，很难创建通用的构件，所以最终并未取得较大成功。

现如今，相比作为一种技术，“组件”这个词更多地被用于营销领域。

6.9 设计模式是优秀的设计思想集

接下来，我们介绍一下另外一个重要的可重用技术——设计模式。

类库和框架等前面介绍的可重用构件群的开发者们在工作过程中发现，

即使编程语言、操作系统等开发环境和软件的目标领域不同，也会有共同的设计思想。因此，大家就考虑将这种设计思想以某种形式进行重用。这就是设计模式。

顾名思义，**设计模式**（design patterns）就是“设计的模式”。具体来说，就是不依赖于编程语言和应用程序的应用领域，对在各种情况下反复出现的类结构进行命名，形成模式。与前面介绍的可重用构件群一样，设计模式不采用具体的源代码或者运行形式的模块等方式，而是从中提取出优秀的思想，总结成文档。

设计模式是前人为了创建便于功能扩展和重用的软件而研究出的技术窍门集，对典型的解决方法以及使用它们时需要考虑的要点等都进行了汇总，因此对从事设计、编程的人来说是非常有用的引导。这些技术窍门是前人总结出来的智慧，想要一个人从零开始考虑同样的思想是完全不可能的。

设计模式被大家广泛认知始于1995年*Design Patterns: Elements of Reusable Object-Oriented Software*[1]一书的出版。该书由来自不同国家的具有不同技术背景的4名技术人员精诚合作而成。人们将这4名技术人员亲切地称为GoF（Gang of Four，四人组）[2]，将他们共同发表的23种设计模式称为**GoF设计模式**。GoF设计模式针对经常出现的典型课题编写了3~5个类，并给出了相应的解决方法，总结了课题和解决对策、应用效果和需要考虑的要点等，同时还给出了UML[3]图和示例代码。

---

① 日本早在1995年就出版了该书的日译本，1999年再次出版了修订版，添加了包含Java示例代码的光盘。详细内容请参考本章后面的参考书籍。（中文版名为《设计模式：可复用面向对象软件的基础》，李英军等译，机械工业出版社2000年出版。——译者注）

② 核心成员埃里希·伽玛（Erich Gamma）作为Java的单元测试工具JUnit和综合开发环境Eclipse的开发者而闻名于世。

③ 实际上，由于UML在23种设计模式发表时还未出现，所以当时使用的是UML的前身——OMT表示法。

表 6-2 是 GoF 设计模式的一览表[1]。

表 6-2 GoF 设计模式

No.	分　类	模 式 名	目　的
1	适应设计模式	Iterator	逐个遍历
2		Adapter	加个“适配器”以进行重用
3	交给子类	Template Method	将具体处理交给子类
4		Factory Method	将实例的生成交给子类
5	生成实例	Singleton	只有一个实例
6		Prototype	通过复制生成实例
7		Builder	组装复杂的实例
8		Abstract Factory	将关联部件组装成产品
9	分开考虑	Bridge	将功能层次与实现层次分离
10		Strategy	整体替换算法
11	一致性	Composite	容器与内容的一致性
12		Decorator	装饰边框与被装饰物的一致性
13	访问数据结构	Visitor	访问数据结构并处理数据
14		Chain of Responsibility	责任循环
15	简单化	Façade	简单窗口
16		Mediator	只有一个仲裁者
17	管理状态	Observer	发送状态变化的通知
18		Memento	保存状态
19		State	用类表示状态
20	避免浪费	Flyweight	共享对象，避免浪费
21		Proxy	只在必要时生成实例
22	用类来表现	Command	命令也是类
23		Interpreter	语法规则也是类

在 20 世纪 90 年代前半期，这些技术窍门还不为人所知。据说大概 1000 个程序员中只有 1 个能编写出像类库这样通用性高的可重用构件群。

① 这里引用了《图解设计模式》(结城浩著，杨文轩译，人民邮电出版社 2016 年 12 月出版。——译者注)中的表述。

直到 1995 年 GoF 设计模式出现，许多开发者才知道这些有用的技术窍门。在技术革新非常快的软件领域，GoF 设计模式至今仍不过时，成为经典。

6.10　设计模式是类库探险的路标

本章开头介绍过，设计模式是基于类库和框架的开发经验形成的，这里，我们反过来介绍一下类库和框架对设计模式的利用。

首先来看一下 Java 类库中使用设计模式的示例。如图 6-7~ 图 6-9 所示，左边是设计模式，右边是使用设计模式的一部分类库。这些图是 UML 的类图，使用图形来表示类定义和多个类之间的关系。关于 UML，我们将在第 8 章中进行介绍，不了解 UML 的读者可以对比一下左右两边图形的不同。

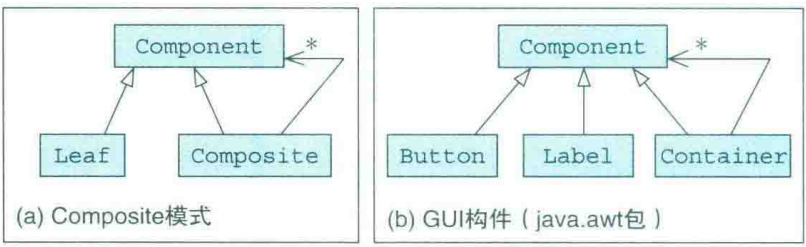

图 6-7　Java 类库中的设计模式使用示例 1（Composite）

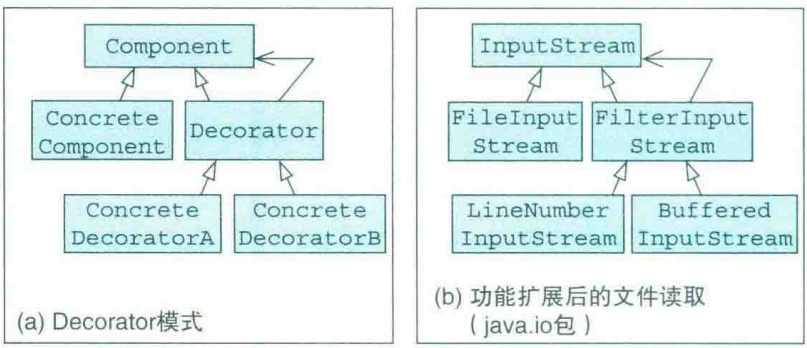

图 6-8　Java 类库中的设计模式使用示例 2（Decorator）

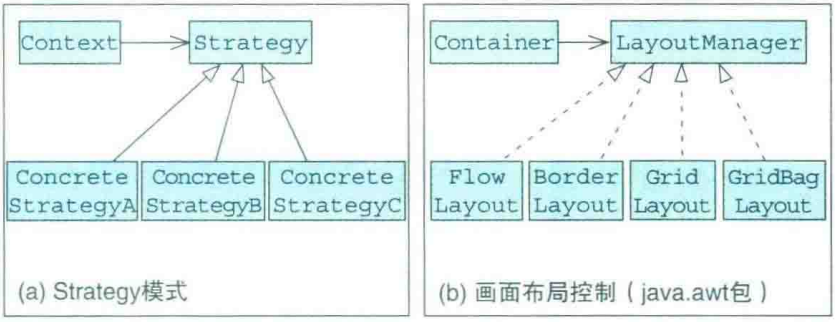

图 6-9 Java 类库中的设计模式使用示例 3（Strategy）

各图的左右两边图形都非常相似。虽然长方形框中写的类名和框的个数有所不同，另外还存在实线和虚线的差别，但是整体形状基本上都是一样的。这是 Java 类库直接使用设计模式的铁证。

有的 Java 类库是在设计模式出现之后编写的，其中充满了设计模式。除此之外，有的类名还直接使用了设计模式的名称，如 Observer、Iterator 等[1]。

像这样，如果大家能够看出其中使用了设计模式，就能够明白开发者的意图，进而也就能够理解正确的使用方法。这是因为，设计模式手册中详细记述了哪个模式适用于哪些课题，以及使用时应该注意的地方。因此，类库开发者为了传达设计意图，经常会在类名中加上设计模式的名称。

6.11 扩展到各个领域的思想的重用

传统的列表和栈等数据结构、冒泡排序和二分查找等算法都是对设计思想进行命名，设计模式也可以看作它们的发展形式。

由于 OOP 中拥有类、多态和继承等优秀结构，程序的表现力大幅提升，所以能够表示为设计模式的内容也得到了大幅扩展。

自从设计模式出现以来，在系统开发的各个领域，以同样的形式对思想和技术窍门进行积累并重用的活动普及开来。实际上，世界上许多技术

① 严格来说，Observer 和 Iterator 并不是类，而是接口。

人员和研究人员都将自己的经验和知识总结成模式并发表，而对其进行讨论和推广的工作坊也开展了起来（表6-3）。

表6-3 扩展到各个领域的模式技术

名　称	说　明
其他设计模式	限定于特定运行环境和用途的技术窍门集（J2EE模式、EJB模式和面向多线程的设计模式等）
分析模式	在业务分析和需求定义阶段编写的表示应用程序的问题领域的模式
架构模式	用于表示软件整体结构的模式
成例	有助于熟练使用Java、C++等特定编程语言的技术
重构	在不修改既有程序功能的情况下改善内部结构的技术
流程模式	与系统开发的推进相关的模式
反面模式	汇总系统开发中经常出现的陷阱及相应的对策

6.12 通过类库和模式发现的重用的好处

本章介绍了两种可重用技术——软件的重用和思想的重用。

虽然使用面向对象的好处是可重用性强，但是如果只有自己使用OOP编程，那么大概也不会感受到这种效果。

正如本章介绍的那样，现实中存在很多像类库、框架和组件那样高质量的可重用构件群。Java和.NET等附带的类库规模庞大，想要用好其中包含的所有功能，恐怕要花几年时间。

另外，除了这些软件构件之外，以GoF的23种设计模式为代表的重用优秀思想的模式技术也被提了出来。

正是得益于可重用技术，我们才得以使用这些可重用构件群和模式。另外，在对其进行使用的过程中积累的技术窍门还可以进一步形成可重用构件群和模式。

如你所见，从Simula 67开始的编程技术的革新已经发展到了这种阶段。