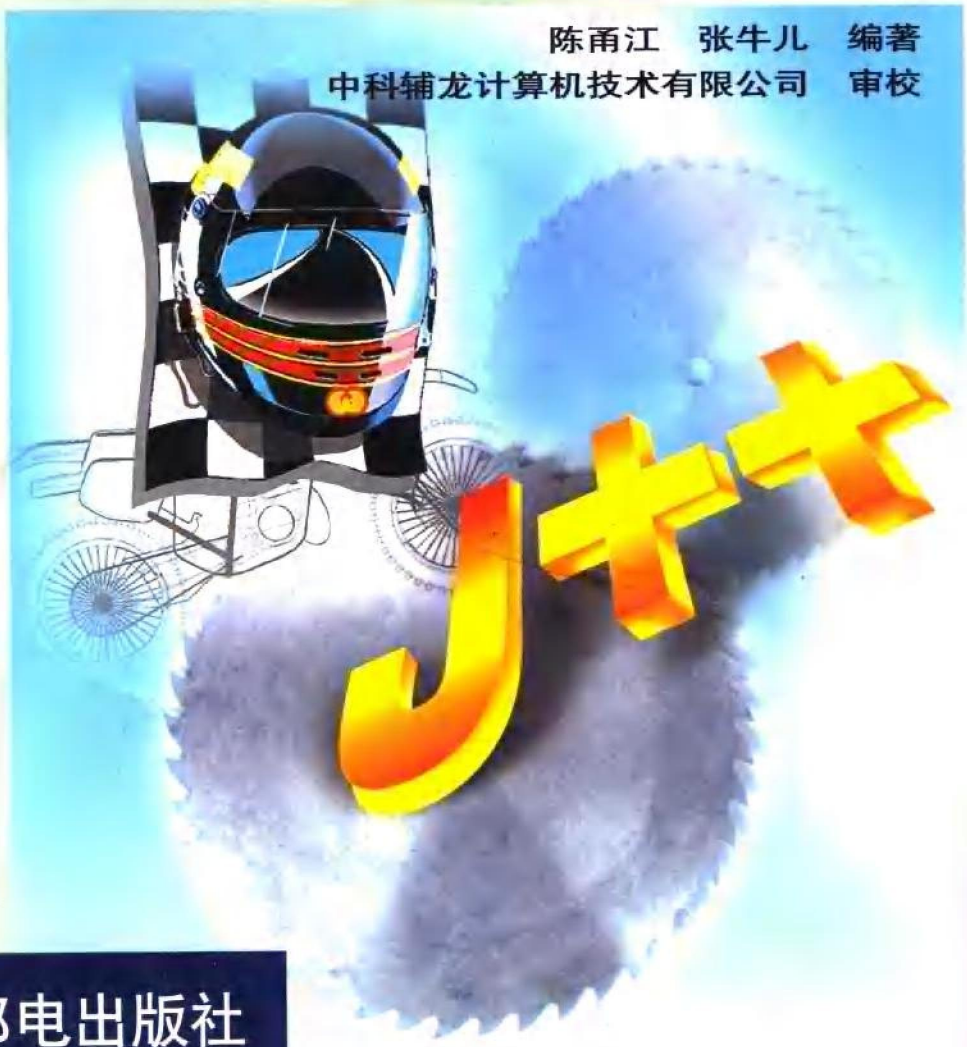
Visual J++ 6.0

在多媒体开发中的应用

陈甬江　张牛儿　编著
中科辅龙计算机技术有限公司　审校



人民邮电出版社

Visual J++ 6.0 在多媒体开发中的应用

陈甬江　张牛儿　　编著

中科辅龙计算机技术有限公司　审校

人 民 邮 电 出 版 社

内 容 提 要

本书结合多个具体实例通俗地介绍 Visual J++ 6.0 的实际应用，全书分为入门篇、进阶篇、高级篇、实践篇 4 部分共 8 章，从 Java 语言概念开始直到高级的多媒体程序设计，全面系统地讲解如何使用 Visual J++ 6.0 制作多媒体软件的方法，内容包括：Java 语言基础，图形与文本，图形界面基础，高级图形界面设计，例外处理，多线程，多媒体技术、图形、动画和声音，Java 的网络应用等。通过实际制作一个多媒体程序，读者将会在无尽的乐趣中获得所需要的知识。

本书内容实用，结构合理，适合于程序开发人员、计算机编程爱好者及大专院校有关专业师生阅读参考。

Visual J + + 6.0 在多媒体开发中的应用

◆ 编　　著　陈甬江　张牛儿
审　　校　中科辅龙计算机技术有限公司
责任编辑　赵宝珊

◆ 人民邮电出版社出版发行　北京市崇文区夕照寺街 14 号
北京鸿佳印刷厂印刷
新华书店总店北京发行所经销

◆ 开本：787 × 1092　1/16
印张：21.5
字数：534 千字　　1999 年 11 月第 1 版
印数：1 - 6 000 册　　1999 年 11 月北京第 1 次印刷

ISBN 7-115-08109-3/TP·1323

定价：32.00 元

前 言

几年之前，刚刚横空出世的网络语言 Java 语言，现已逐渐成长为功能强大、适用于各种环境的编程语言。微软公司开发的 Visual J++ 编程环境，为广大使用 Java 语言进行程序开发的专业人员提供了一个方便、快速、功能强大的工具。去年秋季，微软公司又推出了 Visual J++的升级 6.0 版本，不但强化了原来的可视化编程环境，而且集成了微软公司在软件开发中先进的技术如 ActiveX、COM 等。正确应用 Visual J++ 6.0 编程环境，可以帮助程序开发者实现运用 Java 语言方便地开发多媒体程序和网页。

本书旨在向读者介绍如何应用 Visual J++ 6.0 这个功能强大的编程软件，进行多媒体程序设计，包括图形界面、动画、声音和多媒体网页的设计。全书由浅入深、循序渐进，并结合大量实例（几乎每小节都有实例），这些实例都是笔者为了说明篇中所介绍的知识而精心设计的。对每一个实例，书中都给出了详尽的操作步骤和相应的说明。本书主要面向 Java 语言开发人员和多媒体应用开发人员。

本书主要部分由陈甬江、张牛儿、王利华、徐胜文编写，另外参加编写的有史惠康、郭美山、石利文、徐平、冯金慧、郑红、杨桂莲、闫高峰、马向英、白燕斌、郭志龙。全书由郭美山和史惠康统稿。

由于时间仓促、作者水平有限，本书如有错漏之处，敬请广大读者批评指正。

作 者

1999/4

目 录

第一篇 入门篇

第三篇 高 级 篇

第四篇 实 践 篇

第一篇 入门篇

第一章 Java语言基础

Java 语言是程序设计语言，它具有与平台无关、面向对象、安全保障和一些当今交互式 Web 开发所需的特点。本章将主要介绍有关 Java 的特点以及 Java 语言的主要语法和程序设计方法。

1.1 Java简介

在这一节里，先介绍 Java 语言的历史和现状以及 Java 语言的特点，从而使读者了解 Java 在程序设计语言中的地位及全新程序设计思想的精华。

1.1.1 Java语言的历史与现状

一、 Java 语言的历史

Java 语言的前身是起步于 1990 年的一种专门用于消费类电子产品的小型软件，它的开发者是由 SUN 公司著名程序设计专家 James Gosling 领导的一个工作小组。这个软件的蓝本是当时非常流行的 C 和 C++语言，但是为了使之能够具有更好的移植性、可靠性而做了相应的修改和特别设计。

1993 年 SUN 公司的 Bill Joy 发现这种结构中立的软件很适合 Internet 上动态、交互功能的实现和其他复杂网络应用的开发，于是组织技术力量针对网络应用而对它进行再设计和开发。1995 年 3 月，SUN 公司正式向业界公布了 Java 和其完整的技术规范。1997 年 11 月国际标准化组织正式批准了 SUN 公司的 Java 标准。Java 的标准化为它的进一步发展铺平了道路，标志着 Java 正逐步走向成熟。

二、 Java 语言的现状和发展趋势

目前，Java 语言正以惊人的速度发展着，全球一半以上的大型或较有影响的计算机企业正在或准备研制开发 Java 产品。目前 Java 的应用领域主要有以下几个方面：

1. Java 平台

主要是指 Java 虚拟机和 Java API 等软件产品。Java 平台包括：台式机上的 JDK、486 以上微机的个人 Java、家电类产品上的嵌入式 Java 以及用于智能卡的 Java CARD。

2. Java STATION

Java STATION 是一种运行 Java 的 NC(网络计算机)。

3. Java 应用

Java 应用较多的是网络应用，如电子商务、远程教育、中间件产品等等。总之，作为新生力量，Java 的发展前景是非常客观的。

1.1.2 Java 语言的特点

一、 Java 语言的工作机制

对于一个运用于 Internet 上的网络应用程序，对其性能的基本要求是良好的可移植性。因为全球最大的异构互连网络 Internet 是由各种各样的终端、服务器和 PC 机组成的，所以 Internet 上的网络应用程序需具有可在各种不同的软硬件平台上均能正常工作的能力。而传统的应用程序开发工具，如 C 或 C++只能实现源代码级的可移植性，造成了开发人员的极大负担，同时应用程序的维护和升级也面临着同样被放大了若干倍的工作量，这些都在很大程度上制约了网络应用程序的发展和推广。

Java 的诞生为解决这个问题提供了一个崭新而有效的方法。Java 的工作机制是这样的：编程人员首先编好源代码，然后经编译生成一种二进制的中间码，称为字节码，最后再通过运行于操作系统平台上的一种 Java 解释器的运行机构来执行编译生成的字节码。这种运行机制与传统高级语言有所不同，如图 1-1 所示。

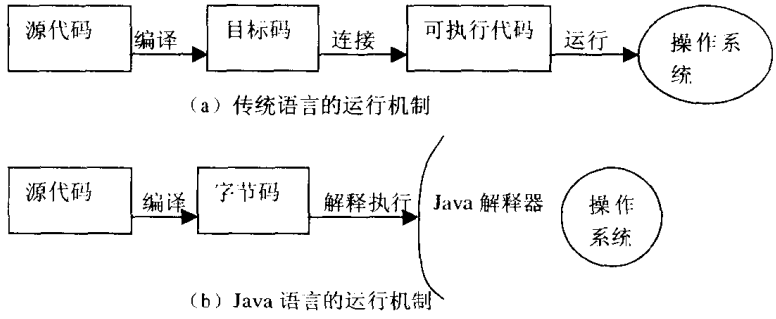


图 1-1 传统语言与 Java 的各自运行机制

传统语言的可移植性差，是因为它们的可执行程序直接作用于不同的操作系统上，所以需要能适应不同的平台环境的可执行程序；Java 的字节码则是运行在操作系统之上的 Java 解释器，不同的平台环境需要不同的解释器，但任何一个平台上的解释器，对于同一段 Java 程序的字节码来说却是完全一样的，也就是说，Java 的运行机制利用解释器来隐藏网络平台上环境的差异性，这就作到了 Java 语言二进制代码级的可移植性，在网络上成功地实现了跨平台的特性，相对于 C 和 C++之类的传统语言有着不可比拟的优势。Java 语言的跨平台运行机制如图 1-2 所示。

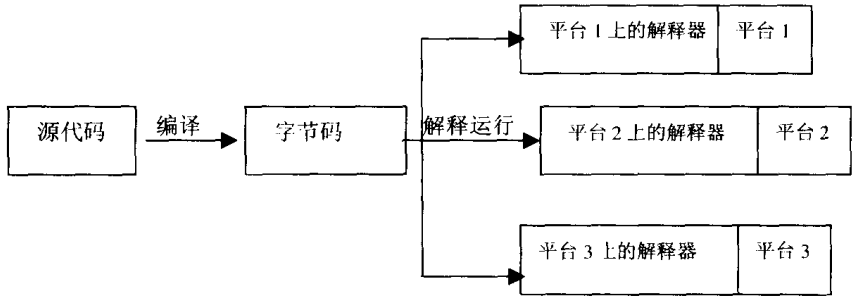


图 1-2 Java 的跨平台运行机制

Java 的解释器又被称为“Java 虚拟器”或 JVM（Java Visual Machine），是驻留在计算机内存的虚拟计算机或逻辑计算机，实际上就是一段小程序，专门负责解释执行 Java 程序编译而成的字节码。

如果 Java 解释器是一个独立的应用程序并可以在操作系统下直接运行，那么它所执行的 Java 程序被称为“Java Application”；如果 Java 解释器包含在一个 WWW 的客户端浏览器内部，使得这个浏览器能够解释字节码，则这种浏览器能够自动执行的 Java 程序称为“Java Applet”。

二、Java 语言的特点

Java 是定位于网络计算的计算机语言，同时作为面世较晚的编程语言，Java 语言集中体现和应用了若干当今软件技术新成果，如面向对象、多线程等。Java 主要特点如下：

1. 平台无关性

Java 语言独特的运行机制，使得它具有良好的二进制的可移植性，利用 Java，开发人员可以编写出与具体平台无关、普遍适用的应用程序，大大降低了开发、维护和管理的开销。

2. 面向对象

Java 是面向对象的编程语言。而面向对象技术的核心上以更接近于人类思维的方式建立计算机逻辑模型，它利用类和对象的机制将数据与其上的操作封装在一起，并通过统一的接口与外界交互，使反映现实世界实体的各个类在程序中能够独立、自治、继承；这种面向对象的方法非常有利于提高程序的可维护性和可重用性，大大提高了开发效率和持续的可管理性。

3. 安全稳定

对网络上应用程序的另一个要求是具有较高的可靠性。Java 特有的“沙箱”机制是其安全性的保障，此外，去除 C++中易造成错误的指针、增加自动内存管理等措施则保证了 Java 程序运行的可靠性。

4. 支持多线程

多线程技术允许同一个程序有两个执行线索，即同时做两件事情，Java 不但内置多线程功能，而且提供语言级的多线程支持，使得开发具有多线程功能的程序变得简单、容易和有效。

5. 简单易学

出于安全性的考虑，去除了 C++中不容易理解和掌握的部分，降低了学习的难度，同时 Java 还有一个特点就是它的基本部分与 C 语言几乎一模一样。这样无论是学过了 Java 再学 C 语言，还是已经掌握了 C 语言再来学 Java，都会感到容易入门。

1.2 面向对象程序设计的基本概念

在面向对象程序设计思想出现以前，程序员都是采用过程化程序设计方法（自顶向下或自下向上）。随着软件工业的迅速发展，原来的程序设计方法已经远远不能满足大规模生产的需要，程序员的负担加重，而编制出的程序质量却没有明显的提高。在这种情况下，面向对象程序设计方法的出现使得程序员的工作量大大减少、错误率大大降低，将程序完

全采用模块封装的概念，从而形成了软件工业的大规模生产。可以说，面向对象程序设计方法的出现完全是生产力发展到一定程度引起的，不理解面向对象概念就不能顺利进行现代程序设计，设计出的程序也不能满足软件工业在程序开发的规模性、复杂性、可靠性、质量、效率上的各种要求。了解 Java 的先进程序设计思想，从而了解、学习这一先进的程序设计语言，对于程序员和非程序员来说都是非常重要和必须的。

1.2.1 对象、实体和类

对象的概念是从生活实际中抽象而来的，它是整个面向对象程序设计技术的核心概念。当我们从面向对象的观点来看待面向对象程序时，程序中所有的元素都是由对象构成或者本身就是对象，这些对象不是孤立的、机械的，它们拥有自己管理自己的能力，能够根据外部条件的变化自身做出相应的改变，同时在这些对象之间还可以互相通信、协同工作，这样才能共同完成整个程序预定的任务与功能。

那么，如何才能够理解对象的概念呢？其实，通过现实生活中的例子我们不难理解对象这一来源于生活的概念。对象的本质是现实生活中某个具体物理实体在面向对象程序设计方法中的映射和表达。通俗一点说，当我们谈论一辆汽车时，我们所说的汽车就是某一个拥有方向盘、轮子、能够在陆地上开动的，而且可能还具有其他属性（如颜色为红色）的汽车的描述。这时的汽车就是面向对象程序设计中的一个对象。这个对象有许多属性，如有质量、方向盘、轮子、红色，而且具有一定的内部行为，可以在公路上行驶、也可以刹车、启动。这些属性和行为都可以在面向对象程序中通过对对象的定义模拟出来。

类的概念则可以理解为面向对象中某些具有某种相同属性的对象的集合或者说是抽象。类的概念非常重要，因为面向对象的程序都是由类构成的，而我们将要介绍的 Java 程序也是由一个个类组成的。例如，我们前面提到的某一对象——汽车，当我们说它是一个对象时，是因为我们所指的汽车是具有某一特定属性的汽车，如颜色为红色就是它的特定属性，而很多其他汽车颜色不一定为红色，它们可能是白色、黑色也可能是蓝色。同样，汽车又有大客车、卡车、小轿车，也有公共汽车，这些汽车分开来时是一个个的对象，而当我们把它们当作整体看待时就成为一个类，可以命名为汽车类，这个汽车类统指具有方向盘、轮子、能够在陆地上开动的物体，而不特指一个现实存在的汽车。因此，从某种意义上说，类是一个抽象的概念，它在面向对象中是具有某一共同属性的对象的集合，在面向对象程序设计中把属于类的某一对象称为类的一个实例（instance），是类的一次实例化结果。如汽车是一个类，而一辆红色的奔驰小轿车则是汽车类的一个对象，是一个具体存在的汽车。图 1-3 示出了对象与类的关系。

从图 1-3 中可以看到，当我们用面向对象的思维方式解决实际生活中的问题时，第一步是先把现实生活中存在的实际客体经过一定的抽象转化为概念中的抽象类，这个抽象类包含了现实客体的主要特性和相应的数据，然后通过面向对象程序设计的工具，如 JAVA 语言将抽象类表达为面向对象程序中的类，这样一来就可以在面向对象程序中通过实例化类（即构造对象）来实现现实世界到虚拟的面向对象之间的映射关系，通过对程序中的对象进行操作就可以解决现实生活中与之对应的问题。

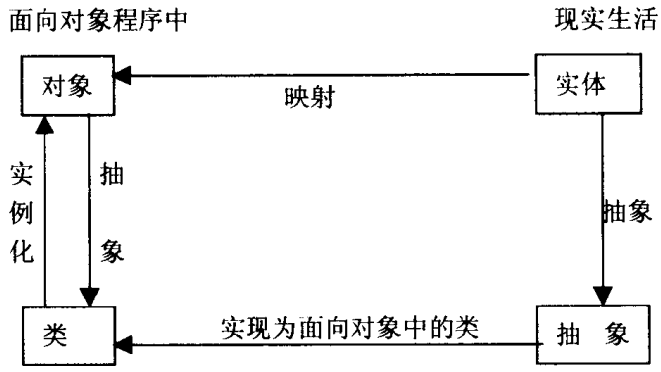


图 1-3　对象与类

因此，面向对象程序设计的主要工作，或者是第一步工作就是如何将现实生活中的物理实际客体和问题转化为虚拟计算机逻辑中的类和对象。这与传统程序设计中，将客观世界转化为计算机能够处理的数据结构的方法完全不同。面向对象从一开始就对客观物体进行对象化、模型化和封装，然后才考虑如何处理这些对象的问题。这样就大大缩小了计算机处理的抽象问题与实际生活中的现实物体之间的差别，提高了计算机解决实际问题的能力，而且在维护、修改、扩充原程序等各方面都做出了很大的提高，避免了重新设计程序的传统设计方法，减少了传统程序设计方法中不可避免的种种麻烦，加快了程序设计的速度，提高了程序设计的效率。

1.2.2　对象的属性

抽象的概念往往需要具体的实例来加以说明，才能够对它产生比较深刻的印象，理解也会容易一些。虽然人们对于对象这一概念有许多不同定义，大体说来这些定义的对象都具有以下几个特点（或者说是属性）：

（1）对象的状态；

（2）对象的行为；

（3）对象的标志。

我们将从上面三个方面来说明对象这个抽象概念，以使读者对它有一个具体的了解。

对象的状态又称为对象的静态属性，它是对象的基本属性，主要指对象内部所包含的各种信息，对象的状态（静态属性）就是我们通常所说的变量。一般而言，每个对象都有自己固定的状态（内部变量），这些内部变量表明了对象所处的状态。例如我们定义了汽车的一个对象 QCH1，那么 QCH1 的型号、颜色、速度等就是它的静态属性，当 QCH1 处于某一特定的速度，具有特定的型号、颜色时，这个对象就处于一种特定的状态。当对象经过某种操作和行为而发生状态改变时，这些改变就具体表现为这个对象的属性变量的值的改变。通过检查对象的属性变量，我们就可以了解这个对象当前所处的状态比如 QCH1 对象的状态可能为：速度 50km/h，颜色为红色，型号 XXX。当汽车速度变化时，可能从 50km/h 变为 60km/h，则 QCH1 就处于另一种状态：速度 60km/h，颜色为红色，型号 XXX。

对象的第二个属性称为行为，又称为对象的操作。这个属性用于表示对象的动态特征，操作的作用是设置或者改变对象的状态。例如汽车的操作有启动、行驶、停止等，通过这

些操作，汽车对象的状态将发生改变，如有静止变为运动，行为的作用一般都是基于本对象内部的变量即状态，用于改变对象的状态。如果要改变汽车静止的状态，则可以对汽车对象进行启动操作，对象的状态由静止变为运动。行为与状态两者是相互影响的，对象的状态也可能改变其行为，比如当汽车的速度（汽车对象的状态变量之一）为零时，汽车处于静止的状态。如果对象由速度不为零变为零，则状态的改变导致汽车对象产生刹车这一操作。对象的状态在程序中用变量来表示，对象的行为则用方法来表示。方法其实类似于传统过程程序设计中的函数，但与函数不同的是方法是对象的内部属性，而函数不能对对象本身产生影响。

对象的行为属性用方法来表示使得我们可以把对象内部的变量用方法进行封装，使其与外部完全隔绝，只有对象自身的方法才能对这些内部变量进行操作，方法同时提供了一个接口以允许对象与外界进行信息交互，使得我们通过外部环境调用对象自身的方法操纵对象的行为进而改变对象的状态成为可能。

仅仅使用对象的行为和状态并不能把各个对象完全的区分开来，比如有同一厂家出厂的同一型号的汽车，用同样的速度行驶在相同的路面上，我们就不可能将其分开。而对象的标志就是用于区分不同的对象，每一个对象度有仅属于自己的唯一的标志。国际性组织 CORBA 为对象定义了专用的 128 比特的标志量，该标志量在全球范围内通用而不会重复，从而保证了对象在不同的环境下被重复使用而不至于混淆，提高了程序开发的效率和质量，但在某些情况下为了简单起见，对象的标志也可以用程序的对象名来表示。

综上，面向对象技术中的对象具有状态、行为、标志三个基本属性，在 JAVA 程序中我们分别用对象的变量、方法和对象名来表示。这样，我们只要在程序中定义了对象的内部变量、方法和对象名就可以建立一个对象，从而将现实社会中的客体事物模拟为计算机逻辑中的对象。实际上，在程序的编制过程中，我们首先定义的是一个类（具有共同属性的对象的集合），同时定义类中对象的公共属性，包括变量和方法，然后用对象名创建类的实例——对象。

1.3 面向对象程序设计的特点

掌握面向对象程序设计的特点有利于我们设计出高效率、高质量的面向对象程序。面向对象程序设计的主要特点包括：封装、继承与多态性。

1.3.1 封装

通过对对象的属性——行为的讲解，我们可以体会到方法在封装中的作用。封装就是指利用抽象数据类型将数据和基于数据的操作封装在一起，数据被保护在抽象数据类型的内部，系统的其他部分只有通过包裹在数据外面的被授权的操作，才能与这个抽象数据类型交流和交互。

在面向对象的程序中，抽象数据类型是用“类”来表示的，前面讲过，类的实例——对象，自身具有状态和行为，因此在每个类中都封装了相关的数据和操作，对象与对象之间的相互交互只能通过被授权的行为（对象的方法）进行。封装使得类中的数据被严密保护，模块与模块之间仅通过严格控制的界面进行交互，使得传统程序设计中常常出现的模

块之间相互耦合、交叉的现象大大减少，从而降低了开发成本和过程的复杂性，提高了效率和质量，减少了错误，进而保证了程序中数据的完整性和安全性。

封装的另一个重要意义在于使程序中的抽象数据类型即类的可重用性大大提高。由于每一个类都是针对具体客观对象建立的抽象映射，它不具有个别物体所具有的特殊性，因此当我们需要在不同的场合使用同一客体对象时，完全不必象传统程序那样重新编制程序以适应环境的变化，只需要把原来设计的类拿过来用就可以了。封装使得类成为一个高度自治、自我管理结构完整的单元整体。每个类对于外部环境都只有单一的接口和功能，可适应于各种环境。大大降低了开发程序的成本和周期。

1.3.2 继承

继承可以说是面向对象中一个崭新的概念。它所表示的是面向对象程序中类与类之间的关系。当一个类拥有另一个类所有的数据和方法时，则称这两个类之间有继承的关系。例如，我们在前面定义了一个汽车类，现在我们还可以定义一个轿车类，按照大家日常的生活观念，轿车显然拥有汽车所拥有的一切属性，如可以在陆地上行驶，有四个轮子等。但是轿车拥有的属性汽车不一定有。如有的汽车体型很大。这时我们说轿车类继承了汽车类，它是汽车类的子类，而汽车类成为轿车类的父类或是超类。子类拥有父类或超类的一切属性，子类继承了父类，子类实际上是扩展、延伸了父类的基本属性。一个父类可以有许多子类，而每一个子类都是父类的特殊化，父类的属性是其所有子类属性的公共属性的集合。每一个子类又可以衍生出自己的子类，如此反复，我们可以仅仅依靠一个最基本的父类派生出许多的子类来，Java语言的类结构正是这样的。

使用继承的优点主要在于使得程序的结构更加清楚、降低代码的编制和维护工作。另一个需要注意的地方是在类的继承中有单重继承和多重继承的概念。单重继承是指任何一个类都只有一个单一的父类；而多重继承是指一个类可以有一个以上的父类，它的属性和操作从所有的父类中继承。在Java程序中为了简化程序设计，提高程序的安全性，不支持多重继承，仅支持单重继承。

综上所述，在面向对象的程序设计中，采用继承来组织和设计系统中的类，可以大大提高编制程序的效率，改善程序的结构，降低维护工作量。

1.3.3 多态

面向对象程序设计的封装特性使得多态在面向对象程序中的实现成为可能。所谓多态就是指一个程序中同名的不同方法共存的情况。当我们在编制面向对象的程序时，由于工作量巨大，单独依靠一个人或者是几个人是无法完成一个大型软件的编制工作的，而这许多人在编制过程中，很可能使用了相同的方法名，则在一个程序中可能出现多个重名的方法，如果不对这种情况作出规定，则在程序编译的过程中就会产生错误。如果我们能够允许重名方法存在，就可以大大提高程序的设计能力。这就是通过多态技术来实现的。面向对象程序中多态的情况有多种，可以通过子类对父类方法的覆盖实现，也可以利用重载在同一个类中定义多个同名的不同方法。

什么是覆盖呢？举个例子来说，汽车具有“刹车”这一操作，汽车的子类小轿车也具

有“刹车”这一操作，但在小轿车类中可以重新定义“刹车”这一操作。如果汽车的“刹车”操作为手动，则小轿车的“刹车”操作可以改为自动，这时子类（小轿车类）实际上是把父类（汽车类）的方法（操作）覆盖（重定义）了。这就是多态技术的一种形式。

在程序中如何区分这些重名的方法呢？这些方法虽然重名，但它们都分布在不同的类中，如果我们在调用重名的方法时，如果指明该方法所属的类名，就不会引起混淆了。

多态的另一种形式叫作重载，是同一类中定义同名方法的情况。这些方法同名的原因是由于其最终的目的和功能都是一样的，但在完成同一功能时，每个方法所遭遇的环境不同，其解决问题的办法也不一样，因此针对不同的环境就产生了同名但过程不同的方法。这就叫重载。比如在小轿车类中可以定义两个同名的方法“刹车”，一个是通过手动刹车，一个则通过自动刹车。

在同一类中又如何区分同名的方法呢？当然不能使用类名来区分重载的方法了，因为它们都属于同一类。一般而言，我们都根据同名的方法中参数的个数、类型以及顺序的不同来区分各种同名方法。在实际调用同名方法时，根据调用命令所指定的参数的个数、类型、顺序来启动相应的方法。

可以说，多态技术大大提高了程序的抽象性和简洁性，最大限度的降低了类和程序模块之间的相互耦合，使得模块之间的原始联系大大减少，具体细节都封装在每个类的内部，工作时通过参数来传递信息，从而为程序的设计、开发和维护提供了良好的基础。

1.4 Java 语法与语义

Java 语言中的语法与语义与其他语言没有很大的区别。介绍的目的主要是使初识 Java 的用户能够有一个全面的认识。

1.4.1 标识符、变量、常量

一、 标识符命名

标识符是对变量、类、标号和其他各种用户自定义对象的命名，在 Java 中，标识符必须以字母、下划线（_）或美元符（$）开头、后面跟 0 个或多个由字母、下划线、美元符或数字组成的字符串。由于 Java 使用了 unicode 字符集，因此字母不仅可以是英文字母还可以是其他语言中的文字如汉字等。

以下的几种字符组合都是 Java 的合法标识符：

idea、$(var)、_thread_type、变量 a

注意：

Java 语言中有不少关键字是保留字，因此用户自定义的标识符不能和这些关键字重名，否则会出现编译错误。同时为了增加程序的可读性，在 Java 中，变量名一般以小写字母开头，类名一般以大写字母开头。

二、 变量类型

变量是内存中一个用标识符命名的位置，其中存储的是能被程序修改的值，Java 语言