目　　录

第1章　绪论

本章先向读者介绍Java语言的基础知识，再介绍如何使用命令行输出程序，最后介绍Eclipse开发工具的使用。

1.1　Java语言简介

Java语言是一门面向对象的编程语言，它不仅吸收了C++语言的各种优点，还摒弃了C++语言中难以理解的多继承、指针等概念，所以Java语言具有功能强大和简单易用两个特征。Java语言作为静态面向对象编程语言的代表，极好地实现了面向对象理论，允许程序员以优雅的思维方式进行复杂的编程。

1.1.1　Java语言平台无关性

1. 什么是平台

Java语言是可以跨平台的编程语言，那什么是平台呢？无论使用哪种编程语言编写的应用程序，都需要经过操作系统和处理器来完成程序的运行，所以这里的平台由操作系统（Operating System，OS）和处理器（Central Processing Unit，CPU）构成。与平台无关是指因操作系统和处理器不同造成的编译程序不能运行或运行错误。

2. Java语言跨平台的原理

C语言编译执行过程如图1-1所示。

如果您有使用C语言的开发经历，看到图1.1将会非常轻松。我们知道，只要是用标准C开发的程序，使用不同的编译器编译后的可执行文件是可以在对应平台运行的，例如，

Windows可以使用Visual C++编译，编译后的．exe文件就可以在Windows下运行；Linux下可以使用GCC编译，生成的可执行文件就可以在Linux上运行。不同平台的编译软件不能相互使用。

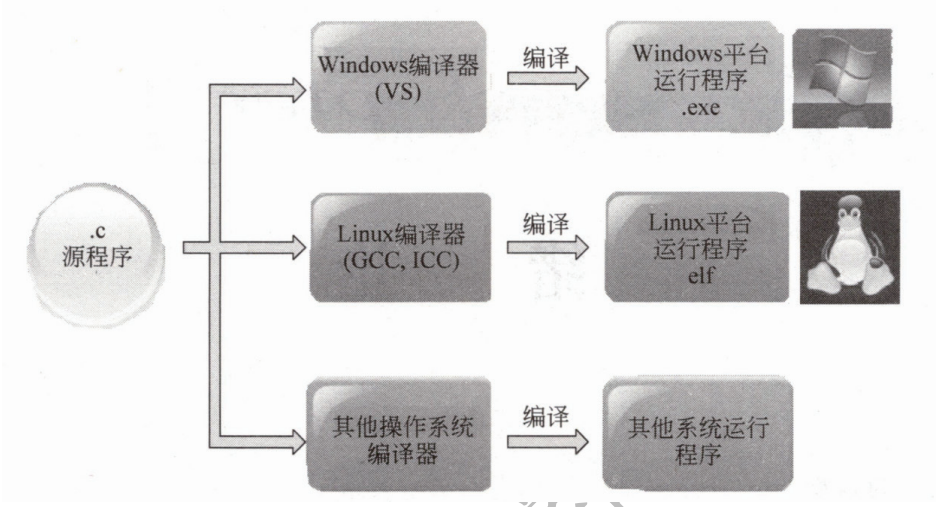

图1-1　C语言编译执行过程

Java程序实际上是在Java虚拟机（JRE是软件实现）中运行，Java虚拟机类似于一个模拟执行环境，在不同的操作系统上拥有不同的Java虚拟机实现，但是这些Java虚拟机遵循统一的规范来解释．class文件，并将．class文件中的指令转换为本地操作系统对应的指令，这样就实现了相同的．class文件。可以通过Java虚拟机转换为对应操作系统上的对应指令实现．class文件，也就是Java程序的跨平台性。Java语言代码执行过程如图1-2所示。

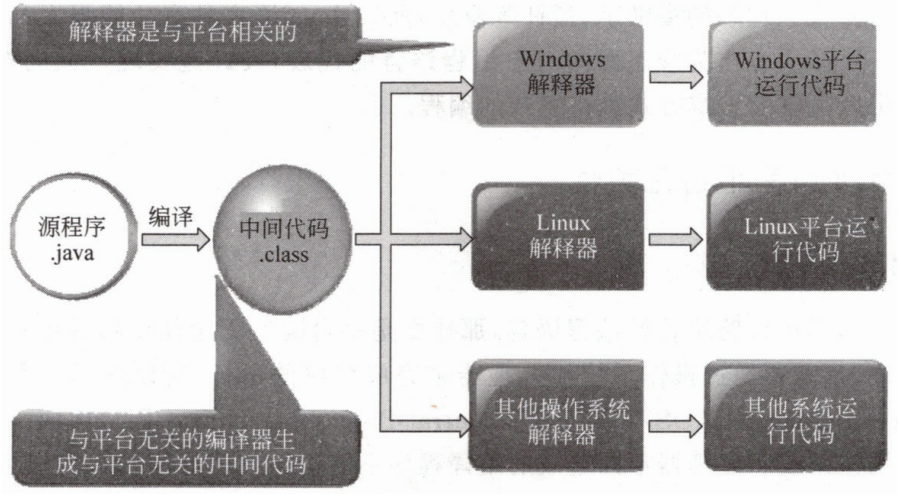

图1-2　Java语言代码执行过程

图1–2中的．java就是源程序，类似于C语言的．c，生成的中间代码是．class，通过Java虚拟机翻译成不同平台的机器执行代码。同一个．class文件在不同的虚拟机中会得到不同的机器指令（Windows和Linux的机器指令不同），但是最终的执行结果都是相同的。

第一种是编译执行，如上文中说到的C语言，它把源程序由特定平台的编译器一次性编译为平台相关的机器码，它的优点是执行速度快，缺点是无法跨平台；第二种是解释执行，如HTML、JavaScript，它使用特定的解释器，把代码一行行解释为机器码，类似于同声翻译，它的优点是可以跨平台，缺点是执行速度慢，暴露源程序；第三种是从Java语言开始引入的“中间代码+虚拟机”的方式，它既整合了编译语言与解释语言的优点，同时又如虚拟机那样可以解决如垃圾回收、安全性检查等这些传统语言中难以解决的问题，所以其后微软公司的．NET平台也使用了这种方式。

1.1.2　Java语言的发展历史

Java语言有二十几年的历史，已发展成为人类计算机史上影响深远的编程语言。Java语言所崇尚的开源、自由等精神，吸引了全世界无数优秀的程序员。事实上，从来没有一门编程语言能吸引这么多的程序员，也没有一门编程语言能衍生出如此多的开源框架。

Java语言是一门非常纯粹的面向对象的编程语言，它吸收了C++语言的各种优点，又摒弃了C++语言中难以理解的多继承、指针等概念，所以Java语言具有功能强大和简单易用两个特征。

从某种程度上来看，精通了Java语言的相关知识，相当于系统地学习了软件开发的相关知识，而不是仅仅学完了一门编程语言。

1. Java语言的由来

Java语言最开始只是Sun公司在1990年12月开始研究的一个内部项目。Sun公司的一位名叫帕特里克·诺顿的工程师被自己开发的C语言和C语言编译器搞得焦头烂额，因为其中的API极其难用。帕特里克决定改用NeXT，同时他也获得了研究公司的一个“Stealth计划”项目的机会。

“Stealth计划”后来改名为“Green计划”，詹姆斯·高斯林和麦克·舍林丹也加入了帕特里克的工作小组。他们和其他几个工程师一起在加利福尼亚州门罗帕克市沙丘路的一个小工作室里面研究开发新技术，瞄准下一代智能家电（如微波炉）的程序设计，Sun公司预料未来科技将在家用电器领域大显身手。团队最初考虑使用C语言，但是很多成员包括Sun公司的首席科学家比尔·乔伊，发现C语言和可用的API在某些方面存在很大问题。

工作小组使用的是内嵌类型平台，可以用的资源极其有限。很多成员发现C语言太复杂以至于经常被错误使用。他们还发现C语言缺少垃圾回收系统，以及可移植的安全性、分布程序设计和多线程功能。最后，他们想要一种易于移植到各种设备上的平台。

根据可用的资金，比尔·乔伊决定开发一种集C语言和Mesa语言搭成的新语言，在一份报告上，乔伊把它叫作“未来”，他提议Sun公司的工程师应该在C语言的基础上，开发一种面向对象的环境。最初，高斯林试图修改和扩展C语言的功能，他称这种新语言为C--，但是他放弃了。后来，他创造出一种全新的语言——Oak（橡树），以他的办公室外的树而命名。

就像很多开发新技术的秘密工程一样，工作小组没日没夜地工作，直到1992年的夏天，他们能够演示新平台的一部分了，包括Green操作系统、Oak的程序设计语言、类库和其硬件。最初的尝试是面向一种类PDA设备，被命名为Star7，这种设备有鲜艳的图形界面和被称为Duke的智能代理来帮助用户。1992年12月3日，这台设备进行了展示。

1992年11月，Sun公司的全资子公司——FirstPerson有限公司，被安排到了帕洛阿尔托。FirstPerson团队对建造一种高度互动的设备感兴趣，当时代华纳发布了一个关于电视机顶盒的征求提议书（Request for Proposal）时，FirstPerson改变了他们的目标，作为对征求意见书的响应，他们提出了一个机顶盒平台的提议。但是有线电视业界觉得FirstPerson的平台给予用户过多的控制权，所以FirstPerson的投标败给了SGI，与3DO公司的另外一项关于机顶盒的交易也没有成功。由于他们的平台不能在电视工业产生任何效益，因此该子公司被并回Sun公司。

1994年6～7月，约翰·盖吉、詹姆斯·高斯林、比尔·乔伊、帕特里克·诺顿、韦恩·罗斯因和埃里克·斯库米，在经历了一场历时三天的头脑风暴之后，团队的决定再一次改变了他们努力的目标，这次他们决定将该技术应用于万维网。他们认为随着Mosaic浏览器的到来，因特网正在向同样的高度、互动的远景演变，而这一远景正是他们在有线电视网中看到的。作为原型，帕特里克·诺顿写了一个小型万维网浏览器——WebRunner，后来改名为Hotjava。同年，Oak改名为Java。商标搜索显示，Oak已被一家显卡制造商注册，因此团队找到了一个新名字。这个名字是在很多成员常去的本地咖啡馆中杜撰出来的。名字是不是首字母缩写还不清楚，很大程度上来说不是。虽然有人声称是开发人员名字的组合：James Gosling（詹姆斯·高斯林）、Arthur Van Hoff（阿瑟·凡·霍夫）、Andy Bechtolsheim（安迪·贝克托克姆），或Just Another Vague Acronym（只是另外一个含糊的缩写）。还有一种比较可信的说法是这个名字是出于对咖啡的喜爱，所以以Java咖啡来命名。类文件的前四字节如果用十六进制阅读，分别为CA FEBA BE，就会拼出两个单词CAFE BABE（咖啡宝贝）。

1994年10月，HotJava和Java平台为公司高层进行演示。1994年，Java 1.0a版本已经可以提供下载，但是Java和Hotjava浏览器的第一次公开发布却是在1995年5月23日SunWorld大会上进行的。Sun公司的科学指导约翰·盖吉宣告Java技术的诞生。这个发布是与网景公司的执行副总裁马克·安德森的惊人发布一起进行的，其宣布网景将在其浏览器中包含对Java的支持。1996年1月，Sun公司成立了Java业务集团，专门开发Java技术。

1.1.3　Java语言的应用领域

如果你刚开始学习Java语言，你可能会思考Java语言会确切地运用在哪些地方。除了Minecraft（《我的世界》，一款沙盒游戏），你可能无法看到用Java语言编写的其他游戏吧？像Adobe Acrobat这样的桌面工具、Microsoft办公软件，这些都不是用Java语言编写的，甚至就连Linux或者Windows的操作系统也不是，那么人们到底在哪里使用了Java语言呢？Java语言到底有没有实际的应用程序呢？

1. 安卓App

如果你想知道Java语言应用在哪里，那么答案就会离你不远。打开你的安卓手机或者任何一款App，它们完全是用有谷歌Android API的Java编程语言编写的，这个API和JDK非常相似。前几年安卓刚开始起步，而如今已经有很多Java程序员从事安卓App的开发。顺便一提，虽然安卓使用了不同的Java虚拟机和不同的封装，但是代码仍然是用Java语言编写的。

2. 在金融服务行业的服务器应用

Java在金融服务业有着广泛的应用。很多全球性投资银行，如Goldman Sachs（高盛投资公司）、Citigroup（花旗集团）、Barclays（巴克莱银行）、Standard Charted（英国渣打银行）和一些其他银行，都用Java语言编写前台和后台的电子交易系统，结算、信息确认系统、数据处理项目和其他的项目。Java语言被运用于编写服务器端应用，但大多数没有前端，都是从一个服务器端（上一级）接收数据，处理数据后发向其他处理系统（下一级）。Java Swing由于能开发出图形用户界面的客户端供交易者使用而备受欢迎，但是现在C#语言正在快速地取代Swing的市场，这让Swing更有压力。

3. 网站应用

Java同样在电子商务和网站开发上有着广泛的应用。你可以运用很多RESTfull架构，这些架构是用Spring MVC、Struts 2.0和类似的框架开发出来的。甚至简单的Servlet、JSP和Struts在各种政府项目也备受欢迎，许多政府、医疗、保险、教育、国防和其他部门的网站都是建立在Java语言之上的。

4. 软件工具

很多有用的软件和开发工具都是运用Java语言编写和开发的，例如Ecilpse、InetelliJ Idea和Netbans IDE。这些都是经常使用的用Java语言编写的桌面应用程序，就如上面所说，Swing曾经在图形用户界面的客户端开发领域非常流行，它们大多数应用在金融服务领域以及投资银行。虽然现在Java Fx正在逐渐地流行起来，却仍然无法替代Swing，但C#语言已经在大部分金融领域中代替了Swing。

5. 交易系统

第三方交易系统是金融服务行业的一大部分，同样也是使用Java语言编写的。例如Murex这种受欢迎的交易系统，运用于与许多的银行前端连接，同样也是用Java语言编写的。

6. J2ME App

虽然iOS和Android的到来几乎扼杀了J2ME的市场，但是仍然有很多的低端手机在使用J2ME。曾经有段时间大部分的游戏和手机应用都是利用MIDP和CLDC，或者J2ME部分平台编写的，以适用于Android系统。J2ME依然在蓝光、磁卡、机顶盒等产品中流行着。App之所以如此流行是因为对于所有的低端手机，App仍然适用于J2ME。

7. 嵌入式领域

Java在嵌入式领域也有广泛的应用。你只需要130KB就能够使用Java技术（在一块小的芯片或者传感器上），这显示了这个平台的可靠性。Java最初是为了嵌入式设备而设计的。事实上，这也是Java最初的一项“编写一次，到处运行”主旨的一部分。

8. 大数据技术

Hadoop和其他的大数据技术也在不同程度上使用着Java，例如Apache的基于Java的Hbase、Accumulo（开源）以及ElasticSearch。但是Java并没有占领整个领域，还有其他大数据技术，例如MongoDB就是使用C++语言编写的。如果Hadoop和ElasticSearch逐渐发展，那么Java就能有潜力在大数据技术领域上得到更大的发展空间。

9. 高频交易领域

Java平台已经大大提高了其性能特点和JITS，并且Java也拥有C++级别的传输性能。所以，Java也流行于编写高并发系统。虽然Java的传输性能比不上C++，但用户可以不用考虑Java的安全性、可移植性和可维护性等问题（Java内部已经实现好了），而且Java有着更快的运行速度。安全性等问题会使一个没有经验的C++程序员编写的应用程序变得更加缓慢和不可靠。

10. 科学应用

现在Java经常是科学应用的默认选择，包括自然语言处理。这最主要的原因是Java比起C++或者其他语言有更高的安全性、可移植性、可维护性，而且Java有着更好的高级并发工具。

20世纪90年代，由于Applet的原因，Java在互联网中占据了重要地位，但是这几年，因为Applet的沙箱模型存在各种安全隐患，Applet越来越失去人气，导致如今桌面Java和Applet几乎都消失了，但是Java仍然作为软件行业默认的应用开发语言，且在金融服务业、投资银行和电子商务领域有着广泛的应用。任何学习Java的人都认为自己有着光明的未来。另外，Java 8加强了一个概念——在未来的几年，Java将继续统治着软件开发领域。

1.1.4　Java语言的地位

1. 网络地位

网络已经成为信息时代最重要的交互媒介，那么基于网络的软件设计就成为软件设计领域的核心。Java的平台无关性让Java成为编写网络应用程序的佼佼者．而且Java也提供了许多以网络应用为核心的技术，使得Java特别适用于网络应用软件的设计与开发。

2. 语言地位

Java语言面向对象编程，并涉及网络、多线程等重要的基础知识，是一门很好的面向对象语言。通过学习Java语言不仅可以掌握使用对象来完成某些任务及面向对象编程的基本思想，而且会为今后进一步学习设计模式奠定一个较好的语言基础。C语言无疑是非常基础和实用的语言之一。目前，Java语言已经获得了和C语言同样重要的地位，即其不仅是一门正在被广泛使用的编程语言，而且已经成为软件设计开发者应当掌握的一门基础语言。

3. 需求地位

目前，由于很多新技术领域都涉及Java语言，例如用于设计Web应用的JSP、设计手机应用的Android等，使得IT行业对Java人才的需求正在不断地增长。人们可以经常看到许多培训或招聘Java软件工程师的广告，所以，掌握Java语言及其相关技术意味着会拥有较好的就业前景和工作酬金。

1.2　Java语言的特性

Java语言是一种跨平台、适用于分布式计算环境的面向对象的编程语言。具体来说，它具有如下特性：简单性、面向对象、分布性、编译和解释性、稳健性、安全性、可移植性、高性能、多线程性和动态性等。

1. 简单性

Java语言的设计类似于C++语言，但是为了使语言小和容易熟悉，设计者们把C++语言中许多可用的特征去掉了，这些特征是一般程序员很少使用的。例如，Java不支持go to语句，代之提供break和continue语句以及异常处理。Java还剔除了C++的操作符过载（overload）和多继承特征，并且不使用主文件，免去了预处理程序。因为Java没有结构，

数组和串都是对象，所以不需要指针。Java能够自动处理对象的引用和间接引用，实现自动的“无用单元收集”，使用户不必为存储管理问题烦恼，能将更多的时间和精力花在研发上。

2. 面向对象

Java语言是一种面向对象的语言。对程序员来说，这意味着要注意其中的数据和操纵数据的方法（method），而不是严格地用过程来思考。在一个面向对象的系统中，类（class）是数据和操作数据的方法的集合。数据和方法一起描述对象（object）的状态和行为。每一对象是其状态和行为的封装。类是按一定体系和层次安排的，使得子类可以从超类继承行为。在这个类层次体系中有一个根类，它是具有一般行为的类。Java程序是用类来组织的。

Java还包括一个类的扩展集合，分别组成各种程序包（package），用户可以在自己的程序中使用。例如，Java提供产生图形用户接口部件的类（java. awt包），这里awt是抽象窗口工具集（abstract window toolkit）的缩写，处理输入输出的类（java. io包）和支持网络功能的类（java. net包）。

3. 分布性

Java语言支持在网络上应用，它是分布式语言。Java语言既支持各种层次的网络连接，又以Socket类支持可靠的流（stream）网络连接，所以用户可以产生分布式的客户机和服务器。

网络变成软件应用的分布运载工具，Java程序只要编写一次，就可到处运行。

4. 编译和解释性

Java编译程序生成字节码（byte-code），而不是通常的机器码。Java字节码提供对体系结构中性的目标文件格式，代码设计成可有效地传送程序到多个平台。Java程序可以在任何实现了Java解释程序和运行系统（run-time system）的系统上运行。

在一个解释性的环境中，程序开发的标准连接阶段大大消失了。如果说Java还有一个连接阶段，它只是把新类装进环境的过程，它是增量式的，轻量级的过程。所以，Java支持快速原型和容易试验，它将导致快速程序开发。这是一个与传统的、耗时的“编译、连接和测试”形成鲜明对比的精巧的开发过程。

5. 稳健性

Java语言原来是用作编写消费类家用电子产品软件的语言，所以它被设计成编写具有高可靠性和稳健性的软件。Java语言消除了某些编程错误，用它写可靠软件相当容易。

Java语言是一个强类型语言，它具有允许扩展编译时检查潜在类型不匹配问题的功能。Java语言要求显式的方法声明，它不支持C语言风格的隐式声明。这些严格的要求保证编译程序能捕捉调用错误，确保程序的可靠性。

可靠性方面最重要的增强之一是Java的存储模型。Java不支持指针，它消除重写存储

和讹误数据的可能性。类似地，Java自动的“无用单元收集”预防存储泄漏及其他有关动态存储分配和解除分配的有害错误。Java解释程序也执行许多运行时的检查，诸如验证所有数组和串访问是否在界限之内。

异常处理是Java中使得程序更稳健的另一个特征。异常是某种类似于错误的异常条件出现的信号。使用try-catch-finally语句，程序员可以找到出错的处理代码，这就简化了出错处理和恢复的任务。

6. 安全性

Java的存储分配模型是它防御恶意代码的主要方法之一。Java没有指针，所以程序员不能得到隐蔽起来的内幕和伪造指针去指向存储器。更重要的是，Java编译程序不处理存储安排决策，所以程序员不能通过查看声明去猜测类的实际存储安排。编译的Java代码中的存储引用在运行时由Java解释程序决定实际存储地址。

Java运行系统使用字节码验证过程来保证装载到网络上的代码不违背任何Java语言限制。这个安全机制部分包括类如何从网上装载。例如，装载的类是放在分开的名字空间而不是局部类，预防恶意的小应用程序用它自己的版本来代替标准Java类。

7. 可移植性

Java语言使得语言声明不依赖于实现的方面。例如，Java显式说明每个基本数据类型的大小和它的运算行为（这些数据类型由Java语法描述）。

Java环境本身对新的硬件平台和操作系统是可移植的。Java编译程序也用Java编写，而Java运行系统用ANSI C语言编写。

8. 高性能

Java语言是一种先编译后解释的语言，所以它不如全编译性语言快。但是有些情况下性能是很重要的，为了支持这些情况，Java设计者制作了“及时”编译程序，它能在运行时把Java字节码翻译成特定CPU（中央处理器）的机器代码，也就是实现了全编译。

Java语言字节码格式在设计时考虑这些“及时”编译程序的需要，所以生成机器代码的过程相当简单，它能产生相当好的代码。

9. 多线程性

Java语言是多线程语言，它提供支持多线程的执行（也称为轻便过程），能处理不同任务，使具有线程的程序设计很容易。Java的lang包提供一个Thread类，它支持开始线程、运行线程、停止线程和检查线程状态的方法。

Java的线程支持也包括一组同步原语。这些原语是基于监督程序和条件变量风范，由C.A.R. Haore开发的广泛使用的同步化方案。利用关键词synchronized，程序员可以说明某些方法在一个类中不能并发地运行。这些方法在监督程序控制之下，确保变量维持在一个一致的状态。

10. 动态性

Java语言适应于变化的环境，它是一个动态的语言。例如，Java语言中的类是根据需要载入的，甚至有些是通过网络获取的。

1.3 Java编程规范

建立一个可行可操作的编程标准、约定和指南，以规范人们的代码开发工作，提高代码的可读性，提高系统的健壮性、稳定性、可靠性。通过遵循这些程序设计标准，作为一个Java软件开发者的生产效率会有显著提高。经验表明，若从一开始就花时间编写高质量的代码，则在软件开发阶段，对代码的修改要容易很多。最后，遵循一套通用的程序设计标准将带来更大的一致性，使软件开发团队的效率明显提高。

1.3.1 包的命名与注释

1. 包的命名

包的命名都是由小写的英文字母组成，每个包名称之间用点号分隔开来，如java.lang和javax. servlet. http等。

全局包的名字用所在机构的Internet保留域名开头，如com. sun和com. oracle等。

我们的命名约定：

com. excellence. common：excellence下的通用文件，可能被以后不同项目通用的类文件。

com. excellence. exoa4j：exoa2004forjava项目的Java分模块分类存放。

在com. excellence. exoa4j下面，我们按照模块分开各自的包。

2. 注释包

应保证有一个或者多个外部文档以说明相应机构所创建的包的用途。对于每个包，应说明如下内容。

（1）包的基本原理。其他开发者需要了解一个包是用来做什么的，这样他们才能判断是否可以用它；如果是一个共享包，他们可以判断是否需要改进或是扩展它。

（2）包中的类。在包中要包含一个类和接口的列表，每个类或接口用简短的一行文字来说明，以便让其他的开发者了解这个包中含有什么。

技巧：生成一个以包名命名的HTML文件，将它放到包的适当的目录中去。这个文件应具有扩展名．html。

1.3.2　类、接口的命名及注释

1. 命名类

标准Java类约定使用完全的英文描述符，所有单词的第一个字母要大写，并且单词中大小写混合。

类名应是单数形式。

示例：

Customer Employee　Order　OrderItem　FileStream　String

相关的类如果使用了某些已经约定的开发模式，应该使用相关约定的命名，如LoggerFactory、UserDAO等。

在同一个模块中的相关类，可以采用同样开头的名称以区分它们的相关性，如UserValue、UserDAO等。

2. 命名接口

采用完整的英文描述符说明接口封装，所有单词的第一个字母大写。习惯上，名字后面加上后缀able、ible或者er。

示例：

Runnable　Contactable　Compressible　Prompter

3. 命名编译单元

编译单元在这种情况下是一个源码文件，应被赋予文件内定义的主要的类或接口的名字。用与包或类相同的名字命名文件，大小写也应相同。.java应作为文件名的扩展名。

示例：

Customer.java

Singleton.java

DeptInfo.java

4. 类及接口的注释

以下信息应写在文档注释中紧靠类的定义的前面。

（1）类的目的和作用。开发者需要了解一个类的一般目的，以判断这个类是否满足他们的需求。开发者应养成一个注释与类有关的任何事项的习惯。

（2）已知的问题。如果一个类有任何突出的问题，应说明出来，以让其他开发者了解这个类的缺点／难点。此外，还应注明为什么不解决该问题的原因。

（3）类的开发／维护历史。通常要包含一个历史记录表，列出日期、类的作者和修

改概要。这样做的目的是让进行维护的程序员了解过去曾对一个类所做的修改，是谁做了什么样的修改。

（4）版权信息。

对于以上几点，其中类的目的和作用是必须要填写的。采用javadoc形式标志的一个示例如下：

```
/**
* <p>类说明：LoggerFactory，创建Logger的工厂类</p>
*  < p> Copyright  2003:  Excellence Network Co., LTD< /p>
* @ author fjy
* @ version1.0
* @ since 2017-06-10
* @modified fjy 2017-06-13修改了××××
* @modified fjy 2017-06-23修改了××××
*/
```

1.3.3 成员函数的命名及注释

1. 命名成员函数

成员函数的命名应采用完整的英文描述符，大小写混合使用，所有中间单词的第一个字母大写。成员函数名称的第一个单词常常采用一个具有强烈动作色彩的动词。

示例：

```
openAccount ( )     printMailingLabel ( )     createUser ( )     delete ( )
```

这种约定常常使人一看到成员函数的名称就能判断它的功能。虽然这种约定要使开发者多做一些输入工作，而且函数名常常较长，但是可以提高代码的可理解性。

2. 获取函数

获取函数作为一个成员函数，返回一个字段的值。除了布尔字段之外，还应采用get作为字段的前缀，布尔字段采用is作为前缀。

示例：

```
public String getUserCode ( )  {
   return userCode;
)
public String getUserName ( )  {
   return userName;
)
getFirstName ( )
```

```
getAccountNumber ( )
isPersistent ( )
isAtEnd ( )
```

遵循这个命名约定，显然，成员函数将返回对象的字段，布尔型的获取函数将返回布尔值“真”或者“假”。这个标准的另一个优点是它遵循Beans Development Kit（BDK）对获取成员函数采用的命名约定（DES97）。它的一个主要的缺点是get是多余的，需要额外地录入。

按照JBuilder中bean / properties工具生成的设置和获取函数。

3. 设置函数

设置函数也称为变值函数，是可以修改一个字段值的成员函数。无论何种字段类型，都要在字段名的前面加上set前缀。

示例：

```
public void setUserCode （String userCode） {
   this.userCode=userCode;
}
public void setUserName （String userName） {
   this .userName=userName;
}
setFirstName （String name）
setAccountNumber （int accountNumber）
setReasonableGoals （Vector goals）
setPe rsis tent （boolean is Persis tent）
setAtEnd （boolean isAtEnd）
```

按照这种命名约定，显然是一个成员函数设定一个对象的字段值。这个标准的另一个优点是：它遵循Beans Development Kit（BDK）对设置函数采用的命名约定（DES97）。它的一个主要的缺点是set是多余的，需要额外地录入。

按照JBuilder中bean / properties工具生成的设置和获取函数。

4. 命名构造函数

构造函数是在一个对象初次生成时，完成所有必需的初始化的成员函数。构造函数与它所属类的名字总是相同的。例如，类Customer的构造函数是Customer（ ）。注意大小写一致。

示例：

```
Customer ( )
SavingsAccount ( )
```

PersistenceBroker（ ）

这个命名约定由Sun公司设定，必须严格遵守。

5. 成员函数的可见性

良好的程序设计应该尽可能地减小类与类之间耦合，所遵循的经验法则是：尽量限制成员函数的可见性。如果成员函数没必要公共（public），就定义为保护（protected）；没必要保护，就定义为专用（private），如表1.1所示。

表1–1　成员函数的可见性

可见性	Java关键字	说　明	正确用法
公共	public	公有成员函数可被任何其他对象和类的成员函数调用	当该成员函数必须被该函数所在的层次结构之外的其他对象和类访问时
受保护	protected	被保护的成员函数可被它所在的类或该类的子类的任何成员函数调用	当该成员函数提供的行为被它所在类的层次结构内部而非外部需要时
专用	private	私有成员函数只可以被该类所在的其他成员函数调用，该类的子类不可以调用	当该成员函数所提供的行为明确针对定义它的类时。私有成员函数常常是重新分配要素的结果。重新分配要素又称为“重组”，指类内其他成员函数封装某一个特定行为的做法
缺省	无关键字，简单地使其为空白	成员函数对于同一包中的其他所有类实际上都是公共的，但是对该包外部的类是专用的。有时，它称为包可见性或友好的可见性	这是一个有趣的功能，但要小心使用。一般不建议使用

6. 成员函数的参数

成员函数的参数在采用javadoc@param标识的头文件中注释。应说明：

（1）参数用来做什么。需要注释出参数用来做什么，以便其他开发者了解使用参数的上下文。

（2）任何约束或前提条件。如果一个参数的值域不能被成员函数接收，则应让调用者知道。可能一个成员函数只接收正数，或者字符个数小于5的字符串。

（3）示例。如果应传递的参数不明显，那么应该在注释中给出一个或多个例子。

1.3.4　字段、属性的命名及注释

field这个词在这里指的是字段，Beans Development Kit （BDK）称为“属性”（DES97）。字段是说明一个对象或者一个类的一段数据。字段可以是像字符串或者浮点数这样的基本数据类型，也可以是一个对象，例如一个消费者或者一个银行账户。

1. 命名字段

应采用完整的英文描述符来命名字段（GOS96，AMB98），以便使字段所表达的意思一目了然。像数组或者矢量这样是集合的字段，命名时应使用复数来表示它们代表多值。

示例：

firstName　　zipCode unitPrice　　discountRate　　orderItems（复数）

2. 命名组件（部件）

应采用完整的英文描述符命名组件（接口部件），名字的扩展名是组件类型名。这让用户容易区分一个组件的目的和它的类型，容易在一个表里找到各个组件（许多可视化的编程环境在一个Applet程序或者一个应用程序中提供了一个所有组件的列表。如果所有名字都是类似于button1、button2或&，则容易混淆）。

示例：

okButton　　customerList　　fileMenu　　newFileMenuItem

3. 字段都应很好地加以注释，以便其他开发者理解它

要想有效地注释，以下部分需要说明。

（1）字段的说明。需说明一个字段，才能使人了解如何使用它。

（2）注释出所有采用的不变量。字段中的不变量是指永远为“真”的条件。例如，字段dayOfMonth的不变量可能是它的值只能在1～31（显然，可以用基于某一年里的某个月份来限制这个字段值，使其变得更加复杂）。通过说明字段值的限制条件，有助于定义重要的业务规则，使代码更易理解。

示例：

```
// 终执行的SQL语句
protected String  strSQL=null;
// 需要连接数据源的名称
private String  dsName;
```

1.3.5　局部变量命名及注释

一般说来，命名局部变量遵循与命名字段一样的约定，即使用完整的英文描述符，任何非开头的单词的第一个字母都要大写。

但是为了方便起见，对于如下几个特殊的局部变量类型，这个约定可以放宽。

（1）流。

（2）循环计数器。

（3）异常。

1. 命名流

当有一个单输入和/或单输出流在一个成员函数中被打开、使用和关闭时，通常的约定是对这些流分别采用in和out来命名（GOS96）。对于既用于输入又用于输出的流，采用inOut来命名。

一个常用的取代这种约定的方法是分别采用inputStream、outputStream和ioStream这样的名字，而不是in、out和inOut，虽然这与Sun公司的建议相抵触。

2. 命名循环计数器

因为局部变量常用作循环计数器，并且它为C/C++所接受，所以在Java编程中，可以采用i、j或k作为循环计数器（GOS96）。若采用这些名字作为循环计数器，要始终使用它们。

概括起来说，i、j、k作为计数器时，它们可以很快地被输入，也可以被广泛地接受。

3. 命名异常对象

因为在Java代码中异常处理也非常普遍，所以字母e作为一般的异常符被广泛地接受。

如果有多个异常同时出现，应该使用该类型异常全称的缩写表示。例如：

catch（SQLException sqlexception）

4. 声明和注释局部变量

在Java中声明和注释局部变量有以下几种约定。

（1）一行代码只声明一个局部变量。这与一行代码应只有一个语句相一致，并使得对每个变量采用一个行内注释成为可能。

（2）用一个行内注释语句说明局部变量。行内注释是一种紧接在同一行的命令代码后，用符号//标注出来的单行注释风格（也称“行末注释”）。应注释出一个局部变量用于做什么、在哪里适用、为什么要用等，使代码易读。

例如：

```
int count=1;                    //记录循环次数
int pageSize=5;                 //页面的大小
```


1.4 搭建 Java 开发环境

对 Java 有了一个基本了解后，现在就来学习一下如何进行 Java 程序开发。Java 开发环境的搭建相对其他语言可能有些复杂，因为 Java 本身提供了很多机制，从而方便学习。Java的开发环境可以用 JDK 来代表，在本节中就来看一下如何下载、安装和配置 JDK。

1.4.1　下载 JDK

JDK 是 Sun 公司提供的一种免费的 Java 软件开发工具包，里面包含了很多用于 Java 程序开发的工具，最常用的是编译和运行工具。现在就先来看一下该工具包的下载， JDK 的下载可以通过 Sun 官方网站或者通过搜索引擎下载。

1.4.2　安装

JDK下载 JDK 后，双击下载的 EXE 文件，即可开始安装 JDK。首先是弹出许可证协议窗口，其中给出了 Sun 公司的一些开发协议，单击其中的“接受”按钮，就会弹出如图 1–3 所示的“自定义安装”窗口。

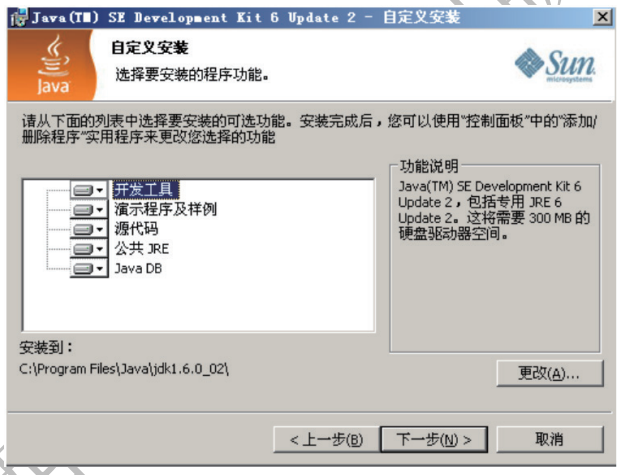

图1–3 “自定义安装”窗口

在窗口中可以选择要安装的 Java 组件和 JDK 文件的安装路径。这里可以采用默认安装Java 的所有组件并在 C 盘安装。在后面的配置中，也将按照默认安装进行配置。单击“下一步”按钮后就开始安装 JDK，稍后，单击窗口中的“完成”按钮就正式完成了 JDK 的安装。

1.4.3　配置 JDK

下载和安装 JDK 后，只是完成 Java 开发环境搭建的前半部分，最关键的部分还是配置JDK。配置 JDK 的目的是能够在命令提示符中运行 JDK 中的命令，例如编译和运行。配置JDK 的操作步骤如下。

（1）在“我的电脑”桌面图标上，单击鼠标右键，在弹出菜单中选择“属性”→“高级”→“环境变量”命令，弹出“环境变量”窗口，在该窗口中就可以进行环境变量的设置，如图 1–4 所示。

（2）单击“系统变量”选项组中的“新建”按钮，弹出如图 1–5 所示的“新建系统变量”窗口。

图1–4　环境变量设置

图1–5　新建系统变量

（3）在“变量名”文本框中输入“PATH”，在“变量值”文本框中输入“C:\Program Files\Java\jdk1.6.0_02\bin;”，注意要以分号结尾。

（4）重复步骤（3）的操作，在“变量名”文本框中输入“CLASSPATH”，在“变量值”文本框中输入“C:\Program Files\Java\jdk1.6.0_02\lib\tools.jar;”。单击“确定”按钮。

（5）配置 JDK 完成之后，这时候就可以测试一下是否配置正确。选择“开始”→“运行”命令，弹出“运行”命令框，如图 1–6 所示。

（6）在“运行”命令框中输入“cmd”，进入命令提示符界面。在该界面中输入javac命令，如果出现如图 1-7 所示的结果，则表示 JDK 配置成功；如果提示错误，则表示配置失败，就需要重新配置，查找哪一步发生了错误。

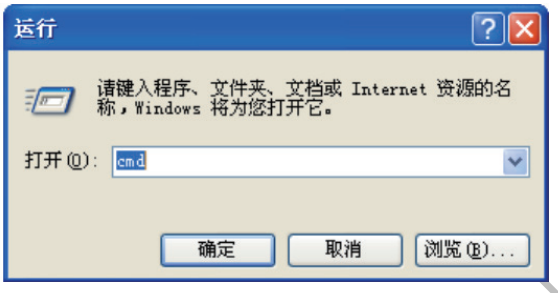

图1-6　“运行”命令框

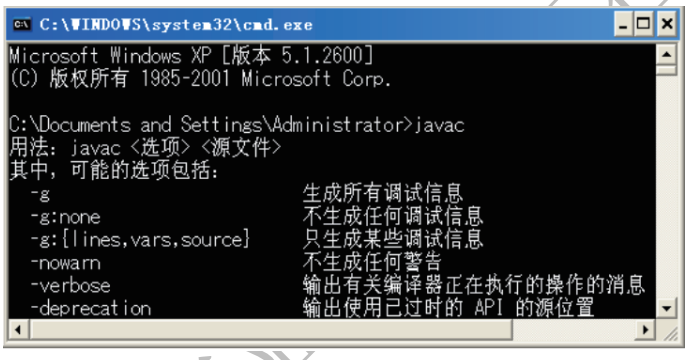

图1-7　测试

1.5　开发 Hello World 程序

搭建 Java 的开发环境后，现在是否已经迫不及待地想开发一个 Java 程序了？在本节中就来开发一个非常简单的输出“Hello World”内容的程序。通过该程序来演示 Java 程序的编写、编译和运行，从而了解 Java 程序的开发过程。

1.5.1　编写 Java 程序

开发 Java 程序，首先要编写一个 Java 程序。在 F 盘下，新建一个文本文档，将初始的“新建文本文档.txt”名称重命名为“Hello World.java”名称。在有些计算机中，默认是

没有扩展名的，所以要首先将扩展名设置出来。选择菜单栏中的“工具”→“文件夹选项”→“查看”命令，如图 1-8所示。

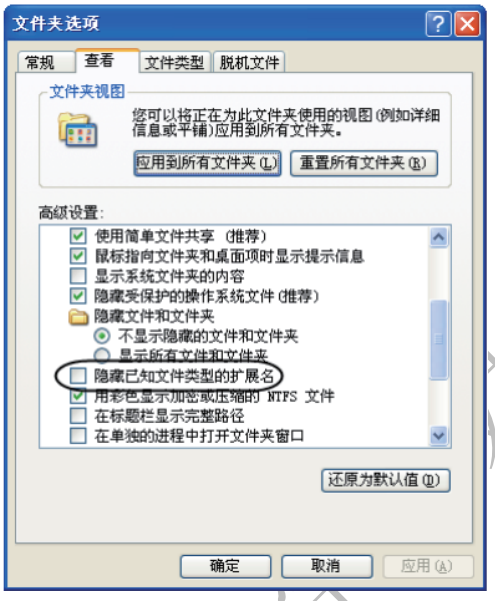

图1-8　扩展名设置

在图1-8中取消“隐藏已知文件类型的扩展名”复选框的勾选，单击“确定”按钮，这样系统中的所有文件就都具有扩展名了。

现在已经有了一个名称为“Hello World.java”的文本文档，使用系统自带的记事本程序将其打开，在其中输入如下代码。

```
public class Hello World
{
   public static void main ( String args [  ] )
   {
          System.out.println ( "Hello World" ) ;
   }
}
```

输入代码后，不要忘记保存。

1.5.2　编译 Java 程序

编写并保存 Java 程序后， 选择“开始”→“运行”命令， 在“运行”命令框中输入“cmd”，弹出命令提示符界面。在该界面中的操作方式如图 1-9 所示。

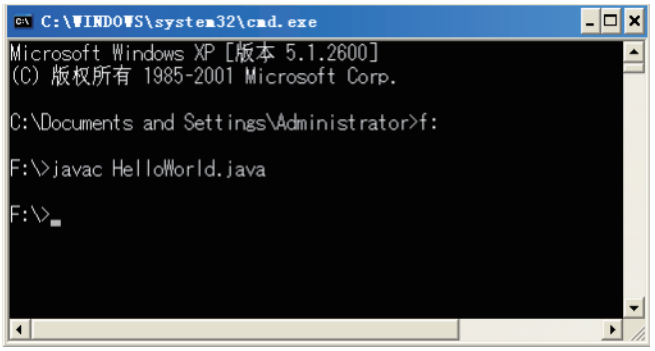

图1-9　编译 Java 程序

在该界面中，首先输入“f：”命令，这样就切换到 F 盘下，这是因为上一节开发的程序保存在 F 盘中。如果读者编写的程序不在 F 盘中，这里就输入所编写程序所在的位置。

进入 F 盘后，输入“javac Hello World.java”命令，其中 javac 是 JDK 中的编译命令，而Hello World.java 是上一节中编写的 Java 程序的文件名。执行“java Hello World.java”命令后，会在 F 盘下产生一个名称为 Hello World.class 的文件，它是执行编译命令所产生的文件。

注意：在 javac 命令后输入的文件名中一定要有.java 扩展名，否则会发生错误。

1.5.3　运行 Java 程序

编译 Java 程序后，产生一个以.class 为扩展名的文件，运行 Java 程序就是运行该文件。在图 1-9 所示界面的命令输入下继续输入“java Hello World”命令，如图 1-10 所示。

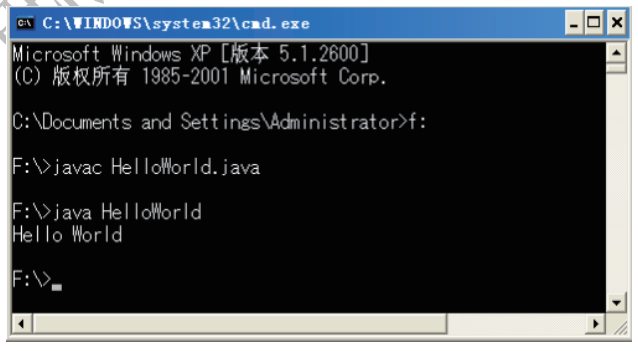

图1-10　运行 Java 程序

从运行结果中可以看到输出了“Hello World”信息，这就是开发的该程序的功能。运行Java 程序是通过 java 命令来完成的。

注意：在 java 命令后输入的文件名没有扩展名，如果有，则会发生错误。

1.5.4 简单讲解一下 Hello World 程序

在以上三个小节中，对开发 Java 程序的流程进行了讲解，在本节中将简单地讲解一下前面开发的 Hello World 程序，让读者对 Java 程序先有一个初步了解。

Hello World 程序中的第一行的内容是“public class Hello World”，其中“Hello World”是一个类名，“class”是判断“Hello World”为一个类名的关键字，而“public”是用来修饰类的修饰符。每一个基础类都有一个类体，使用大括号包括起来。

程序中的第三行为“public static void main（String args［ ］）”，它是一个特殊方法，主体是“main”，其他的都是修饰内容。这条代码语句是一个 Java 类固定的内容，其中 main 定义一个 Java 程序的入口。和类具有类体，方法具有方法体一样，其同样也要使用大括号括起来。程序的第五行为“System.out.println（"Hello World"）；”，该语句的功能是向输出台输出内容。在该程序中输入的是“Hello World”信息，从而才有了图 1-10 的运行结果。

1.6 使用集成开发工具 Eclipse 开发

在前面讲解了使用记事本来开发 Java 程序，因为要调用命令提示符界面，所以显得有些麻烦。而 Java 的一些集成开发工具解决了这一问题。目前 Java 的集成开发工具有很多，这里采用开发中最常用 Eclipse 来进行讲解。

1.6.1 下载和安装 Eclipse

到作者成书时，Eclipse 集成开发工具的最新版本为 3.4。但是作为一本 Java 语言学习的入门书，新版本的新功能是使用不到的，而 3.2 版本的工具都是可用的。由于读者都是初学者，所以作者决定采用中文版的 Eclipse 来进行讲解，而 3.2 以上版本都没有专门提供 Eclipse中文包，所以这里采用 Eclipse 3.2 版本进行讲解。

Eclipse 可以通过 Eclipse 的官方网站“http：//www.eclipse.org/”来进行下载，也可以通过搜索引擎进行下载，获取资源的渠道是很多的。Eclipse 是绿色软件，直接解压就可以使用，解压完成也就完成了安装。通常将解压后的 Eclipse 文件直接复制到某一盘下，例如复制到 D盘下，Eclipse 的文档结构图如图 1-11 所示。

图1–11　Eclipse 文档结构图

1.6.2　下载和安装 Eclipse 中文包

Eclipse中文包可以通过Eclipse的官方网站“http：//www.eclipse.org/”来进行下载，也可以通过搜索引擎进行下载。Eclipse中文包同样是一个压缩文件。解压缩Eclipse中文包，将plugins目录下的所有文件和文件夹复制解压到D：\eclipse\plugins 目录，然后将features目录下的所有文件和文件夹复制解压到D：\eclipse\features 目录。运行D：\eclipse\eclipse.exe即可启动中文版的Eclipse。

1.6.3　启动 Eclipse

下载和安装中文版 Eclipse 后，就可以启动中文版 Eclipse。在 Eclipse 文件下，有一个eclipse.exe 文件，双击该文件，就可以启动 Eclipse。第一次启动 Eclipse 时，首先会出现如图1–12 所示的窗口。

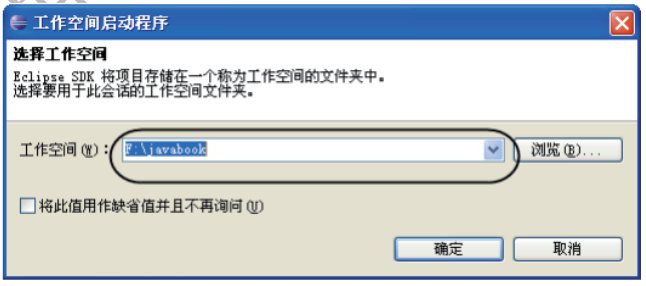

图1–12　Eclipse程序路径

读者第一次启动Eclipse的时候显示的窗口和图1–12不同，窗口中“工作空间”文本框显示的通常是C盘下的位置，这里是经过修改后的，读者也可以进行修改来确定通过Eclipse开发的项目和程序保存的位置。单击“确定”按钮后，弹出如图1–13所示的Eclipse欢迎窗口。

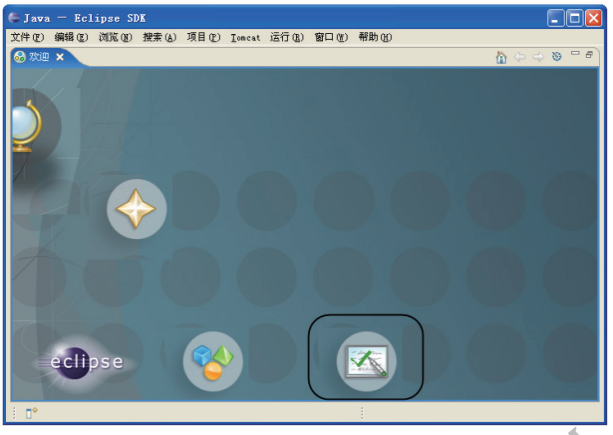

图1-13　Eclipse欢迎窗口

有些读者往往忽略该欢迎窗口，其实该窗口的功能很多，例如图1-13选中的就是Eclipse的教程，读者可以通过该教程了解很多Eclipse的知识。本书主要讲解Java的相关内容，一些Eclipse方面的知识，读者可通过该教程进行学习。关闭欢迎窗口后，就会弹出真正用于开发的窗口，如图1-14所示。

如图1-14所示的窗口基本分为5部分，最上面的是菜单栏，其中包括了Eclipse的所有开发工具。左边是项目结构区，会在其中显示一个项目的结构。中间是编码区，在其中就可以编写Java的程序，和记事本很相似。右边是大纲区，其中显示一个程序的结构。最下面是提示区，最常见到的是在下面输出结果和提示错误。

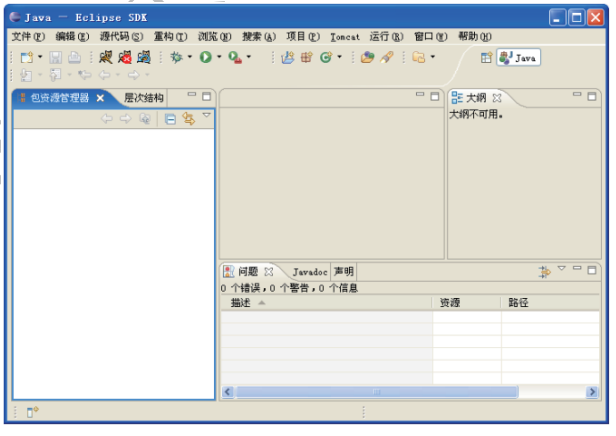

图1-14　Eclipse开发窗口

1.6.4　使用Eclipse开发Java程序

搭建Eclipse开发环境并对Eclipse有一个基本了解后，就可以使用Eclipse集成开发工具来开发Eclipse中的Hello World程序了。在这里开发的Hello World程序看不出比使用记事

本开发有什么优越的地方，但是当开发大型程序时，使用Eclipse集成开发工具就要比直接使用记事本容易得多。现在就来看一下使用Eclipse集成开发工具开发Hello World程序的步骤。

（1）选择菜单栏中“文件”→“新建”→“项目”命令，弹出如图1-15所示的“新建项目”窗口。

（2）选择“Java项目”选项，单击“下一步”按钮，弹出如图1-16所示的“新建Java项目”窗口。

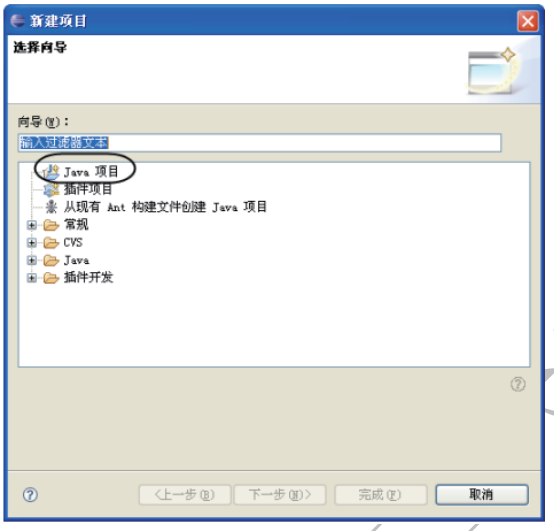

图1-15　“新建项目”窗口

图1-16　“新建Java项目”窗口

（3）在“新建Java项目”窗口的“项目名”文本框中输入自己要创建的项目名。由于这里是本书的第一章，就设置创建的项目名为“chap1”，单击“完成”按钮，这样就创建了一个名称为“chap1”的Java项目。此时就会在Eclipse开发窗口的项目结构区显示该项目，单击前面的加号，就会显示出该项目的结构，如图1-17所示。

从项目结构图中可以看到，在Eclipse中自动导入JDK的类包，从而可以运行Java程序。

（4）在“chap1”项目上右击，选择“新建”→“类”命令，弹出如图1-18所示的“新建Java类”窗口。

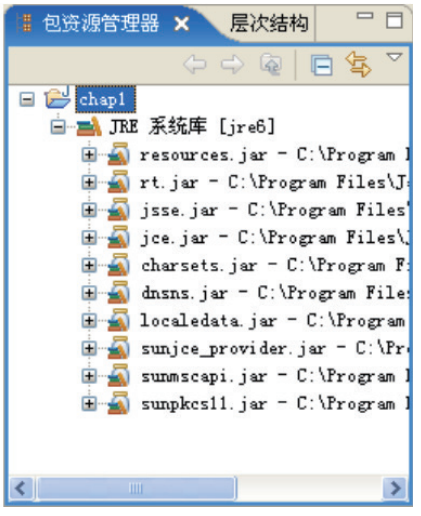

图1-17　项目结构

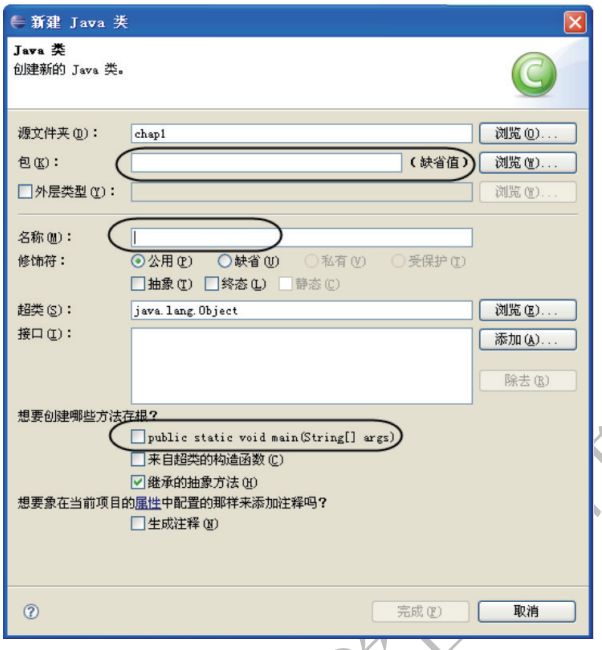

图1-18 “新建Java类”窗口

在“新建Java类”窗口中有很多需要填写的选项。首先是填写包，包的概念会在后面进行讲解，如果这里不填，则采用默认值，也就是不使用包。下面需要填写的就是Java类的名称，在名称文本框输入“Hello World”。最后在“想要创建哪些方法存根”中勾选第一个复选框，也就是main方法，因为它是一个类的入口。设置好这些选项后，单击“完成”按钮，将会在编码区出现如下代码。

```
public class Hello World {
    /**
    * @param args
    */
    public static void main ( String [  ] args ) {
            //TODO自动生成方法存根
    }
}
```

将该代码和前面的代码相比较，会发现在Eclipse中自动生成了大部分代码。

注意：由于Eclipse自动生成一些注释和空格，为了少占篇幅和方便学习，在后面的代码中会将这些东西去掉，然后加上一些更易懂的注释。如果发现自己开发的代码和书中的不太一样，也不要奇怪。

（5）创建程序的基本框架后，就可以添加功能程序代码。这个程序的功能是输出

“Hello World”信息，添加功能语句后的代码如下。

```
public class Hello World {
    public static void main ( String [  ]  args ) {
            System.out.println ( "Hello World" ) ;  //功能语句
    }
}
```

（6）完成Java程序的编写后，就可以编译和运行该Java程序。在Eclipse集成开发工具中，编译和运行是一体的，不需要分别执行。选择菜单栏中的“运行”→“运行方式”→“Java运行程序”命令，将弹出如图1–19所示的“保存并启动”窗口。

选择要运行的Hello World.java程序，单击“确定”按钮，即可运行Java程序。

说明：并不是每一次运行Java程序都是这么复杂的。第一次这样运行后，如果后面代码被修改需要再次运行，可以直接选择“运行”→“运行上次启动”命令来运行该程序。也可以通过单击Eclipse工具栏中的按钮运行程序。

运行Java程序后，在提示区出现如图1–20所示的运行结果。

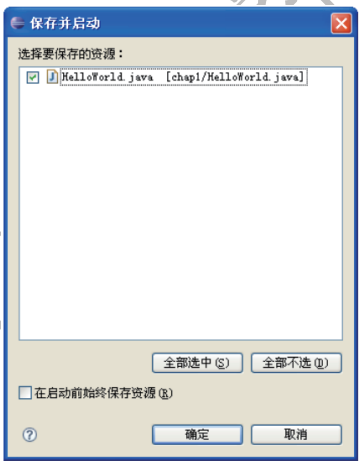

图1–19　“保存并启动”窗口

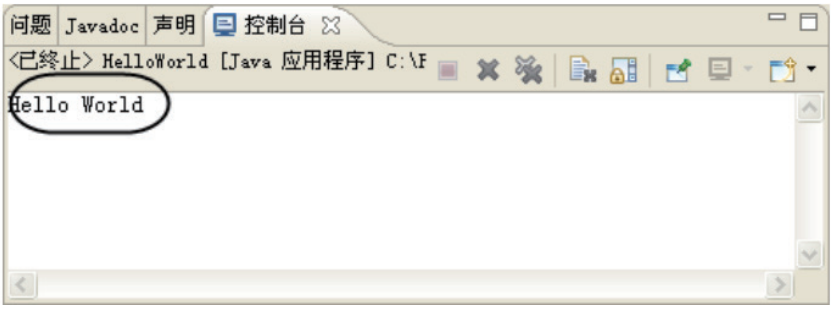

图1–20　运行结果

从运行结果中可以看到输出了“Hello World”信息，从而完成了该程序的开发。如果没有出现如图1-18所示的运行结果，读者就需要认真查一下什么地方出了问题，或者重新开发。

1.7 小　结

在本章中初步介绍了Java程序开发的相关知识和过程。首先简单地讲解了读者最关心的几个问题，并没有过多地讲解Java发展和起源等内容。接着讲解了Java开发环境的搭建，以及如何使用该开发环境进行Java程序开发。最后讲解了Eclipse这一集成开发工具的基本功能及如何在集成工具中进行程序开发。

习　题

编写程序，在命令行使用*输出一个菱形，具体如下：

```
    *
  *   *
*       *
  *   *
    *
```

第2章　Java语言编程基础

本章主要介绍Java语言的基本数据类型和控制结构。假设你有其他的语言经验，例如熟悉变量、循环、数组和流程控制等概念，也许你会发现它们之间有类似的地方，但同时还需要注意它们之间的不同。本章会加速你的Java语言学习进程。

2.1　基本数据类型及转换

基本数据类型也称为简单数据类型。Java语言有8种基本数据类型，分别是boolean、byte、short、char、int、long、float、double，这8种基本数据类型习惯上可分为以下几类。

- 整数类型：byte、short、int、long。
- 浮点类型：float、double。
- 逻辑类型：boolean。
- 字符类型：char。

同时这几个类型之间的相互转换也在本章中学习。

2.1.1　整数类型

整数类型分为4种。

1. int型

常量：123（十进制），077（八进制），0X3ABC（十六进制）。

变量：使用关键字int来声明int型变量，声明时也可以赋给初值，例如：

```
int x=12, y=9898, z;
```

对于int型变量，分配4个字节的内存，占32位。最大数据存储容量是$2^{32}-1$，数据范围是$-2^{31} \sim 2^{31}-1$。

2. long型

常量：long型常量用L来表示，例如108L（十进制），07123L（八进制），0x3ABCL（十六进制）。

变量：使用关键字long来声明long型变量，例如：

```
long width=12L, height=2005L, length;
```

对于long型变量，分配8个字节的内存，占64位。最大数据存储容量是$2^{64}-1$，数据范围为$-2^{63} \sim 2^{63}-1$。

3. byte型

常量：Java语言中不存在byte型常量的表示方法，但是可以把一定范围内的int型常量赋值给byte型变量。

变量：使用关键字byte来声明byte型变量，例如：

```
byte x=-12, tom=28, 漂亮=98;
```

对于byte型变量，分配1B（字节）的内存，占8b（位）。最大存储数据量是255，存放的数据范围是-128 ~ 127。

4. short型

常量：和byte型类似，Java语言中也不存在short型常量的表示方法，但是可以把一定范围内的int型常量赋值给short型变量。

变量：使用关键字short来声明short型变量，例如：

```
short x=12, y=1234;
```

对于short型变量，分配2B的内存，占16b。最大数据存储量是65 536，数据范围是-32 768 ~ 32 767。

绝大多数情况下，int类型最实用。如果你想用数字代表地球上的总人口数，则需要使用long类型，如果遇到long类型还不够用的情况，则使用BigInteger类。byte和short类型主要用在特殊应用中，例如当存储空间非常珍贵时的底层文件处理，或大数组处理。

在Java中，整型的范围不依赖于程序所运行的机器。毕竟，Java是按照“编写一次，到处运行”的原则设计的语言。相比之下，C和C++程序依赖编译程序的处理器。

注意：Java没有无符号的byte、short、int和long，这一点和C语言有很大的不同。所以，“unsigned int m;”是错误的变量声明。

2.1.2 字符类型

字符型描述了Java使用了UTF-16字符编码中的“编码单元”。

常量：‘A’，‘b’，‘！’，‘好’等，即用单引号（需用英文输入法输入）括起来的Unicode表中的一个字符。

变量：使用关键字char来声明char型变量，例如：

```
char ch='A', home='家', handsome='酷';
```

对于char型变量，分配2B的内存，占16b。最高位不是符号位，没有负数的char。char型变量的取值范围是0～65 535。对于

```
char x='a';
```

内存中存储的是97，97是字符a在Unicode表中的排序位置。所以，允许将上面的变量声明写成

```
char x=97;
```

要观察一个字符在Unicode表中的顺序位置，可以使用int类型转换，如“（int）'A'”。如果要得到一个0～65 535之间的数所代表的Unicode表中相应位置上的字符，必须使用char型类型转换，如“（char）65”。

注意：Java中的char型数据一定是无符号的，而且不允许使用unsigned来修饰所声明的char型变量（这一点和C语言是不同的）。

2.1.3　浮点类型

浮点类型是指有分数的数字。两种浮点类型是：float（单精度）和double（双精度）。

1. float型

常量：453.5439f，21379.987F，231.0f（小数点表示法），2e40f（2×10^{40}，指数表示法）。需要特别注意的是常量后面必须加后缀f或F。

变量：使用关键字float来声明float型变量，例如：

```
float x=22.76f, tom=1234.987f, weight=1e-12F;
```

float变量在存储float型数据时保留8位有效数字（相对double型保留的有效数字，称之为单精度）。对于float型变量，分配4B的内存，占32b。数据范围在3.4e-45～1.4e38，直接赋值时必须在数字后加上f或F。

2. double型

常量：453.5439d，21379.987，0.05（小数点表示法），1e-90（1×10-90，指数表示法）。

变量：使用关键字double来声明double型变量，例如：

```
double height=23.345, width=34.56D, length=1e12;
```

对于double型变量，分配8B的内存，占64b。数据范围在4.9e-324～1.8e308，赋值时可以加d或D，也可以不加。double变量在存储double型数据时保留16位有效数字（相对float型

保留的有效数字，称之为双精度）。

需要特别注意的是，比较float和double型数据必须注意数据的实际精度，例如，对于

```
float x=0.4f;
double y=0.4f;
```

那么实际存储在变量x中的数据是（这里我们将小数点保留16位）：0.400 000 005 960 464 5，存储在y变量中的数据是（小数点保留16位）：0. 400 000 000 000 000 000，所以，y中的值小于x中的值。

许多年前，内存还是稀缺资源，最常使用4个字节的浮点型。但是7位小数位数表示的精度有限。现在，默认使用“double精度”的数字。只有当用户需要存储比较大的浮点数时，使用float才有意义。

注意：浮点数不适合做金融运算，因为金融计算中的舍入误差是不被接受的。如果需要精确而无舍入误差的数字运算，则可以使用BigDecimal类。

2.1.4 逻辑类型

常量：true，false。用来判定逻辑条件。

变量：使用关键字boolean来声明逻辑变量，声明时也可以赋给初值，例如：

```
boolean male=true,  on=true,  off=false,  isTriangle;
```

在Java中，布尔类型不是数字类型。布尔值与整数0和1没有关系。

2.1.5 类型转换运算

当把一种基本的数据类型变量的值赋给另一种基本类型运算时，就涉及数据类型转换。例如，一方面int类型的值将会自动地转换为double类型。但另一方面，有时也需要将double类型的值转换成int类型。在Java运算中，允许进行这种数值之间的类型转换。当然，这有可能会造成数据丢失，此时就需要强制转换。将这些类型按精度从低到高排列：byte、short、char、int、long、float、double。

当把级别低的变量的值赋给级别高的变量时，系统会自动完成转换。例如：

```
float x=100;
```

如果输出x的值，结果将是100.0。

例2.1

```
int x=50;
float y;
y=x;
```

如果输出y的值，结果将是50.0。

当把级别高的变量的值赋给级别低的变量时，必须使用类型强制转换，格式如下：

（类型名）要转换的值；

例2.2

```
int x=（int）34.89;
long y=（long）56.98F;
int z=（int）1999L;
```

如果输出x、y、z的值，结果将是34、56和1 999，类型转换运算结果的精度可能低于原数据的精度。

注意：如果试图将一个数值从一种类型强制转换为另一种类型，而又超出了目标类型的表示范围，结果就会截断成一个完全不同的值。例如："（byte）300"的实际值为44。

不要在boolean类型与任何数据类型之间进行强制类型转换，这样可以防止发生错误。只有极少数的情况才需要将布尔类型转换为数值类型，这时可以使用条件表达式"b?1：0"。

2.2　变量与常量

在程序中存在大量的数据来代表程序的状态，其中有些数据在程序的运行过程中值会发生改变，而有些数据在程序运行过程中值不能发生改变，这些数据在程序中分别被称作变量和常量。

2.2.1　变量

为了方便引用变量的值，在程序中需要为变量设定一个名称，这就是变量名。例如在2D游戏程序中，需要代表人物的位置，则需要2个变量，一个是x坐标，一个是y坐标，在程序运行的过程中，这两个变量的值会发生改变。

由于Java语言是一种强类型的语言，所以变量在使用之前必须要先声明，在程序中声明变量的语法格式如下：

```
数据类型　变量名称;
```

例如：

```
int x;
```

在该语法格式中，数据类型可以是Java语言中任意的类型，包括前面介绍的基本数据类型以及后续将要介绍的复合数据类型。变量名称是该变量的标识符，需要符合标识符的命名规则，在实际使用中，该名称一般和变量的用途对应，这样便于程序的阅读。数据类型和变量名称之间使用空格进行间隔，空格的个数不限，但是至少需要一个。语句使用“；”作为结束。

也可以在声明变量的同时，设定该变量的值，语法格式如下：

```
数据类型  变量名称=值;
```

例如：

```
int x=10;
```

在该语法格式中，前面的语法和上面介绍的内容一致，后续的“=”代表赋值，其中的“值”代表具体的数据，注意“==”代表为判断是否相等。在该语法格式中，要求值的类型需要和声明变量的数据类型一致。

在程序中，变量的值代表程序的状态，在程序中可以通过变量名称来引用变量中存储的值，也可以为变量重新赋值。例如：

```
int n=5;
n=10;
```

在实际开发过程中，需要声明什么类型的变量，需要声明多少个变量，需要为变量赋什么数值，都根据程序逻辑决定，这里列举的只是表达的格式。

2.2.2 常量

常量代表程序运行过程中不能改变的值。

常量在程序运行的过程中主要有两个作用。

（1）代表常数，便于程序的修改（例如：圆周率的值）。

（2）增强程序的可读性（例如：常量UP、DOWN、LEFT和RIGHT分别代表上、下、左、右，其数值分别是1、2、3和4）。

常量的语法格式和变量类型，只需要在变量的语法格式前面添加关键字final即可。在Java编码规范中，要求常量名必须大写。

常量的语法格式如下：

```
final数据类型 常量名称=值;
final数据类型 常量名称1=值1，常量名称2=值2，…，常量名称n=值n;
```

例如：

```
final  double  PI=3.14;
final char MALE= 'M', FEMALE= 'F';
```

在Java语法中，常量也可以首先声明，然后再进行赋值，但是只能赋值一次，示例代码如下：

```
final int UP;
UP=1;
```

2.3　运算符和字符串

Java提供了丰富的运算符，如算术运算符、关系运算符、逻辑运算符、位运算符等。并且，Java语言中绝大多数的运算符和C语言类似。此外，从概念上讲，Java字符串就是Unicode字符序列。Java没有内置的字符串类型，而是在标准Java类库中提供了一个预定义类String。

2.3.1　运算符

1. 算术运算符

在Java中，使用算术运算符+、-、*、/表示加、减、乘、除运算。当参与/运算的两个操作数都是整数时，表示整数除法；否则，表示浮点除法。整数的求余（有时候称为取模）操作用%表示。

加减运算符是二目运算符，它的操作元是整型或者浮点型数据，加减运算符的优先级是4级；乘，除和求余运算的操作元是整型或浮点型数据，优先级是3级。

2. 自增、自减运算符

自增、自减运算符++，一一是单目运算符，可以放在操作元之前，也可以放在操作元之后。作用是使变量的值增1或减1，如：++x（--x）表示在使用x之前，先使x的值增（减）1；x++（x--）表示在使用x之后，使x的值增（减）1。粗略地看，++x和x++的作用相当于x=x+1。但两者的不同之处在于++x是先执行x=x+1再使用x的值，x++是先使用x的值再执行x=x+1。

3. 关系运算符

关系运算符是二目运算符，用来比较两个值之间的关系。关系运算符的结果是布尔型，当运算符对应的关系成立时，运算结果是true，否则是false。因为算术运算符的级别高于关系运算符，故10>20-17相当于10>（20-17），其结果是true。

4. 逻辑运算符

逻辑运算符包括&&、||、！。其中&&、||为二目运算符。实现逻辑与、逻辑或；！为单目运算符，实现逻辑非。逻辑运算符的操作元必须是布尔型数据，真值表如表2-1所示。

表2-1　逻辑运算符的真值表

A	B	A&&B	A||B	!A	A^B	A&B	A|B
false	false	false	false	true	false	false	false
true	false	false	true	false	true	false	true
false	true	false	true	true	true	false	true
true	true	true	true	false	false	true	true

例如，2>8&&9>2的结果为false，2>8||9>2的结果为true。由于关系运算符的级别高于&&、||的级别，故2>8&&9>2相当于（2>8）&&（9>2）。

逻辑运算符&&和||也称为短路逻辑运算符，这是因为当A为false时，&&运算符在进行运算时不再去计算B的值，直接就得出A&&B的结果是false；当A的值为true时，||运算符在进行运算时不再去计算B的值，直接就得出A||B的结果是true。

5. 位运算符

位运算是以二进制位为单位进行的运算，其操作数和运算结果都是整型值，主要包括如下几类：位与&、位或|、位非～、位异或^、右移>>、左移<<、0填充的右移>>>。

位运算的位与&、位或|、位非～、位异或^与逻辑运算的相应操作的真值表完全相同，其区别是位运算操作的操作数和运算结果都是二进制整数，而逻辑运算相应操作的操作数和运算结果都是逻辑值boolean型，位运算符示例如表2.2所示。

表2-2　位运算示例

运算符	名　称	示　例	说　明
&	位与	x&y	把x和y按位求与
|	位或	x|y	把x和y按位求或
～	位非	～x	把x按位求非
^	位异或	x^y	把x和y按位求异或
>>	右移	x>>y	把x的各位右移y位
<<	左移	x<<y	把x的各位左移y位
>>>	0填充的右移	x>>>y	把x的各位右移y位，左边填0

6. instanceof运算符

该运算符是二目运算符，左面的操作元是一个对象，右面是一个类。当左面的对象是右面的类或子类创建的对象时，该运算符的结果是true，否则是false。

7. 运算符综述

Java表达式就是用运算符连接起来的符合Java规则的式子。运算符的优先级决定了表达式中运算执行的先后顺序。例如，x<y&&!z相当于（x<y）&&（!z）。没有必要去记忆运算符的优先级别，在编写程序时尽量使用括号“（）”运算符号来实现想要的运算次序，以免产生难以阅读或者含糊不清的计算顺序。运算符的结合性决定了并列的相同级别运算符的先后顺序，例如，加减的结合性是从左到右，8-5+3相当于（8-5）+3；逻辑否运算符“！”的结合性是从右向左，！！x相当于！（!x）。表2.3是Java所有运算符的优先级和结合性，有些运算符和C语言类同，不再赘述。

表2-3　运算符优先级表

优先级	运算符	结合性
1	（）［］.	从左向右
2	1+（正）-（负）~++--	从右向左
3	* / ? %	从左向右
4	+（加）-（减）	从左向右
5	<<>>>>>	从左向右
6	<<=>>=instanceof	从左向右
7	==!=	从左向右
8	&（按位与）	从左向右
9	^	从左向右
10	|	从左向右
11	&&	从左向右
12	||	从左向右
13	?:	从右向左
14	=+=-=*= / =%=&=|=^= ~ =<<=>>=>>>=	从右向左

2.3.2　字符串

字符串就是一系列字符。在Java中，字符串可以包含任意的Unicode字符。例如，字符串“JavaTM”或“Java\u2122”是由字符J、a、v、a和TM组成的。最后一个字符是“u+2122”商标。每个用双引号括起来的字符串都是String类的一个实例。

```
String e=""; //一个值为空的字符串
String greeting= "Hello";
```

1. 字符串连接

使用操作符+可以将两个字符串连接起来。例如：

```
String location= "Java";
```

```
String greeting="Hello"+location;
```

设置greeting的内容是Hello Java（注意，第一个操作数末尾有空格）。

当将一个字符串和另外一个值连接起来时，该值转换为字符串。

```
int age=42;
String output=age+"years"
```

要将几个以分隔符隔开的字符连接起来，使用join方法。例如：

```
String names=String.join（ ¨ ， ¨ ，"Peter"，"Paul"，"Mary"）；
//将names设置为"Peter、Paul、Mary"
```

第一个参数是字符串分隔符，后面紧跟的字符是想要连接的那些字符串。可以包含任意多个字符串，也可以由用户自己提供字符串数组。

如果只是想要最终的结果，那么连接大量的字符就会引起效率低下。这种情况下，最好使用StringBuilder代替。

```
StringBuilder builder=new StringBuilder（ ）；
While （more strings）{
   Builder.append（next string）；
}
String result=builder. toString（ ）；
```

2. 子字符串

要将字符串拆开，可以使用subString方法。例如：

```
String greeting="Hello，World";
String location= greeting. subString（7，12）；        //设置location为"World"
```

subString方法的第一个参数是要提取的子字符串的起始位置。字符串的位置从零开始。

subString方法的第二个参数是子字符串中不应该包含的字符的起始位置。在上面的例子中，字符串greeting的位置12上的字符是“！”，也就是我们不想包含的字符。指定不想要的字符的位置，可能看起来比较奇怪，但是这样是有好处的，12与7之差就是子字符串的长度。

有时，可能需要将分隔符分隔的字符串中所有的子字符串提取出来。split方法就是执行这种任务的，用于返回一个子字符串数组。

```
String names= ¨Peter，  Paul，  Mary";
String［ ］result=names.split（ ¨ ， ¨ ）；
                    //结果为由［"Peter"，"Paul"，"Mary"］三个字符串组成的数组
```

分隔符可以是正则表达式。例如，“input.split（"＼＼s+"）”以空格将input分隔。

1）字符串比较

熟悉C++的人对于两个字符串比较的代码一定很了解：

（string1==string2）

但在Java中，这个代码即使在两个字符串完全相同的情况下也会返回false。

Java中提供了两种比较方法：

• 利用“string1.equals（string2）”来进行判断；

• 利用==来进行比较。

如果

```
string s1=new String （"Hello" ）;
string s2=new String （"Hello" ）;
```

则

```
（s1==s2）=false;
```

如果

```
string s1= "Hello";
string s2= "Hello";
```

则

```
（s1==s2）=true;
```

因为它们指向的是同一个对象。

如果把其他变量的值赋给s1和s2，即使内容相同，由于不是指向同一个对象，也会返回false。

对应上面示例，第一种情况：

（1）声明一个String类型的对象str1，这个操作会在内存中实例化一个String类型的对象。

（2）再声明一个String类型的对象str2，这个操作会在内存中实例化另一个String类型的对象，与第一个对象相互独立。

（3）比较两个对象的引用，因为它们是相互独立的，所以输出false。

（4）比较两个对象的值，它们虽然是独立的内存区域但是有共同的内存结构（值），因此输出true。

第二种情况：

（1）声明一个字符串str1会实例化一个String类型的对象。

（2）声明另一个字符串，现在会检查在同一作用域中是否有相同的内存结构的变量，如果有就直接将地址指向它。

（3）一一比较它们是否有相同的地址。

（4）equals比较它们是否有相同的内存结构（值）。

3. 数字与字符串转换

将整数转换为字符串，可调用静态的Integer.toString方法：

```
int n=42;
String str=Integer.toString （n）；      //设置str为字符串"42";
```

该方法的变种有第二个参数，可以指定基数，范围为2～36。

```
str=Integer.toString （n，2）；          //设置str为字符串"101010"
```

相反地，将包含整数的字符串转换为数字，使用Integer.parseInt方法：

```
n=Integer.parseInt（str）；              //设置n为101010
```

可以指定基数：

```
n=Integer.parseInt（str，42）；          //设置n为42
```

对于浮点数，使用Double. toString和Double. parseDouble方法：

```
String str=Double.toString（3.14）；     //设置str为字符串"3.14"
double x=Double.parseDouble（"3.14"）；  //设置x为3.14
```

2.4 表达式和语句

Java除了在基本数据类型上和C语言有相似之处以外，Java中的绝大多数表达式甚至基本语句也与C语言类似，所以，本节就主要的知识点进行简单的介绍。

2.4.1 表达式

用算术运算符和括号连接起来的符合Java语法规则的式子，称为算术表达式。例如：

```
x+2*y-30+3*（y+5）
```

结果为数值型的变量或表达式可以通过关系运算符形成关系表达式。例如：

```
4>8，（x+y）>80
```

结果为布尔型的变量或表达式可以通过逻辑运算符来形成逻辑表达式（关系表达式）。

2.4.2 语句

Java中的语句可以分为以下6类。