



目录



第 1 章 Java 概述 / 1

1.1 Java 语言概述	2	1.2.3 测试开发环境	8
1.1.1 Java 语言的历史	2	1.3 编写一个 Java 程序	8
1.1.2 Java 语言的现状	2	1.3.1 编写源代码	8
1.1.3 Java 语言的面向对象特性	3	1.3.2 编译源代码	9
1.1.4 Java 与其他编程语言之间的关系	4	1.3.3 运行 class 文件	9
1.2 搭建 Java 环境	5	本章小结	9
1.2.1 JDK 的下载和安装	5	实战训练	9
1.2.2 配置 JDK	6		



第 2 章 Eclipse 开发工具 / 11

2.1 熟悉 Eclipse	12	2.2.3 运行 Java 程序	27
2.1.1 Eclipse 简介	12	2.3 程序调试	27
2.1.2 下载安装 Eclipse	12	2.3.1 设置断点	27
2.1.3 为 Eclipse 安装中文语言包	15	2.3.2 以调试方式运行 Java 程序	27
2.1.4 Eclipse 工作台	17	2.3.3 程序调试	28
2.2 使用 Eclipse	21	本章小结	28
2.2.1 创建 Java 项目	21	实战训练	28
2.2.2 使用编辑器编写程序代码	26		



第 3 章 Java 语言基础 / 29

3.1 Java 程序的基本结构	30	3.3.2 浮点类型	34
3.2 标识符和关键字	31	3.3.3 字符类型	35
3.2.1 标识符	31	3.3.4 布尔类型	37
3.2.2 关键字	32	3.4 变量与常量	37
3.3 基本数据类型	33	3.4.1 声明变量	37
3.3.1 整数类型	33	3.4.2 声明常量	38

3.4.3 变量的有效范围	39	3.6 类型转换	48
3.5 运算符	40	3.6.1 自动类型转换	48
3.5.1 赋值运算符	40	3.6.2 强制类型转换	49
3.5.2 算术运算符	41	3.7 代码注释和编码规范	50
3.5.3 自增和自减运算符	42	3.7.1 代码注释	50
3.5.4 比较运算符	42	3.7.2 编码规范	51
3.5.5 逻辑运算符	43	3.8 经典范例	52
3.5.6 位运算符	44	3.8.1 判断某一年是否是闰年	52
3.5.7 三元运算符	46	3.8.2 求球形的体积	52
3.5.8 运算符优先级	47	本章小结	53
3.5.9 范例 1: 不用其他变量实现两变量互换	47	实战训练	53
3.5.10 范例 2: 判断数字的奇偶性	48		



第 4 章 流程控制 / 55

4.1 复合语句	56	4.4.1 break 语句	69
4.2 条件语句	57	4.4.2 continue 语句	70
4.2.1 if 条件语句	57	4.4.3 return 语句	71
4.2.2 switch 多分支语句	60	4.4.4 范例 5: 终止循环体	71
4.2.3 范例 1: 验证登录信息的合法性	62	4.4.5 范例 6: 循环体的过滤器	72
4.2.4 范例 2: 为新员工分配部门	63	4.5 经典范例	73
4.3 循环语句	64	4.5.1 经典范例 1: 使用 for 循环输出空心的 菱形	73
4.3.1 while 循环语句	64	4.5.2 经典范例 2: 使用 for 循环输出杨辉 三角	74
4.3.2 do...while 循环语句	65	本章小结	75
4.3.3 for 循环语句	66	实战训练	75
4.3.4 范例 3: 使用 while 循环遍历数组	68		
4.3.5 范例 4: 使用 for 循环输出九九乘法表	68		
4.4 跳转语句	69		



第 5 章 数组 / 77

5.1 数组概述	78	5.3.4 范例 4: 求方阵的迹	84
5.2 一维数组	78	5.4 数组的基本操作	85
5.2.1 创建一维数组	78	5.4.1 遍历数组	85
5.2.2 初始化一维数组	79	5.4.2 填充(替换)数组元素	87
5.2.3 范例 1: 求一维数组各元素的和	80	5.4.3 对数组进行排序	88
5.2.4 范例 2: 获取一维数组的最小值	80	5.4.4 复制数组	89
5.3 多维数组	81	5.4.5 范例 5: 对比一维数组、二维数组所占 内存	91
5.3.1 二维数组	81	5.4.6 范例 6: 使用直接插入排序法排序	91
5.3.2 三维数组	83	5.5 经典范例	92
5.3.3 范例 3: 对矩阵进行转置运算	84		

5.5.1 经典范例 1: 使用冒泡排序法排序	92	本章小结	94
5.5.2 经典范例 2: 输出九宫格	93	实战训练	94



第 6 章 字符串 / 95

6.1 创建字符串	96	6.4 正则表达式	115
6.2 字符串操作	96	6.4.1 判断是否符合正则表达式的方法	115
6.2.1 字符串连接	96	6.4.2 正则表达式的元字符	115
6.2.2 获取字符串信息	97	6.4.3 正则表达式的限定符	116
6.2.3 去除字符串中的空格	99	6.4.4 方括号中元字符的含义	117
6.2.4 字符串替换	101	6.4.5 范例 5: 验证 IP 地址的合法性	118
6.2.5 判断字符串	102	6.4.6 范例 6: 计算文章中汉字的个数	119
6.2.6 字母大小写转换	104	6.5 字符串生成器	120
6.2.7 字符串分隔	105	6.5.1 StringBuilder 类	120
6.2.8 范例 1: 根据指定分隔符把字符串分行	106	6.5.2 StringBuilder 类的常用方法	121
6.2.9 范例 2: 判断字符串是否是数字格式	107	6.5.3 范例 7: 将汉字和字母转换成 Unicode 码	123
6.3 格式化字符串	107	6.5.4 范例 8: 去掉字符串中重复的字符	124
6.3.1 格式化方法	107	6.6 经典范例	124
6.3.2 日期格式化	108	6.6.1 经典范例 1: 实现字符串加密	124
6.3.3 时间格式化	109	6.6.2 经典范例 2: 验证字符串是否回文	125
6.3.4 日期时间组合格式化	110	本章小结	126
6.3.5 常规类型格式化	111	实战训练	126
6.3.6 范例 3: 将金额格式化成大写	112		
6.3.7 范例 4: 将数字格式化成货币格式	114		



第 7 章 类与对象 / 127

7.1 面向对象概述	128	7.2.9 this 关键字	139
7.1.1 对象	128	7.2.10 范例 1: 自定义图书类	140
7.1.2 类	129	7.2.11 范例 2: 温度单位转换工具	140
7.1.3 面向对象的特点	130	7.3 对象的应用	141
7.2 类	132	7.3.1 对象的创建	141
7.2.1 类的创建方法	132	7.3.2 访问对象的属性和行为	142
7.2.2 类的方法	133	7.3.3 对象的引用	145
7.2.3 成员变量	134	7.3.4 对象的比较	145
7.2.4 成员方法	134	7.3.5 对象的销毁	146
7.2.5 局部变量	135	7.3.6 范例 3: 统计图书数量	147
7.2.6 局部变量的有效范围	135	7.3.7 范例 4: 重新计算对象的哈希码	147
7.2.7 静态变量、常量和方法	136	7.4 经典范例	149
7.2.8 权限修饰符	138	7.4.1 经典范例 1: 汉诺塔问题求解	149

7.4.2 经典范例 2: 单例模式的应用	150	实战训练	151
本章小结	151		



第 8 章 继承与多态 / 153

8.1 Java 中的继承	154	8.2.2 引用类型检查	161
8.1.1 调用父类构造方法	156	8.2.3 引用类型转换	163
8.1.2 成员变量隐藏和方法覆盖	157	8.2.4 final 关键字	165
8.1.3 方法的覆盖	158	本章小结	166
8.2 多态	159	实战训练	166
8.2.1 多态概念	159		



第 9 章 抽象类与接口 / 169

9.1 Java 抽象类	170	9.2.3 接口与多继承	173
9.1.1 抽象类概念	170	9.2.4 接口继承	175
9.1.2 抽象类声明和实现	170	9.2.5 没有静态方法	176
9.2 使用接口	172	9.3 抽象类与接口的区别	178
9.2.1 接口概念	172	本章小结	178
9.2.2 接口声明和实现	172	实战训练	178



第 10 章 异常处理 / 179

10.1 Java 异常	180	10.2.5 多重捕获	189
10.1.1 Java 异常处理机制	181	10.3 释放资源	189
10.1.2 异常类继承层次	182	10.3.1 finally 代码块	190
10.1.3 Throwable 类	182	10.3.2 自动资源管理	192
10.1.4 Error 和 Exception	183	10.3.3 throws 与声明方法抛出异常	193
10.1.5 受检查异常和运行时异常	184	10.3.4 自定义异常类	195
10.2 捕获异常	184	10.3.5 throw 与显式抛出异常	195
10.2.1 try-catch 语句	185	10.3.6 throw Throwable 或其子类的实例	195
10.2.2 处理 ParseException	186	本章小结	196
10.2.3 多个 catch 代码块	186	实战训练	197
10.2.4 try-catch 语句嵌套	188		

参考文献	198
------------	-----

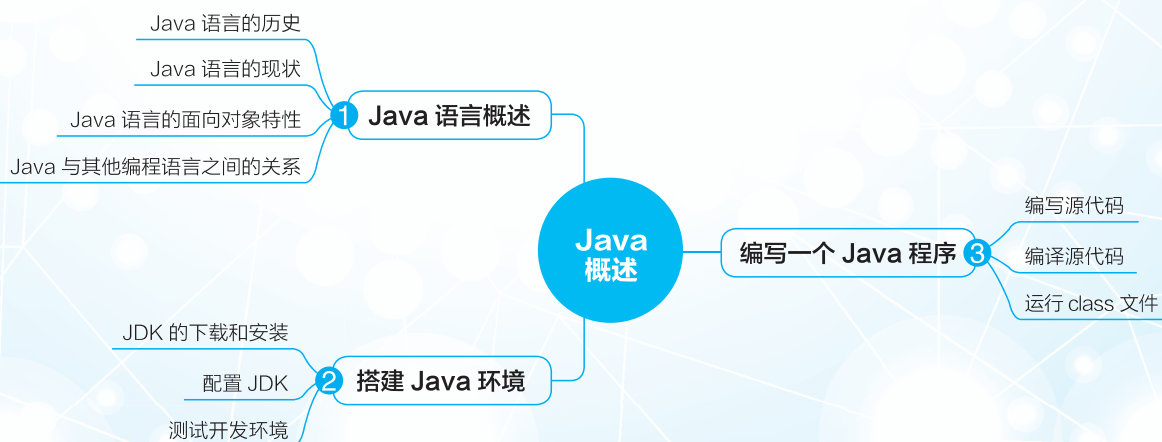
第 1 章

Java 概述

学习目标 >

- ① 了解 Java 语言的历史。
- ② 了解 Java 语言的现状。
- ③ 了解 Java 语言的面向对象特性。
- ④ 掌握不同平台的 JDK 环境的搭建。
- ⑤ 掌握 Java 程序的编写方法。

知识导图 >



本章将向读者简单介绍 Java 语言及其相关特性，Java 不同的版本，以及学好 Java 语言的方法等。本章的主要目的是让读者对 Java 语言有一个整体的了解，然后慢慢地学习具体内容，最后掌握 Java 语言。

1.1 Java 语言概述

Java 是一种跨平台的、面向对象的程序设计语言。很难想象在 Window 环境下编写的程序可以不加修改就可以在 Linux 系统中运行，因为计算机硬件只识别机器指令，而不同操作系统中的机器指令是不同的。所以，要把一种平台下的程序迁移到另一个平台中，必须针对目标平台进行修改。如果想使程序能够运行在不同的操作系统下，就要求程序设计语言能够跨平台，即可以跨越不同的硬件、软件环境，而 Java 语言就能够满足这种要求。

Java 自问世以来，受到越来越多的开发者喜爱，其不仅吸收了 C++ 语言的各种优点，还摒弃了 C++ 中令人难以理解的多继承、指针等概念，因此 Java 语言具有功能强大且简单易用等特点。本章将从 Java 的起源讲起，详细介绍 Java 的发展历程与特点，并对 Java 开发工具及 Java 程序开发的基本步骤进行详细讲解，此外还创建一个简单的 Hello World 程序。

1.1.1 Java 语言的历史

1991 年，美国 Sun 公司开始研究家用消费类电子设备。在 Sun 公司内部，James Gosling 领导的 Green 小组专注于软件方面的研究，该小组在开始阶段选择已经非常成熟的 C/C++ 语言进行开发和设计，可是却发现执行 C++ 程序需要消耗大量的内存，而且还不能兼容不同的设备。因此，在 C/C++ 语言的基础上，Green 小组开发出一种名为 Oak 的新语言（Java 语言的前身）。当时，由于 Green 小组并未将 Oak 语言产品化，因而未能给 Sun 公司带来经济效益，导致了 Oak 语言面临夭折的危险。直到 1995 年，伴随着互联网的出现及蓬勃发展，急需一种可面向网络编程，且能够在不同终端设备、不同的操作系统上运行的语言。与此同时，Marc Andreessen 开发的 Mosaic（后来称为 Netscape）启用 Oak 项目组成员，开发出基于 Oak 语言的 HotJava 浏览器，并得到 Sun 公司首席执行官 Scott McNealy 的全力支持，从而开启了 Oak 语言进军 Internet 的契机。由于 Oak 名称的版权问题，Sun 公司将 Oak 语言重新命名为 Java，同时图标也设计成冒着热气的咖啡形状（以爪哇岛咖啡命名）。

1.1.2 Java 语言的现状

借助 Java，程序开发人员可以自由地使用现有的硬件和软件系统平台。由于 Java 独立于平台，还可以应用于计算机之外的领域，如 Java 程序可以在便携式计算机、电视、电话、手机和其他设备上运行。Java 的用途数不胜数，与其他编程软件相比，它拥有无可比拟的优势，使用它所能节省的时间和费用也是十分可观的。

如果仔细观察就会发现，Java 就在人们身边。使用 Java 语言编写的常见开源软件包

括 NetBeans 和 Eclipse 集成开发环境、JBoss 和 GlassFish 应用服务器；商业软件则包括 Office 和 Chrome、Websphere 和 Oracle Database ligo。此外，各手机厂商都为自己的产品提供了对 Java 技术的支持，手机上的 Java 程序和游戏已经不胜枚举。

为了满足不同开发人员的需求，Java 开发分成了以下 3 个方向：Java ME、Java SE 和 Java EE，它们的侧重点各有不同，现列举如下。

(1) Java ME (Java Micro Edition) 是 Java 的微型版本，常用于嵌入式设备及消费类电器（如手机等）上的开发，主要用于嵌入式系统程序的开发。

(2) Java SE (Java Standard Edition) 是 Java 的标准版，用于针对普通 PC 的标准应用程序开发，主要用于桌面程序的开发，它是学习 Java EE 和 Java ME 的基础，也是本书的重点内容。

(3) Java EE (Java Enterprise Edition) 是 Java 的企业版，用于针对企业级应用服务的开发。Java ME、Java SE、Java EE 是 Java 针对不同的应用而提供的不同服务，即提供不同类型的类库。初学者一般可从 Java SE 入手来学习 Java 语言。Java SE 是一个优秀的开发环境，开发者可以基于这一环境创建功能丰富的交互式应用，并且可以把这些应用配置到其他平台上。Java SE 是多种不同风格软件的开发基础，包括客户端 Java 小程序和应用程序，以及独立的服务器应用程序等，同时 Java SE 也是 Java ME 和 Java EE 的基础。事实上，大部分非企业级软件还是在 Java SE 上开发和部署得比较多。首先，这是因为很多的应用软件都是在 Java SE 上开发的；其次，Java SE 和 Java EE 是兼容的，企业版是对标准版的扩充，在 Java SE 版本上开发的软件拿到企业版上一样可以运行；最后，通常的手机及嵌入式设备的应用开发也是在 Java SE 环境中完成的，因为 Java ME 提供的只是微型版的一个环境，而人们完全可以在 Java SE 中将这个环境虚拟出来，然后将开发出来的应用软件拿到微型版的实际环境中去运行。其主要用于网页程序的开发，并且随着互联网的发展，越来越多的企业使用 Java 语言来开发自己的官方网站，其中不乏世界 500 强的企业。

1.1.3 Java 语言的面向对象特性

1. 简单性

Java 语言的语法简单明了、容易掌握，而且是纯面向对象的语言。Java 语言的简单性主要体现在以下几点。

(1) 语法规则和 C++ 类似。从某种意义上讲，Java 语言是由 C 和 C++ 语言转变而来的，所以 C 程序设计人员可以很容易地掌握 Java 语言的语法。

(2) Java 语言对 C++ 进行了简化和提高。例如，Java 使用接口取代了多重继承，并取消了指针，因为指针和多重继承通常会使程序变得复杂。Java 语言还通过垃圾自动回收机制，大大简化了程序设计人员对资源进行释放的管理工作。

(3) Java 提供了丰富的类库和 API 文档以及第三方开发包，另外，还有大量的基于 Java 的开源项目，现在 JDK 也开放源代码了，读者可以通过分析项目的源代码来提高自己的编程水平。

2. 面向对象

Java 语言本身是种面向对象的程序设计语言。Java 提倡万物皆对象，语法中规定不能



笔记



在类外面定义单独的变量和方法，也就是说，Java 语言最外部的数据类型是对象，所有的元素都要通过类和对象来访问。

3. 分布性

Java 的分布性包括操作分布和数据分布，其中，操作分布是指在多个不同的主机上布置相关操作，而数据分布是指将数据分别存放在多个不同的主机上，这些主机是网络中的不同成员。Java 可以凭借 URL 对象访问网络对象，访问方式与访问本地系统相同。

4. 可移植性

Java 程序具有与体系结构无关的特性，从而使 Java 程序可以很方便地移植到网络中的不同计算机上。同时，Java 的类库中也实现了针对不同平台的接口，使这些类库也可以移植。

5. 解释性

运行 Java 程序需要解释器，任何移植了 Java 解释器的计算机或其他设备都可以用 Java 字节码进行解释执行。字节码独立于平台，其本身携带了许多编译信息，使得连接过程更加简单，开发过程也就更加迅速，更具探索性。

6. 安全性

Java 语言删除了类似 C 语言中的指针和内存释放等语法，从而有效地避免了非法操作内存。Java 程序代码要经过代码校验、指针校验等测试步骤才能够运行，所以未经允许的 Java 程序不可能出现损害系统平台的行为，而且使用 Java 可以编写防病毒和防修改的系统。

7. 健壮性

Java 的设计目标之一是编写多方面可靠的应用程序，Java 将检查程序在编译和运行时产生的错误，并进行消除。类型检查能帮助用户检查出许多在开发早期出现的错误，同时，很多集成开发工具 IDE（如 Eclipse、NetBeans 等）的出现使编译和运行 Java 程序更加容易。

8. 多线程

多线程机制能够使应用程序在同一时间并行执行多项任务，而且相应的同步机制可以保证不同线程能够正确地共享数据。使用多线程，可以带来更好的交互能力和实时行为。

9. 高性能

Java 编译后的字节码是在解释器中运行的，所以它的速度比多数交互式应用程序提高了很多。另外，字节码可以在程序运行时被翻译成特定平台的机器指令，从而进一步提高运行速度。

10. 动态

Java 在很多方面比 C 和 C++ 更能够适应环境的发展，可以动态调整库中的方法和变量的增加，而客户端却不需要进行任何更改。在 Java 中，动态调整是非常简单、直接的。

1.1.4 Java 与其他编程语言之间的关系

程序开发语言可分为 4 代：机器语言、汇编语言、高级语言和面向对象程序设计语言。机器语言是机器最终执行时所能识别的二进制序列，任何其他语言编写的程序最后都



要转换为相应的机器语言才能运行。在电子计算机刚刚诞生的一小段时间内，人们只能用 0、1 进行编程，后来为了提高编程效率，引入了英文助记符，才出现了汇编语言。汇编语言的出现大大提升了代码的编写速度，同时也使代码的可读性和可维护性大大提高。直到今天，仍然有人在用汇编语言进行编程，当然这主要是为底层使用（如一些硬件驱动），毕竟汇编语言的执行效率高。但是，汇编语言对于程序员的自身要求还是很高的，一般需要程序员是专业出身的，这就限制了其他领域的科技工作者们想利用计算机进行辅助工作。因此，为了普及计算机的使用，使其成为社会各行各业都可以应用的一种工具，需要开发语法简单、编写容易的高级编程语言。Bill Gates 设计开发的 Basic 语言就是一种高级汇编语言。除了 Basic，还有很多其他的高级语言，如 Pascal、Fortran、C 等。随着软件行业的不断发展，软件规模也变得越来越来大，迫切需要更高效的编程语言。基于此，Java、C++、Visual Basic 和 Delphi 等面向对象语言应运而生。除此之外，世界上还有很多其他编程语言，只不过它们不是很流行，并不被人们熟知。每一种流行的开发语言都有其优势：C 语言适合用来开发系统程序，很多的操作系统及驱动程序都是使用 C 语言编写的；Fortran 适合用来进行数值计算；Pascal 语言结构严谨，适合作为教学语言；Visual Basic 和 Delphi 适合用来开发中小型应用程序；C++ 适合开发大型应用程序；Java 适合开发跨平台的应用程序。总之，每种语言都有自己的特色，至于选用什么语言作为开发工具，关键是要看具体的开发任务。没有最好的，只有合适的。很多开发任务可能需要同时使用几种开发语言一起来完成。

1.2 搭建 Java 环境

在学习 Java 语言之前，必须了解并搭建好它所需要的开发环境。要编译和执行 Java 程序，Java 开发工具包（Java Developers Kits，JDK）是必备的。本节将具体介绍下载并安装 JDK 和配置环境变量的方法。

1.2.1 JDK 的下载和安装

JDK 的安装文件可以从 Oracle 网站（www.oracle.com）下载，直接在首页搜索“Java”，然后单击“Java Download Central”按钮进入下载页面，选择需要下载的版本即可。JDK 的安装文件有多个版本，每个版本对应不同的操作系统又分为多个文件。本书选择下载的是 Java SE 15 对应的 Windows 64 位版本，下载得到的文件是 `jdk-15.0.1-windows-x64_bin.exe`。下面以该文件为例介绍 JDK 的安装方法。

`jdk-15.0.1-windows-x64_bin.exe` 是一个自解压文件，双击即可解压缩，并进行安装工作。安装程序首先收集一些信息，用于安装的选择，然后才开始复制文件、设置 Windows 注册表等。安装过程中，只需要按照提示一步一步操作即可。默认的安装路径为 `C:\Program Files\Java\jdk-15.0.1\`（该路径在后面配置环境变量时要用到），读者可根据自己需要更改安装路径。安装完毕后，切换至安装目录，可以发现有如下一些子文件夹。

（1）bin 文件夹：包含编译器（`javac.exe`）、解释器（`java.exe`）、Applet 查看器（`appletviewer.exe`）等 Java 命令的可执行文件。

（2）lib 文件夹：存放了一系列 Java 类库。

（3）jre 文件夹：存放了 Java 运行时可能需要的一些可执行文件和类库。如果发现没有此文件夹，可在命令行窗口执行以下命令手动添加。



```
cd C:\Program Files\Java\jdk-15.0.1\ (此处是安装目录, 读者可根据具体的安装目录调整。)  
bin\jlink.exe --module-path jmods --add-modules bin\jlink.exe --module-path jmods  
--add-modules
```

(4) include 文件夹: 存放了一些头文件。

以上文件夹中, bin 文件夹是需要特别注意的。因为这个文件夹中的编译器和解释器是后面需要用到的。另外, 最好将这个文件夹的绝对路径 (C:\Program Files\Java\jdk-15.0.1\bin) 设置到环境变量 path 中, 这样在进入命令行窗口后就可以直接调用编译和执行命令了。

1.2.2 配置 JDK

配置环境变量主要是为了让程序知道到哪儿去找到它所需要的文件, 设置的内容是一些路径。在 Windows 操作系统中, 环境变量的具体操作步骤如下。

- (1) 在桌面上右击“计算机”图标, 在弹出的快捷菜单中选择“属性”选项, 单击“高级系统设备”按钮, 弹出“系统属性”对话框, 如图 1-1 所示。
- (2) 在“系统属性”对话框“高级”选项卡中单击“环境变量”按钮, 弹出“环境变量”对话框, 如图 1-2 所示。该对话框分为两部分, 上半部分用于设置用户变量, 下半部分则用于设置系统变量。它们的区别是用户变量只对本用户有效, 且设置后无须重新启动计算机; 系统变量则对任何用户均有效, 但设置后需要重启计算机才能生效。一般情况下, 配置为用户变量即可。这里共需要配置两个用户变量: path 和 classpath。



图 1-1 “系统属性”对话框

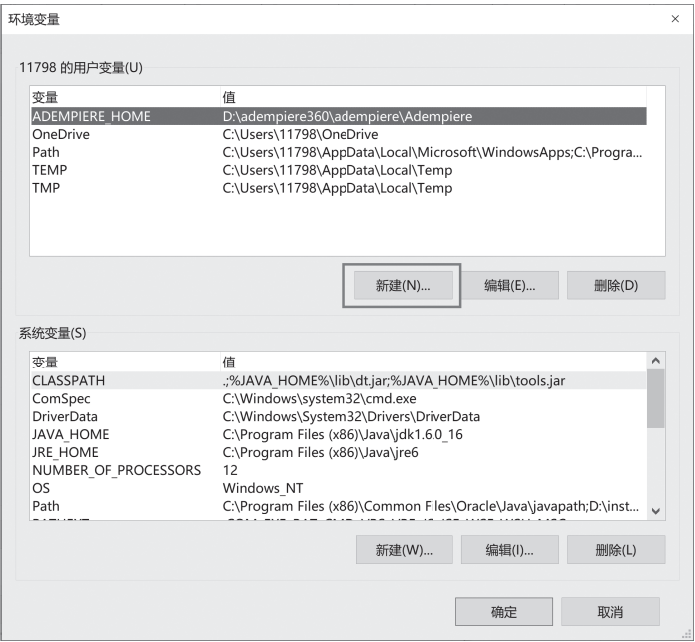


图 1-2 “环境变量”对话框

- (3) 若原本没有 path 用户变量, 就新建一个, 并将变量值设置为“C:\Program Files\Java\jdk-15.0.1\bin”, 如图 1-3 所示。如果已经存在 path 用户变量, 则直接将 JDK 的 bin 文件夹配置到环境变量 path 中即可。首先找到 path 变量, 然后单击“编辑”按钮, 弹出“编辑用户变量”对话框, 在“变量值”文本框中的最前面添加如下内容: “C:\Program

Files\Java\jdk-15.0.1\bin;”（注意在引号中，最后有一个英文分号，这个一定不能缺少）。最后单击一系列“确定”按钮后即可生效。

（4）若原本没有 classpath 用户变量，则新建一个，设置变量值为“C:\Program Files\Java\jdk-15.0.1\lib”。此外，当运行所编写的 Java 程序时，一般还需要将相应的工作目录（即存放 Java 程序及编译过的字节码文件的目录）也添加到 classpath 变量值中，以便程序运行时能找到用户所编写的 Java 类。这一点一定要特别注意，因为很多人在开始学 Java 时会忘记，从而导致程序运行失败。

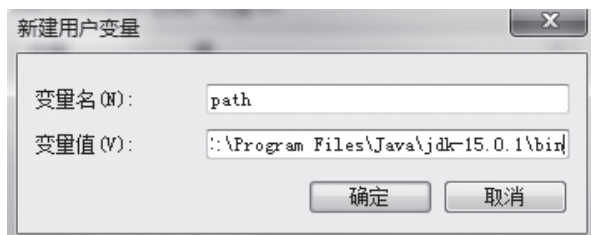


图 1-3 “新建用户变量”对话框

设置完上述环境变量后，可以选择“开始”→“运行”选项，打开“运行”程序并输入“cmd”，在打开的命令行窗口中执行“set”命令，验证刚才 Java 的设置是否成功，如图 1-4 所示。

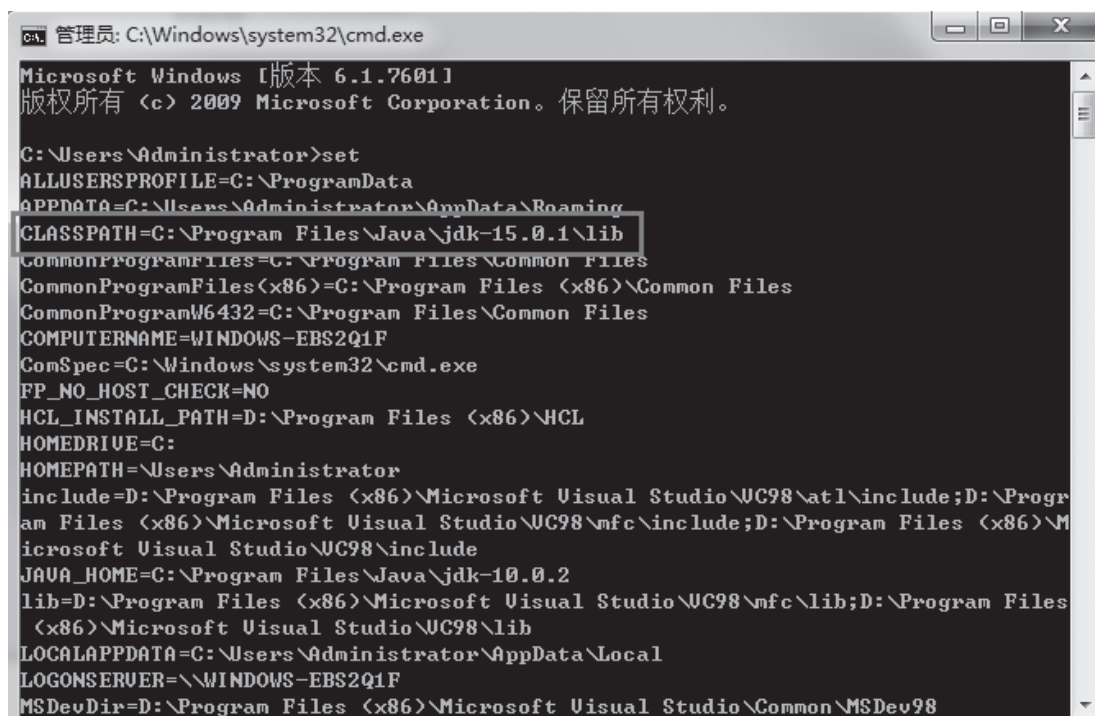
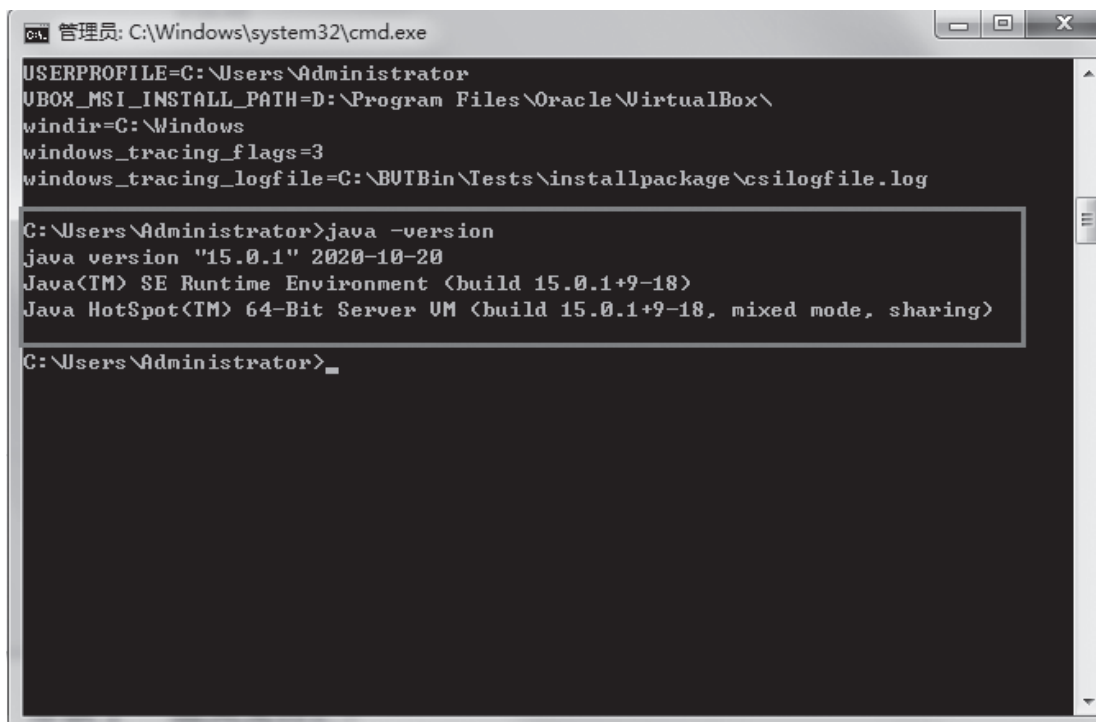


图 1-4 执行 set 命令

继续在命令行窗口中输入“java -version”，如果可以得到如图 1-5 所示的界面，则说明 JDK 的安装配置已经成功。



笔记



```
C:\Windows\system32\cmd.exe
USERPROFILE=C:\Users\Administrator
UBOX_MSI_INSTALL_PATH=D:\Program Files\Oracle\VirtualBox\
windir=C:\Windows
windows_tracing_flags=3
windows_tracing_logfile=C:\BUTBin\Tests\installpackage\csilogfile.log

C:\Users\Administrator>java -version
java version "15.0.1" 2020-10-20
Java(TM) SE Runtime Environment (build 15.0.1+9-18)
Java HotSpot(TM) 64-Bit Server VM (build 15.0.1+9-18, mixed mode, sharing)

C:\Users\Administrator>
```

图 1-5 测试 JDK 是否配置成功

1.2.3 测试开发环境

JDK 配置完成后，需确认是否配置准确。在 Windows 系统中测试 JDK 环境需要选择“开始”→“运行”选项（没有“运行”选项可以按“Windows+R”快捷键），然后在“运行”程序中输入“cmd”并单击“确定”按钮启动控制台，在控制台中输入 javac，按 Enter 键，将输出 JDK 的编译信息，其中包括修改命令的语法和参数选项等信息，这说明 JDK 环境搭建成功。

1.3 编写一个 Java 程序

在完成 JDK 的下载和安装后，就可以开始编写 Java 程序了。下面将编写第一个 Java 程序，其用途是在 DOS 控制台上显示“我能学好 Java 语言！”。本节将初步介绍开发 Java 程序的流程。

1.3.1 编写源代码

使用 Java 编程的第一步是编写源代码，这里需要使用到文本编辑器。目前有各种各样的文本编辑器，如“记事本”工具、Office 等。这里使用 Windows 系统自带的“记事本”工具。

选择“开始”→“所有程序”→“附件”→“记事本”选项，打开“记事本”工具，然后输入以下内容。

```
public class MyApp{
    public static void main(String[] args){
        System.out.println("我能学好 Java 语言!");
    }
}
```



按“Ctrl+S”快捷键保存源代码，弹出文件保存页面，将文件保存到 D 盘中，文件名使用 MyApp.java。默认记事本软件会为文件增加扩展名 .txt，而 Java 程序的源代码扩展名要使用 .java，所以在修改文件名时需要使用双引号将文件名括起来，这样就不会增加新的扩展名了。

文件名需要与源代码中公共类的名称相同，这里公共类的名字是 MyApp，因此文件名也只能是 MyApp。还需注意，Java 语言是区分大小写的，因此文件名必须是 MyApp，而不能是 Myapp，或者其他。

1.3.2 编译源代码

(1) 选择“开始”→“所有程序”→“附件”→“命令提示符”选项，将路径切换到 D 盘，并使用 dir 命令查看 D 盘中文件。

(2) 接着执行 javac -encoding UTF-8 .\MyApp.java 命令，然后使用 dir 命令查看 D 盘中的文件。

(3) 此时可以发现，在 D 盘中出现了一个名称为 MyApp、类型为 class 的文件，其中，javac 是 JDK 中提供的编译源代码的命令，由于已经配置了环境变量，因此可以直接使用。注意，javac 命令和 MyApp.java 之间存在一个空格。

1.3.3 运行 class 文件

输入“java MyApp”来运行 class 文件，这里可以看到，在 DOS 控制台上输出了“我能学好 Java 语言！”。使用 java 命令时，不需要输入文件的扩展名；而使用 javac 命令时，则需要输入文件的扩展名。

本章小结

本章首先简单介绍了 Java 语言的发展历史、现状以及各种特性；接着介绍了 JDK 的安装与配置，这是进行 Java 程序开发的第一步，也是本章的重点；最后介绍了如何开发第一个 Java 应用程序，读者可以从中学习到 Java 开发的一般流程：编写源代码、编译源代码和运行 class 文件。

实战训练

在不参考 1.3 节的情况下，自己动手编写一个 Java 程序，以熟悉 Java 程序的编写流程。

第 2 章

Eclipse 开发工具

学习目标 >

- ① 掌握安装 Eclipse 国际化语言包的方法。
- ② 学会 Eclipse 的菜单和工具栏的使用。
- ③ 学会使用 Eclipse 编写程序代码。
- ④ 掌握 Eclipse 中调试器的使用方法。

知识导图 >





本章导入

学习 Java 语言程序设计必须选择一个功能强大、使用简单，并且能够辅助程序设计的 IDE（集成开发工具）。Eclipse 是目前最流行的 Java 语言开发工具，其强大的代码辅助功能可以帮助程序开发人员自动完成语法修正、补全文字、代码修正、API 提示等编码工作，大大节省了程序开发所需的时间和精力。本章将简要介绍 Eclipse 开发工具，使读者能够初步了解 Eclipse 并使用它完成最基础的程序设计工作。

2.1 熟悉 Eclipse

虽然使用记事本和 JDK 编译工具已经可以编写 Java 程序，但在项目开发过程中必须使用大型的集成开发工具（Integrated Development, IDE）来编写 Java 程序，这样可以避免编码错误，方便管理项目结构，而且使用 IDE 的代码辅助功能可以快速地输入程序代码。本节将介绍 Eclipse 开发工具，包括其安装、配置与启动，以及其菜单栏、工具栏与以及各种视图的作用等。

2.1.1 Eclipse 简介

Eclipse 是由 IBM 公司投资 4000 万美元开发的集成开发工具。它基于 Java 语言编写，并且是开放源代码的、可扩展的，也是目前最流行的 Java 集成开发工具之一。另外，IBM 公司捐出 Eclipse 源代码，组建了 Eclipse 联盟，由该联盟负责该工具的后续开发。Eclipse 为编程人员提供了一流的 Java 程序开发环境，其平台体系结构是在插件概念的基础上构建的，插件是 Eclipse 平台最具特色的特征之一，也是其区别于其他开发工具的特征之一。学习了本章之后，读者应对 Eclipse 有一个初步的了解，以便为后续深入学习做准备。

2.1.2 下载安装 Eclipse

Eclipse 的安装文件可以到官网（<http://www.eclipse.org>）下载，具体的下载和使用步骤如下。

（1）打开浏览器，直接搜索 Eclipse，单击访问 Eclipse 的官网首页，然后单击官网首页上的“DownLoad Now”按钮，如图 2-1 所示。

（2）单击“DownLoad Now”按钮后，进入 Desktop IDE 页面，单击“DownLoad 2020-09”按钮，如图 2-2 所示。

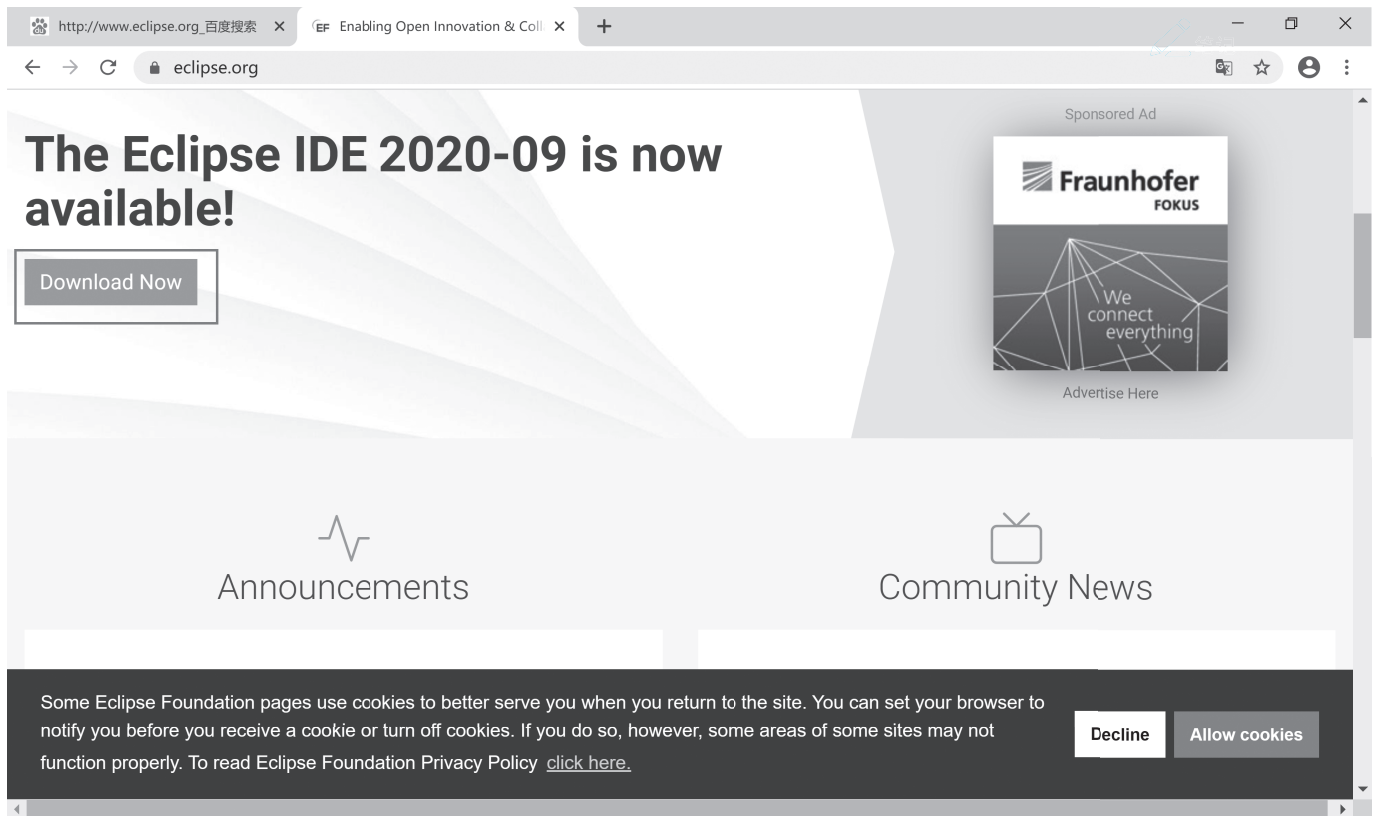


图 2-1 Eclipse 首页

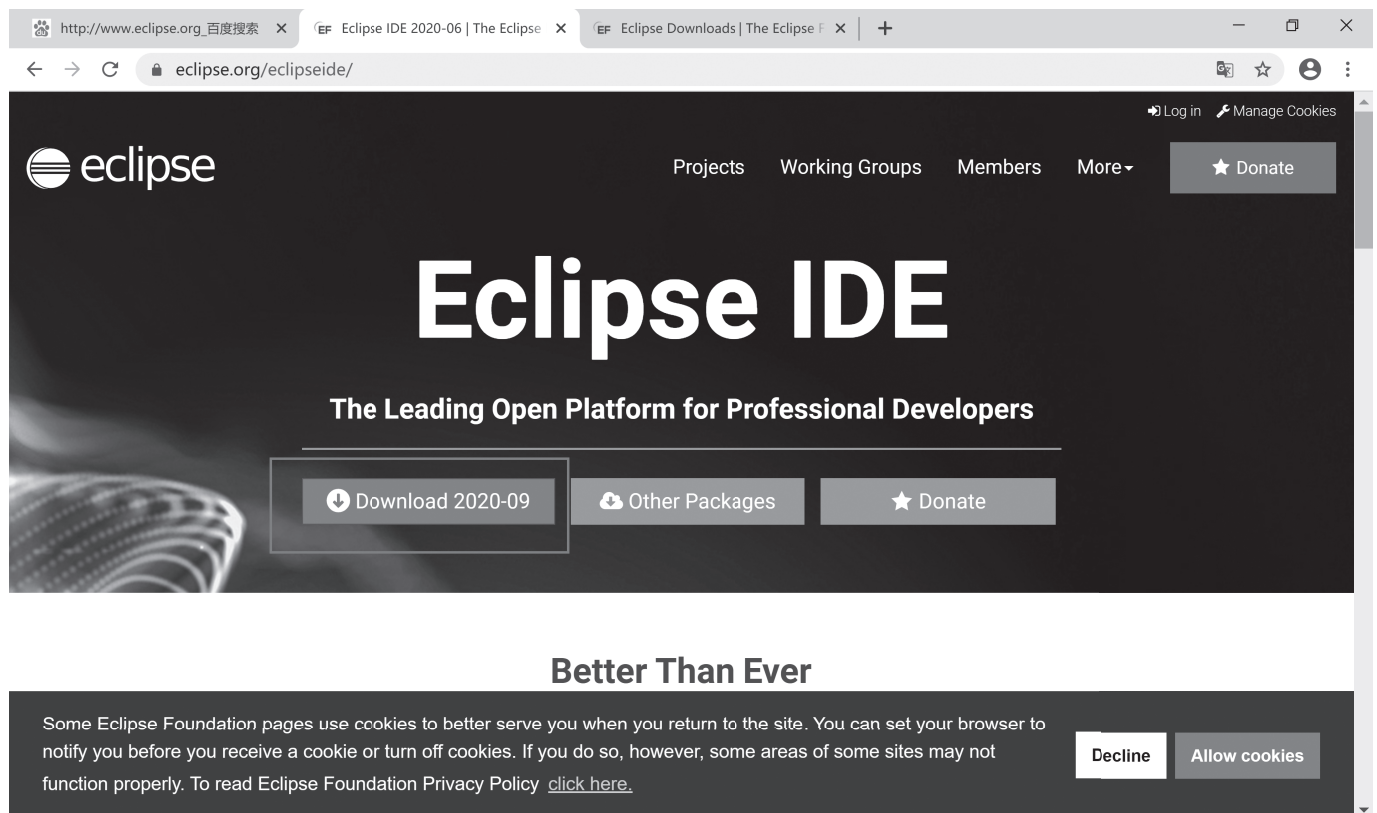


图 2-2 选择版本

(3) 选择 Eclipse IDE for Java Developers 对应的计算机操作系统及位数。这里选择的是 64 位的 Windows 操作系统, 如图 2-3 所示。

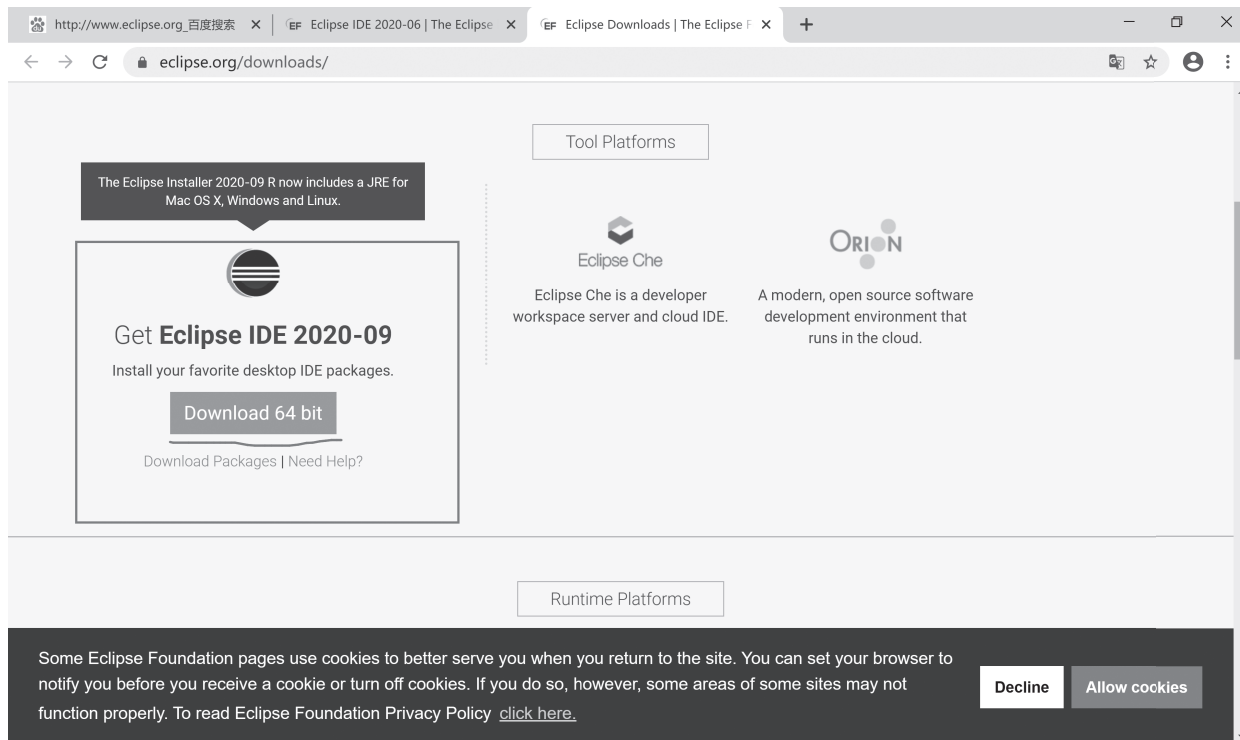


图 2-3 选择对应的操作系统及位数

(4) 在具体的下载页面单击“Download”按钮(见图 2-4), 打开下载窗口, 并显示付款界面, 也可以单击“Select Another Mirror”选择其他镜像进行下载。

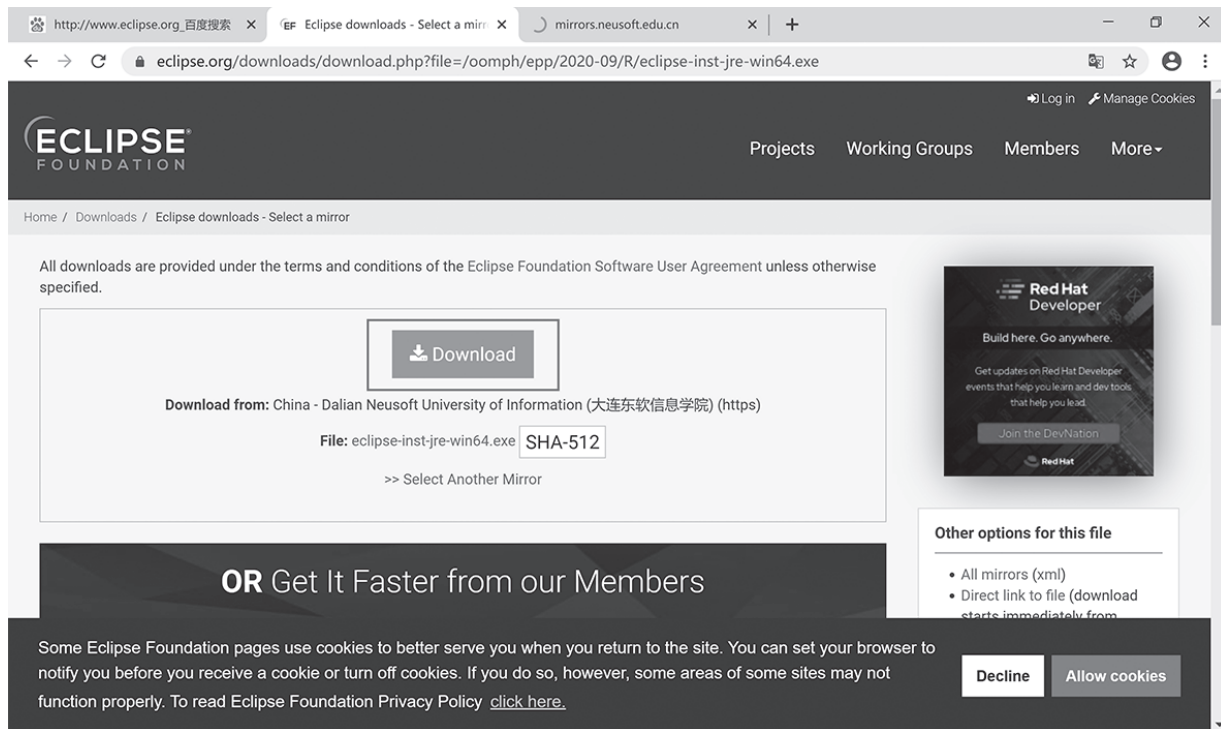


图 2-4 下载文件

从网站中下载的 Eclipse 安装文件是一个压缩包，将其解压缩到指定的文件夹，然后运行文件夹中的 Eclipse.exe 文件，即可启动 Eclipse 开发工具。但是在启动 Eclipse 之前需要安装中文语言包，以降低学习难度。



笔记

2.1.3 为 Eclipse 安装中文语言包

Eclipse 官网提供中文语言包的下载，具体下载步骤如下。

(1) 访问 <http://www.eclipse.org/babel/>，打开下载页面，单击“Downloads”按钮，如图 2-5 所示。

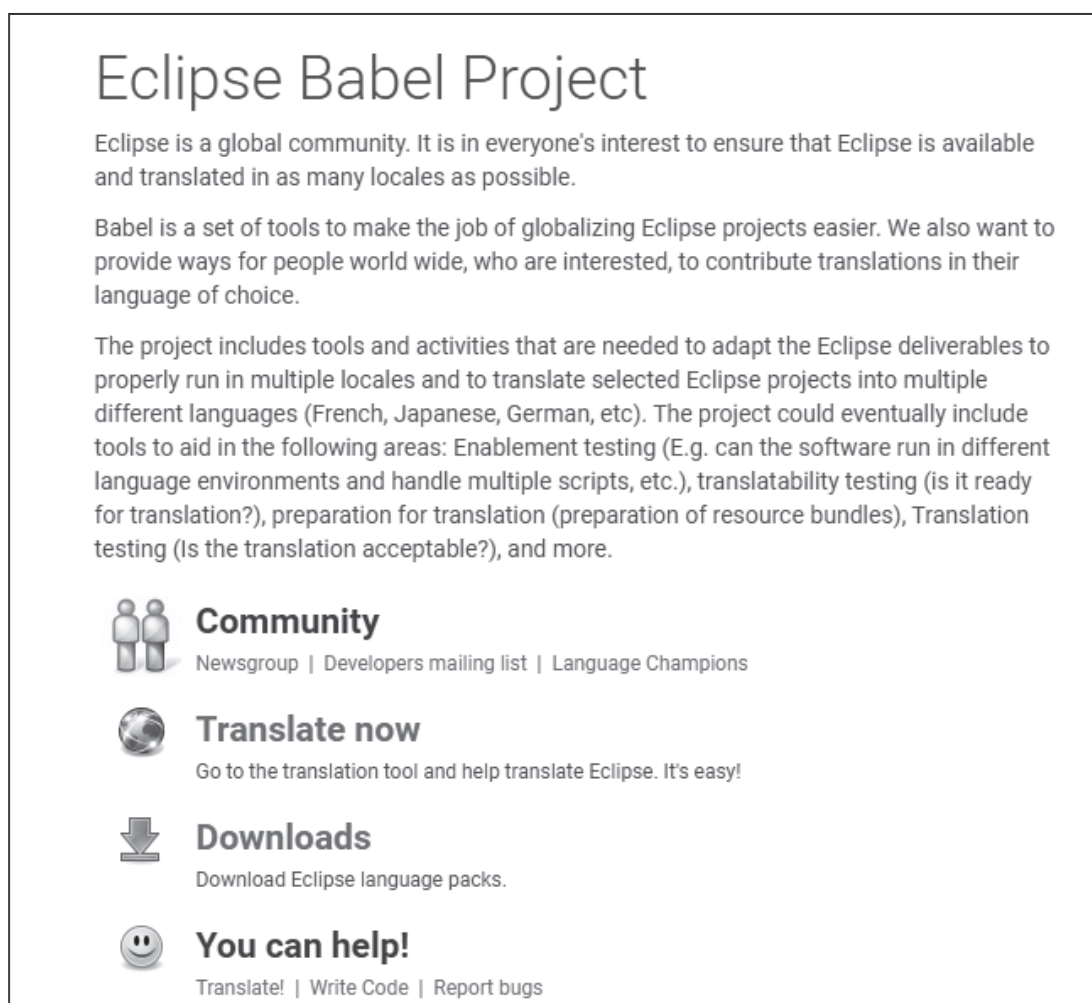


图 2-5 下载语言包首页

(2) 选择自己要下载的版本，一般选择最新版本，这里选择“2020-09”版本，如图 2-6 所示。

Babel Language Pack Zips and Update Sites - R0.18.1 (2020/10/14)

 **Babel Language Pack Zips**
2020-09 | 2020-06 | 2019-12

 **Babel Language Pack Update Site for 2020-09**
<https://download.eclipse.org/technology/babel/update-site/R0.18.1/2020-09/>
Zipped p2 repository for 2020-09 (149 MB)

 **Babel Language Pack Update Site for 2020-06**
<https://download.eclipse.org/technology/babel/update-site/R0.18.1/2020-06/>
Zipped p2 repository for 2020-06 (145 MB)

 **Babel Language Pack Update Site for 2019-12**
<https://download.eclipse.org/technology/babel/update-site/R0.18.1/2019-12/>
Zipped p2 repository for 2019-12 (138 MB)

图 2-6 选择语言包版本

(3) 选择中文语言包，如图 2-7 所示。

Language: Catalan	
• BabelLanguagePack-datatools-ca_4.17.0.v20201010020001.zip (0.01%)	
• <u>BabelLanguagePack-eclipse-ca_4.17.0.v20201010020001.zip (0.27%)</u>	
• BabelLanguagePack-modeling.tmf.xtext-ca_4.17.0.v20201010020001.zip (0.56%)	
• BabelLanguagePack-mylyn-ca_4.17.0.v20201010020001.zip (0.05%)	
• BabelLanguagePack-rt.rap-ca_4.17.0.v20201010020001.zip (0.09%)	
• BabelLanguagePack-technology.egit-ca_4.17.0.v20201010020001.zip (0.03%)	
• BabelLanguagePack-tools.cdt-ca_4.17.0.v20201010020001.zip (0.1%)	
• BabelLanguagePack-webtools-ca_4.17.0.v20201010020001.zip (0.01%)	
Language: Chinese (Simplified)	
• BabelLanguagePack-datatools-zh_4.17.0.v20201010020001.zip (75.71%)	
• <u>BabelLanguagePack-eclipse-zh_4.17.0.v20201010020001.zip (84.38%)</u>	
• BabelLanguagePack-modeling.emf-zh_4.17.0.v20201010020001.zip (58.69%)	
• BabelLanguagePack-modeling.mdt.bpmn2-zh_4.17.0.v20201010020001.zip (30.66%)	
• BabelLanguagePack-modeling.tmf.xtext-zh_4.17.0.v20201010020001.zip (56.33%)	
• BabelLanguagePack-mylyn-zh_4.17.0.v20201010020001.zip (45.49%)	
• BabelLanguagePack-rt.rap-zh_4.17.0.v20201010020001.zip (89.22%)	
• BabelLanguagePack-soa.bpmn2-modeler-zh_4.17.0.v20201010020001.zip (20.25%)	
• BabelLanguagePack-technology.egit-zh_4.17.0.v20201010020001.zip (21.1%)	
• BabelLanguagePack-technology.handly-zh_4.17.0.v20201010020001.zip (67.44%)	
• BabelLanguagePack-technology.jgit-zh_4.17.0.v20201010020001.zip (3.85%)	
• BabelLanguagePack-technology.lsp4e-zh_4.17.0.v20201010020001.zip (41.1%)	
• BabelLanguagePack-technology.packaging-zh_4.17.0.v20201010020001.zip (22.91%)	
• BabelLanguagePack-technology.packaging.mpc-zh_4.17.0.v20201010020001.zip (9.59%)	
• BabelLanguagePack-technology.passage-zh_4.17.0.v20201010020001.zip (18.08%)	
• BabelLanguagePack-technology.tm4e-zh_4.17.0.v20201010020001.zip (24.03%)	
• BabelLanguagePack-tools.cdt-zh_4.17.0.v20201010020001.zip (55.92%)	
• BabelLanguagePack-tools.gef-zh_4.17.0.v20201010020001.zip (4.92%)	
• BabelLanguagePack-tools.tracecompass-zh_4.17.0.v20201010020001.zip (20.69%)	
• BabelLanguagePack-webtools-zh_4.17.0.v20201010020001.zip (74.05%)	

图 2-7 选择中文语言包

(4) 将下载的压缩包中的两个文件(见图 2-8)复制到 Eclipse 安装目录下的 dropins 文件中,启动 Eclipse,就会自动加载汉化包(这里没有将这两个文件直接覆盖 Eclipse 安装目录下的两个文件,因为覆盖以后启动 Eclipse 程序会报错)。



笔记

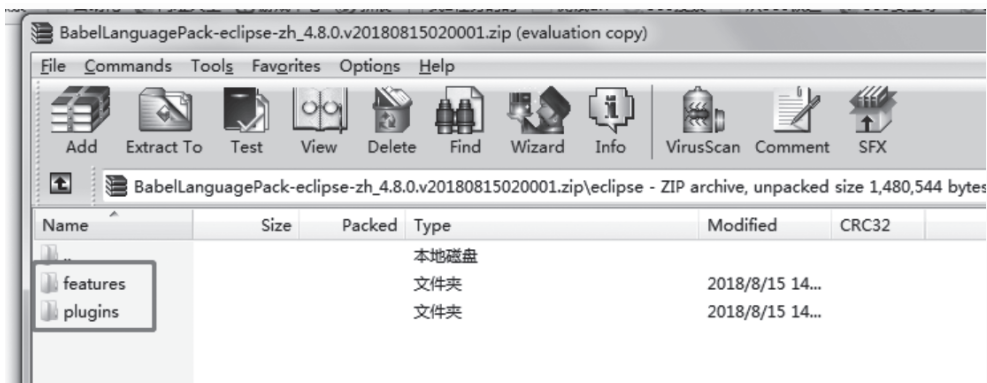


图 2-8 中文语言包文件

2.1.4 Eclipse 工作台

在 Eclipse 的欢迎界面中,单击“工作台”(Workbench)按钮或关闭欢迎界面,将显示 Eclipse 工作台,它是程序开发人员开发程序的主要场所。Eclipse 还可以将各种插件无缝地集成到工作台中,也可以在工作台中开发各种插件。Eclipse 工作台主要包括标题栏、菜单栏、工具栏、编辑器、透视图和相关的视图等,如图 2-9 所示。



图 2-9 Eclipse 工作台

1. 透视图与视图

本小节将分别介绍透视图和视图在 Eclipse 中的作用。

(1) 透视图。透视图是 Eclipse 工作台提供的附加组织层,它可以实现多个视图的布局 and 可用操作的集合,并为这个集合定义一个名称,起到一个组织的作用。例如, Eclipse 提供的 Java 透视图组织了与 Java 程序设计有关的视图和操作的集合,而“调试”透视图负责组织与程序调试有关的视图和操作集。在 Eclipse 的 Java 开发环境中提供了几种常用



的透视图，如“Java”透视图、“资源”透视图、“调试”透视图、“小组同步”透视图等。不同的透视图之间还可以进行切换，但是同一时刻只能使用一个透视图。

(2) 视图。视图多用于浏览信息的层次结构和显示活动编辑器的属性，例如，“控制台”视图用于显示程序运行时的输出信息和异常错误，而“包资源管理器”视图可以浏览项目的文件组织结构。视图可以单独出现，也可以与其他视图以选项卡样式叠加在一起，它们可以有自己独立的菜单和工具栏，并且可以通过拖动随意改变布局位置。

在视图标题上右击，在弹出的快捷菜单中选择“已拆离”选项可以将视图从 Eclipse 工作台界面中分离，让其以单独窗口存在，同样也可以将视图再次合并到工作台界面中。大部分视图都是从 Eclipse 中分离后截取的界面效果。

2. 菜单栏

Eclipse 的菜单栏包含了 Eclipse 的基本命令，在使用不同的编辑器时，还会动态地添加有关该编辑器的菜单。基本菜单栏中除了常用的“文件”“编辑”“窗口”和“帮助”等菜单以外，还提供了一些功能菜单，如“源代码”和“重构”等。

每个菜单中都包含不同的命令，这些命令用于完成最终的操作，如文件的打开与保存、代码格式化、程序的运行与分步调试等。每个菜单所包含的命令也都有所区别。菜单中的命令虽然很多，但并不是所有命令都经常使用。本小节将介绍几个最常用的菜单及其命令，其他不常用的菜单，读者可以在日后程序开发过程中慢慢掌握。

(1) “文件”菜单。“文件”菜单中包含“新建”“保存”“关闭”“打印”“切换工作空间”和“属性”等命令。菜单中包含的内容虽然很多，但也有常用的和不常用的。例如，不常用的打印功能。“文件”菜单中的常用命令都列在表 2-1 中，以方便阅读和查询。

表 2-1 “文件”菜单中的常用命令

命令	说明	快捷键
新建	创建新项目、元素或资源	Alt+Shift+N
关闭	关闭当前编辑器	Ctrl+W
全部关闭	关闭所有编辑器	Ctrl+Shift+W
保存	保存当前编辑器的内容	Ctrl+S
刷新	刷新所选元素的内容	F5
属性	打开所选元素的属性对话框	Alt+Enter

(2) “编辑”菜单。“编辑”菜单用于辅助程序代码设计工作，除了常用的“剪切”“复制”和“粘贴”命令之外，还提供了“快速修正”“将选择范围扩展到”和“内容辅助”等高级命令。“编辑”菜单中的常用命令如表 2-2 所示。

表 2-2 “编辑”菜单中常用的命令

命令	说明	快捷键
将选择范围扩展到	将选择编辑内容的范围扩大到外层元素、下一个元素、上一个元素或者恢复上一次选择的元素	-
查找 / 替换	搜索编辑器中的内容片段并根据需要替换为新的内容	Ctrl+F
查找下一个	搜索当前所选内容下一次出现的地方	Ctrl+K
查找上一个	搜索当前所选内容上一次出现的地方	Ctrl+Shift+K

续表



命令	说明	快捷键
添加书签	在当前光标所在行添加书签	-
添加任务	在当前光标所在行添加任务	-
灵活插入方式	切换插入方式。当禁用灵活插入方式时，将禁用自动缩进、添加右方括号等辅助功能	Ctrl+Shift+Insert
内容辅助	在当前光标位置打开内容辅助对话框	-
文字补全	补全当前编辑器中正在输入的文字	Ctrl+Alt+/
快速修正	如果光标位于问题代码附件中，则打开一个解决方案对话框	Ctrl+1

（3）“源代码”菜单。“源代码”菜单中包含的命令都是和代码编写相关的命令，主要用于辅助编程。“源代码”菜单的常用命令如表 2-3 所示。

表 2-3 “源代码”菜单中常用的命令

命令	说明	快捷键
切换注释	注释或取消注释当前选择的所有行	Ctrl+/
添加块注释	在当前选择的多行代码周围添加块注释	Ctrl+Shift+/
除去块注释	从当前选择的多行代码中除去块注释	Ctrl+Shift+\
更正缩进	更正当前选择的代码行的缩进	Ctrl+I
格式	使用代码格式化程序来格式化当前 Java 代码	Ctrl+Shift+F
组织导入	导入当前类所使用的类包	Ctrl+Shift+O
覆盖 / 实现方法	使用向导覆盖父类或实现接口中的方法	-
生成 Getter 和 Setter	使用向导创建成员变量的 getXXX()/setXXX() 方法	-
生成 hashCode() 和 equals()	打开生成 hashCode() 和 equals() 方法	-
使用字段生成构造函数	添加构造函数，这些构造函数初始化当前所选类型的字段。可用于类型、字段或类型中的文本选择	-
从超类中生成构造函数	对于当前所选类型，按照超类中的定义来添加构造函数	-
包围方式	使用代码模板包围所选语句	Alt+Shift+Z
外部化字符串	打开“将字符串外部化”向导，此向导允许通过使用语句访问属性文件来代替代码中的所有字符串	-

（4）“重构”菜单。“重构”菜单是 Eclipse 中最关键的菜单，主要包括项目重构的相关命令，应该重点掌握。“重构”菜单中常用的命令如表 2-4 所示。

表 2-4 “重构”菜单中常用的命令

命令	说明	快捷键
重命名	重命名所选择的 Java 元素	Alt+Shift+R
移动	移动所选择的 Java 元素	Alt+Shift+V
抽取方法	创建一个包含当前所选语句或表达式的新方法，并创建相关的引用	Alt+Shift+M
抽取局部变量	创建为当前所选表达式指定的新变量，并将选择替换为对新变量的引用	Alt+Shift+L

笔记

续表

命令	说明	快捷键
抽取常量	从所选表达式中创建静态终极字段并替换字段引用，并且可以选择重写同一表达式的其他出现位置	-
内联	直接插入局部变量、方法或常量	Alt+Shift+I
将匿名类转换为嵌套类	将匿名内部类转换为成员类	-
将成员类型转换为顶级	为所选成员类型创建新的 Java 编译单元，并根据需要更新所有引用	-
将局部变量转换为字段	如果该变量是在创建时初始化的，则此操作将把初始化移至新字段的声明或类的构造函数中	-
抽取超类	从一组同类型中抽取公共超类	-
抽取接口	根据当前类的方法创建接口，并使该类实现这个接口	-
包括字段	将对变量的所有引用替换为 getXXX()/setXXX() 方法	-
历史记录	浏览工作空间重构历史记录，并提供用于从重构历史记录中删除重构的选项	-

3. 工具栏

Eclipse 与大多数软件的布局格式相同，Eclipse 的工具栏位于菜单栏的下方。工具栏中的按钮都是菜单命令对应的快捷图标，在打开不同的编辑器时，还会动态地添加与编辑器相关的新工具栏按钮。另外，除了菜单栏下面的主工具栏，Eclipse 中还有视图工具栏、透视图工具栏和快速视图工具栏等多种工具栏。

（1）主工具栏。主工具栏就是位于 Eclipse 菜单栏下方的工具栏，其内容将根据不同的透视图和不同类型的编辑器显示相关的工具按钮。

（2）视图工具栏。Eclipse 界面中包含多种视图，这些视图都有不同的用途（有关视图的概念已在前文中进行过介绍），可以根据视图功能需求在视图的标题栏位置添加相应的视图工具栏。例如，“控制台”视图用于输出程序运行中的输出结果和运行时的异常信息。

（3）透视图工具栏。透视图工具栏主要包括切换已经打开的不同透视图的缩略按钮，以及打开其他视图的按钮。在相应的工具按钮上右击就会弹出透视图的管理菜单，可以实现透视图的定制、关闭、复位、布局设置、是否显示文本等操作。

（4）快速视图工具栏。快速视图工具栏包含了一组快速视图的图表按钮，该工具栏中的视图会以图标按钮形式隐藏，单击指定的工具按钮时就会显示指定的视图，该视图失去焦点时则会自动隐藏到快速工具栏中。默认情况下，快速视图工具栏出现在工作台的左下角。

4. “包资源管理器”视图

该视图用于浏览项目结构中的 Java 元素，包括包、类、类库的引用等，但最主要的用途还是操作项目中的源代码文件。“包资源管理器”视图界面如图 2-10 所示。

5. “控制台”视图

“控制台”视图用于显示程序运行的输出结果和异常信息（Runtime Exception）。在学习 Swing

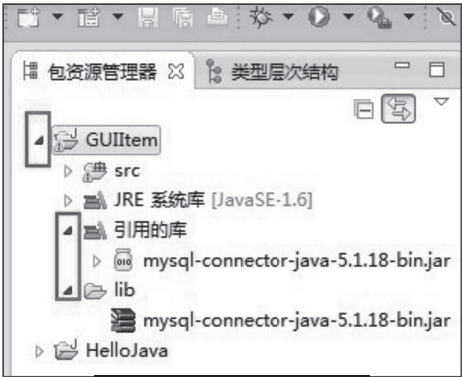


图 2-10 “包资源管理器”视图界面

程序设计之前，必须使用控制台实现与程序的交互。例如，为方便调试某个方法，该视图在方法执行前后会分别输出“方法开始”和“方法结束”信息。



笔记

Eclipse 的英文开发环境中，“控制台”视图称为“Console”。因为英文版的开发环境更加稳定，所以推荐读者使用英文原版的 Eclipse 来学习 Java。

2.2 使用 Eclipse

相信读者现在对 Eclipse 工具已经有了大体的认识，本节将介绍如何使用 Eclipse 完成 HelloJava 程序的编写和运行。

2.2.1 创建 Java 项目

要在 Eclipse 中编写程序，必须先创建项目。在 Eclipse 中可以创建很多种项目，其中 Java 项目用于管理和编写 Java 程序。创建该项目的步骤如下。

(1) 在如图 2-9 所示的界面中选择“File”(文件)→“New”(新建)→“Java Project”(Java 项目)选项，弹出“New Java Project”(新建 Java 项目)对话框。

(2) 设置“Project name”(项目名称)为 HelloJava，选中“Use default location”(使用默认位置)复选框，将项目保存到工作空间中，其他暂时不用设置，如图 2-11 所示。

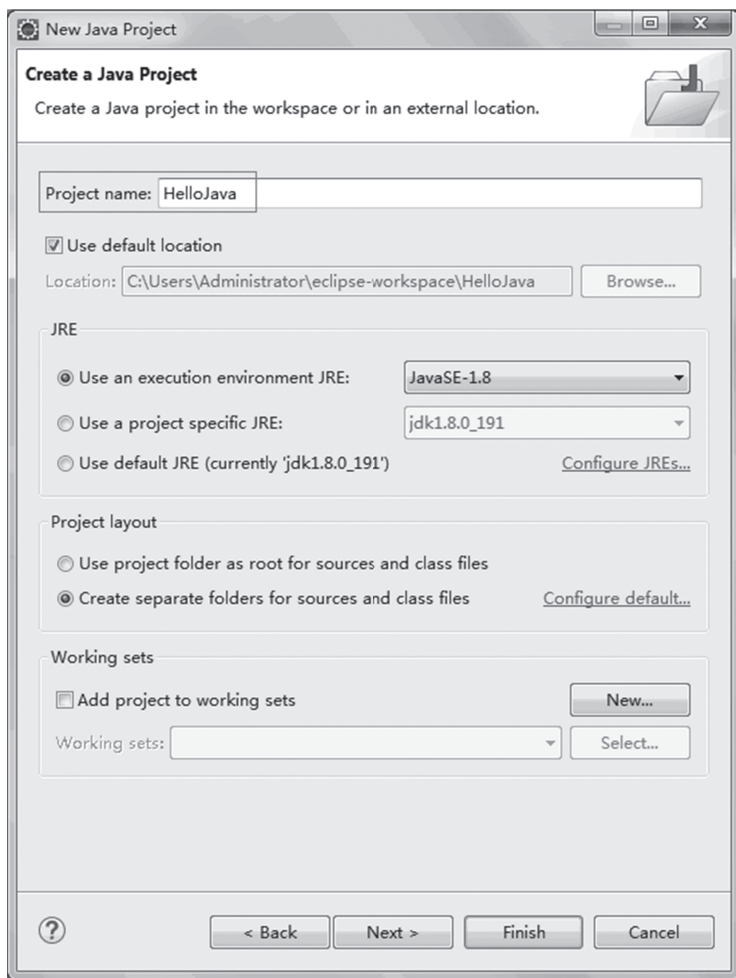


图 2-11 “New Java Project”(新建 Java 项目)对话框



(3) 单击“Next”(下一步)按钮,在进入的对话框中更改项目的 Source(源码)选项,同时还可以设置 Projects(项目)、Libraries(库)、Order and Export(排序和导出)方面的选项,分别如图 2-12 至图 2-15 所示。

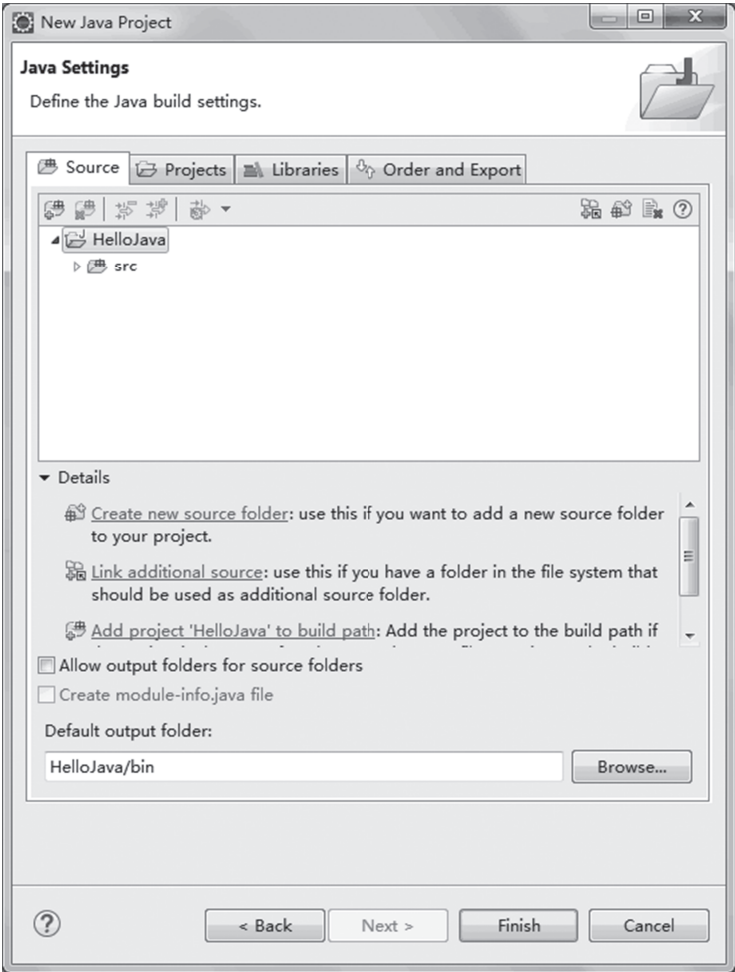


图 2-12 设置 Source(源码)选项

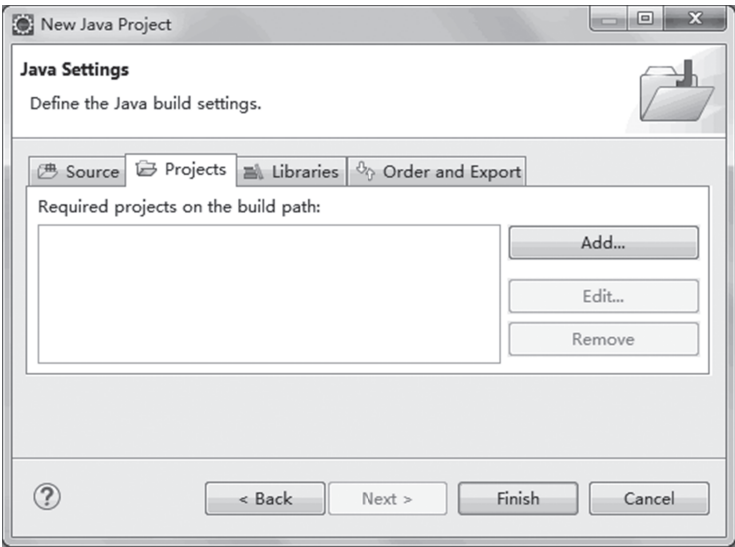


图 2-13 设置 Projects(项目)选项



笔记

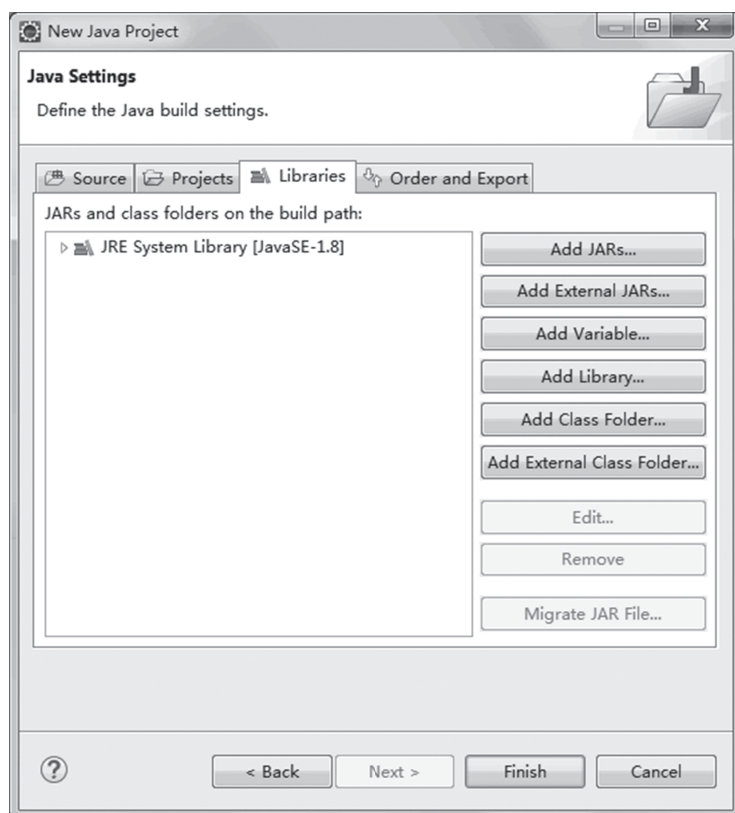


图 2-14 设置 Libraries (库) 选项

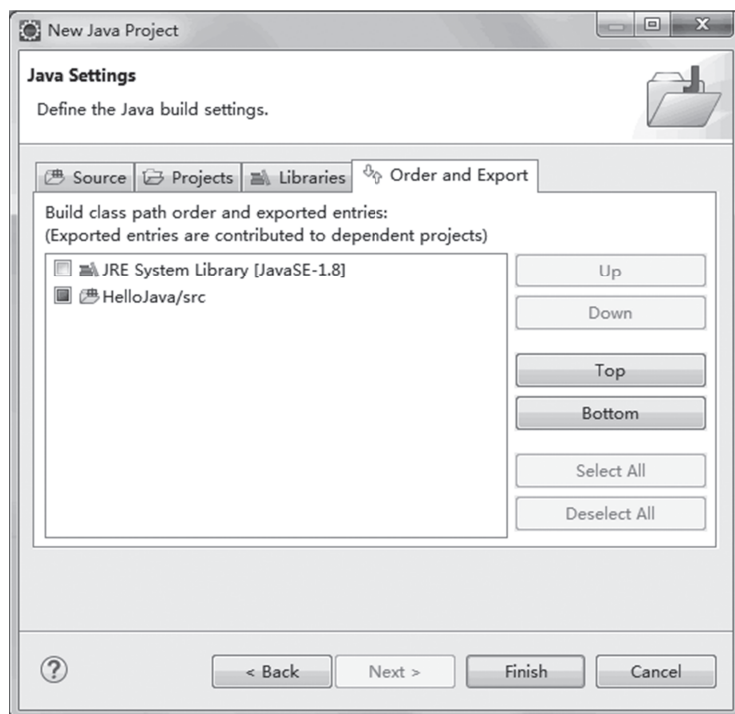


图 2-15 设置 Order and Export (排序和导出) 选项

(4) 单击“Finish”(完成)按钮,即可创建名称为 HelloJava 的项目,Eclipse 会自动生成相关的代码和布局结构。在 Eclipse 左侧“Package Explorer”(包资源管理器)窗格中会显示整个 Java 项目的目录结构,默认为空项目。

笔记

(5) 右击 src 目录，在弹出的快捷菜单中选择“New”(新建)→“Class”(类)选项，弹出“New Java Class”(新建 Java 类)对话框。设置类的名称为“HelloJava”，并勾选“public static void main (String[] args)”复选框，为新类生成 main() 方法，如图 2-16 所示。

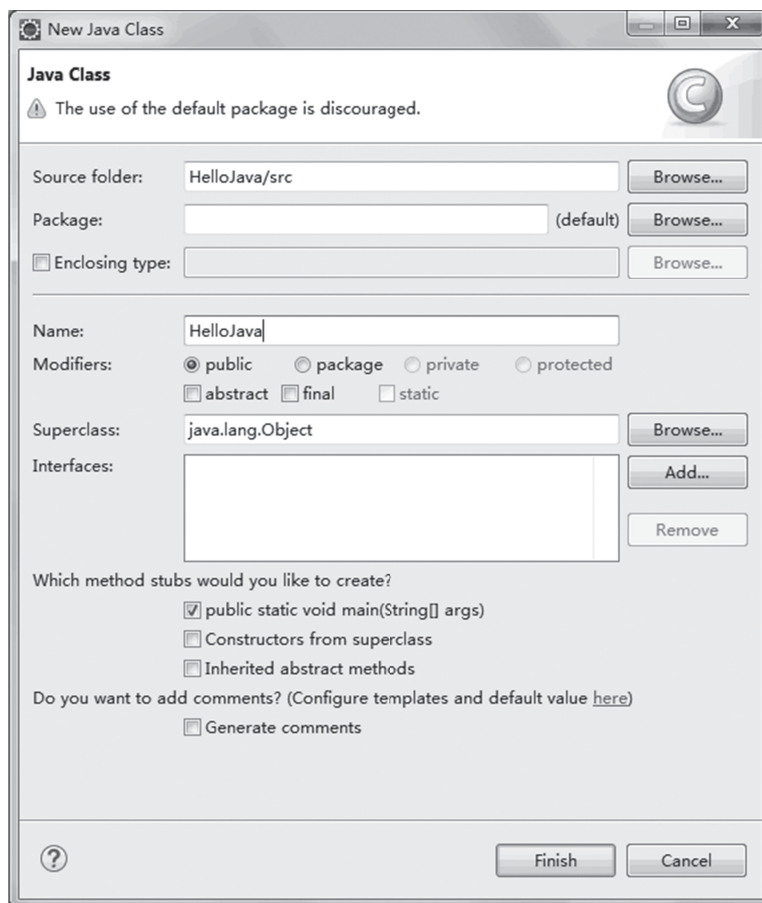


图 2-16 “New Java Class”(新建 Java 类)对话框

(6) 最后单击“Finish”(完成)按钮，会看到生成的 HelloJava.java 文件的内容，其处于编辑状态，如图 2-17 所示。

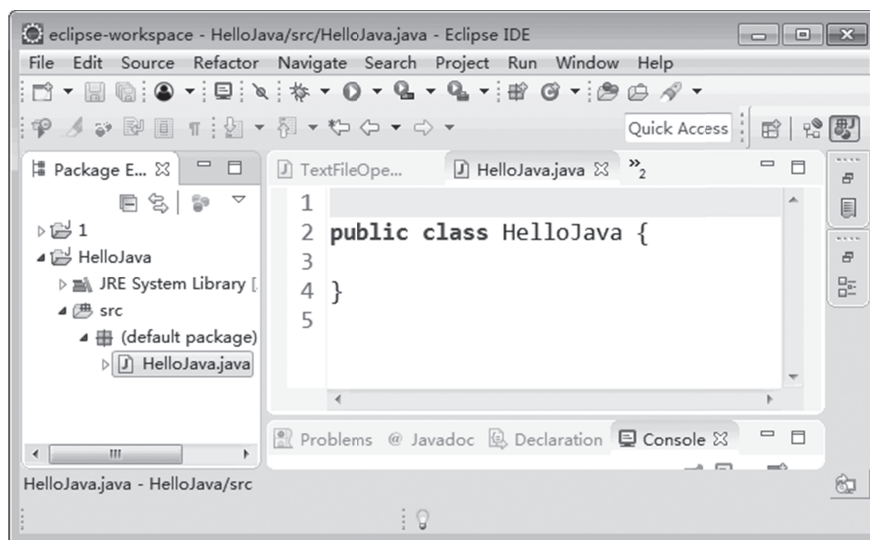


图 2-17 HelloJava.java 文件的内容

【例 2.1】对生成的源代码进行修改，最终文件代码如下：

```
public class HelloJava
{
    public static void main(String[] args)
    {
        //TODO 自动生成的方法存根
        System.out.println(" *");
        System.out.println(" ***");
        System.out.println(" *****");
        System.out.println(" *****");
        System.out.println(" *****");
    }
}
```



读者输入上述代码时，会充分体会到 Eclipse 强大的代码辅助功能。图 2-18 所示为代码的自动提示补全功能。

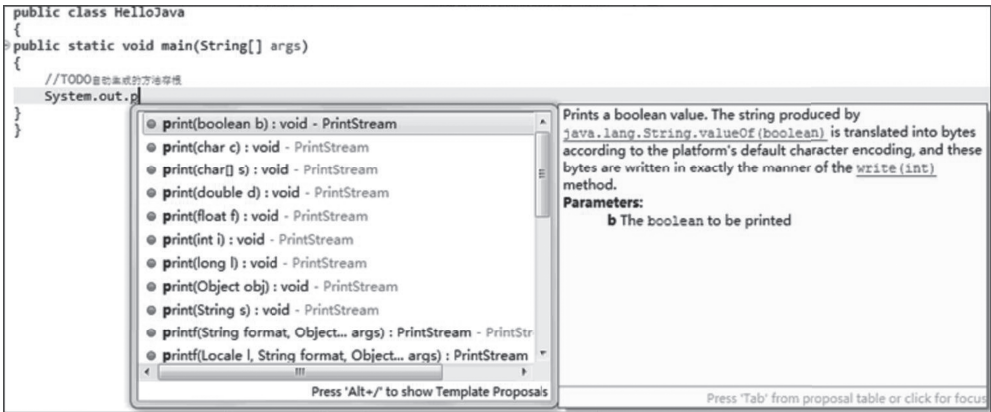


图 2-18 自动补全代码

编写完成之后按“Ctrl+S”快捷键保存修改。在工具栏中单击“运行”按钮后，如果程序没有编译错误，在底部的“Console”（控制台）窗格中会看到输出的等腰三角形，这就是程序的运行结果，如图 2-19 所示。

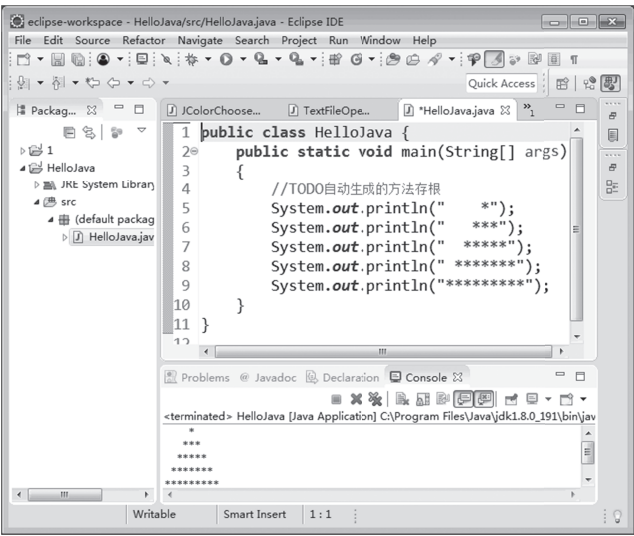


图 2-19 查看程序运行结果



2.2.2 使用编辑器编写程序代码

编辑器总是位于 Eclipse 工作台的中间区域，该区域可以重叠放置多个编辑器。编辑器的类型可以不同，但它们的主要功能都是用于完成 Java 程序、XML 配置等代码的编写或可视化设计工作。本小节将介绍如何使用 Java 编辑器和其他代码辅助功能快速编写 Java 程序。

1. 打开 Java 编辑器

在使用向导创建 Java 类文件后，会自动打开 Java 编辑器编辑新创建的 Java 类文件。除此之外，打开 Java 编辑器最常用的方法是在“包资源管理器”视图中双击 Java 源文件或在 Java 源文件处右击并在弹出的快捷菜单中选择“打开方式”→“Java 编辑器”命令。

Java 编辑器以不同的样式和颜色突出显示 Java 语法。这些突出显示的语法包括以下几个方面：

- (1) 程序代码注释。
- (2) Javadoc 注释。
- (3) Java 关键字。

在 Java 编辑器左侧右击，在弹出的快捷菜单中选择“显示行号”选项，可以开启 Java 编辑器显示行号的功能。

2. 编写 Java 代码

Eclipse 的强大之处并不在于编辑器能突显 Java 语法，而在于它强大的代码辅助功能。在编写 Java 程序代码时，可以使用“Ctrl+Alt+/”快捷键自动补全 Java 关键字，也可以使用“Alt+/”快捷键启动 Eclipse 代码辅助菜单。

在使用向导创建 HelloJava 类之后，向导会自动构建 HelloJava 类结构的部分代码，并建立 main() 方法，程序开发人员需要做的就是将代码补全，为程序添加相应的业务逻辑。本程序的完整代码如下：

```
public class HelloJava{
    private static String say ="我要学会你。";
    /**
     * @param args
     */
    public static void main (String[] args){
        System.out.println("你好 Java"+ say);
    }
}
```

在安装 Eclipse 后，Java 编辑器文本字体为 Consolas 10。应用该字体时，中文显示比较小，不方便查看。这时，可以选择主菜单上的“窗口”→“首选项”选项，在弹出的“首选项”对话框左侧的列表中选择“常规”→“外观”→“颜色和字体”节点，在右侧选择“Java 编辑器文本字体”节点，并单击“编辑”按钮，在弹出的对话框中选择 Courier New 字体，单击“确定”按钮，返回到“首选项”对话框中，单击“应用”按钮即可。另外，“调试”→“控制台字体”节点也需要进行以上修改。

在 HelloJava 程序代码中，第 1、3、4、5、6 行是由向导创建的，完成这个程序只要编写第 2 行和第 7 行代码即可。