
# 第1章
# 零起步学Java

## 内容概要

Java 是一种可以编写跨平台应用程序的面向对象程序设计语言。本章将对 Java 语言的发展历史、运行机制、开发环境，以及如何编译和执行 Java 应用程序等内容进行介绍。通过本章的学习，读者将会对 Java 语言有一个初步的了解，并能够顺利地搭建 Java 应用程序的运行开发环境。

## 学习目标

- 了解 Java 语言的发展历程
- 了解 Java 语言的特点
- 理解 Java 的工作原理
- 掌握 Java 开发环境的搭建

## 课时安排

- 理论学习 1 课时
- 上机操作 1 课时

## 1.1 Java 语言的发展历史和特点

### 1. Java 语言的发展历史

Java 语言的历史要追溯到 1991 年，当时美国 Sun 公司的 Patrick Naughton 及其伙伴 James Gosling 带领的工程师小组（Green 项目组）准备研发一种能够应用于智能家电（如电视机、电冰箱）的小型语言。由于家电设备的处理能力和内存空间都很有限，所以要求这种语言必须非常简练且能够生成非常紧凑的代码。同时，由于不同的家电生产商会选择不同的中央处理器（CPU），因此还要求这种语言不能与任何特定的体系结构捆绑在一起，也就是说必须具有跨平台能力。

项目开始时，项目组首先从改写 C/C++ 语言编译器着手，但是在改写过程中感到仅仅使用 C 语言无法满足需要，而 C++ 语言又过于复杂，安全性也差，无法满足项目设计的需要。于是项目组从 1991 年 6 月开始研发一种新的编程语言，并命名为 Oak，但后来发现 Oak 已被另一个公司注册，于是又将其改名为 Java，并配了一杯冒着热气的咖啡图案作为它的标志。

1992 年，Green 项目组发布了它的第一个产品，称之为“*7”。该产品具有非常智能的远程控制。遗憾的是当时的智能消费型电子产品市场还很不成熟，没有一家公司对此感兴趣，该产品以失败而告终。到了 1993 年，Sun 公司重新分析市场需求，认为网络具有非常好的发展前景，而且 Java 语言似乎非常适合网络编程，于是就将 Java 语言的应用背景转向了网络市场。

1994 年，在 James Gosling 的带领下，项目组采用 Java 语言开发了功能强大的 HotJava 浏览器。为了炫耀 Java 语言的超强能力，项目组让 HotJava 浏览器具有执行网页中内嵌代码的能力，为网页增加了“动态的内容”。这一“技术印证”在 1995 年的 SunWorld 上得到了展示，同时引发了人们延续至今的对 Java 语言的狂热追逐。

1996 年，Sun 公司发布了 Java 的第 1 个版本 Java 1.0，但它不能用来进行真正的应用开发，后来的 Java 1.1 弥补了其中大部分明显的缺陷，大大改进了反射能力，并为 GUI 编程增加了新的事件处理模型。

1998 年，Sun 公司发布了 Java 1.2 版，这个版本取代了早期玩具式的 GUI，并且它的图形工具箱更加精细而且具有较强的可伸缩性，更加接近“一次编写，随处运行”的承诺。

1999 年，Sun 公司发布 Java 三个版本：标准版（J2SE）、企业版（J2EE）和微型版（J2ME）。

2005 年，Sun 公司发布 Java SE 6。此时，Java 的各种版本已经更名，取消了其中的数字 2。J2EE 更名为 Java EE，J2SE 更名为 Java SE，J2ME 更名为 Java ME。

2010 年，Sun 公司被 Oracle 公司收购，交易金额达到 74 亿美元。

2011 年，Oracle 公司发布 Java 7.0 正式版。

2014 年，Oracle 公司发布 Java SE 8 正式版。

目前 Java SE 最新开发包是 Oracle 公司于 2018 年发布的 JDK 10.0.2。

本书主要介绍的是 Java SE，也就是 Java 标准版。

### 2. Java 语言的特点

Java 语言的特点与其发展历史是紧密相关的。它之所以能够受到如此多的好评以及拥有如此迅猛的发展速度，与其语言本身的特点是分不开的。其主要特点总结如下。

1）简单性

Java 语言是在 C++ 语言的基础上进行简化和改进的一种新型编程语言，它去掉了 C++ 中最难正确应用的指针和最难理解的多重继承技术等内容，因此，Java 语言具有功能强大和简单易用两个特征。

2）面向对象性

Java 语言是一种新的编程语言，没有兼容面向过程编程语言的负担，因此 Java 语言和 C++ 相比，其面向对象的特性更加突出。

Java 语言的设计集中于对象及其接口，它提供了简单的类机制及动态的接口模型。与其他面向对象的语言一样，Java 具备继承、封装及多态等核心技术，更提供了一些类的原型，程序员可以通过继承机制实现代码的复用。

3）分布性

Java 从诞生之日起就与网络联系在一起，它强调网络特性，从而使之成为一种分布式程序设计语言。Java 语言包括一个支持 HTTP 和 FTP 等基于 TCP/IP 协议的子库，它提供一个 Java.net 包，通过它可以完成各种层次上的网络连接。因此 Java 语言编写的应用程序可以凭借 URL 打开并访问网络上的对象，其访问方式与访问本地文件系统几乎完全相同。Java 语言的 Socket 类提供可靠的流式网络连接，使程序设计者可以非常方便地创建分布式应用程序。

4）平台无关性

借助于 Java 虚拟机（JVM），使用 Java 语言编写的应用程序不需要进行任何修改，就可以在不同的软、硬件平台上运行。

5）安全性

安全性可以分为四个层面，即语言级安全性、编译时安全性、运行时安全性、可执行代码安全性等。语言级安全性指 Java 的数据结构是完整的对象，这些封装过的数据类型具有安全性。编译时要进行 Java 语言和语义的检查，保证每个变量对应一个相应的值，编译后生成 Java 类。运行时，Java 类需要载入类加载器，并经由字节码校验器校验之后才可以运行。 Java 类在网络上使用时，对它的权限进行了设置，保证了被访问用户的安全性。

6）多线程

多线程机制使应用程序能够并行执行，通过使用多线程，程序设计者可以分别用不同的线程完成特定的行为，而不需要采用全局的事件循环机制，这样就很容易实现网络上的实时交互行为和实时控制性能。

大多数高级语言（包括C、C++等）都不支持多线程，用它们只能编写顺序执行的程序（除非有操作系统 API 的支持）。而 Java 却内置了语言级多线程功能，提供了现成的 Thread 类，只要继承这个类就可以编写多线程的程序，使用户程序并行执行。Java 提供的同步机制可

保证各线程对共享数据的正确操作，完成各自的特定任务。在硬件条件允许的情况下，这些线程可以直接分布到各个 CPU 上，充分发挥硬件性能，减少用户等待的时间。

7）自动废区回收性

在用 C 及 C++ 编写大型软件时，编程人员必须自己管理所用的内存块，这项工作非常困难并往往成为出错和内存不足的根源。在 Java 环境下，编程人员不必为内存管理操心，Java 语言系统有一个叫做“无用单元收集器”的内置程序，它扫描内存，并自动释放那些不再使用的内存块。Java 语言的这种自动废区收集机制，对程序不再引用的对象自动取消其所占资源，彻底消除了存储器泄露之类的错误，并免去了程序员管理存储器的繁琐工作。

## 1.2 Java 程序的运行机制

Java 语言比较特殊，其编写的程序需要经过编译步骤，但这个编译步骤不会产生特定平台的机器码，而是生成一种与平台无关的字节码（也就是 .class 文件），这种字节码不是可执行性的，必须使用 Java 解释器来解释执行，也就是需要通过 Java 解释器将字节码转换为本地计算机可执行的机器代码。

Java 语言里负责解释执行字节码文件的是 Java 虚拟机，即 Java Virtual Machine（JVM）。JVM 是可以运行 Java 字节码文件的虚拟计算机，所有平台上的 JVM 向编译器提供相同的编程接口，而编译器只需要面向虚拟机，生成虚拟机能理解的代码，然后由虚拟机来解释执行。不同平台上的 JVM 都是不同的，但他们都提供了相同的接口。JVM 是 Java 程序跨平台的关键部分，只要为不同的平台实现了相应的虚拟机，编译后的 Java 字节码就可以在该平台上运行。

Java 虚拟机执行字节码的过程由一个循环组成，它不停地加载程序，进行合法性和安全性检测，以及解释执行，直到程序执行完毕（包括异常退出）。

Java 虚拟机首先从后缀为 .class 的文件中加载字节码到内存中，接着在内存中检测代码的合法性和安全性。例如，检测 Java 程序用到的数组是否越界、所要访问的内存地址是否合法等，然后解释执行通过检测的代码，并根据不同的计算机平台将字节码转化成为相应的机器代码，再交给相应的计算机执行。如果加载的代码不能通过合法性和安全性检测，则 Java 虚拟机执行相应的异常处理程序。Java 虚拟机不停地执行这个过程直到程序执行结束。Java 程序的运行机制和工作原理如图 1-1 所示。

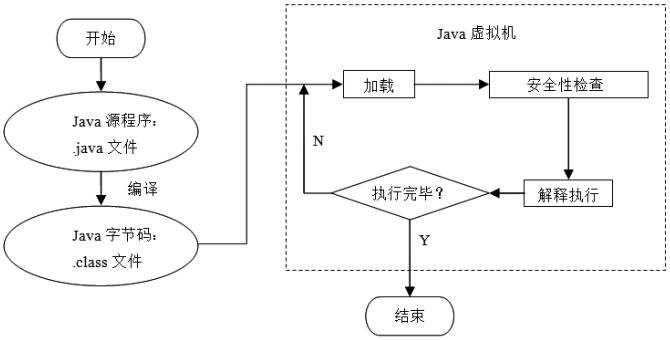


图 1-1　Java 程序的运行机制和工作原理

# 1.3 Java 开发环境的建立

经过前面几节的介绍，相信读者已经对 Java 语言的特点、运行机制等有了一定的了解，本节将详细介绍如何在本地计算机上搭建 Java 程序的开发环境。

## 1.3.1 JDK 的安装

JDK（Java Development Kit）是 Oracle 公司发布的免费的 Java 开发工具，它提供了调试及运行一个 Java 程序所有必需的工具和类库。在正式开发 Java 程序前，需要先安装 JDK。JDK 的最新版本可以到 http://www.oracle.com/technetwork/java/javase/downloads/index.html 上免费下载。目前 JDK 最新版本是 Oracle 公司于 2018 年 3 月发布的 JDK 10 正式版。根据运行时所对应的操作系统，JDK 10 可以划分为 for Windows、for Linux 和 for MacOS 等不同版本。

**说明：**从 2018 年开始，JDK 的发布周期由以前的数年一个大版本变化为 6 个月一个小版本。目前，业界使用最多的仍是 JDK 6、JDK 7、JDK 8 三个版本。本书实例基于的 Java SE 平台是 JDK 8 for Windows。

下面就以 JDK 8 for Windows 为例，来介绍它的安装和配置。

（1）通过网址 http://www.oracle.com/technetwork/java/javase/downloads/index.html 进入 Java SE 下载页面，可以找到最新版本的 JDK，如图 1-2 所示。


图 1-2　Java SE 下载页面

（2）单击 Java Platform（JDK）8u121 上方的 DOWNLOAD 按钮，打开 Java SE 下载列表页面，其中包括 Windows、Solaris 和 Linux 等平台的不同环境 JDK 的下载，如图 1-3 所示。

（3）在下载之前，选中 Accept License Agreement 单选按钮，接受许可协议。由于本书中使用的是 64 位版本的 Windows 操作系统，因此需要选择与平台对应的 Windows x64 类型的 jdk-8u121-windows-x64.exe 超链接，进行 JDK 的下载，如图 1-4 所示。


图 1-3　Java SE 下载列表页面


图 1-4　JDK 下载页面

（4）下载完成后，在计算机硬盘中可以发现一个名称为 jdk-8u121-windows-x64.exe 的可执行文件，双击该文件，将会弹出 JDK 安装程序的“欢迎”对话框，如图 1-5 所示。

（5）单击“下一步”按钮，进入“定制安装”界面，可以选择要安装的模块和路径，如图 1-6 所示。


图 1-5　欢迎对话框


图 1-6　“定制安装”界面

**注意：**在上述安装界面中，“开发工具”是必选的，“源代码”是给开发者作参考的，如果硬盘剩余空间比较多的话，最好选择安装；“公共 JRE”是一个独立的 Java 运行时环境（Java Runtime Environment，JRE），任何应用程序均可使用此 JRE，它会向浏览器和系统注册 Java 插件和 Java Web Start，如果不选择此项，IE 可能会无法运行 Java 编写的 Applet 程序。安装路径默认的是 C:\Program Files\Java\jdk1.8.0_121，如果需要更改安装路径，可以单击“更改”按钮，输入你想要的安装路径。

（6）单击“下一步”按钮，进入“进度”界面，如图 1-7 所示，可以了解 JDK 安装进度。

（7）JDK 安装完毕后，自动进入自定义安装 JRE 界面，如图 1-8 所示。通过此界面可以选择 JRE 的安装模块和路径，通常情况下，不需要用户修改这些默认选项。默认的安装路径是 C:\Program Files\Java\jre1.8.0_121，也可以通过单击“更改”按钮进行修改。


图 1-7 安装进度


图 1-8 自定义安装 JRE

（8）单击“下一步”按钮，开始 JRE 的安装，如图 1-9 所示。

（9）JRE 安装结束后，自动进入安装完成界面，如图 1-10 所示。单击“关闭”按钮，完成安装。单击“后续步骤”按钮，可以访问教程、API 文档、开发人员指南等内容。在这里，直接单击“关闭”按钮，完成 JDK 的安装。


图 1-9 安装 JRE


图 1-10 安装完成

JDK 安装完成后，会在安装目录下多出一个名称为 jdk1.8.0_121 的文件夹，打开该文件夹，如图 1-11 所示。

从图 1-11 中可以看出，JDK 安装目录下存在多个文件夹和文件，下面对其中一些比较重要的目录和文件进行简单介绍。

（1）bin 目录。JDK 开发工具的可执行文件，包括 java、javac、javadoc、appletviewer 等可执行文件。

（2）lib 目录。开发工具需要的附加类库和支持文件。

（3）jre。Java 运行环境，包含 Java 虚拟机、类库及其他文件，可支持执行用 Java 语言编写的程序。

（4）demo。带有源代码的 Java 平台编程示例。

（5）include。存放用于本地访问的文件。

（6）src.zip。Java 核心 API 类的源代码压缩文件。

注意：和一般的 Windows 程序不同，JDK 安装成功后，不会在“开始”菜单和桌面生成快捷方式。这是因为 bin 文件夹下面的可执行程序都不是图形界面，它们必须在控制台中以命令行方式运行。另外，还需要用户手工配置一些环境变量，才能方便地使用 JDK。

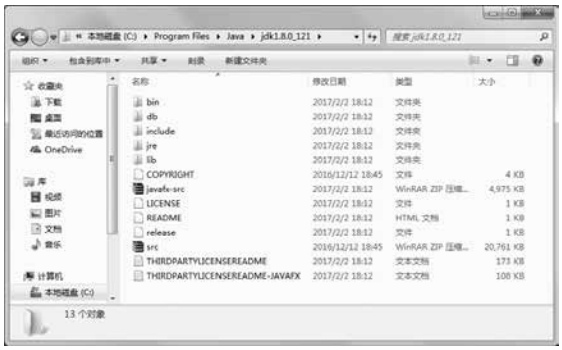


图 1-11　JDK 安装目录

## 1.3.2　系统环境变量的设置

环境变量是包含关于系统及当前登录用户的环境信息的字符串，一些程序使用此信息确定在何处放置和搜索文件。对于 Java 程序开发而言，主要会使用 JDK 的两个命令：javac.exe、java.exe，路径是 C:\Program Files\Java\jdk1.8.0_121\bin，但是它们不是 Windows 的命令，所以，要想使用此命令必须对其进行配置。和 JDK 相关的环境变量主要是 path 和 Classpath，JDK 1.5 以后，不设置 Classpath 也可以，所以此处只介绍 path。path 变量记录的是可执行程序所在的路径，系统根据这个变量的值查找可执行程序，如果待执行的可执行程序不在当前目录下，那就会依次搜索 path 变量中记录的路径；而 Java 的各种操作命令在其安装路径中的 bin 目录下，所以在 path 中设置了 JDK 的安装目录后就不用再把 Java 文件的完整路径写出来了，它会自动去 path 中设置的路径中寻找。

下面以 Windows 7 操作系统为例来介绍如何设置和 Java 有关的系统环境变量，假设 JDK 安装在系统默认目录下。

### 1.path 的配置

（1）右击“计算机”，在弹出的快捷菜单中选择“属性”选项，然后在打开的窗口中选择左侧导航栏里面的“高级系统设置”，弹出“系统属性”对话框，如图 1-12 所示。


图 1-12　“系统属性”对话框

（2）单击“环境变量”按钮，弹出“环境变量”对话框，选中“系统变量”列表框中的 Path 变量，如图 1-13 所示。

（3）单击“系统变量”列表框下方的“编辑”按钮，弹出“编辑系统变量”对话框，对环境变量 Path 进行修改，如图 1-14 所示。

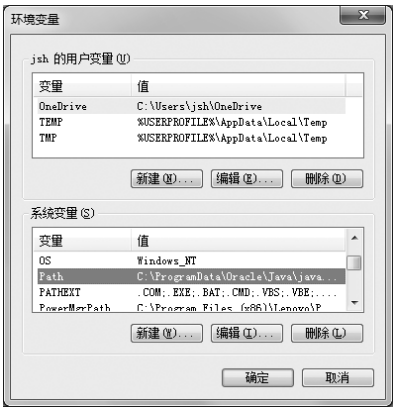


图 1-13 “环境变量”对话框


图 1-14 “编辑系统变量”对话框

（4）在“变量值”的尾部添加“; C:\Program Files\Java\jdk1.8.0_121\bin;”，然后单击“确定”按钮，完成对 Path 环境变量的设置。

### 2. 测试环境变量配置是否成功

（1）按组合键 Win+R，在弹出的“运行”对话框中输入 cmd，如图 1-15 所示。

（2）单击“确定”按钮，打开 dos 命令行窗口，输入 javac 命令，然后按 Enter 键，出现图 1-16 所示的信息，表示环境变量配置成功。

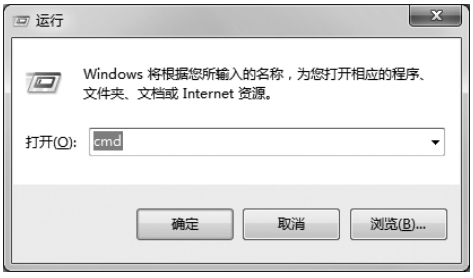


图 1-15 “运行”对话框


图 1-16 javac 命令执行结果

## 1.4 创建第一个 Java 应用程序

Java 开发环境建立好以后，就可以开始编写 Java 应用程序了。为了使读者对开发 Java 应用程序的步骤有一个初步的了解，本节将向读者展示一个完整的 Java 应用程序的开发过

程，并给出一些开发过程中应该注意的事项。

### 1.4.1 编写源程序

Java 源程序的编辑可以在 Windows 的记事本中进行，也可以在诸如 Edit Plus、Ultra Edit 之类的文本编辑器中进行，还可以在 Eclipse、NetBeans、JCreator、MyEclipse 等专用的开发工具中进行。

现在假设在记事本中进行源程序的编辑，启动记事本应用程序，在其窗口中输入如下程序代码：

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

程序代码输入完毕后，将该文件另存为 HelloWorld.java，“保存类型”选择“所有文件”，然后单击“保存”按钮，把文件保存到 D:\chapter1 文件夹中，如图 1-17 所示。


图 1-17　保存 HelloWorld.java 文件

**注意：**

（1）存储文件时，源程序文件的扩展名必须为 .java，且源程序文件名必须与程序中声明为 public class 的类的名字完全一致（包括大小写一致）。

（2）程序中的 public class HelloWorld 表示要声明一个名为 HelloWorld 的类，其中 class 是声明一个类必需的关键字。类由类头和类体组成，类体部分的内容由一对大括号括起来。类中不能嵌套声明其他类。类体内容包括属性和方法，具体内容将在第 4 章中介绍。

（3）Java 应用程序可以由若干类组成，每个类可以定义若干个方法，但必须有一个类中包含有一个且只能有一个 public static void main(String[] args) 方法，main 方法是所有 Java 应用程序执行的入口点，当运行 Java 应用程序时，将从 main 方法开始执行。

（4）System.out 是 Java 提供的标准输出对象，println 是该对象的一个方法，用于向屏幕输出。

### 1.4.2 编译和运行执行程序

JDK 所提供的开发工具主要有编译程序、解释执行程序、调试程序、Applet 执行程序、文档管理程序、包管理程序等，这些都是控制台程序，要以命令的方式执行。其中，编译程序和解释执行程序是最常用，它们都在 JDK 安装目录下 bin 文件夹中。

**1．编译程序**

JDK 的编译程序是 javac.exe，该命令将 Java 源程序编译成字节码，生成与类同名但扩展名为 .class 的文件。通常情况下，编译器会把 .class 文件放在和 Java 源文件相同的一个文件夹里，除非在编译过程中使用了 -d 选项。javac 的一般用法如下：

```
javac [ 选项…] file.java
```

其中，常用选项如下。

-classpath：该选项用于设置路径，在该路径上 javac 寻找需被调用的类。该路径是一个用分号分开的目录列表。

-d directory：该选项用于指定存放生成的类文件的位置。

-g：该选项在代码产生器中打开调试表，以后可凭此调试产生字节代码。

-nowarn：该选项用于禁止编译器产生警告。

-verbose：该选项用于输出有关编译器正在执行的操作的消息。

-sourcepath < 路径 >：该选项用于指定查找输入源文件的位置。

-version：该选项标识版本信息。

虽然 javac 的选项众多，但对于初学者而言，并不需要一开始就掌握全部选项的用法，只需要掌握最简单的用法就可以了。例如，编译 HelloWorld.java 源程序文件，只需在命令行输入如下命令即可：

```
javac HelloWorld.java
```

**注意**：javac 和 HelloWorld.java 之间必须用空格隔开，文件扩展名 .java 不能省略。

编译 HelloWorld.java 的具体步骤如下。

（1）利用第 3 节介绍的方法进入 dos 命令行窗口。

（2）在命令行窗口，输入“d:”，按 Enter 键转到 D 盘，然后再输入“cd chapter1”，按 Enter 键进入 Java 源程序文件所在目录。

（3）输入命令“javac HelloWorld.java”，按 Enter 键，稍等一会儿，如果没有任何其他信息出现，表示该源程序已经通过了编译。

具体操作过程如图 1-18 所示。

**注意**：如果编译不正确，则给出错误信息，程序员可根据系统提供的错误提示信息修改源代码，直到编译正确为止。

成功编译后，可以在 D:\chapter1 文件夹中看到一个名为 HelloWorld.class 的文件，如图 1-19 所示。

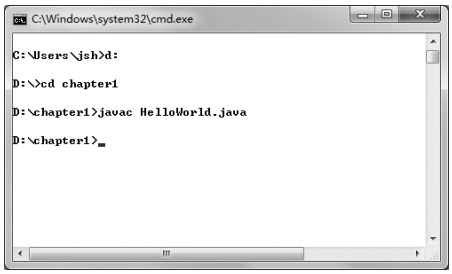


图 1-18　编译程序的命令行窗口


图 1-19　chapter1 文件夹

### 2．解释执行程序

JDK 的解释执行程序是 java.exe，该程序用于执行编译好的 class 文件。它的一般用法如下。

```
java [ 选项…] file [ 参数…]
```

其中，常用选项如下。

-classpath：用于设置路径，在该路径上 javac 寻找需被调用的类。该路径是一个用分号分开的目录列表。

-client：选择客户虚拟机（这是默认值）。

-server：选择服务虚拟机。

-hotspot：与 client 相同。

-verify：对所有代码使用校验。

-noverify：不对代码进行校验。

-verbose：每当类被调用时，向标准输出设备输出信息。

-version：输出版本信息。

初学者只要掌握最简单的用法就可以了。

例如，要执行 HelloWorld.class 文件，只需要在命令行输入如下命令即可：

```
java HelloWorld
```

然后按 Enter 键，稍等一会儿，如果在窗口中出现“hello world！”字符串，说明程序执行成功，执行结果如图 1-20 所示。


图 1-20　程序执行结果

**注意**：java HelloWorld 的作用是让 Java 解释器装载、校验并执行字节码文件 HelloWorld.class，在输入文件名时，大小写必须严格区分，并且文件扩展名 .class 必须省略，否则无法执行该程序。

# 1.5 初次使用 Eclipse

Eclipse 最初是由 IBM 公司开发的用于替代商业软件 Visual Age for Java 的下一代 IDE。2001 年 11 月，IBM 公司将 Eclipse 作为一个开放源代码的项目发布，将其贡献给开源社区。现在它由非营利软件供应商联盟 Eclipse 基金会（Eclipse Foundation）管理。

Eclipse 只是一个框架和一组服务，它通过各种插件来构建开发环境。Eclipse 最初主要用于 Java 语言开发，但现在可以通过安装不同的插件使其支持不同的计算机语言，比如 C++ 和 Python 等。

Eclipse 本身只是一个框架平台，但是众多插件的支持使其拥有其他功能相对固定的 IDE 软件很难具有的灵活性。现在，许多软件开发商以 Eclipse 为框架开发自己的 IDE。

## 1.5.1 Eclipse 下载与安装

读者可以到 Eclipse 的官方网站下载最新版本的 Eclipse 软件，具体步骤如下。

（1）打开浏览器，在地址栏中输入“http://www.eclipse.org/downloads/”，按 Enter 键，进入 Eclipse 官方网站的下载页面，如图 1-21 所示。

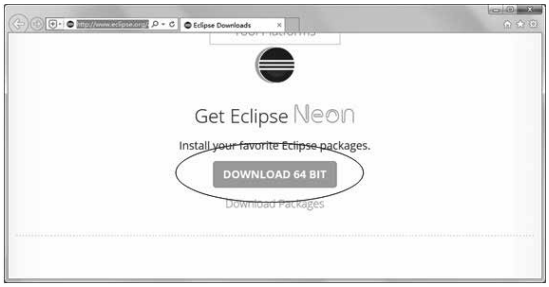


图 1-21 Eclipse 下载页面

（2）单击 DOWNLOAD 64 BIT 按钮，下载页面会根据客户所在的地理位置，分配合理的下载镜像站点，如图 1-22 所示。单击 DOWNLOAD 按钮，开始进行软件下载，读者只需耐心等待下载完成即可。


图 1-22 选择下载镜像站点

（3）下载完成后，在本地计算机中会出现一个 eclipse-inst-win64.exe 可执行文件，如图 1-23 所示。


图 1-23　下载到本地的可执行文件

Eclipse 是基于 Java 的可扩展开发平台，所以读者在安装 Eclipse 前要确保自己的计算机上已安装 JDK。我们的计算机上安装的 JDK 是 64 位的 jdk-8u121-windows-x64 正式版，该 JDK 和已下载的 64 位的 eclipse-inst-win64 是完全兼容的。

Eclipse 的具体安装步骤如下。

（1）双击已下载的 eclipse-inst-win64.exe 可执行文件，打开下载列表窗口，如图 1-24 所示。在下载列表中列出了不同语言的 Eclipse IDE，其中第一个是 Java 开发 IDE，第二个是 Java EE 开发 IDE，第三个是 C/C++ 开发 IDE，第四个是 Java web 开发 IDE，第五个是 PHP 开发 IDE。本书使用的是第一个版本，即 Java 开发 IDE。

（2）单击超链接 Eclipse IDE for Java Developers，进入选择安装路径界面，如图 1-25 所示。选择安装路径，建议不要安装在 C 盘，路径下面的两个选项分别是创建开始菜单和创建桌面快捷方式，读者可根据需要自行选择。

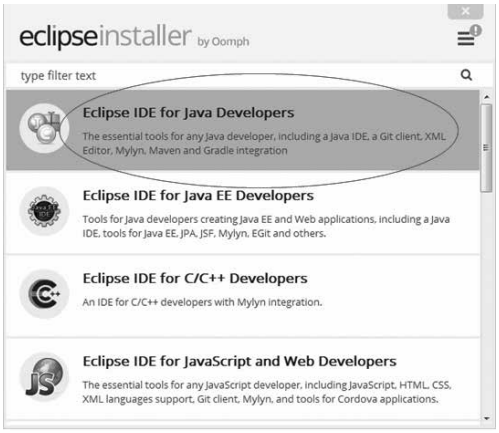


图 1-24　Eclipse 下载列表


图 1-25　选择安装路径

（3）单击 INSTALL 按钮，开始安装 Eclipse，在安装过程中会出现安装协议，如图 1-26 所示。

（4）单击 Accept 按钮，耐心等待安装即可。安装完成后进入图 1-27 所示的界面。

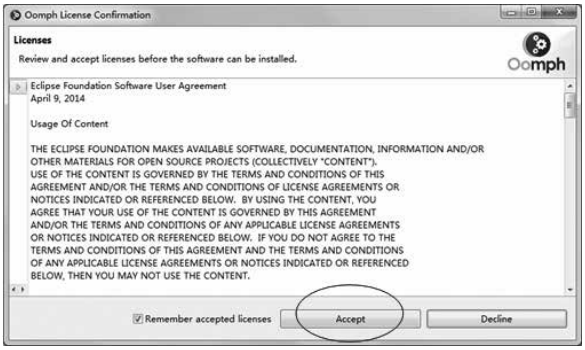


图 1-26　Eclipse 安装协议


图 1-27　Eclipse 安装完成

关闭窗口，安装过程结束。

## 1.5.2　Eclipse 配置与启动

Eclipse 安装结束后，可以按照如下的步骤启动 Eclipse。

（1）在 Eclipse 的安装目录 D:\eclipse_neon\，找到 eclipse 目录下的 eclipse 图标，如图 1-28 所示。


图 1-28　Eclipse 的安装文件夹

（2）双击 eclipse 图标，启动 Eclipse，弹出选择工作空间对话框，如图 1-29 所示。第一次打开 Eclipse，需要设置 Eclipse 的工作空间（用于保存 Eclipse 建立的项目和相关设置），读者可以使用默认的工作空间，或者选择新的工作空间。我们的工作空间是 c:\workspace，并且将其设置为默认工作空间，下次启动时就无须再配置工作空间了。

（3）单击 OK 按钮，即可启动 Eclipse。Eclipse 首次启动时，会显示欢迎页面，其中包括 Eclipse 概述、新增内容、示例、教程、创建新工程、导入工程等相关按钮，如图 1-30 所示。

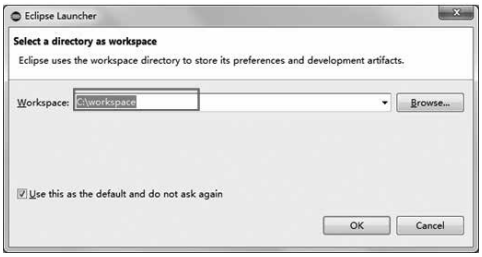


图 1-29　选择工作空间


图 1-30　Eclipse 欢迎界面

（4）关闭欢迎界面，将打开 Eclipse 工作台，如图 1-31 所示。Eclipse 工作台是程序开发人员开发程序的主要场所。

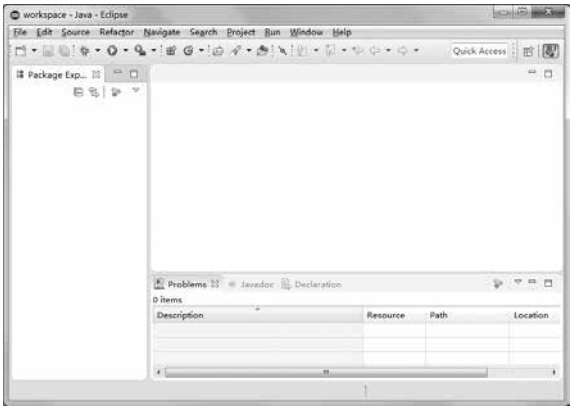


图 1-31　Eclipse 工作台

### 1.5.3　Eclipse 开发 Java 应用程序

此时，开发前的一切准备工作都已经就绪，本节将通过一个实例来与读者一起体验使用 Eclipse 开发 Java 应用程序的便捷。

**1．选择透视图**

透视图是为了定义 Eclipse 在窗口里显示的最初的设计和布局，主要用于控制在菜单和工具上显示什么内容。比如，一个 Java 透视图包括常用的编辑 Java 源程序的视图，而用于调试的透视图则包括调试 Java 程序时要用到的视图。可以转换透视图，但是必须为一个工作区设置好初始的透视图。

打开 Java 透视图的具体步骤如下。

（1）在 Eclipse 菜单栏选择 Window → Perspective → Open Perspective → Other 命令，如图 1-32 所示，弹出“打开透视图”对话框，如图 1-33 所示。

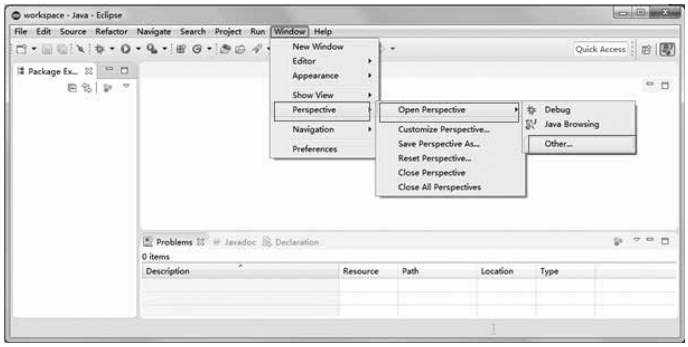


图 1-32　选择 Other 菜单命令


图 1-33　“打开透视图”对话框

（2）选择 Java（default）选项，然后单击 OK 按钮，即可打开 Java 透视图，如图 1-34 所示。

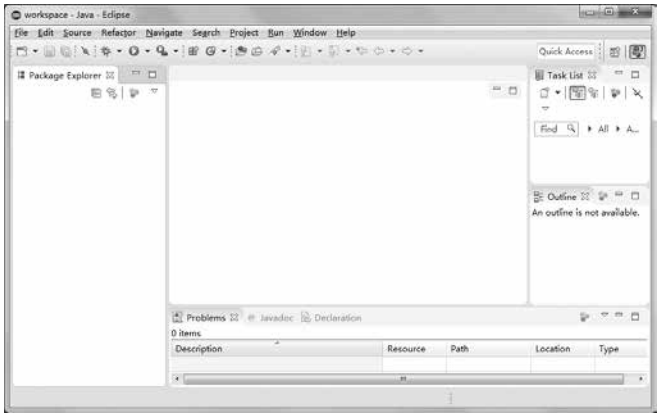


图 1-34　Java 透视图

## 2．新建 Java 项目

通过新建 Java 项目向导可以很容易地创建 Java 项目。

（1）选择 Eclipse 菜单栏中的 File → New → Java Project 命令，如图 1-35 所示。

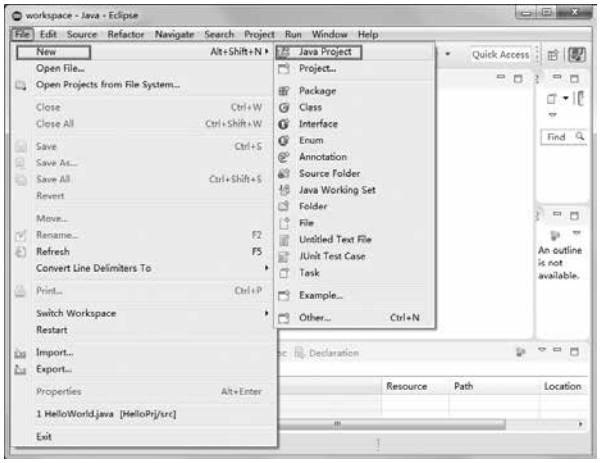


图 1-35　选择新建 Java 项目菜单命令

（2）在打开的“新建 Java 项目”窗口中输入项目名称、选择 JRE 版本和项目布局。通常情况下，读者只需要输入项目名称，其他内容直接采用默认值即可，如图 1-36 所示。

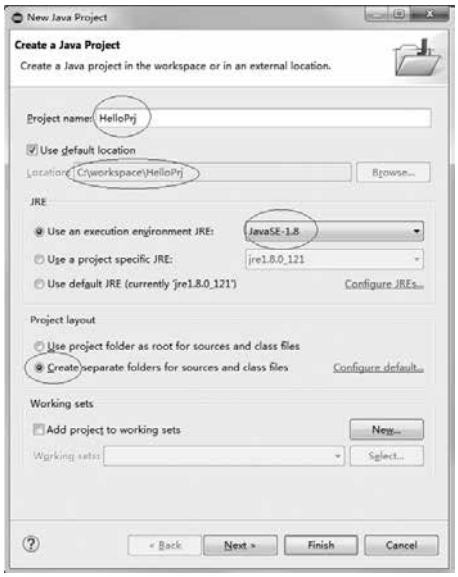


图 1-36　创建 Java 项目窗口

（3）单击 Next 按钮，进入 Java 构建路径设置窗口，在该窗口中可以修改 Java 构建路径等信息。对于初学者而言，可以直接单击 Finish 按钮完成项目的创建，新建项目会自动出现在包浏览器中，如图 1-37 所示。

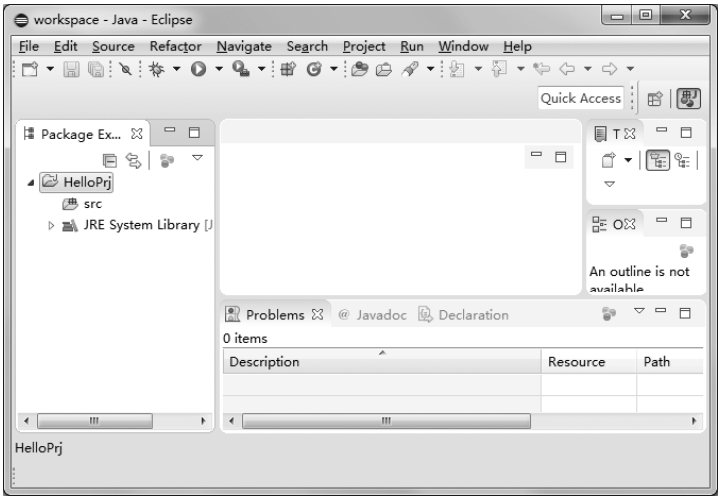


图 1-37　查看新建的 Java 项目 HelloPrj

### 3．编写 Java 代码

创建的 HelloPrj 项目还只是一个空项目，没有实际的源程序。下面就在该项目中建立一个 Java 源程序文件，体验一下在 Eclipse 中编写代码的乐趣。

（1）右击项目名称 HelloPrj，在弹出的快捷菜单中选择 New → Class 命令，如图 1-38 所示。

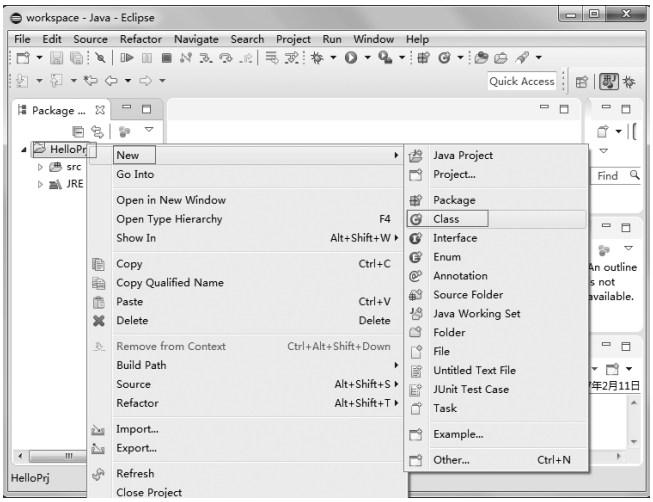


图 1-38　选择 Class 命令

（2）在打开的新建 Java 类窗口中输入包名、类名、修饰符，选择要创建的方法等内容。在这里，我们输入了类名，并选择了创建 main 方法，具体情况如图 1-39 所示。


图 1-39　新建 Java 类窗口

（3）单击 Finish 按钮，系统将创建一个 Java 文件 HelloWorld.java，如图 1-40 所示。

（4）编辑 Java 源程序文件。在源程序的 main 方法中添加下面的语句：

```
System.out.println("Hello World ！ ");
```

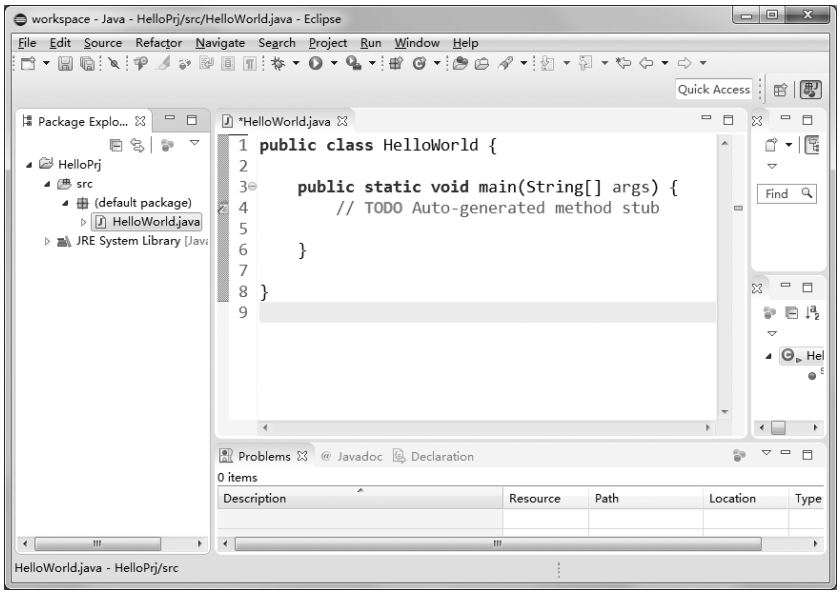


图 1-40　编辑 Java 类

### 4．编译和执行程序

Eclipse 会自动编译 Java 源程序，如果源程序有错误，Eclipse 会自动给出相应的提示信息。

运行程序前要确保程序已经成功编译。

右击要执行的程序，在弹出的快捷菜单中选择 Run As → Java Application 命令，如图 1-41 所示。稍后即可在下方的控制台中可以看到程序的执行结果，如图 1-42 所示。

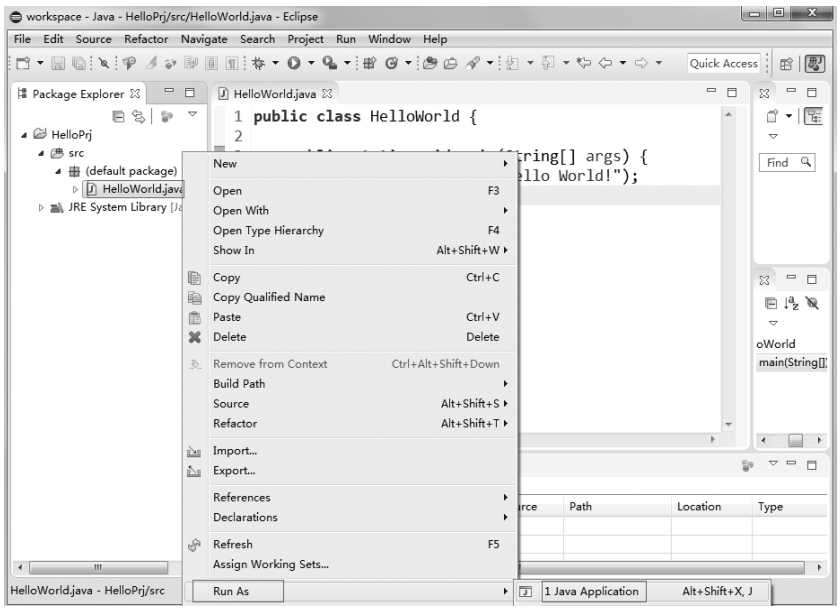


图 1-41　执行程序

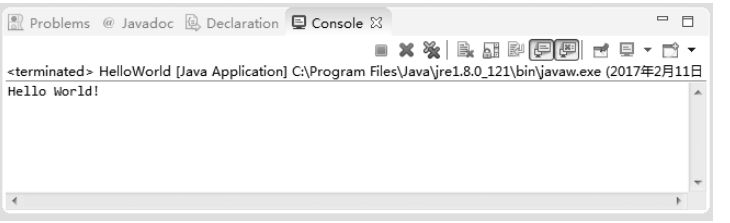


图 1-42　程序执行结果

### 5．调试程序

Eclipse 还集成了程序调试工具，开发者不用离开集成开发环境就能通过 Eclipse 调试器找到程序的错误。

Eclipse 调试器提供了断点设置的功能，可以一行一行地执行程序。在程序执行过程中，可以查看变量的值，研究哪个方法被调用了，并且知道程序将要发生什么事件。

下面通过一个简单的例子介绍如何使用 Eclipse 调试器来调试程序。

```
public class DebugTest {
    public static void main(String[] args) {
            int sum =0;
            for(int i=1;i<=5;i++){
                sum=sum +i;
            }
            System.out.println(sum);
    }
}
```

上述代码的核心功能是：计算 1 ～ 5 的所有整数之和，并输出计算结果，即 sum 的值。但对于初学者来说，可能对 sum 值的变化过程不是非常了解，接下来通过 Eclipse 调试器来了解 sum 的变化过程。

（1）设置断点。双击要插入断点的语句前面的蓝色区域，这时该行最前面会出现一个蓝色的圆点，这就是断点，如图 1-43 所示。如果要取消该断点，直接双击断点处即可。

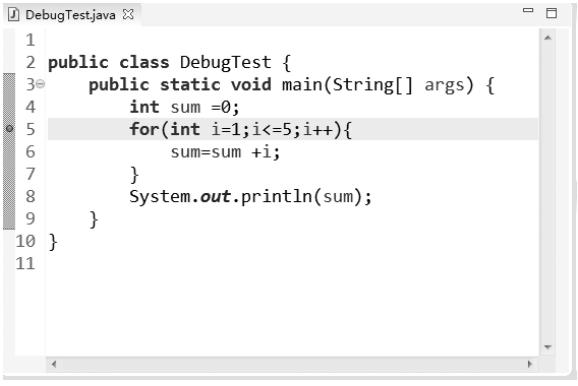


图 1-43　设置断点

（2）调试程序。右击要调试的程序，在弹出的快捷菜单中选择 Debug As → Java Application 命令，如图 1-44 所示。

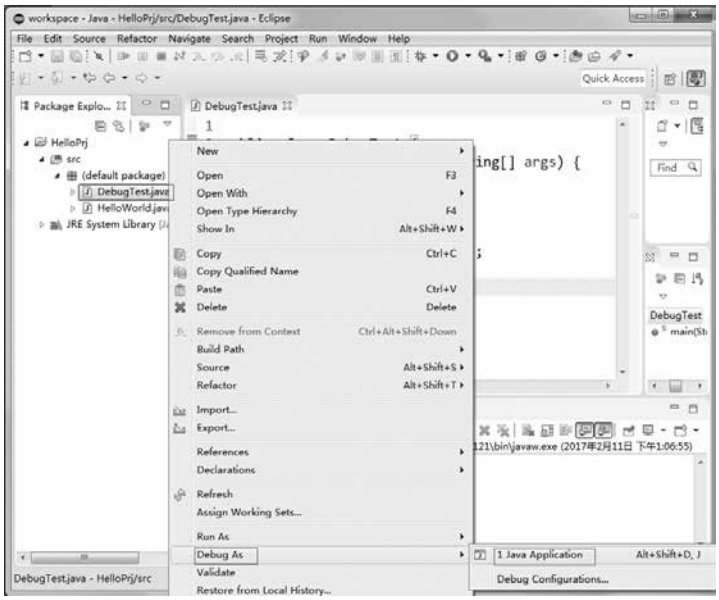


图 1-44　调试程序

（3）程序开始执行，执行到断点位置，弹出图 1-45 所示的对话框，单击 Yes 按钮进入 Debug 透视图模式，如图 1-46 所示。从图中可以发现，设置了断点的语句已经被绿色光带覆盖。

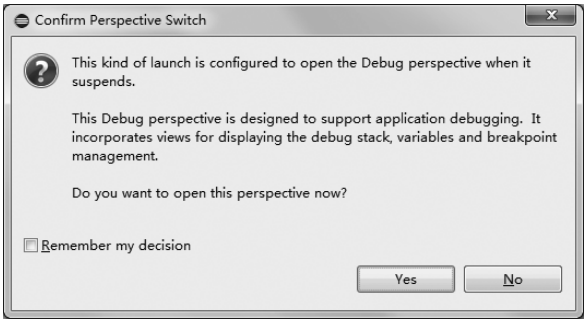


图 1-45　提示信息

（4）逐行执行代码。选择 Run → step Over 菜单命令或者直接按 F6 键，程序开始单步执行，这时可以看到 Variables 窗口中 sum 的值是 0。继续执行程序，这时会发现重新回到了 for 循环开始的位置，准备下一次的执行了。

（5）继续执行程序，sum 的值变成 1，且 Variables 窗口中 sum 所在行被黄色光带覆盖，如图 1-47 所示。

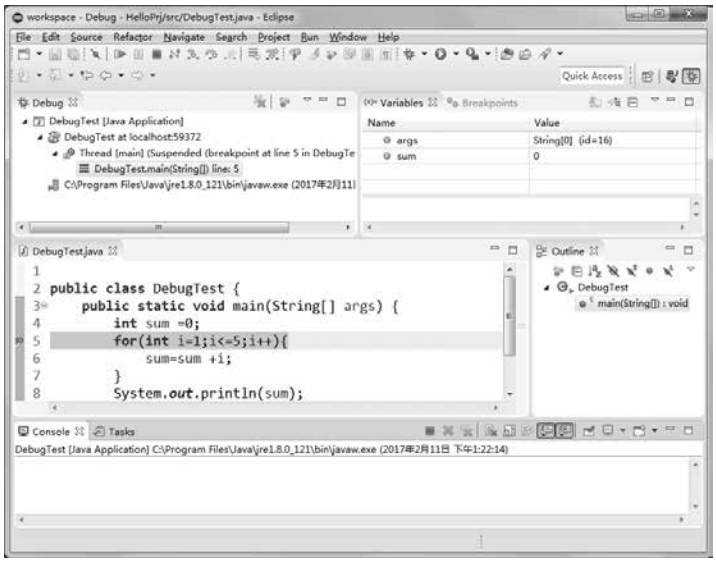


图 1-46　Debug 透视图

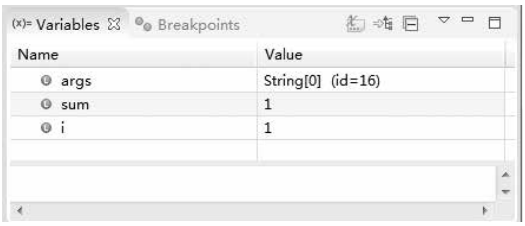


图 1-47　Variables 透视图

（6）继续按F6键，程序继续执行，直到执行完毕。在此过程中，sum的值从1依次变成3、6、10、15，程序执行结束，并在控制台输出 sum 的值 15，如图 1-48 所示。

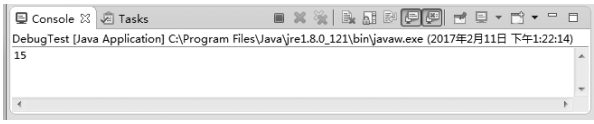


图 1-48　程序执行结果

**注意**：Eclipse 调试器是一个不可缺少的、功能强大的工具，它可以帮助读者快速提高自己的编程水平。一开始读者需要花费一些时间去熟悉它，但是你的努力会在将来得到很好的回报。

## 强化练习

通过对本章内容的学习，能对 Java 语言的发展历史、编译、运行等有大概的了解。刚开始编译运行程序时，可能会出现较多的错误，其中大多数是由环境变量配置错误造成的，所以需要读者熟练掌握环境变量的配置方法。通过以下练习可以更好地帮助读者打牢基础。

### 练习 1：

到 Oracle 公司的官方网站上下载最新的 JDK 安装文件，运行该文件，建立 Java 应用程序的开发运行环境，编辑系统环境变量 path 的值，使系统在任何目录下都能识别 javac 指令。

### 练习 2：

利用记事本编写一个简单的 Java 应用程序，并用命令行方式对其进行编译和运行。

### 练习 3：

从官方网站上下载 Eclipse 的安装文件，运行该文件并在本机上安装 Eclipse 软件。

### 练习 4：

利用 Eclipse 开发环境，编写一个简单的 Java 应用程序，并用 Eclipse 调试器调试程序，观察程序执行过程。