Java

第一部分　实验指导

第1章

Java 语言概述

本章知识点

Java 语言是一种跨平台、适合于分布式计算环境的面向对象编程语言。Java 开发工具（Java Development Kit，JDK）可在 Oracle 公司的网站免费取得，它与 JDK 的参考文件（Java docs）同样是编写 Java 程序必备的工具。

本章将指导读者在计算机上安装和配置 JDK 的运行环境，了解 Java 应用程序的编辑和运行过程。

实验 1.1　Java 语言开发环境的配置

1. 实验目的

（1）学习下载并安装 JDK。

（2）学习设置系统变量 Path 和 ClassPath。

（3）解决 JDK 开发环境配置中的常见问题。

2. 实验指导

步骤 1：下载 JDK 安装文件。进入 Oracle 公司 Java SE 的下载页面 https://www.oracle.com/java/technologies/downloads/#jdk17-windows，下载 JDK 安装文件。JDK 按安装方法的不同，可分为 .zip 格式的压缩文件、.exe 格式的安装文件和 .msi 格式的安装文件，根据需要可以下载不同安装文件。这里下载得到的是 JDK 压缩包文件 jdk-17_windows-x64_bin.zip，如图 1.1 所示。

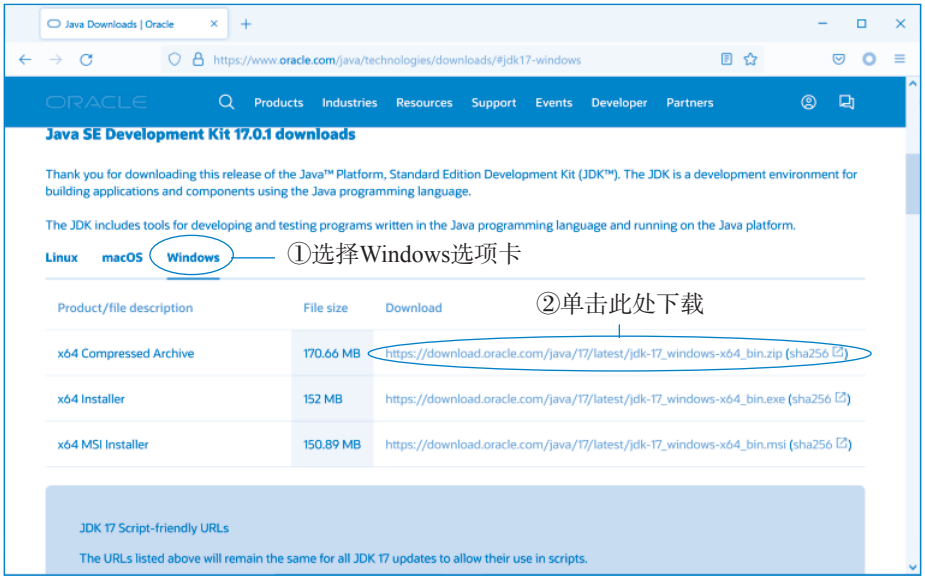

图 1.1　下载 JDK 压缩包文件

步骤 2：安装 JDK。在 C 盘的根目录下新建一个文件夹，命名为 jdk。将 JDK 压缩包文件 jdk-17_windows-x64_bin.zip 解压到文件夹 C:\jdk 中，称 C:\jdk 为安装路径。

步骤 3：配置 JDK 运行环境。因为是利用压缩包文件安装的 JDK，所以需要人工配置运行环境。在 Windows 11 操作系统的任务栏中单击“文件资源管理器”按钮，在弹出窗口的左窗格中右击“此电脑”选项，在弹出的快捷菜单中选择“属性”命令。然后在弹出的“关于”窗口中选择“相关链接”组中的“高级系统设置”选项，弹出“系统属性”对话框。在该对话框中选择“高级”选项卡，单击“环境变量”按钮，弹出“环境变量”对话框。在该对话框中单击“系统变量”选项区域下面的“新建”按钮，添加系统变量 Java_Home，在弹出的“新建系统变量”对话框中的“变量名”文本框中输入 Java_Home，在“变量值”文本框中输入 C:\jdk，该值就是 JDK 的安装路径。单击“确定”按钮，返回“环境变量”对话框，然后在该对话框的“系统变量”选项区域中选择 Path 选项，单击“编辑”按钮（因为系统变量 Path 在 Windows 安装后就已经生成，所以不需要新建 Path，只需编辑为其添加新值即可），弹出“编辑系统变量”对话框。单击“新建”按钮，将 %Java_Home%\bin 添加到当前值中，单击“确定”按钮，如图 1.2 所示。由于系统变量 Java_Home 的值设置为 C:\jdk，因此可以用 %Java_Home% 代替 C:\jdk，所以为 Path 添加的新值就是 C:\jdk\bin。

说明：如果不想创建系统变量 Java_Home，则必须将“C:\jdk\bin;”添加到已存在的 Path 路径值的最前面。设置系统变量 Java_Home 的好处是便于维护系统变量 Path。

步骤 4：检验 JDK 设置是否成功。在 Windows 11 桌面的任务栏上单击“搜索”按钮，在弹出的窗格上面的搜索栏“在此键入进行搜索”命令框中输入 cmd 并按 Enter 键（或按

Win+R 组合键打开“运行”对话框，输入命令 cmd），弹出命令行窗口，在该窗口中输入 javac 并按 Enter 键，若输出如图 1.3 所示的内容，则表示 JDK 安装和配置成功；若输出其他内容，则表示不成功。这时，首先检查 JDK 是否安装在 C 盘的 jdk 目录下，再检查路径系统变量 Path 的值是否包含有 %Java_Home%\bin。Path 路径修改后，需要把命令行窗口先关闭再打开，Path 路径修改才能生效。

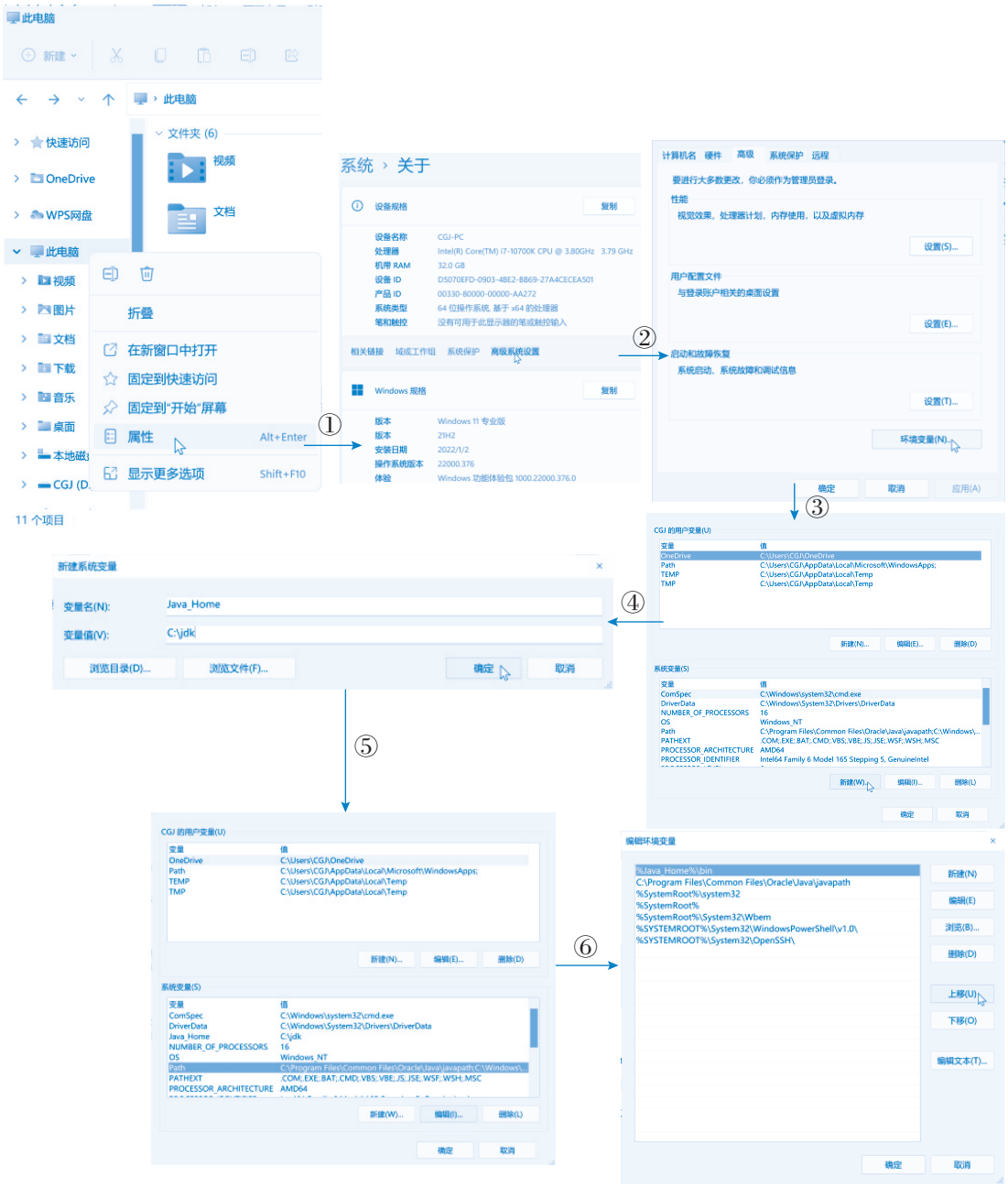

图 1.2　设置 JDK 运行路径

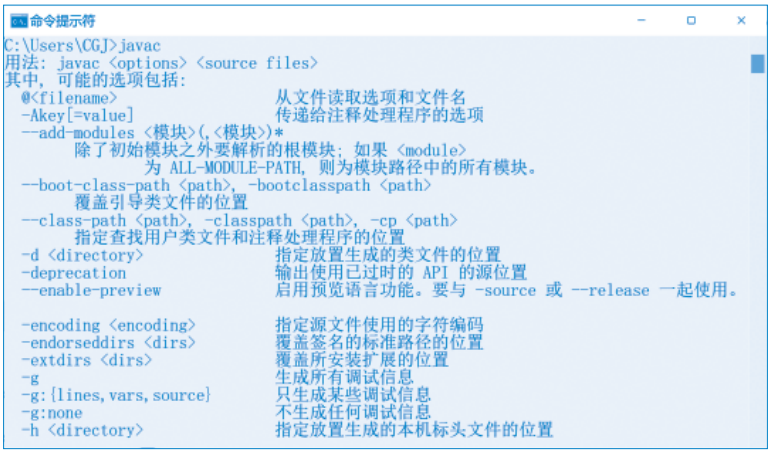

图 1.3　JDK 路径设置成功

步骤 5：设置系统变量 ClassPath。在图 1.4 所示的“新建系统变量”对话框中的“变量名”文本框中输入 ClassPath，同时在“变量值”文本框中输入 Java 类库所在的路径值“.;%Java_Home%\lib”。

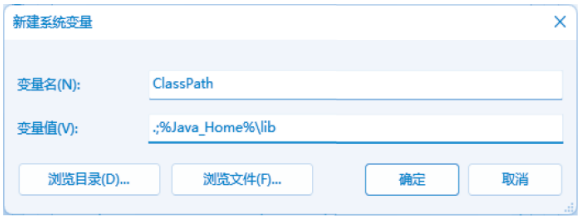

图 1.4　创建类路径系统变量 ClassPath

若想在命令行窗口查看、删除或设置系统变量 Path 和 ClassPath 的值，可使用如下方法。

`set path` 或 `echo %path%`

`set classpath` 或 `echo %classpath%`

实验 1.2　JDK 参考文档的使用

1. 实验目的

（1）学习下载 JDK 参考文档。

（2）学习使用 JDK 参考文档。

（3）了解 JDK 参考文档的组成。

2. 实验指导

步骤 1：通过浏览器访问 Oracle 公司 Java SE 的下载页面。进入 Java SE 的下载页面 https://www.oracle.com/java/technologies/javase-jdk17-doc-downloads.html，下载 JDK 参考文档，如图 1.5 所示。

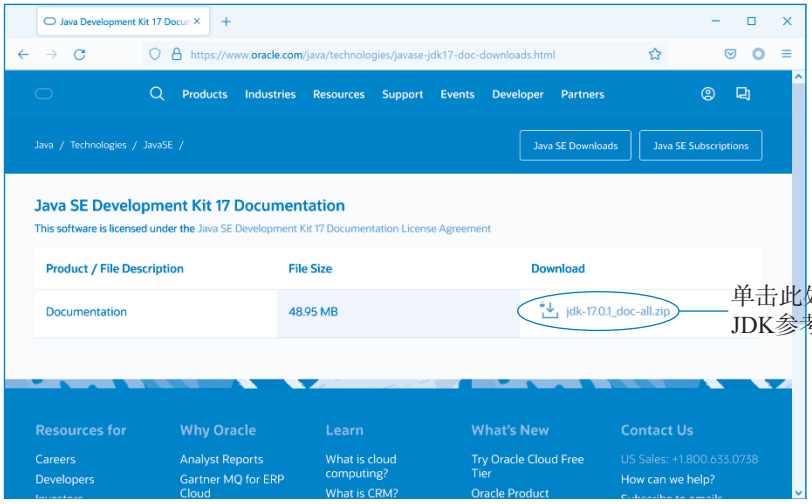

图 1.5　JDK 参考文档下载页面

步骤 2：下载得到的 JDK 参考文档是一个名为 jdk-17.0.1_doc-all.zip 的压缩包文件。

步骤 3：将该文件解压到 C:\jdk\docs 目录下后，在该目录下有一个 index.html 文件，用浏览器打开该文件，出现 JDK 参考文档的首页，如图 1.6 所示。

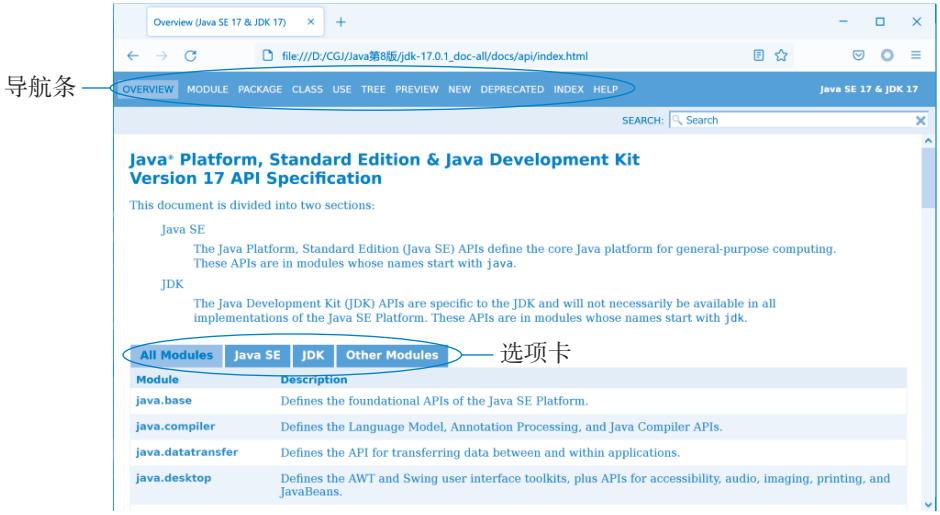

图 1.6　JDK 参考文档的首页

该文档的顶部为导航条，分别对应 OVERVIEW、MODULE、PACKAGE、CLASS、USE、TREE、PREVIEW、NEW、DEPRECATED、INDEX、HELP 页面，下面简单介绍各个页面的作用。

（1）OVERVIEW 页面。OVERVIEW 页面描述了该文档所有模块被分成 Java SE（Java 平台标准版）和 JDK（Java 开发工具包）两部分并对其简介。

Java SE：Java SE API 定义了通用计算的核心 Java 平台。这些 API 在模块中，名称以

java 开头。

JDK：JDK API 是特定于 JDK 的，并且在 Java SE 平台的所有实现中不一定都可以使用。这些 API 在模块中，名称以 jdk 开头。

（2）MODULE 页面。当选择一个模块后，可在 MODULE 页面中显示该模块的描述。Java 17 中的模块是代码、数据和有些资源自描述的集合。JDK 系统较为庞大，因此 Java 17 将 JDK 划分成小模块。这样可以方便开发者使用任何想要的模块，易将 Java 应用程序缩减到小设备。

（3）PACKAGE 页面。在 PACKAGE 页面中提供了对所选包的描述，包括包的类型、相关包和层次结构等，并列出该包中所包含的接口和类，同时在这些内容的旁边给出概要说明。

（4）CLASS 页面。在 CLASS 页面中提供了对所选类的描述，并列出该类中包含的属性和方法，同时在这些内容的旁边给出概要说明。例如选择 java.lang 包，然后选中 System 类，显示如图 1.7 所示的页面。从图 1.7 中可以看出 System 类继承了 Object 类，它的完整定义为

```
public final class System extends Object
```

图 1.7　JDK 参考文档的 CLASS 页面

System 类为最终类，不能再派生子类。在 System 类的页面中，包括对 System 类的概要说明。

System 类里面包含了哪些属性和哪些方法，通过拖动界面中的滚动条可以详细了解。如看到 System 类有 err、in、out 三个静态属性，分别是标准错误输出、标准输入和标准输出。

（5）USE 页面。该页面说明哪些包、类、方法使用了该类。System 类的 USE 页面如图 1.8 所示。由此页面看到没有其他类使用了 System 类。

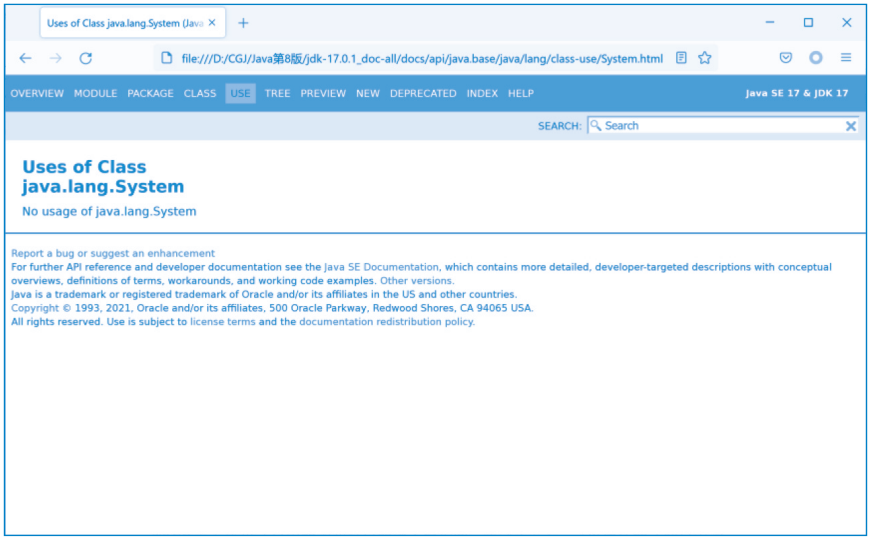

图 1.8　System 类的 USE 页面

（6）TREE 页面。该页面给出了包中的类和接口的继承层次图，通过该页面，可以直观了解包中类和接口的继承关系。

（7）PREVIEW 页面。该页面指出该页面中所列出的接口、类、方法和枚举在未来可能会被删除，如图 1.9 所示。

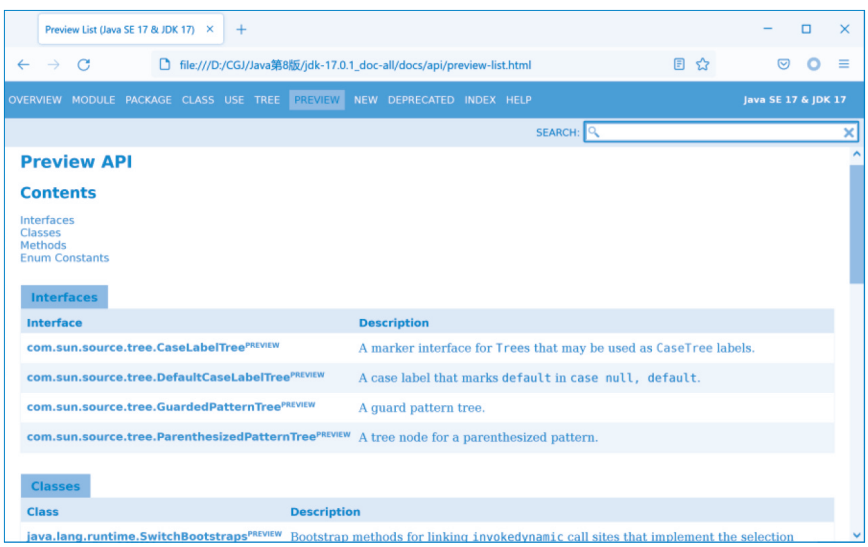

图 1.9　JDK 的 PREVIEW 页面

（8）NEW 页面。该页面中显示了若干选项卡，如图 1.10 所示。最左边的选项卡 New Modules 表示不区分版本添加的新元素，即所有这些新元素，无论它们是在哪个版本中添加的都显示在该选项卡中。其他选项卡中的 Added in 表示在特定版本中添加的新元素。任何元素显示在最左侧选项卡下同时也显示在右侧某选项卡之下。

图 1.10　JDK 的 NEW 页面

（9）DEPRECATED 页面。该页面列出了所有已过时的类、接口、方法等元素。一般是有缺点，通常会给出替换 API。在下面显示的若干选项卡中，最左侧的选项卡给出了已不再使用的元素，而其他选项卡则指出是在哪个 JDK 版本中被弃用的。

（10）INDEX 页面。该页面按照字母顺序列出了 JDK 中所有的类、接口、方法、属性等内容，如图 1.11 所示，通过该页面可以在 JDK 参考文档中快速查找到需要了解的细节。

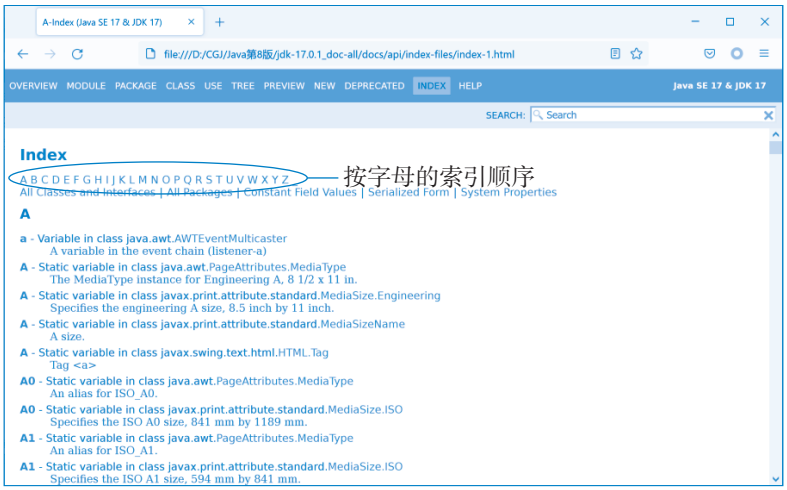

图 1.11　JDK 的 INDEX 页面

例如，查看 println() 方法的细节，通过该页面可以找到 println() 方法，如图 1.12 所示。由此可知 println() 方法是定义在 PrintStream 类中的，并且有许多重载的方法，再单击 PrintStream 类，就可以找到 println() 方法的定义。

（11）HELP 页面。该页面是 JDK 参考文档的帮助页面，概要描述了文档中各个页面的内容。

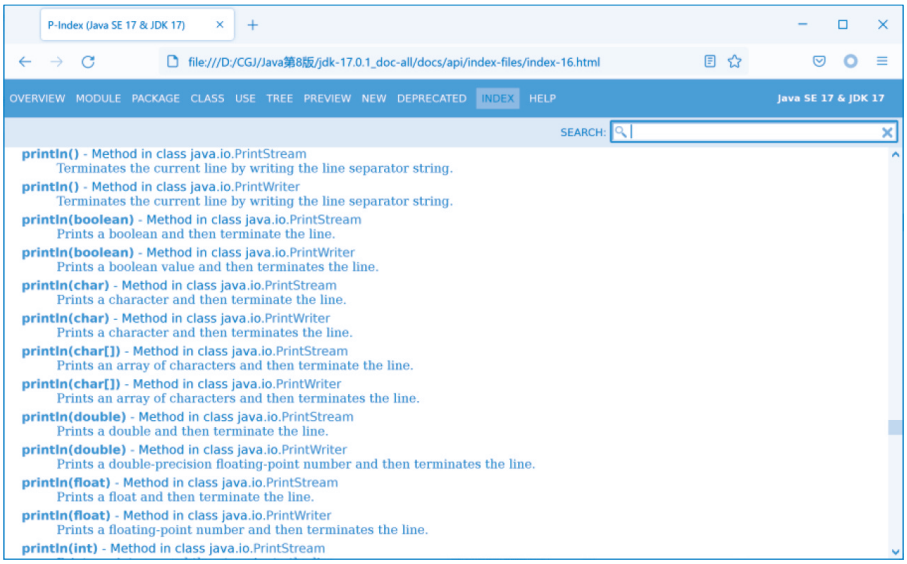

图 1.12　println() 方法的 INDEX 页面

实验 1.3　Eclipse 集成开发环境

1. 实验目的

（1）学习下载并安装 Eclipse。

（2）学习使用 Eclipse 创建项目。

（3）学习使用 Eclipse 环境创建 Java 应用程序。

（4）学习在 Eclipse 环境下调试 Java 应用程序。

2. 实验说明

Eclipse 集成开发环境（Eclipse IDE）虽然是一个很优秀的集成开发工具，但还是建议初学者直接使用 Java SE 提供的 JDK，因为无论哪种集成开发环境都将 JDK 作为核心，而且集成开发环境界面操作复杂，会屏蔽掉一些知识点，不利于初学者掌握基础知识。本实验之所以介绍 Eclipse 集成开发环境，是应某些已经掌握了 Java 基础知识的读者的要求介绍 Eclipse 集成开发环境的用法。

3. 实验指导

Eclipse 是 IBM 公司“日食”计划的产物。2001 年 6 月，IBM 公司将价值 4000 万美元的 Eclipse 捐给了开源组织。Eclipse 是一个免费的、开放源代码的、著名的跨平台的集成开发环境。经过发展，Eclipse 是目前最流行的 Java 集成开发环境，已成为业界的工业标准。要想获得 Eclipse，进入 Eclipse 官方网站 http://www.eclipse.org 可以下载最新版本的 Eclipse 安装文件。本书下载得到的安装文件为 eclipse-inst-win64.exe。

（1）安装 Eclipse IDE。

双击 Eclipse 安装文件 eclipse-inst-win64.exe，进入如图 1.13 所示的安装界面。在该界面中根据需要进行选择安装，这里选择第一项，然后开始安装。

图 1.13　Eclipse IDE 安装界面

（2）启动 Eclipse IDE。

安装完后，启动 Eclipse IDE。第一次运行 Eclipse IDE，启动向导会弹出如图 1.14 所示的让用户选择 Workspace（工作区）的界面。所谓 Workspace 就是所有 Eclipse 项目的工作目录，表示接下来的代码和项目设置都将保存到该工作目录下。这里输入的是 D:\CgjWS，若该目录不存在则系统会自动生成，当然也可以单击 Browse 按钮，在弹出的对话框中选择已存在的目录。

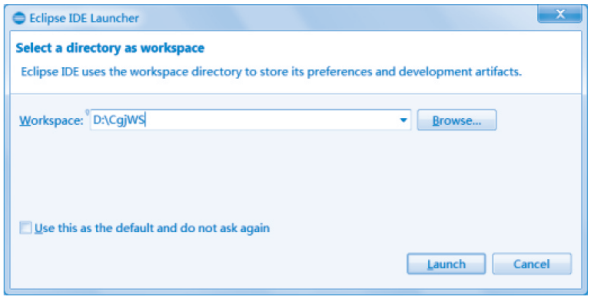

图 1.14　工作区目录设置

设置完工作区后，单击 Launch 按钮，即可进入如图 1.15 所示的欢迎界面。关闭欢迎界面的选项卡后就进入如图 1.16 所示的开发环境布局界面。

图 1.15　Eclipse 的欢迎界面

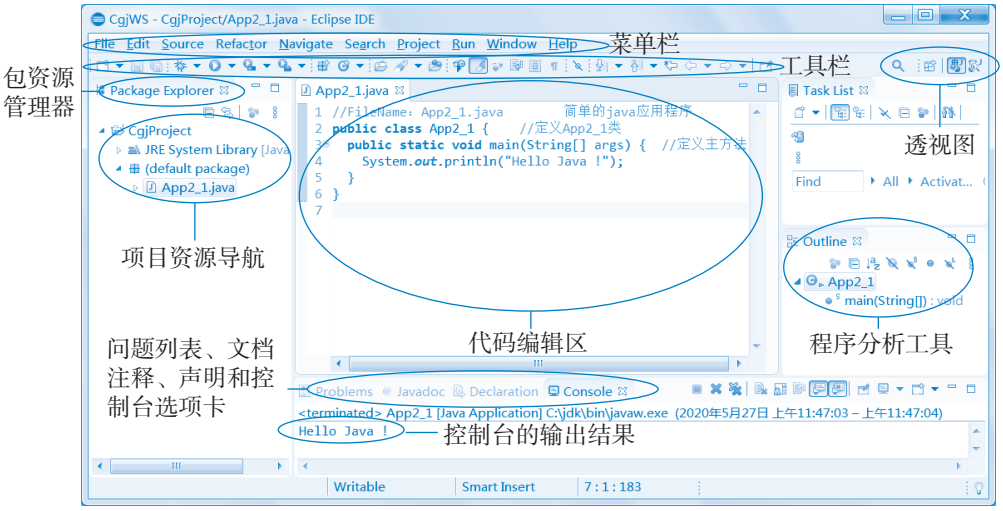

图 1.16　Eclipse 开发环境布局界面

Eclipse 的开发环境包含如下几部分。

- 顶部为菜单栏、工具栏。
- 右上角为 IDE 的透视图，Java 透视图是 Eclipse 专门为 Java 项目设置的开发环境布局，用于切换 Eclipse 不同的外观。通常根据开发项目的需要切换不同的视图。
- 左侧窗格为项目资源导航，主要有包资源管理器。
- 右侧窗格为程序分析工具，主要有大纲、任务列表等。
- 底部为显示区域，主要有问题列表、文档注释、声明和控制台选项卡等。
- 中间区域为代码编辑区。

使用 Eclipse 集成开发环境常用的操作集中在新建 Java 项目、新建类、编译与运行 Java 程序、调试程序这几方面，下面针对这些常用操作进行简单介绍。

（3）新建 Java 项目。

创建 Java 程序前，首先需要创建一个项目。项目类似一个文件夹，用于包含 Java 程序以及所有支持的文件，项目是开发特定应用软件所需的源文件、库文件等多个文件的集合。项目只需创建一次。

选择 File → New → Java Project 选项，弹出如图 1.17 所示的新建 Java 项目对话框。

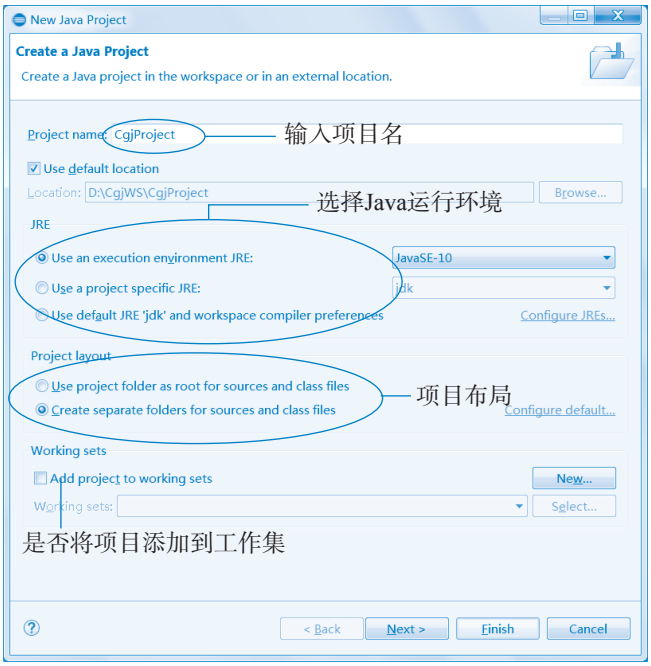

图 1.17　新建 Java 项目对话框

在 Project name 文本框中输入项目名，本例输入的项目名是 CgjProject。Location 文本框自动设置默认保存位置。也可以单击 Browse 按钮为项目自定义保存位置。

在选择 Java 运行环境选项区域中，选择第一项 Use an execution environment JRE（使用执行环境）即可。

在项目布局选项区域中，若选择第一项，则将 .java 文件和 .class 文件均放在项目文件夹 CgjProject 下，方便访问。若选择第二项，则会在项目文件夹 CgjProject 下生成 scr 和 bin 两个子文件夹，scr 用于存放 .java 源文件；bin 用于存放 .class 类文件。这里选择了第二项。

单击 Next 按钮，进入如图 1.18 所示的 Java 构建设置对话框。该对话框中的设置都取默认值，其中在源码选项卡 Source 中可以看到源文件程序保存在 CgjProject/scr 文件夹下，默认输出的 .class 类文件保存在 CgjProject/bin 文件夹下。单击 Finish 按钮，弹出如图 1.19 所示的询问是否创建模块对话框，可以根据需要进行选择，这里选择不创建，所以单击 Don’t Create 按钮。完成项目的创建后，返回如图 1.16 所示的开发环境窗口中。

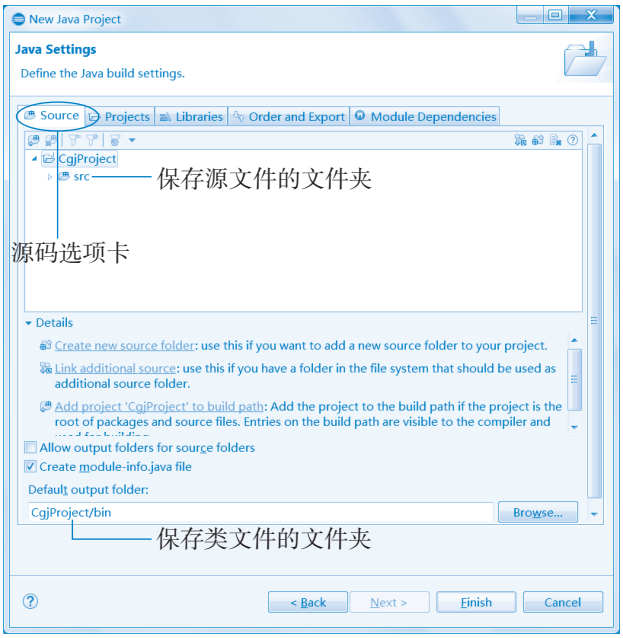

图 1.18　Java 构建设置对话框

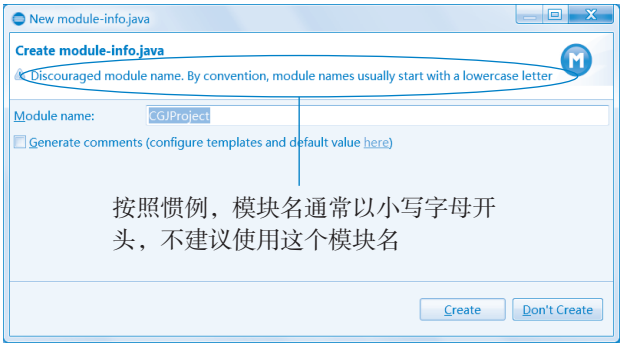

图 1.19　询问是否创建模块对话框

（4）新建类。

项目创建完成后，就可以创建类文件。选择 File → New → Class 选项，弹出如图 1.20 所示的新建 Java 类对话框。

在该对话框的源文件夹项 Source folder 文本框中输入保存源文件的文件夹，这里取默认值 CgjProject/scr 即可。在文件所在包 Package 文本框中输入 chapter2，表示第 2 章程序所在的包名。在类名 Name 文本框中输入 App2_1。修饰符 Modifiers 中根据需要进行选择，这里选择 public 单选按钮。在父类 Superclass 文本框中根据需要选择或输入其父类。在 Which method stubs would you like to create选项区域选择创建哪些方法的存根，选中 public static void main(String[] args) 复选框，Eclipse 将自动生成 main() 方法。单击 Finish 按钮，新类 App2_1.java 创建完毕，然后弹出如图 1.21 所示的 Eclipse 开发环境窗口。

图 1.20　新建 Java 类对话框

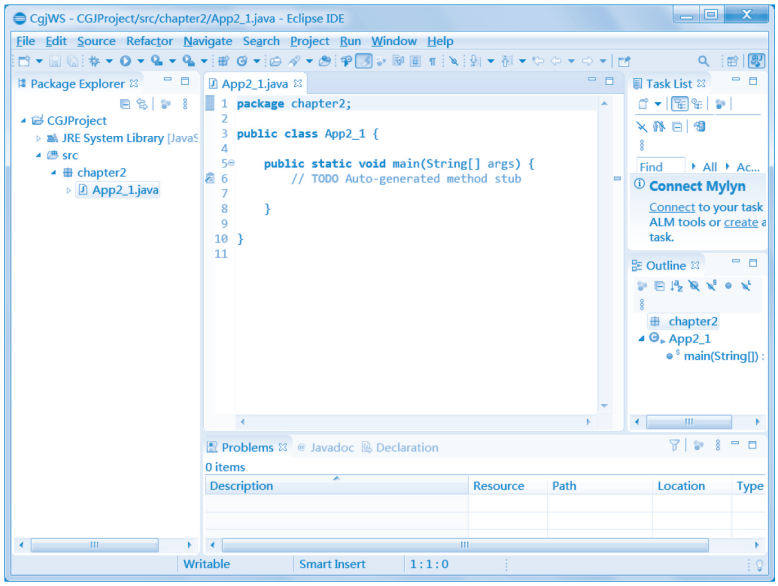

图 1.21　Eclipse 开发环境窗口

说明：存根程序 Stub 其实就是通常所说的桩模块。软件的集成方式分为两种：自顶向下和自底向上。第一种集成方式需要开发桩程序，第二种集成方式需要开发驱动程序。桩程序用来给调用它们的模块传递数据。

（5）编辑与运行 Java 程序。

在该窗口的代码编辑区中输入如图 1.22 所示的程序代码，然后选择 Run → Run As 选项即可编译、运行一步到位，如图 1.22 所示。

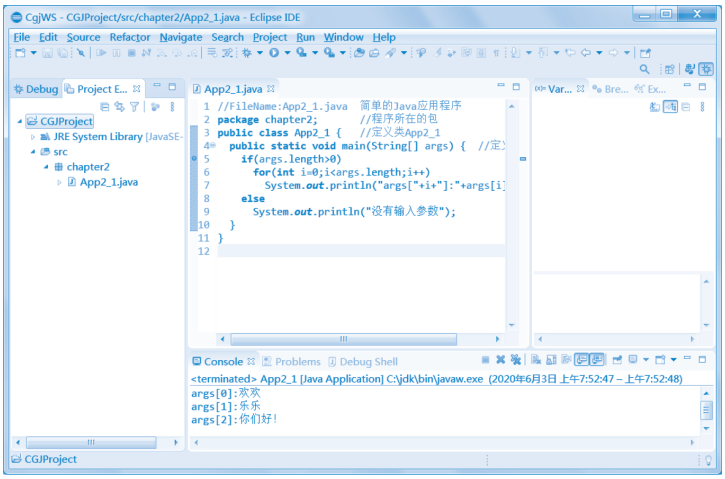

图 1.22　编辑与运行 App2_1.java 程序

如果在 Eclipse 集成开发环境中运行 main() 方法需要参数的程序，如 App2_1.java，则 main() 方法的参数需要在运行前在界面中设置。方法如下，首先选中 App2_1.java 文件，然后选择 Run → Run Configurations 选项，弹出如图 1.23 所示的 Run Configurations 窗口。选择 Arguments 选项卡，然后在 Program arguments 选项区域中输入参数，这里输入的是“欢欢 乐乐 你们好！”三个参数，然后单击 Run 按钮，即可在如图 1.22 所示的控制台中看到输出结果。

图 1.23　在 Eclipse 开发环境中提供命令行参数

（6）调试 Java 程序。

在 Eclipse 的代码编辑区双击需要设置断点的行的左侧边框，会出现一个蓝色的断点标识，如图 1.24 所示。

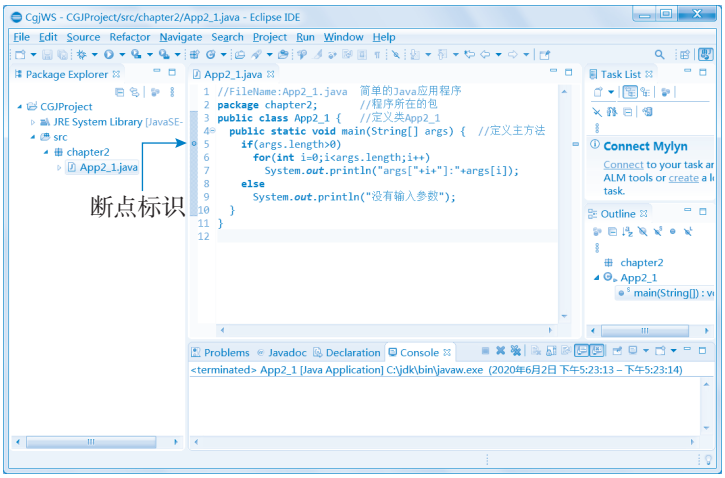

图 1.24　在编辑区边框双击设置断点标识

单击工具栏上的“调试”按钮，然后在下拉菜单中选择 Debug As 选项，选择要调试的程序，则弹出如图 1.25 所示的询问是否切换到 Debug 透视图对话框，单击 Switch 按钮，进入如图 1.26 所示的程序调试界面。单击工具栏中的或按钮，观察 Variables 窗格中的局部变量的变化，以及输出的变化，对代码进行调试并运行。

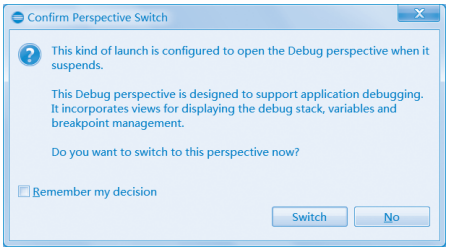

图 1.25　询问是否切换到 Debug 透视图对话框

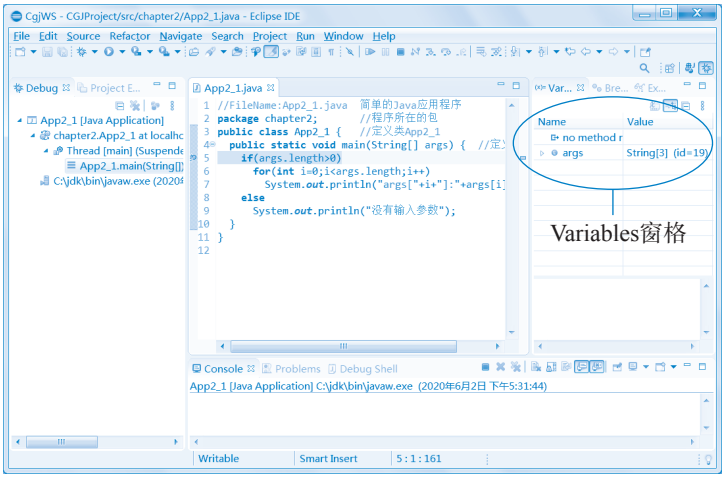

图 1.26　Eclipse 的程序调试界面