模块 1

Java 初步

知识目标

1. 了解 Java 编程开发的基本条件和 Java 程序的运行流程。
2. 掌握 Java 文件的命名原则。
3. 掌握 Java 应用程序的开发过程和简单 Java 应用程序的编写与运行方法。

技能目标

1. 能够在 MyEclipse IDE(integrated development environment,集成开发环境)中配置 JRE。
2. 能够在 MyEclipse IDE 中创建 Java 项目。
3. 能够在 MyEclipse IDE 中编写简单 Java 程序并运行 Java 程序。

1.1 Java 基础知识

Java 自 20 世纪 90 年代起随互联网的兴盛而诞生、发展、壮大。Java 既是一门开发跨平台应用的面向对象程序设计语言,同时又是一种技术标准和体系。Java 具有简单性、面向对象、分布式、健壮性、安全性、平台独立与可移植性、多线程、动态性等特点。可以用 Java 编写桌面应用程序、Web 应用程序、分布式系统和嵌入式系统应用程序等。目前,Java 广泛应用于个人计算机、数据中心、移动通信、金融服务业、电子政务、电子商务、科学计算等诸多领域和方向。在大数据、云计算及人工智能化时代,Java 占据着显著的优势并拥有广阔的前景。

1.1.1 Java 编程开发简介

1. Java 版本

Java 技术标准和体系发展到今天，形成了 3 个独立的版本。它们分别是：Java SE(Java standard edition，Java 标准版，应用于桌面环境)、Java EE(Java enterprise edition，Java 企业版，应用于基于 Java 的应用服务器)和 Java ME(Java micro edition，Java 简化版，应用于移动、无线及有限资源的环境)，代表了桌面应用开发、企业应用开发和移动应用开发 3 个相互区别又紧密联系的开发领域。

2. Java 编程环境

Java 编程开发离不开 JDK(Java development kit，Java 开发包)和 JRE(Java runtime environment ，Java 运行时环境)。JDK 是开发 Java 程序的核心，包含了 JRE、Java 常用工具(如 Java 编译器、Java 解释器、Java 文档生成器等)和 Java 核心类库。JRE 是支持 Java 程序运行的标准环境，包含了 Java SE 核心类库和 JVM(Java virtual machine，Java 虚拟机)。

因为 JRE 是一个运行环境，而 JDK 是一个开发环境，所以编写 Java 程序时需要 JDK，而运行 Java 程序时就需要 JRE。而 JDK 里面已经包含了 JRE，因此只要安装了 JDK，就可以编写 Java 程序，也可以正常运行 Java 程序。但是由于 JDK 包含了许多与运行无关的内容，占用的空间较大，因此，如果只运行普通的 Java 程序，就无须安装 JDK，安装 JRE 即可。

1.1.2 Java 程序运行流程

编写好的 Java 源代码文件(扩展名为“.Java”，也可称为源程序文件或简称为源文件)，经 Java 编译器编译后，生成与平台无关的字节码文件(扩展名为“.class”)，这些字节码能够被安装在不同计算机上的 Java 虚拟机识别，因此，也可称之为“虚拟机代码”。当字节码被装载进入 Java 虚拟机后，再经 Java 解释器转换成对应特定平台或系统的机器码而被解释执行。Java 程序运行流程如图 1-1 所示。

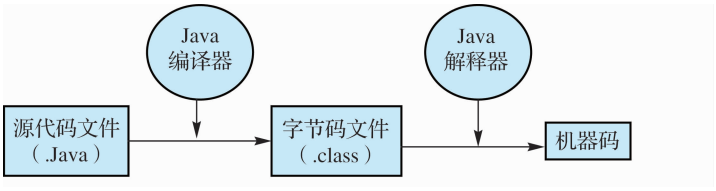

图 1-1 Java 程序运行流程

任何一种可以运行 Java 程序的软件都可以被视为 Java 虚拟机，如浏览器、各种 Java 开发工具等，Java 虚拟机是运行 Java 程序的软件环境。引入了 Java 虚拟机后，Java 程序在不同平台上运行时就不需要被重新编译了。Java 技术使用 Java 虚拟机屏蔽了与具体平台相关的信息。只要当前计算机上安装了针对自身平台的 Java 虚拟机，Java 程序就可以运行，

而不用考虑当前具体的硬件平台和操作系统，也不用考虑字节码文件是如何生成的。Java 虚拟机消除了底层硬件平台和操作系统间的差异，实现了 Java 程序的跨平台移植，大大提高了程序开发的效率。

1.1.3　Java 程序类型

Java 程序主要分为两类：Java Application（Java 应用程序）和 Java Applet（Java 小程序）。Java 应用程序是指能通过 Java 解释器解释独立运行的程序。可执行的 Java 应用程序的主类中必须有 main 方法。main 方法是 Java 应用程序执行时的入口。Java 小程序是用来增强网页功能，产生特殊效果的程序。不能单独运行 Java 小程序，必须将其嵌入用 HTML 编写的 Web 页面中，通过与 Java 兼容的浏览器或小程序查看器来控制执行。Java 小程序的编写方式与 Java 应用程序类似，因此，熟悉了 Java 应用程序的编写，很容易学会编写 Java 小程序。本书将引导读者进行 Java 应用程序的开发设计。

1.2　创建第一个 Java 项目

进行 Java 应用程序开发，需要准备以下开发工具：JDK、Java 集成开发环境、应用数据库等。

1.2.1　Java 应用程序运行环境搭建

1. JDK

首先要安装 JDK。JDK 安装文件可到 Java 官方网站 https://www.oracle.com 下载，如图 1-2 所示，然后运行安装即可。本书建议采用 JDK 1.6 及以上版本。

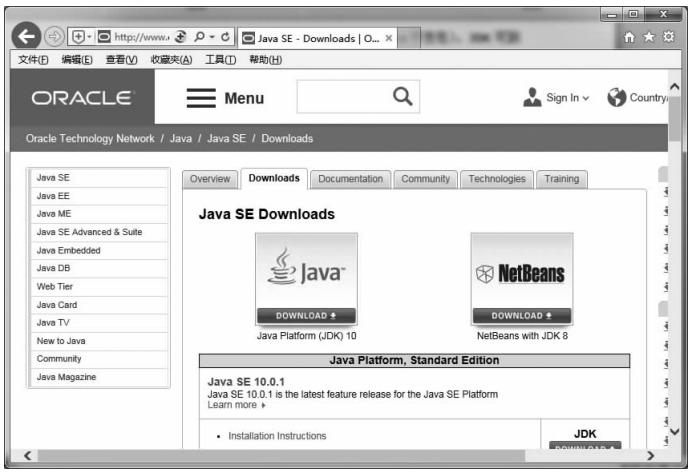

图 1-2　Java 官方网站

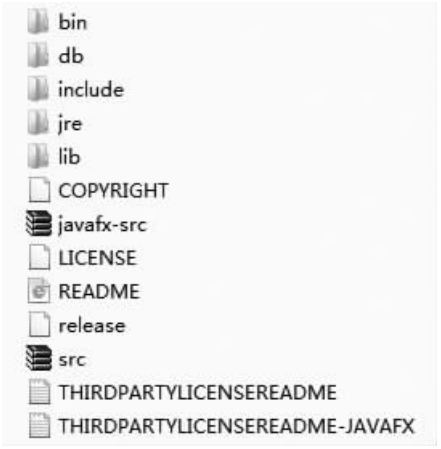

图 1-3　JDK 的目录结构

安装成功后，JDK 的目录结构如图 1-3 所示，其主要部分的描述如下。

(1)bin 文件夹：存放 Java 开发工具的可执行文件(.exe)。

(2)db 文件夹：存放示例的相关数据文件。

(3)include 文件夹：存放用于本地计算机的 C 语言头文件。

(4)jre 文件夹：存放 Java 运行时的环境文件。

(5)lib 文件夹：存放 JDK 的类库文件。

(6) javafx — src. zip：存放用于创建 Rich Internet Applications 的源代码。

(7)src. zip：存放 JDK 的源代码。

2. 应用数据库

Java 应用程序通过 JDBC 驱动可访问多种数据库管理系统。在实验室学习环境下，数据存取规模不大，故本书选用 MySQL 数据库管理系统提供应用数据库服务。MySQL 数据库管理系统可到 MySQL 官方网站 https://www.mysql.com 下载，如图 1-4 所示。本书采用 MySQL 5.5 版本。

图 1-4　MySQL 官方网站

1.2.2　Java 应用程序集成开发环境 MyEclipse

MyEclipse IDE 是一个成熟的用于 Java 项目开发的企业级平台。MyEclipse IDE 可到 MyEclipse 官方网站 https://www.genuitec.com 或 http://www.myeclipsecn.com 下载，如图 1-5 所示。本书采用 MyEclipse 8.5 版本。

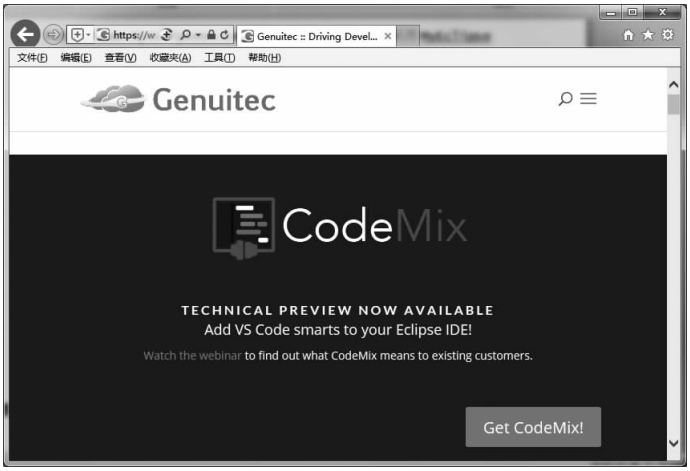

图 1-5　MyEclipse 官方网站

在 Windows 操作系统中，执行“开始”→“所有程序”→“MyEclipse 8.5”命令，启动 MyEclipse 8.5，会出现图 1-6 所示的 MyEclipse 8.5 主窗口。

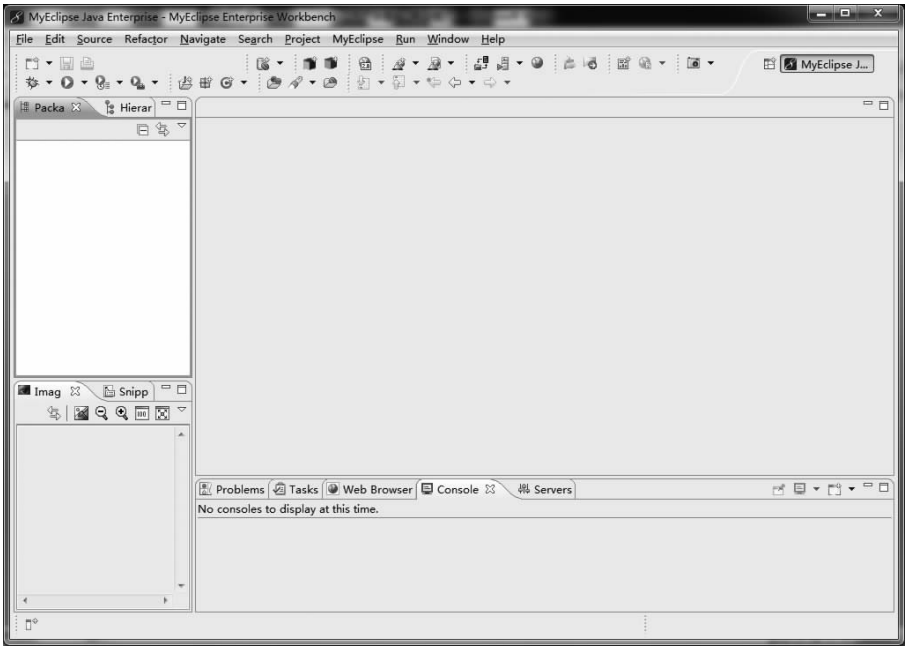

图 1-6　MyEclipse 8.5 主窗口

每次启动 MyEclipse 时，都会提示选择工作区，选定好且以后不打算更改，可选中左下方的复选框，再单击“OK”按钮，如图 1-7 所示。

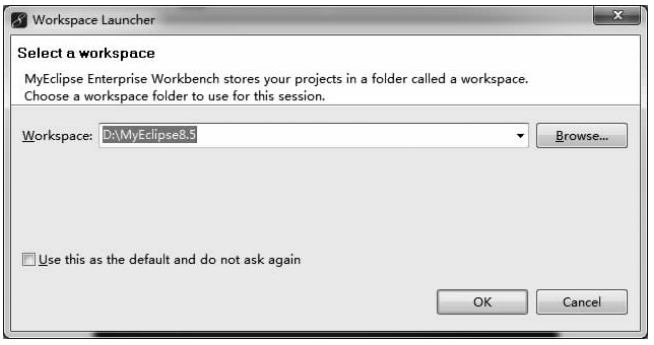

图 1-7　选择 MyEclipse 工作区

1.2.3　在 MyEclipse 下开发 Java 应用程序

在所有需要下载、安装的软件准备好之后，就可以在 MyEclipse 中创建 Java 项目，编写 Java 应用程序了。但是，在 Java 项目创建前后，还需要按以下步骤做一些整合工作，以便所编写的 Java 应用程序能成功运行。

1. 配置 JRE

运行 Java 应用程序时需要 JRE 的支持。可根据需要手动配置已下载的较高版本的 JRE。启动 MyEclipse 8.5，执行“Window”→“Preferences”命令，在打开的“Preferences”窗口中展开左侧目录树中的“Java”项，从中选择“Installed JREs”项，如图 1-8 所示。可以发现 MyEclipse 8.5 内嵌的 JDK 版本为 1.6.0。

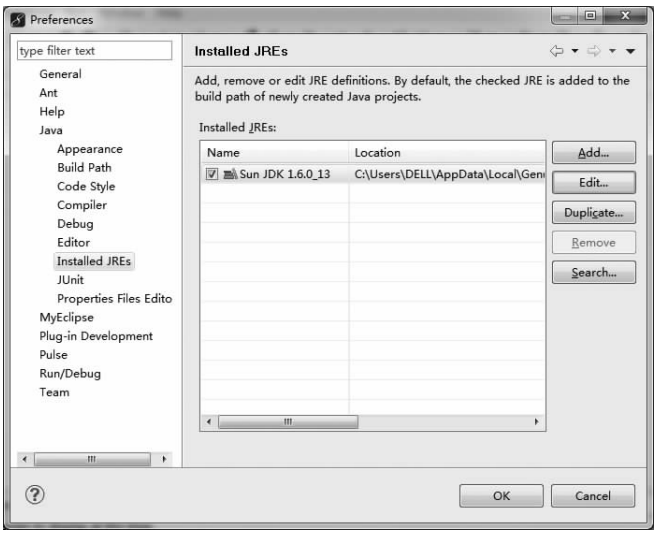

图 1-8　已安装的 JRE

如果需要更换为更高版本的 JRE，可单击“Add”按钮，在打开的“Add JRE”窗口中选择“Standard VM”，单击“Next”按钮，如图 1-9 所示。

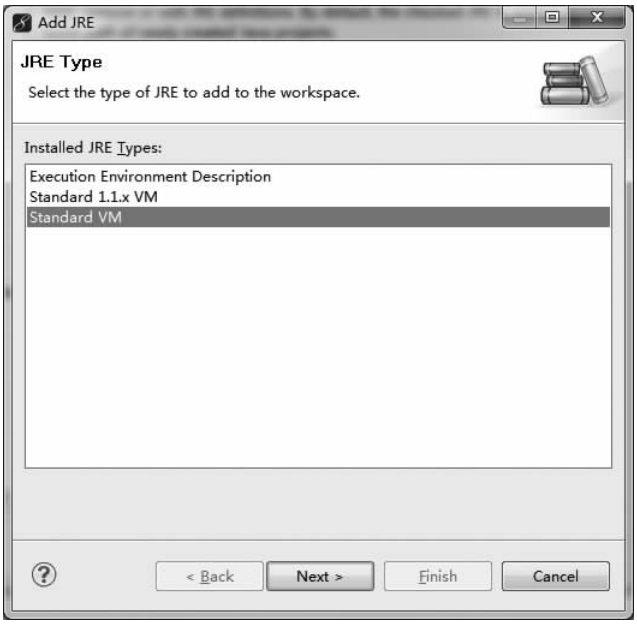

图 1-9　JRE 类型选择

在出现的“JRE Definition”界面中单击“Directory”按钮，如图 1-10 所示。

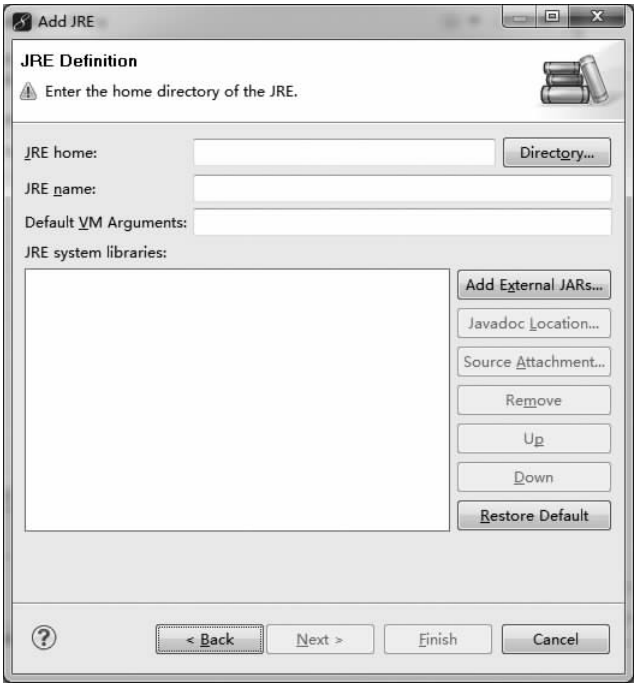

图 1-10　JRE 定义

在打开的“浏览文件夹”对话框中选择已安装的较高版本的 JDK，单击“确定”按钮，如图 1-11 所示。

图 1-11　选择已安装的较高版本的 JDK

返回到添加好路径的“JRE Definition”界面，单击“Finish”按钮完成 JRE 的配置。

注意：不要忘了在“Installed JREs”选项组中选中新添加的 JRE，使配置生效。

2. 创建 Java 项目

在 MyEclipse 8.5 主窗口中，执行“File”→“New”→“Project”命令，打开“New Project”窗口，创建 Java 项目向导，展开“Java”项，从中选择“Java Project”，如图 1-12 所示，单击“Next”按钮，打开如图 1-13 所示的“New Java Project”窗口，在“Project name”文本框中输入项目名称 myPro，其他选项保持默认设置，单击“Finish”按钮，完成 Java 项目的创建。

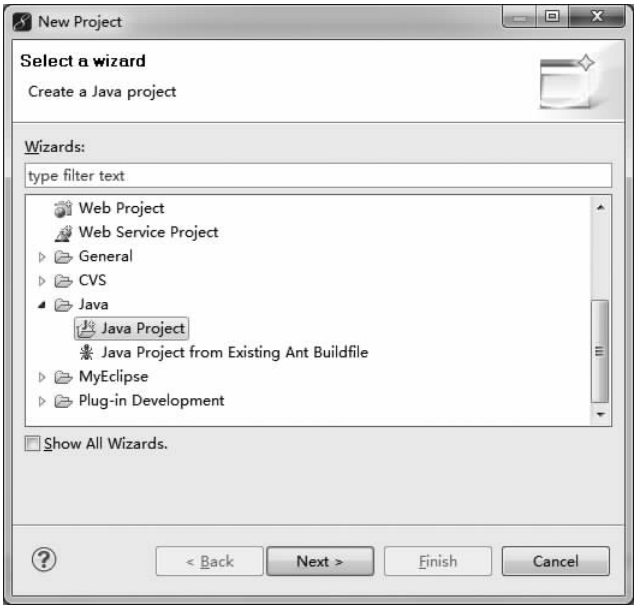

图 1-12　创建 Java 项目

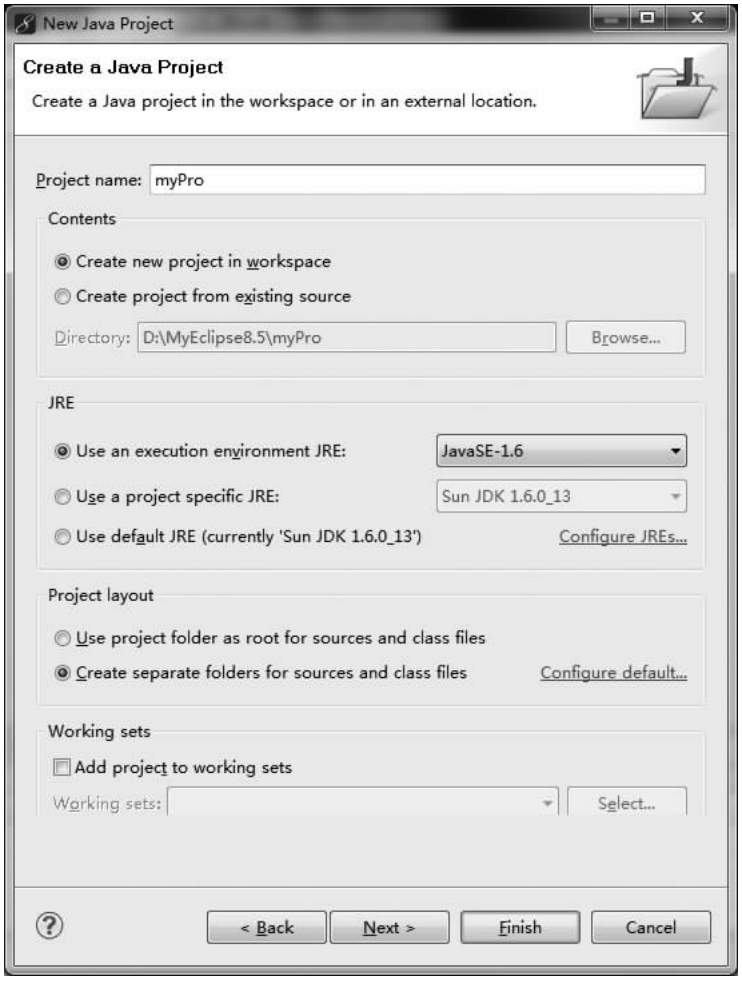

图 1-13　完成 Java 项目创建

3. Java 项目的目录结构

Java 项目要求按特定的目录结构组织文件，当在 MyEclipse 中创建好一个新的 Java 项目后，便可以在 MyEclipse 的包资源管理器(Package Explorer)中看到该 Java 项目的目录结构，如图 1-14 所示。它是由 MyEclipse 自动生成的。

Java 项目的目录和文件包括：

(1)src 目录：用来存放 Java 源文件，在其中可以创建若干个不同层级的包，分别存放功能或用途相近的 Java 源文件。

(2)JRE System Library 目录：包含运行 Java 项目所需的. jar 文件。扩展名为. jar 的文件由若干. class 文件打包生成，名称源自“Java Archive Files”(Java 存档文件)。

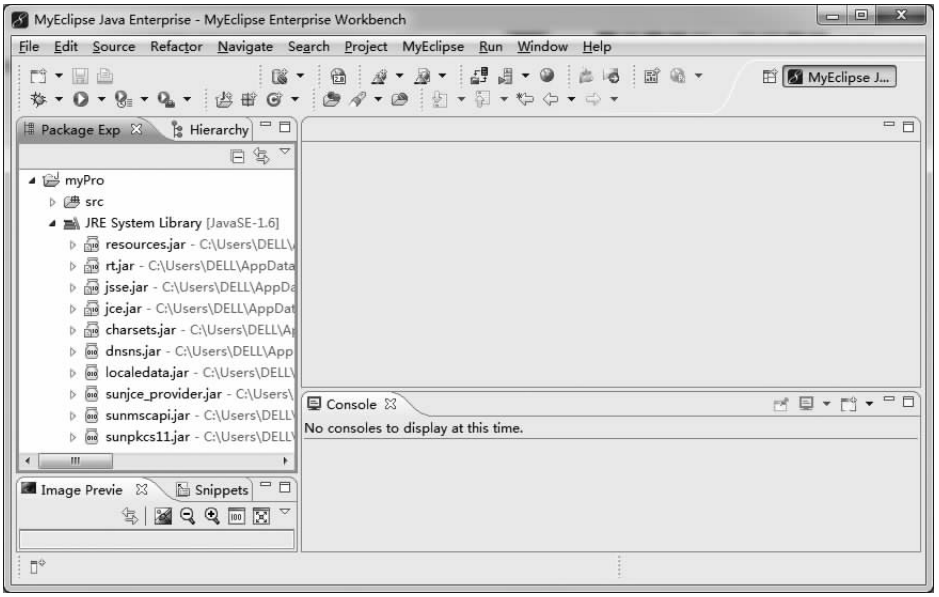

图 1-14　Java 项目的目录结构

4. 编写 Java 程序

首先，创建一个 Java 文件：右击 myPro 项目下的 src 目录，在弹出的快捷菜单中选择“New”→“Class”命令，如图 1-15 所示。

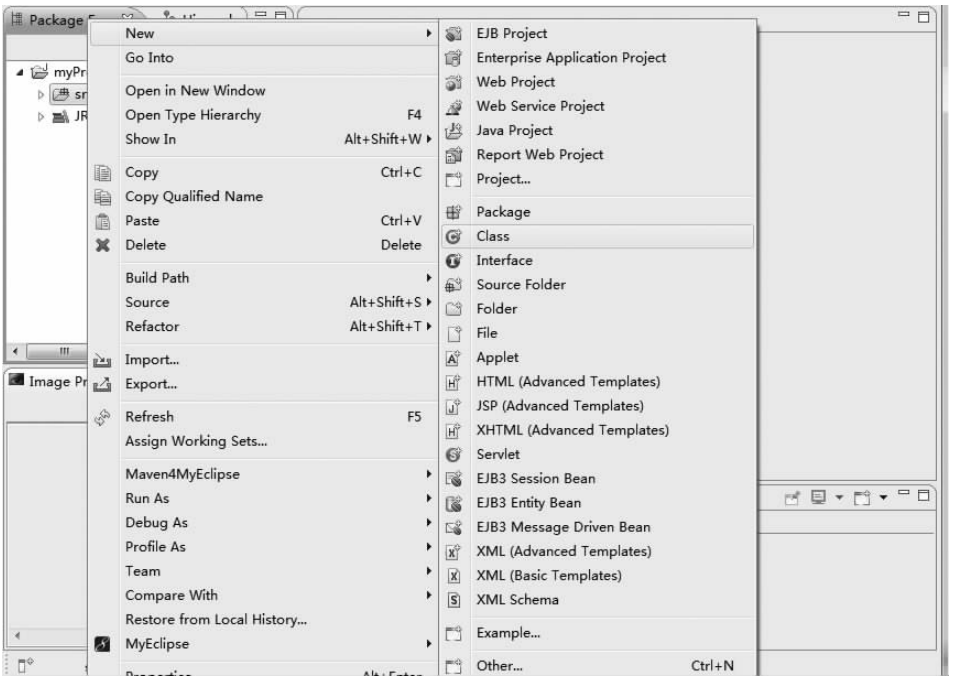

图 1-15　创建 Java 文件

接着，在弹出的“New Java Class”窗口中输入类名 MyFirstClass，选中下方的“public static void main(String[] args)”复选框，如图 1-16 所示。

图 1-16　完成 Java 文件创建

这里的文件名为 MyFirstClass. java，直接放在/myPro/src 目录下。最后，单击“Finish”按钮，完成 Java 文件的创建。MyFirstClass. java 在 MyEclipse 主窗口的编辑区同时被打开，如图 1-17 所示。

在这段代码中，MyFirstClass 就是源文件 MyFirstClass. java 所包含的类。MyFirstClass 是类的名字。每个有意义的 Java 源文件至少要有一个类。类是 Java 源文件的核心组成部分，用一对大括号{}构成类体，大部分程序代码编写在类体内。MyFirstClass 前面的 public 和 class 是两个关键字(“关键字”将在模块 2 做详细解释)。class 表示 MyFirstClass 是一个 Java 类。public 是进行访问控制的修饰符(访问控制将在模块 4 做详细解释)表示 MyFirstClass 是一个公开类，在这个类之外都可以访问它。有的 Java 类前面没有任何修饰符，这其实也是一种访问控制方式，称为 default(默认)访问控制。在用默认访问控制修饰的类之外并不都能访问它。对 Java 类的访问控制只有 public 和 default 这两种方式。

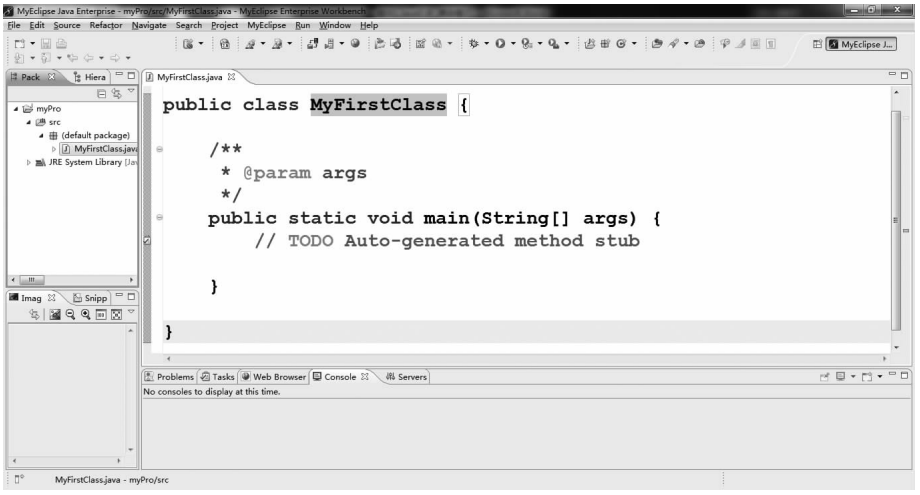

图 1-17　Java 文件编辑区

public static void main(String[] args)是类 MyFirstClass 所包含的 main()方法。main()方法是 Java 程序运行的起点，没有它，Java 程序无法启动。main()方法前面的修饰必须是 public、static 及 void(在模块 4 做详细解释)。main()方法括号中的参数 String[] args 表示程序运行时，所输入的参数会由字符串类型的数组 args 来存放。在 main()方法的大括号{}之间输入以下代码“System. out. println("Hello Java!");”，然后保存(执行“File”→“Save As”命令或在编辑区中按“Ctrl+S”快捷键)，如图 1-18 所示。所输入的代码称为语句。Java 代码中每条语句以分号“;”结束。System. out 指标准输出，其后的 println()方法是由 print 与 line 所组成的缩写，在 MyEclipse 中表示将括号中字符串类型的参数内容输出到控制台上，然后光标移到下一行的开头(换行)。

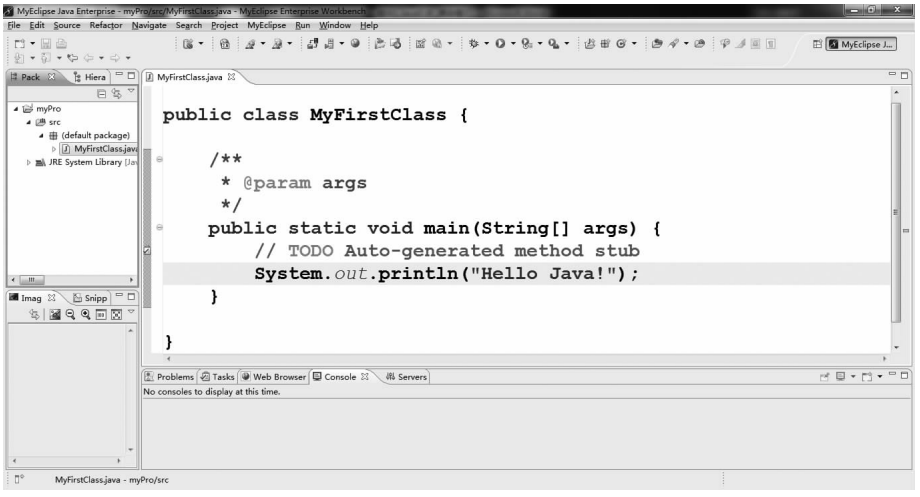

图 1-18　在 Java 文件编辑区编写代码

5. Java 文件的命名

可以把很多类写在一个 Java 源文件中。但是，其中至多只能有一个类用 public 修饰符修饰。Java 源文件的名称必须和用 public 修饰符修饰的类的名称相同。如果文件中所有的类都是默认访问控制，那么 Java 源文件的名称可以不用与类的名称相同。

在组成 Java 源文件的多个类中，如果有一个类的名称与源文件一致且含有 main()方法，这个类通常被称为主类。只有主类才能用 public 修饰符修饰。当然，主类未必一定要用 public 修饰符来修饰。

6. 运行 Java 程序

只有包含主类的 Java 程序才能运行。

将鼠标光标停留在编辑区，在 MyEclipse 主窗口的“Run”菜单项中选择“Run As”→“Java Application”命令，或者单击工具栏中的“Run As”图标右侧的下拉箭头，在弹出的下拉菜单中选择“Run As”→“Java Application”命令，即可运行 Java 程序 MyFirstClass. java，如图 1-19 和图 1-20 所示。

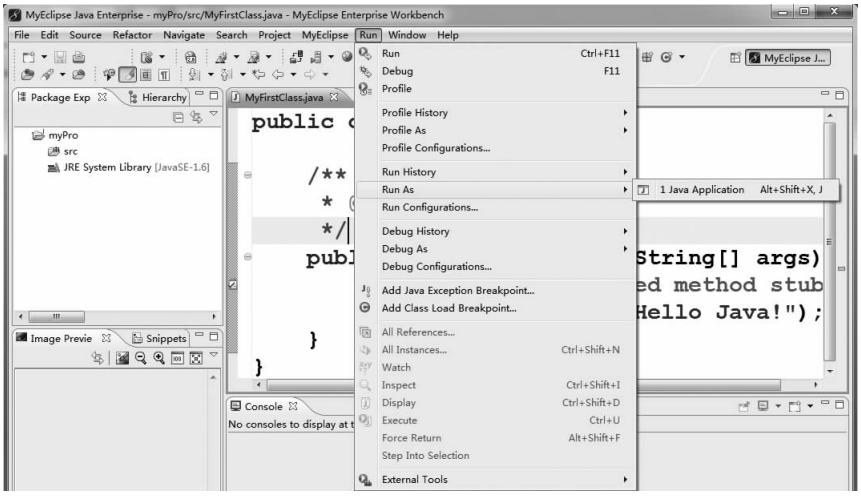

图 1-19　运行 Java 程序的方式之一

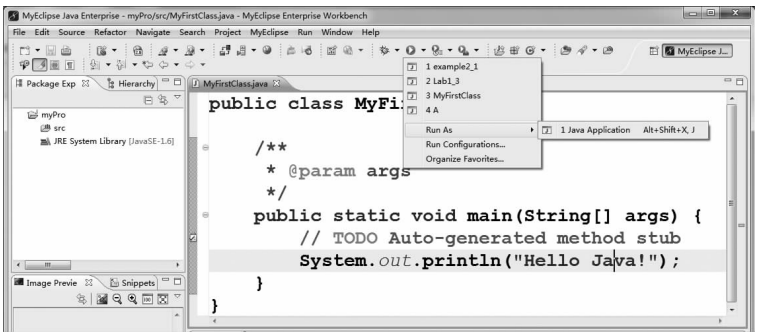

图 1-20　运行 Java 程序的方式之二

在控制台显示 MyFirstClass.java 的运行结果为“Hello Java!”，如图 1-21 所示。

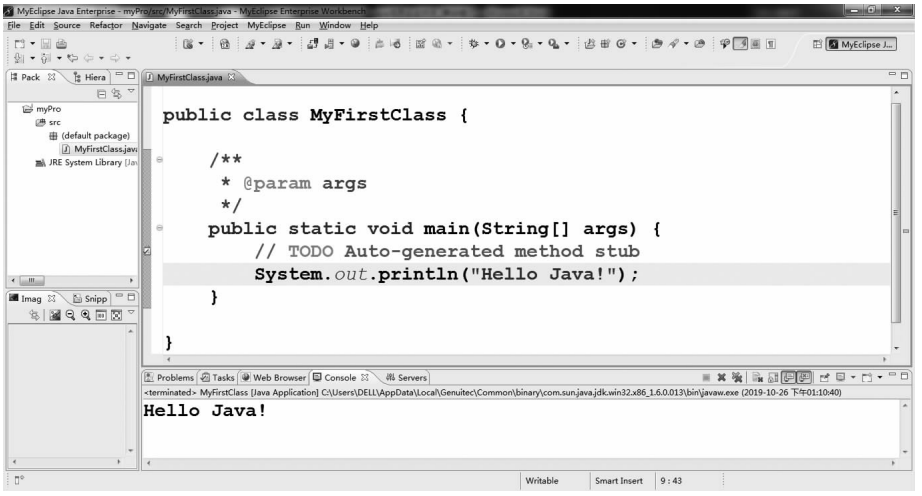

图 1-21　Java 程序运行结果

注意：除“主类”外，Java 源文件中的其他每个类都可以有一个 main()方法。这种 Java 程序仍然可以运行。只不过在 MyEclipse 中启动运行命令时会弹出对话框让用户选择所要匹配的含有 main()方法的类。

7. 有参数的 Java 程序运行

main()方法的参数永远是 String[]类型(字符串数组)。其用意在于以参数的方式接收传入程序的数据。args[0]存放第一个数据，args[1]存放第二个数据，以此类推。这些数据都是 String(字符串)类型。

现在要求通过 MyFirstClass 类中 main()方法的参数向程序中传入数据，在控制台同样能显示“Hello Java!”。

首先，修改 MyFirstClass 类，将其 main()方法的代码“System.out.println("Hello Java!");”改为“System.out.println(args[0] + " " + args[1]);”。

在 MyEclipse 主窗口的“Run”菜单项中选择“Run Configurations”命令，如图 1-22 所示；或者单击工具栏中的“Run As”图标右侧的下拉箭头，在弹出的下拉菜单中选择“Run Configurations”命令。

打开“Run Configurations”窗口，进行 Java 程序运行配置。在右侧界面中选择“Arguments”选项卡，在出现的“Program arguments:”列表框中输入“Hello Java!”，单击“Apply”按钮使输入生效，如图 1-23 所示，最后单击“Run”按钮运行，就可以在控制台看到同样的输出结果。

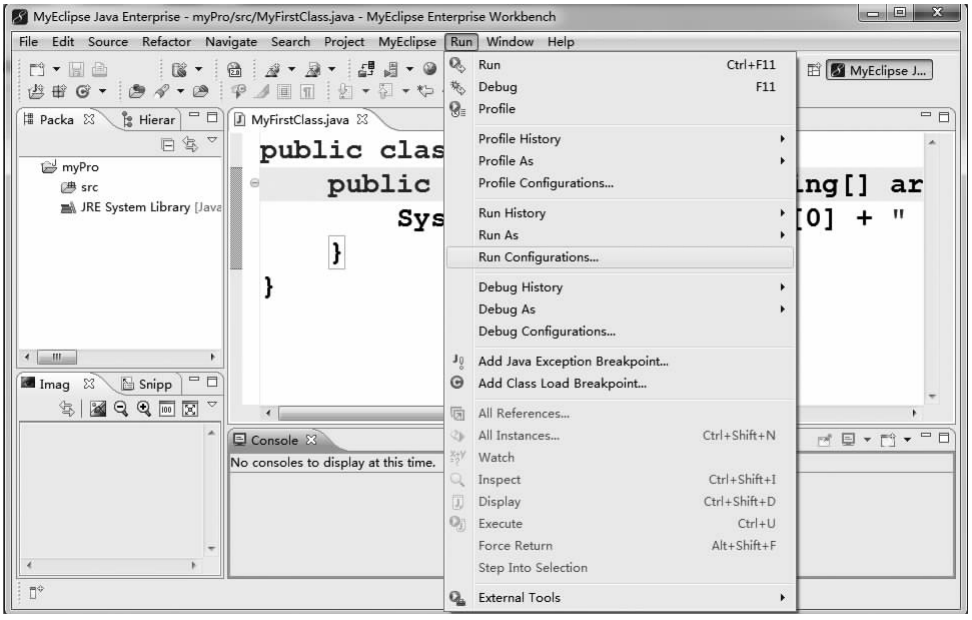

图 1-22　Java 程序运行配置

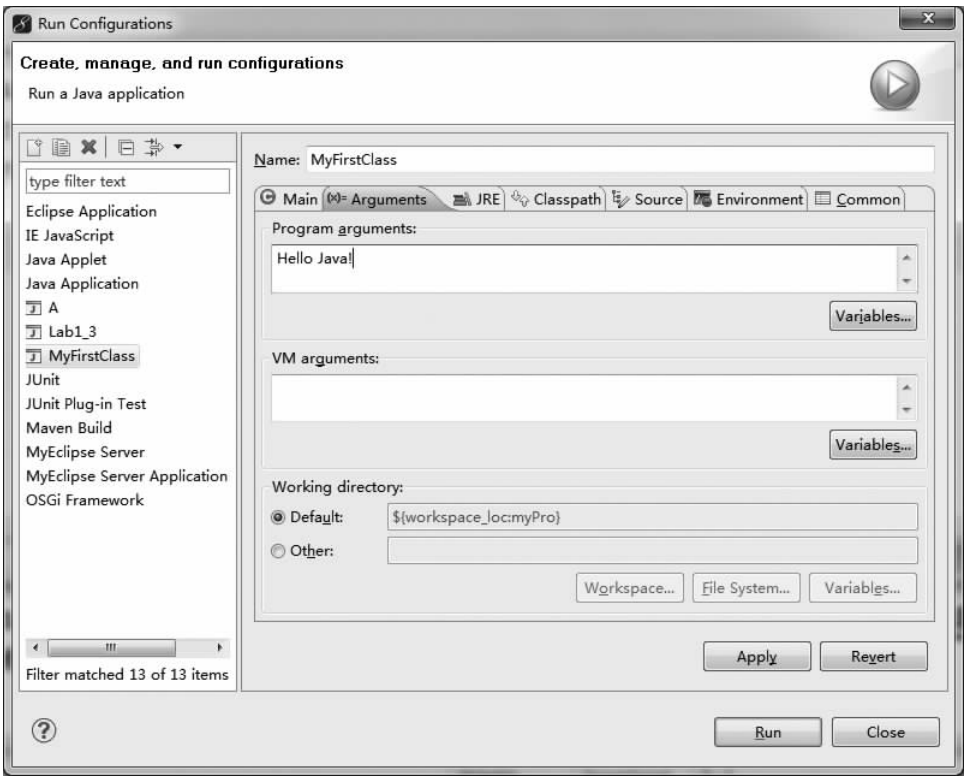

图 1-23　输入程序实际参数

1.3 Java API 帮助文档

API(application programming interface,应用程序编程接口)是预先定义好的功能及服务,使得程序开发人员无须了解源代码或理解内部工作机制的细节,通过外部接口就能直接访问所提供的功能及服务。如果打算在某个平台上支持 Java 程序,必须事先在此平台上实现 Java API 功能及服务。

Java API 以文档的方式为用户提供帮助。可到 Java 官方网站 https://www.oracle.com 在线查看或下载 Java API 文档。Java API 文档是从事 Java 程序开发所必备的权威性参考资料。在 Java 程序开发过程中要养成查阅 Java API 文档的习惯,在 Java API 文档中寻求解决方案。Java API 文档如图 1-24 所示。

图 1-24 Java API 文档

1.4 小 结

Java 是一门面向对象编程语言。Java 具有简单性、面向对象、分布式、健壮性、安全性、平台独立与可移植性、多线程、动态性等特点。Java 有 3 个独立的版本:Java SE、Java EE 和 Java ME。运行 Java 程序时需要 JRE。JDK 是开发 Java 程序的核心。JDK 包含了 JRE。Java 源程序文件经编译器编译后,生成与平台无关的字节码文件,再经解释器解释后,在 Java 虚拟机上执行机器码。Java 虚拟机实现了 Java 字节码的跨平台。Java 程序有两类:

Java 应用程序和 Java 小程序。进行 Java 应用程序开发，需要 JDK、Java 集成开发环境、应用数据库等开发工具。在 MyEclipse 8.5 下开发 Java 项目的步骤为：配置 JRE—创建 Java 项目—创建 Java 文件—编写 Java 程序—运行 Java 程序。Java 主类的名称与源文件一致且含有 main()方法。只有包含主类的 Java 程序才能运行。可以通过 main()方法的参数向程序中传入数据来运行 Java 程序。在 Java 程序开发过程中，可以通过查阅 Java API 文档获取帮助。

1.5　习　　题

1. 用百度搜索 Java、Java SE、JDK、JRE、JVM、MyEclipse、API 等关键词，搜集相关信息进行学习和分析。

2. 回顾并总结在 MyEclipse IDE 中开发 Java 项目的软件安装及配置全过程，并在计算机上进行实践。

3. 文件名为 A.java，其代码如下。合法吗？为什么？

(1)

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

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

class D {
}
```

(2)

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

```
    }
}

public class B {
}
```

1.6 上机实践

1. 在 MyEclipse 8.5 下编写一个 Java 程序，向控制台输出“Hello World!”。

2. 在 MyEclipse 8.5 下编写一个 Java 程序，向控制台输出“这是我的第一个 Java 程序”。

3. 在 MyEclipse 8.5 下编写一个 Java 程序，向控制台输出专业信息，输出结果如图 1-25 所示。

```
--------------------------
        专业信息一览表
--------------------------
    专业          所属学院
--------------------------
计算机网络技术  交通信息工程学院
现代物流技术    运输管理工程学院
市政工程技术      路桥工程学院
车辆维修技术      汽车工程学院
--------------------------
```

图 1-25　上机实践 3 运行结果

模块2

Java 语言基础

知识目标

1. 掌握标识符、关键字、分隔符、变量和常量的含义和使用方法。
2. 了解 Java 注释的格式和代码书写风格。
3. 掌握基本数据类型及数据类型转换。
4. 掌握运算符的分类和使用，以及表达式中运算符的优先级和结合性。
5. 掌握从控制台获取用户键盘输入数据的方法。

技能目标

1. 能够在 MyEclipse IDE 中定义标识符、变量和常量；使用关键字及分隔符。
2. 能够在 MyEclipse IDE 中编写适当的 Java 注释。
3. 能够在 MyEclipse IDE 中运用基本数据类型并进行数据类型转换。
4. 能够在 MyEclipse IDE 中编写表达式并进行各类运算。
5. 能够在 MyEclipse IDE 中编写程序从控制台获取用户键盘输入。

2.1 回顾与思考

在模块 1 中，我们学习了在 MyEclipse 8.5 中开发 Java 项目，编写简单 Java 应用程序和运行 Java 应用程序的详细流程。在了解了 Java 程序基本结构的基础上，本模块我们将学习 Java 语言的基本语法。作为初学者不能急于求成，要认真学习并思考本模块的内容，为后面的学习打好基础。

【例 2-1】 编写一个 Java 应用程序，根据所给圆的半径，计算并输出圆的周长。文件名为 example2_1.java，其代码如下。

```
public class example2_1 {
  /**
   * 例 2-1 根据所给圆的半径，计算并输出圆的周长
   * @author《Java 程序设计案例教程》
   * @version 1.0
   */

  public static void main(String[] args) {
  final double PI = 3.14;//圆周率
  int radius = 10;//半径
  double perimeter = 2 * PI * radius;//周长
  System.out.println("半径为" + radius + "的圆的周长为" + perimeter);
  }
}
```

以上代码中 example2_1、main 、args、PI、radius 和 perimeter 都是标识符；public、class、static、void、final、double 和 int 都是关键字；“{}、()、[]、;”和“.”都是分隔符；radius 和 perimeter 是变量；PI 是常量；double 和 int 属于基本数据类型；PI * radius 涉及数据类型转换；=、* 和+都是运算符；""是字符串定义符；+也是字符串运算符；/** 至 */之间、/* 至 */之间、//之后的同一行的内容为注释。寥寥几行代码就涉及了 Java 中很多基本语法。接下来将逐个分析 Java 语言中的各类基本语法要素。

2.2 变量和常量

2.2.1 标识符

标识符用来表示变量、常量、类、方法、参数、接口、包、文件等元素的名字。Java 语言中的标识符由字母、数字、下划线和美元符号(MYM)组成，并且需要遵守以下规则。

(1)不能以数字开头。

(2)区分大小写。

(3)没有长度限制。

(4)不能使用 Java 语言中的关键字。

用标识符命名惯例上(但不强迫)遵循见名知义原则和驼峰命名法。见名知义原则是指

标识符要能够反映被定义者的含义或作用，使得程序阅读者通过标识符名称就能对程序有所理解。例如，使用 carBrand 定义汽车品牌，使用 bridgeName 定义桥梁名称等。驼峰命名法是指如果标识符是由多个单词组成的，第一个单词以小写字母开始，第二个及后续每一个单词的首字母大写。这样看上去就像驼峰一样起伏，增强了程序的可读性。例如，studentNo、deptManager 等。

2.2.2　关键字

关键字是 Java 语言中已被赋予特定含义的标识符，只能供 Java 编译系统使用。Java 语言中不允许用户对关键字再赋予其他含义。Java 语言中的关键字见表 2-1。

表 2-1　Java 语言中的关键字

abstract	boolean	break	byte	case	catch
char	class	continue	default	do	double
else	extends	false	final	finally	float
for	if	implements	import	instanceof	int
interface	long	native	new	null	package
private	protected	public	return	short	static
super	switch	synchronized	this	throw	throws
transient	true	try	void	volatile	while

另外，还有两个特殊标识符 goto 和 const。它们是 Java 语言中预留的关键字，也称为保留字；目前还没有作为关键字被使用，但是也不能将其当作用户定义标识符来使用。

2.2.3　分隔符

(1)大括号({})：用来定义语句块、类、方法及局部范围；也用于以赋值方式初始化数组。

(2)方括号([])：用来声明数组和引用数组元素。

(3)圆括号(())：用来容纳方法的参数列表；也用于由控制语句和强制类型转换组成的表达式；也用来表示执行或计算的优先级。

(4)分号(;)：用来终止一条语句；也用来分隔 for 循环控制语句圆括号中的表达式。

(5)逗号(,)：用来分隔变量列表中的各个变量或参数列表中的各个参数；也用来分隔 for 循环控制语句圆括号中各表达式部分的语句序列。

(6)圆点(.)：用来分隔包与其子包或类；也用于类和实例对象调用成员变量和成员方法。

(7)空白( )：这类分隔符比较特殊，包括空格(Space)、跳格(Tab)和换行(Enter)。

2.2.4 变量

变量是程序运行过程中值可以发生改变的量。Java 中的变量是程序在计算机内存中的一个基本存储单元。Java 中的变量必须先声明后使用。

声明变量的格式为：

```
数据类型 变量名;
```

变量名必须是 Java 中合法的标识符，可以依据个人的喜好命名变量。习惯上以其所代表的含义给变量命名，如 num 代表数字，age 代表年龄等。尽量不要用简单的英文字母命名变量，因为简单的变量名称所代表的含义往往令人费解，它会增加阅读及调试程序的难度。

变量声明后，可对变量进行初始化，即为其赋初值，格式为：

```
变量名 = 变量值;
```

也可以在声明变量的同时立即对其进行初始化，格式如下。

```
数据类型 变量名 = 变量值;
```

如果需要声明或声明并初始化多个同一类型的变量，可以采用如下方式。

数据类型 变量名 1，变量名 2，...；

数据类型 变量名 1 = 变量值 1，变量名 2 = 变量值 2，...；

因为变量的值在程序运行过程中可以发生改变，所以下面的语句语法是正确的。

```
int i = 10;              //声明一个整型变量 i,并为其赋初值 10
i = 20;                  //给整型变量 i 重新赋值 20
```

2.2.5 常量

常量是程序运行过程中值始终保持不变的量。声明并初始化常量只要在声明变量的数据类型前加上关键字 final 即可，格式为：

```
final 数据类型 常量名;
```

然后：

```
常量名 = 常量值;
```

或者

```
final 数据类型 常量名 = 常量值;
```

常量只能赋一次值，其值在之后的代码中不可被修改。为了与变量名表示进行区分，

Java 语言中约定常量名中的字母全部为大写。例如，例 2-1 中的语句“final double PI = 3.14;”定义了表示圆周率的常量 PI。

注意：所声明的常量若为类的成员常量，必须在声明的同时进行初始化，否则会发生编译错误。

2.3　Java 注释及代码书写风格

2.3.1　Java 注释

注释是对所编写代码的解释和说明。适当的注释有助于阅读和调试程序。注释不属于 Java 程序代码的组成部分，因此，编译器在读到注释时会将其忽略。Java 语言中有以下 3 种形式的注释。

1. 单行注释

以“//”开始，“//”后面整个一行且仅此一行的内容为注释内容。

2. 多行注释

以“/*”开始，“*/”结束，“/*”至“*/”之间（可以写多行）的内容为注释内容。

3. 文档注释

以“/**”开始，“*/”结束，“/**”至“*/”之间（可以写多行）的内容为注释内容。利用 JDK 提供的 javadoc 命令，可将注释内容从程序代码中提取出来，生成 Java 官方格式的 API 帮助文档。注释可包含若干以“@”为前缀的文档注释标签，标明程序作者、版本信息、参数、返回值等数据。

2.3.2　Java 代码书写风格

编写代码必须形成特定的风格。Java 对代码编写格式的要求是非常自由的，在符合语法的前提下，可以把所有代码写成一行，也可以每行只写一个代码。如果这样书写或自由发挥，编写的代码就难以阅读，给程序员的后续工作带来很多麻烦。对于代码的书写，推荐以下风格（在 MyEclipse IDE 中可以设置）。类及其所属包、导入包顶格书写，类体语句块的左大括号在上一行的行尾，右大括号独占一行。类体中的成员变量、构造方法、成员方法等逐级缩进，如图 2-1 所示。

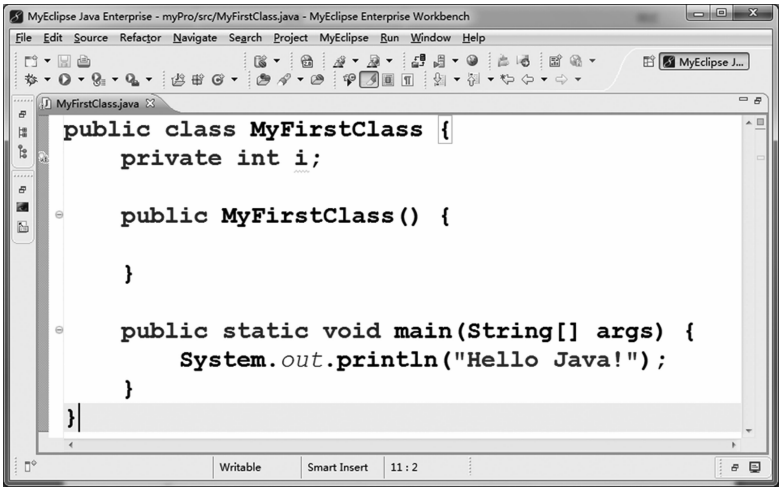

图 2-1　Java 代码书写风格

2.4　数 据 类 型

Java 中的数据类型分为两大类：基本数据类型和引用数据类型。

2.4.1　基本数据类型

基本数据类型有 8 种，分别是：char(字符型)，boolean(布尔型)，4 种整数型 byte(字节型)、short(短整型)、int(整型)和 long(长整型)，两种浮点数型 float(单精度型)和 double(双精度型)。整数型和浮点数型又合称为数值型。

1. 字符型

字符型 char 用来表示单个字符，如字母、数字、符号等。Java 使用 Unicode 字符集，所以字符型数据都是无符号的 16 位整数，范围为 0～65 536，即 0x0000～0xffff(0x 表示十六进制数)。

字符型数据占两个字节，用加一对单引号引起来的 Unicode 字符来表示，如 '南' 'N' '9' 等都是合法的字符型数据。也可以直接用加一对单引号引起来的 Unicode 编码来表示字符型数据，以“\u”开头加上 4 个十六进制数，如 '\u0042' 表示 'B'。

有些字符在程序中难以直接表达，Java 中就定义了这样一些特殊字符，称为转义字符。转义字符以“\”开头，跟在后面的字符含义发生了转变，所以称之为“转义”。常用的转义字符见表 2-2。

表 2-2　常用转义字符表

转义字符	含　义	转义字符	含　义
\	反斜杠	\n	换行
\'	单引号	\t	跳格
\"	双引号	\b	退格
\r	回车	\f	换页

2. 布尔型

布尔型 boolean 用来表示两个逻辑状态，只有 true 和 false 两个取值，分别代表逻辑“真”和逻辑“假”。

3. 整数型

整数型是指没有小数部分，但有符号的数据类型。Java 提供字节型 byte、短整型 short、整型 int 和长整型 long 等 4 种整数型数据。这 4 种整数型数据的区别是它们在内存中所占用的字节数不同，因而它们所能存储的整数的取值范围也相应地不同。整数型数据的基本指标见表 2-3。

表 2-3　整数型数据的基本指标

数据类型	长　度	取值范围
byte	1 字节 8 位	−128～127
short	2 字节 16 位	−32 768～32 767
int	4 字节 32 位	−2 147 483 648～2 147 483 647
long	8 字节 64 位	-2^{63}～$2^{63}-1$

4. 浮点数型

浮点数型是指带有小数部分的数据类型，包括单精度型 float 和双精度型 double 两种浮点数型。默认情况下，浮点数型数据为 double 类型。float 类型的数据须在浮点数后加上后缀 F 或 f，如 10.9f。浮点数型数据的基本指标见表 2-4。

表 2-4　浮点数型数据的基本指标

数据类型	长　度	取值范围
float	4 字节 32 位	-3.4×10^{38}～3.4×10^{38}
double	8 字节 64 位	-1.7×10^{308}～1.7×10^{308}

Java 中还有 3 个特殊的浮点数值：Infinity（正无穷大）、-Infinity（负无穷大）、NaN（非数

字),用于表示溢出和出错。例如,用 0 去除一个正浮点数,所得结果为 Infinity;用 0 去除一个负浮点数,所得结果为-Infinity;用 0 去除 0.0,所得结果为 NaN。

2.4.2 引用数据类型

引用数据类型包含类、接口、数组等,将在后续模块的学习中进行详细讲解。

2.4.3 数据类型转换

在程序设计过程中经常会发生不同类型数据之间的运算。例如,一个整数和一个浮点数相加,两者占用的内存长度不同,相加结果是整数型还是浮点数型呢?这就需要进行数据类型转换处理。Java 中基本数据类型的转换分为以下两种情形。

1. 自动类型转换

取值范围小的数据类型向取值范围大的数据类型转换,Java 编译器自动实现,不需要做特殊处理。这类数据类型转换称为自动类型转换。沿箭头方向取值范围从小到大的顺序为:①byte→short→int→long→float→double,②char→int→long→float→double。沿以上①和②中箭头方向发生的数据类型转换属于自动类型转换。例如,"int i = 10; double d = i;"。

2. 强制类型转换

(1)double→float→long→int→short→byte。

(2)double→float→int→long→char。

沿箭头方向,取值范围大的数据类型向取值范围小的数据类型转换,会导致溢出或精度下降,转换后必然会丢失部分信息。这类数据类型转换称为强制类型转换。强制类型转换有语法格式要求,其格式为:

目标数据类型 变量名 = (目标数据类型)源数据类型;

例如:

```
double d = 5.5; int i = (int)d;
```

当然,强制类型转换后,源数据本身类型不会发生改变。例如,上例中变量 d 的数据类型仍为 double。

注意:char 不能与 byte 和 short 发生自动类型转换,只能相互进行强制类型转换。boolean 布尔型不参与数据类型转换。

2.4.4 包装类

以上 Java 中的 8 种基本数据类型也被称为原始数据类型。为提高程序运行效率,可以不把原始数据类型当成对象考虑。然而,Java 中所有的一切都可作为对象看待,所以 Java

为原始数据类型提供了一些特殊的类，让原始数据类型在使用上如同对象一般。这些特殊的类称为 Wrapper Class(包装类)。包装类的意思就是把原始数据类型包装在类中，并额外提供相关的功能。包装类提供的变量都是“类变量”，提供的方法都是“类方法”。可通过类名接“.”分隔符直接调用。表 2-5 列出了与基本数据类型相对应的包装类。

表 2-5　基本数据类型及其对应的包装类

基本数据类型	包 装 类
boolean	Boolean
char	Character
byte	Byte
short	Short
int	Integer
long	Long
float	Float
double	Double

2.5　运 算 符

在程序设计中会频繁地进行各种运算。要进行运算，就要用到运算符。运算符是用来表示某种运算的符号，它指明了对操作数(数值、字符、对象、方法调用等)所进行的运算。Java 中的运算符由赋值运算符、算术运算符、关系运算符、逻辑运算符、位运算符及其他一些处理特殊情形的运算符组成。

2.5.1　算术运算符

Java 中的算术运算符见表 2-6。

表 2-6　算术运算符

运　算　符	含　　义
+	加法运算符
−	减法运算符
*	乘法运算符
/	除法运算符
%	取余运算符
++	自增运算符
−−	自减运算符

其中，+、-、*、/实现两个操作数的加、减、乘、除四则运算，%取两个操作数相除的余数。+和-也是一元运算符，表示单个操作数的正和负。

++和--也是一元运算符，实现操作数的增 1 和减 1 运算。例如，“x++;”是“x = x + 1;”的等效形式。“--y;”是“y = y - 1;”的等效形式。

++和--既可以放在操作数的前面，也可以放在操作数的后面，但是如果表达式中还有其他运算，两者的运算方式就不一样了。如果放在操作数的前面，操作数先进行加 1 或减 1 运算，再将结果用于表达式操作；如果放在操作数的后面，操作数先参加其他运算，再进行加 1 或减 1 运算。例如：

```
int i = 5, p, q;
p = i++;//先 p = i,再 i = i + 1,结果 p = 5,i = 6
p = ++i;//先 i = i + 1,再 p = i,结果 p = 7,i = 7
q = i--;//先 q = i,再 i = i - 1,结果 q = 7,i = 6
q = --i;//先 i = i - 1,再 q = i,结果 q = 5,i = 5
```

注意：自增运算符和自减运算符不能用于表达式，只能用于简单变量。例如，“++(i + 1);”这种写法编译不能通过。

2.5.2 关系运算符

关系运算符用来比较两个数值的大小，结果返回布尔型的值 true 或 false。关系运算符两边的数据类型要一致。数值型数据的关系运算是比较其值的大小，字符型数据的关系运算是比较其 Unicode 码的大小，布尔型数据只能进行==和!=比较。

Java 中有 6 个关系运算符，见表 2-7。

表 2-7 关系运算符

运 算 符	含 义
>	大于
>=	大于等于
<	小于
<=	小于等于
==	等于
!=	不等于

2.5.3 逻辑运算符

参与逻辑运算的操作数和运算结果都属于布尔型。Java 中有 6 个逻辑运算符，见表 2-

8。其中，只有！是一元运算符，其他都是二元运算符。

表 2-8　逻辑运算符

运　算　符	含　　义
&	逻辑与
|	逻辑或
!	逻辑非
^	异或
&&	简洁与
||	简洁或

简洁运算(&&、||)与非简洁运算(&、|)的运算结果是一样的，区别在于：简洁运算只有在必要时才计算右边的操作数。例如，在进行简洁与运算时，如果左边操作数的值为 false，则运算结果必为 false，就不再计算右边的操作数了。而非简洁运算在任何情况下都要计算两边的操作数。由此可见，简洁运算的效率要高于非简洁运算。简洁运算符也被称为短路运算符。

2.5.4　位运算符

位运算符以二进制为单位对操作数进行运算。位运算符通常与硬件有关，在 Java 应用开发中出现不多。Java 中的位运算符见表 2-9。

表 2-9　位运算符

运　算　符	含　　义
~	按位取反
&	按位与
|	按位或
^	按位异或
<<	按位左移
>>	按位右移
>>>	无符号按位右移

2.5.5　赋值运算符

赋值运算符用一个等号“=”来表示。赋值运算符用于把等号右边的值指定给等号左边的变量或常量。赋值运算符可以与算术运算符和逻辑运算符组成复合赋值运算符，使得程序运行效率更高。复合赋值运算符示例见表 2-10。

表 2-10　复合赋值运算符示例

运　算　符	示　　例	等效表达式
+=	x+=y	x = x + y
-=	x-=y	x = x - y
*=	x*=y	x = x * y
/=	x/=y	x = x / y
%=	x%=y	x = x % y
&=	x&=y	x = x & y
|=	x|=y	x = x | y
^=	x^=y	x = x ^ y

2.5.6　其他运算符

1. 条件运算符

条件运算符由“?”和“:”构成，是一个三元运算符，格式如下。

```
表达式 1 ? 表达式 2 : 表达式 3;
```

表达式 1 的运算结果为布尔型数据。条件运算符的执行过程为：先计算表达式 1 的值，如果结果为 true，则表达式 2 的值作为整个表达式的值；如果结果为 false，则表达式 3 的值作为整个表达式的值。例如：

```
int x = 1, y = 2, z;
z = x > y ? x : y;          //z 的取值为 x 和 y 中的较大者
```

2. 字符串连接符

“+”除了作为算术运算符外，还能作为字符串连接符，实现两个字符串的连接。例如：

```
int a = 123, b = 456;
int c = a + b;                    // 整型数 c 的值为 579
String x = "123", y = "456";
String z = x + y;                 //字符串 z 的值为 123456
```

3. 对象运算符

对象运算符使用关键字 instanceof 来判断对象是否为某种类型，运算结果为布尔型数据。其格式如下。

```
对象标识符 instanceof 类型标识符;
```

例如：