# 第 1 章　Java 程序设计入门

**本章重点**

- Java 语言的特点
- Java 开发环境的配置
- Java 程序运行机制
- Eclipse 开发工具的使用

**本章难点**

- Java 编译过程
- Java 环境变量配置

**本章简介**

Java 语言具有简单易用、可移植性、面向对象的特点，是一种编译+解释型语言，适合分布式计算，具有多线程处理能力。本章简要介绍 Java 程序设计的入门知识，包括 Java 概述、Java 开发环境的配置、Java 程序运行机制、Java 开发工具、Java API 文档。通过本章的学习，读者能够更深入地理解 Java 语言程序设计的基本特征和 Java 程序的编译过程，能够区别 JVM、JRE、JDK 三者的区别。

## 1.1　Java 概述

Java 是一门程序设计语言，它基于面向对象的编程思想，认为一切皆为对象。Java 语言功能强大、简单易用，作为静态的面向对象程序设计语言，能够极好地体现出面向对象思想。Java 语言由美国 Sun 公司于 1995 年发布，是目前 IT 行业应用最广泛、使用人数最多的语言之一。Java 语言广泛应用于企业级软件开发、Android 移动开发、大数据、云计算等领域，成为不少大型开发项目的首选开发语言之一。

### 1.1.1　Java 的发展历史

Java 语言由美国 Sun 公司于 1995 年发布，其主要设计者是 James Gosling。Java 语言最早来源于 Sun 公司的 Green 项目，目的是为家用电子消费产品开发的一个分布式代码系统，用于通过网络对家用电器进行控制。最初，Sun 公司的工程师们准备采用 C++，但由于 C++过于复杂并且安全性差，因此决定基于 C++开发一种新语言 Oak（Java 的前身）。Oak 是一种用于网络的、精巧而安全的语言。Sun 公司曾以此投标一个交互式电视项目，

结果被 SGI 打败，当时的 Oak 几乎无家可归。恰巧这时 Marc Andreessen 开发的 Mosaic 和 Netscape 启发了 Oak 项目组的成员，他们用 Java 语言编制了 HotJava 浏览器，并得到了 Sun 公司首席执行官 Scott MeNealy 的支持，Java 正式进军互联网。

### 1.1.2 Java 的特点

(1) 简单易用。Java 语言语法简单，抛弃了 C++中所有难以理解、容易混淆的特性，例如头文件、指针、结构、单元、虚拟基础类等。

(2) 可移植性。可移植性是 Java 语言的核心优势，Java 程序不需要修改就可以在 Windows、Linux、Mac OS 等不同平台上运行，只需要有对应的 JVM (Java Virtual Machine，Java 虚拟机) 就可以了。

(3) 面向对象。面向对象的特点使 Java 语言非常适合大型软件的设计和开发，Java 语言是完全面向对象的语言。

(4) 解释型语言。Java 语言是一种解释型语言，通过在不同平台上运行 Java 解释器，对 Java 代码进行解释，可以实现“一次编写，到处运行”的目标。

(5) 适合分布式计算。Java 语言是为 Internet 的分布式环境而设计的，它能处理 TCP/IP 协议，通过 URL (Uniform Resource Locator，统一资源定位符) 访问网络资源和访问本地文件一样简单。Java 语言还支持远程方法调用 (Remote Method Invocation，RMI)，使程序能够通过网络调用方法。

(6) 具有多线程处理能力。多线程可以实现更好的交互响应和实时行为。

### 1.1.3 Java 的版本

Java 具有 Java SE、Jave EE、Jave ME 三个版本，三个版本之间的关系如图 1-1 所示。

Java SE (Java Standard Edition)：标准版，前身是 J2SE (Java 2 Platform，Standard Edition)，2005 年更名为 Java SE，该版本定位于个人计算机的应用开发，是 Java 平台的基础核心，它提供了非常丰富的 API 来开发个人计算机上的一般应用程序，包括用户界面接口 AWT 和 Swing、网络功能、图像处理能力及输入输出支持等。

Java EE (Java Enterprise Edition)：企业版，前身是 J2EE (Java 2 Platform，Enterprise Edition)，2005 年更名为 Java EE，该版本定位于服务器端的应用开发，是 Java SE 的扩展版，增加了用于服务器开发的类库。例如，JDBC (Java DataBase Connectivity，Java 数据库连接) 定义了一个支持标准 SQL 功能的通用的应用程序编程 API，让开发者能直接在 Java 中使用 SQL 访问数据库中的数据；Servlet (Server Applet 的简称) 能够延伸服务器的功能，通过“请求—响应”的模式处理客户端的请求。

Java ME (Java Micro Edition)：微型版，前身是 J2ME (Java 2 Platform，Micro Edition)，2005 年更名为 Java ME，该版本定位于消费电子产品的应用开发。Java ME 包含 Java SE 的一部分核心类，也有自己的扩展类，增加了适合微小装置的类库 javax.microedition.io.* 等。该版本针对资源有限的电子消费产品的需求，精简核心类库，并提供了模块化的架构，让不同类型的产品能够随时增加丰富的功能。

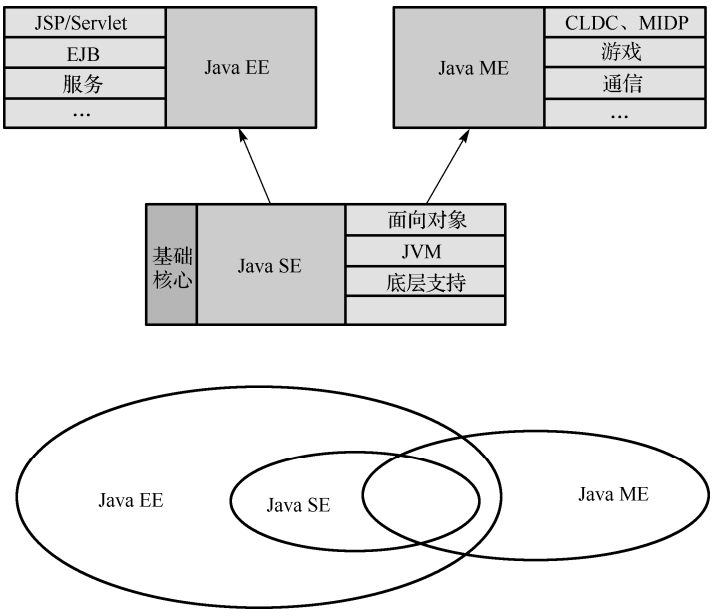


图 1-1　Java 语言三个版本之间的关系

## 1.2　Java 开发环境的配置

### 1.2.1　JDK 概述

JDK(Java Development Kit)是 Java 语言的软件开发工具包，主要用于开发移动设备、嵌入式设备上的 Java 应用程序。在进行 Java 程序设计开发前必须安装 JDK，本书中使用的版本是 JDK 7。JDK 是整个 Java 开发的核心，它包含 Java 的运行环境(JVM+Java 系统类库)和 Java 工具。Java 语言从诞生开始到现在，经历了一系列的发展过程，而 JDK 也在不断发展，其发展过程如表 1-1 所示。

表 1-1　JDK 的发展过程

| 版　本 | 发布时间 | 概　述 |
|---|---|---|
| JDK 1.1 | 1997 年 | 支持 JDBC、内部类、RMI、反射、Java Bean |
| JDK 1.2 | 1998 年 | 支持集合框架、JIT(Just In Time)编译器、对打包的 Java 文件进行数字签名、JFC(Java Foundation Classes，包括 Swing 1.0、拖放和 Java2D 类库)、Java 插件、JDBC 中引入可滚动结果集 |
| JDK 1.3 | 2000 年 | 支持 Java Sound API、jar 文件索引<br>对 Java 的各个方面都做了大量优化和增强 |
| JDK 1.4 | 2002 年 | 支持 XML 处理、Java 打印服务、Logging API、Java Web Start、JDBC 3.0 API |
| JDK 5 | 2004 年 | 支持泛型、增强循环、迭代方式、自动装箱与自动拆箱、类型安全的枚举、可变参数、静态引入、元数据(注解)、Instrumentation |
| JDK 6 | 2006 年 | 支持脚本语言、JDBC 4.0 API、Java Compiler API、可插拔注解、Native PKI(Public Key Infrastructure)、Java GSS(Generic Security Service)、Kerberos、LDAP(Lightweight Directory Access Protocol)、继承、Web Services |
| JDK 7 | 2011 年 | switch 语句块中允许以字符串作为分支条件，支持创建泛型对象时的应用类型推断，支持在一个语句块中捕获多种异常，支持动态语言 |

续表

| 版　本 | 发布时间 | 概　述 |
| --- | --- | --- |
| JDK 8 | 2014 年 | 支持 Lambda 表达式(Lambda 允许把函数作为一个方法的参数，传递进方法中)、方法引用(方法引用提供了非常有用的语法，可以直接引用已有 Java 类或对象的方法或构造方法，与 Lambda 联合使用，方法引用可以使语言的构造更紧凑简洁，减少冗余代码) |
| JDK 9 | 2017 年 | 支持模块化系统(Jigsaw 项目)、REPL(JShell)交互式编程环境 |
| JDK 10 | 2018 年 | 支持 var 局部变量类型推断、合并 JDK 多个代码仓库到一个单独的存储库中、统一的垃圾回收接口 |
| JDK 11 | 2018 年 | 支持 Nest-Based 访问控制、动态类文件常量、无操作垃圾收集器；改善 AArch64 intrinsic |
| JDK 12 | 2019 年 | 新增一个名为 Shenandoah 的垃圾回收器，通过在 Java 线程运行的同时进行疏散(evacuation)工作，来减少停顿时间 |

## 1.2.2　JDK 安装

JDK 可以在 Oracle 官网(www.oracle.com)上下载，如图 1-2 所示，本书使用 JDK 7 版本，操作系统是 64 位 Windows 操作系统，因此选择 64 位版本的 JDK 进行安装。若使用 32 位 Windows 操作系统，则选择对应的 32 位版本的 JDK 进行安装。


图 1-2　下载 JDK 页面

首先，进入 JDK 安装向导，单击“下一步”按钮，如图 1-3 所示。


图 1-3　JDK 安装向导

设置 JDK 的安装路径，为方便以后维护，将 JDK 安装到 D:\Java 目录下，如图 1-4 所示。


图 1-4　JDK 安装路径

安装 JDK 的过程中会提示是否安装 JRE(Java 运行环境)，其主要的功能是解释*.class 程序，此处的安装目的是要更新本机的 JRE 版本，不过 JDK 本身也是可以解释程序的。

安装完成之后会出现如图 1-5 所示的对话框，单击“关闭”按钮。


图 1-5　安装完成

## 1.2.3　环境变量配置

JDK 安装完成后，需要对系统的环境变量进行配制，具体操作步骤如下：右击“计算机”，在弹出的快捷菜单中选择“属性”命令，弹出“查看有关计算机的基本信息”的窗口，如图 1-6 所示。

在窗口左边面板中，选择“高级系统设置”选项，弹出“系统属性”对话框，如图 1-7 所示，选择“高级”选项卡。


图 1-6　查看有关计算机的基本信息

单击“环境变量”按钮，弹出“环境变量”对话框，如图 1-8 所示。


图 1-7　系统属性


图 1-8　环境变量

在“系统变量”列表框中，单击“新建”按钮，弹出“编辑系统变量”对话框，创建 JAVA_HOME 变量，变量值为 JDK 的安装路径，单击“确定”按钮，如图 1-9 所示。

在“系统变量”列表框中，选择 Path，单击“编辑”按钮，向 Path 变量中添加“%JAVA_HOME%\bin”，如图 1-10 所示。


图 1-9　编辑系统变量


图 1-10　编辑系统变量

在“系统变量”列表框中，单击“新建”按钮，弹出“编辑系统变量”对话框，创建

CLASSPATH 变量，变量值为“.;%JAVA_HOME%\lib;%JAVA_HOME%\lib\tools.jar”，单击“确定”按钮，如图 1-11 所示。


图 1-11　编辑系统变量

## 1.2.4　测试 JDK 配置

在 Windows 系统中，打开 CMD 命令行窗口，测试 Java 环境变量是否配置成功，执行“java -version”命令，如图 1-12 所示。


图 1-12　测试 Java 环境变量是否配置成功

若出现 Java 版本号，则表示配置成功。然后可以执行“javac”命令，出现其帮助提示信息，具体如图 1-13 所示。

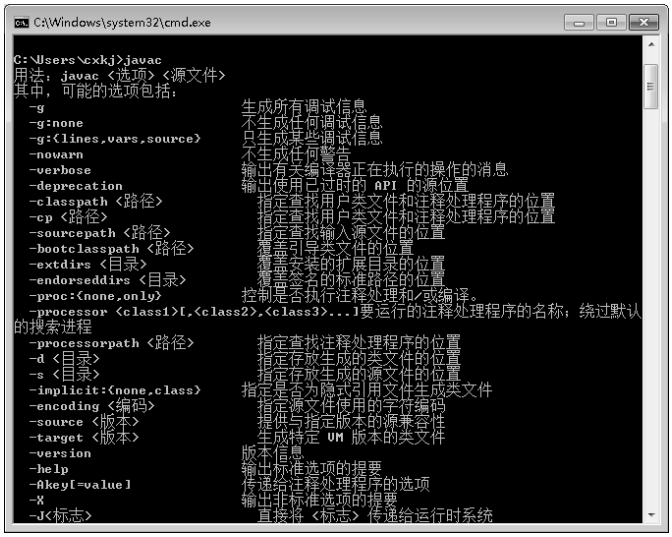


图 1-13　执行“javac”命令

若不能出现图中的提示信息，则表示环境变量配置不正确。

# 1.3 Java 程序运行机制

## 1.3.1 初识 Java 程序

使用 EditPlus 软件创建一个 Hello.java 的源文件，并输入以下代码，如图 1-14 所示。

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

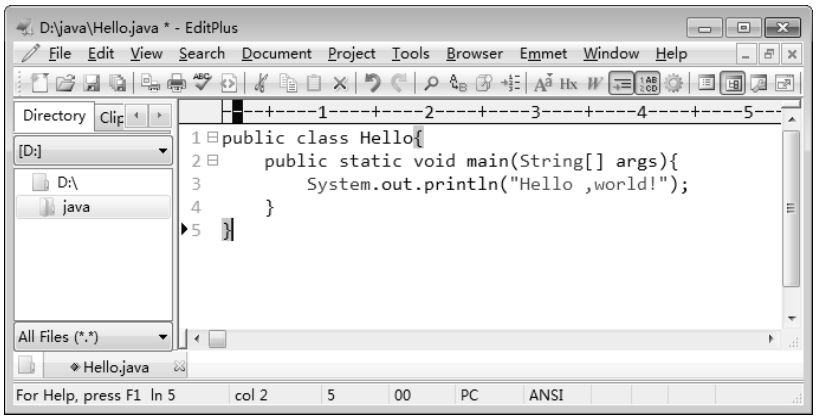


图 1-14　第一个 Java 程序

然后在 CMD 命令行窗口中执行“javac Hello.java”命令对源文件 Hello.java 进行编译，生成 Hello.class 字节码文件，如图 1-15 所示。

接下来，执行 Hello.class 字节码文件，执行“java Hello”命令，查看程序运行结果，具体如图 1-16 所示。

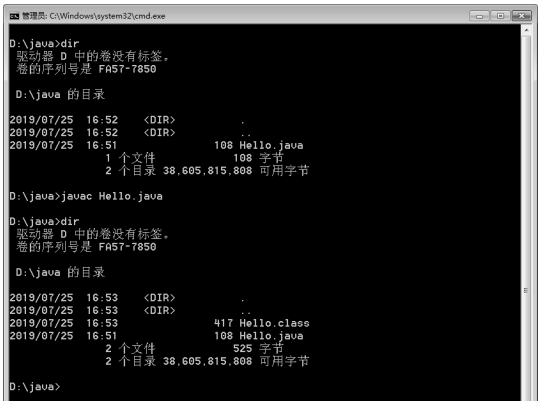


图 1-15　对源文件 Hello.java 进行编译


图 1-16　查看程序运行结果

计算机高级语言的类型主要有编译型和解释型两种，Java 语言将这两种类型有机地结合一起。Java 程序首先利用编译器(javac)将*.java 源文件编译成*.class 字节码文件，然后利用虚拟机解释器(java)将*.class 字节码文件解释执行，其执行过程如图 1-17 所示。

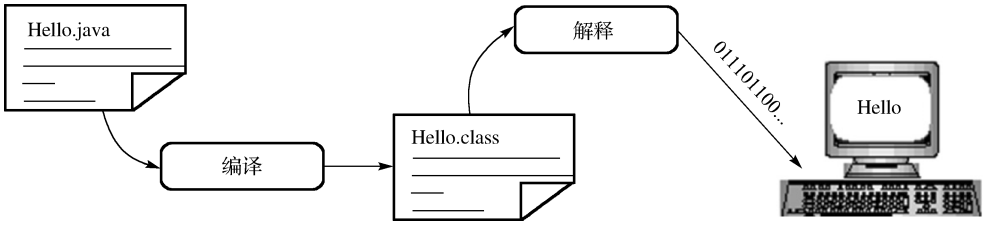


图 1-17　执行过程

JDK、JRE、 JVM 三者之间的关系如图 1-18 所示，可以看出，JDK 包含 JRE，JRE 包含 JVM。三者的概念可总结如下：

JDK(Java Development Kit)：Java 开发工具包，包含 JRE 及编译器和调试器等用于程序开发的文件。

JRE(Java Runtime Environment)：Java 运行环境，包含 JVM、库函数和运行 Java 应用程序所必需的文件。

JVM(Java Virtual Machine)：Java 虚拟机，它也定义了指令集、寄存器集、结构栈、垃圾收集堆、内存区域，负责解释运行*.class 字节码文件，边解释边运行。

Java 虚拟机机制是实现可移植性的核心机制。不同的操作系统有不同的虚拟机，Java 虚拟机机制屏蔽了底层运行平台的差别，使 Java 语言在不同平台上运行时不需要重新编译，实现了一次编写，到处运行，如图 1-19 所示。

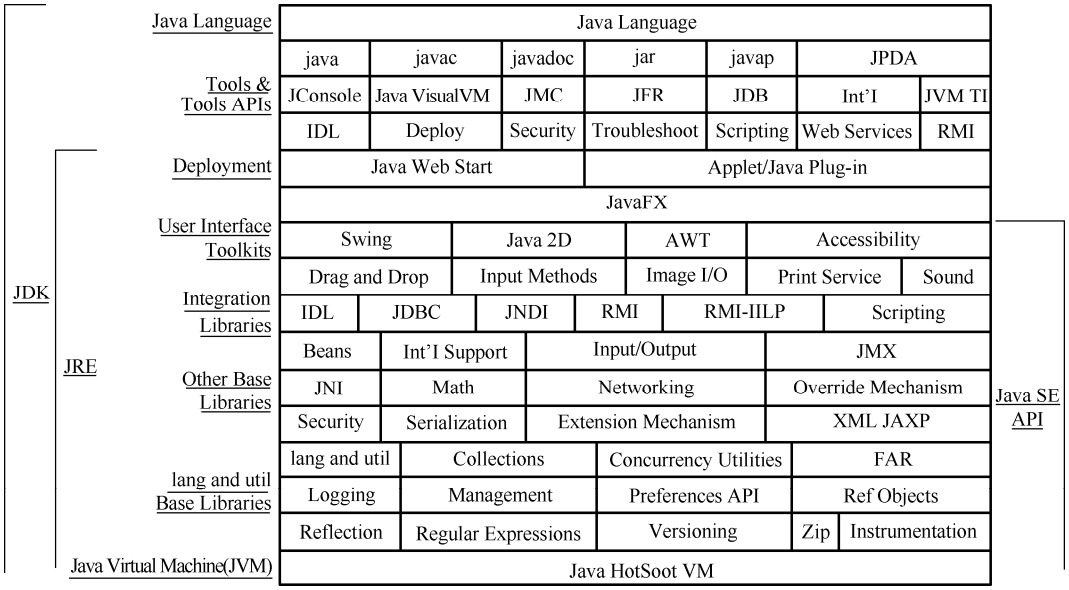


图 1-18　JDK、JVM、JRE 三者之间的关系

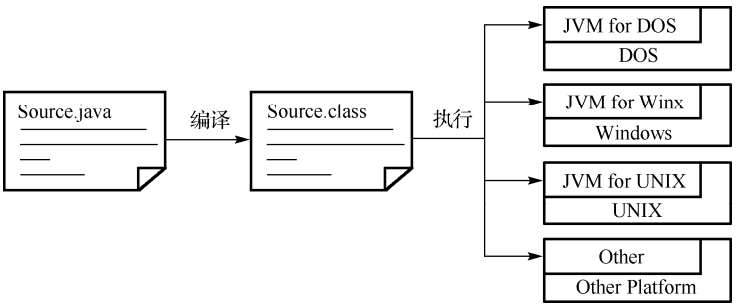


图 1-19　Java 虚拟机机制

可以看出，在 DOS、Windows、UNIX 等系统中，都有一台 JVM，所有的*.class 字节码文件都在 JVM 上运行，即*.class 字节码文件只认识 JVM，由 JVM 适应各个操作系统，读取并处理经过编译的、与平台无关的*.class 字节码文件。只要在不同的操作系统中安装适合其类型的 JVM，*.class 字节码文件不论在哪个操作系统中都可以正确执行，实现了程序的可移植性。

### 1.3.2　Java 的关键字

Java 的关键字对 Java 编译器有特殊的意义，它们用来表示某种数据类型、流程控制、权限控制等，关键字不能作为类名、对象名、变量名、方法名、类型名、数组名及包名。表 1-2 给出了 Java 的关键字。

表 1-2　Java 的关键字

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

### 1.3.3　Java 常用的基本工具

Java 常用的基本工具如下。

(1) javac：Java 编译器，将 Java 源文件*.java 编译成*.class 字节码文件。

(2) java：Java 虚拟机解释器，将*.class 字节码文件进行解释执行。

(3) appletviewer.exe：Java Applet 浏览器，appletviewer 命令可在脱离互联网浏览器环境的情况下运行 Applet。

(4) jar：Java 应用程序打包工具，可将多个类文件合并为单个 jar 归档文件。

(5) javadoc：Java API 文档生成器，从 Java 源程序代码注释中提取文档，生成 HTML 文档。

(6) JDB：Java 调试器(debugger)，可以逐行执行程序，设置断点。

(7) jps：查看 Java 虚拟机进程列表。

## 1.4　Java 开发工具

Eclipse 是由 IBM 公司推出的开源、免费的集成开发工具。提供程序编辑、程序编译、程序调试等功能，方便程序开发，提高实际的开发效率，简化程序设计中的很多操作。

Eclipse 安装程序可以从其官网(http://www.eclipse.org)上下载。Eclipse 是一个使用 Java 语言开发的工具软件，所以在安装 Eclipse 以前，一定要安装和配置 JDK。Eclipse 的安装很简单，只需解压缩后，运行可执行文件即可。

安装完成后，双击 eclipse.exe 或者对应的快捷方式图标，就可以启动 Eclipse 集成开发环境。

要创建一个 Java 项目，选择 File→New→Java Project，启动新项目的创建向导，如图 1-20 所示。在窗口中的 Project name 处输入项目名称(如 myproj)，然后单击“Finish”按钮，完成项目的创建。

项目创建后，就可以在这个项目中创建 Java 程序。一个 Java 项目中可以包含多个 Java 源文件，创建源文件的步骤：首先选择 File→New→Class 命令，弹出“New Java Class”窗口，如图 1-21 所示，然后在 Name 文本框处输入类名(如 Hello)，若希望自动创建 main()方法框架，则勾选 public static void main(String[] args)复选框，最后单击“Finish”按钮。

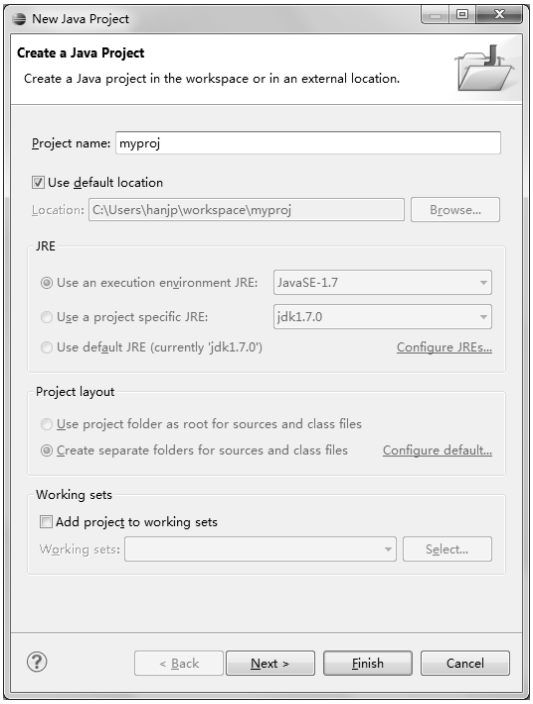


图 1-20　创建 Java 项目

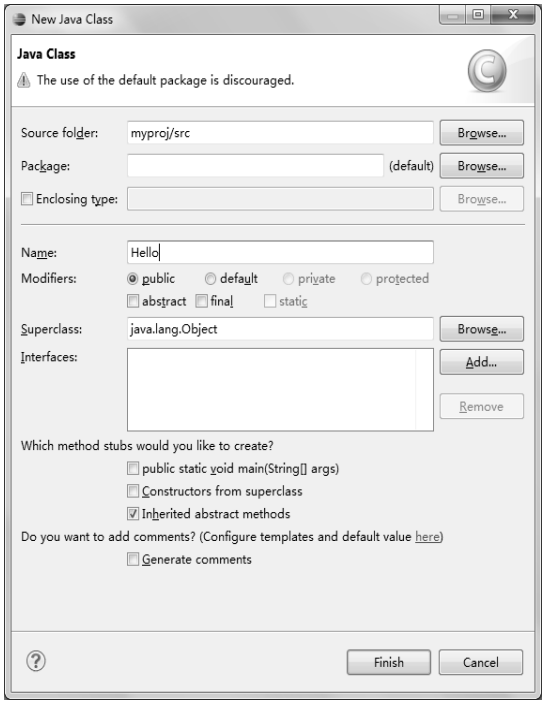


图 1-21　创建 Java 程序

Eclipse 会自动编译代码，如图 1-22 所示。若有语法错误，则以红色波浪线进行提示。

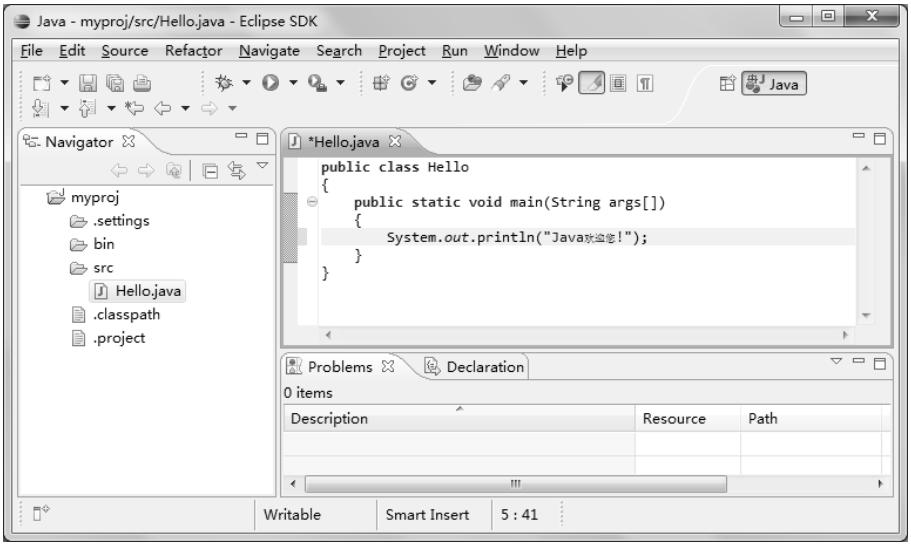


图 1-22　程序编辑

选择 Run→Run as→Java application 命令，运行程序，运行结果如图 1-23 所示。

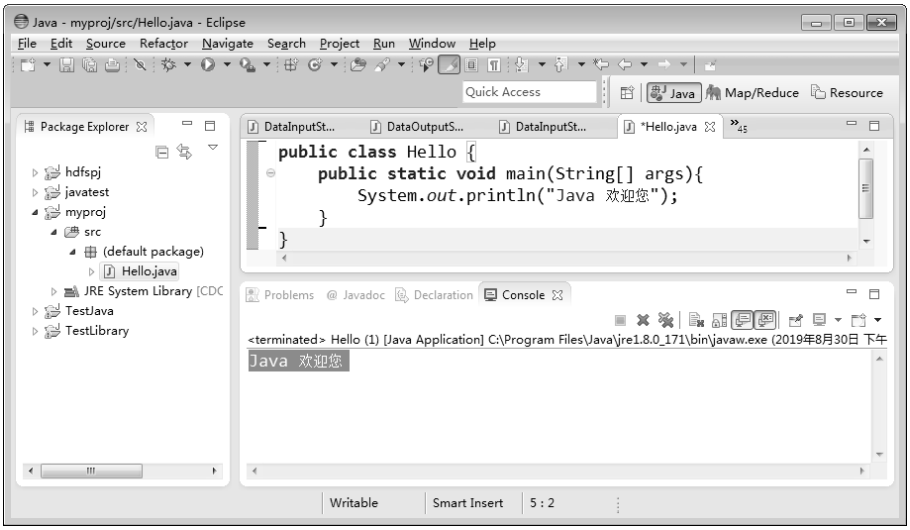


图 1-23　程序运行结果

## 1.5　Java API 文档

Java API（Application Programming Interface，,应用程序编程接口）是一些预先定义的函数，其目的是提供应用程序与开发人员基于某软件或硬件得以访问一组例程的能力，而又无须访问源码或理解其内部工作机制的细节。用户在调用 Java 预先定义的函数时，如果不了解其调用方法，可以查看 Java API 文档。

打开 Oracle 官网，进入下载页面，如图 1-24 所示，单击页面右侧区域中的“Java

Resources”下的“Java APIs”命令，在如图 1-25 所示的页面中，选择需要的版本，这里选择“Java SE 7”，进入 Java API 文档，如图 1-26 所示。

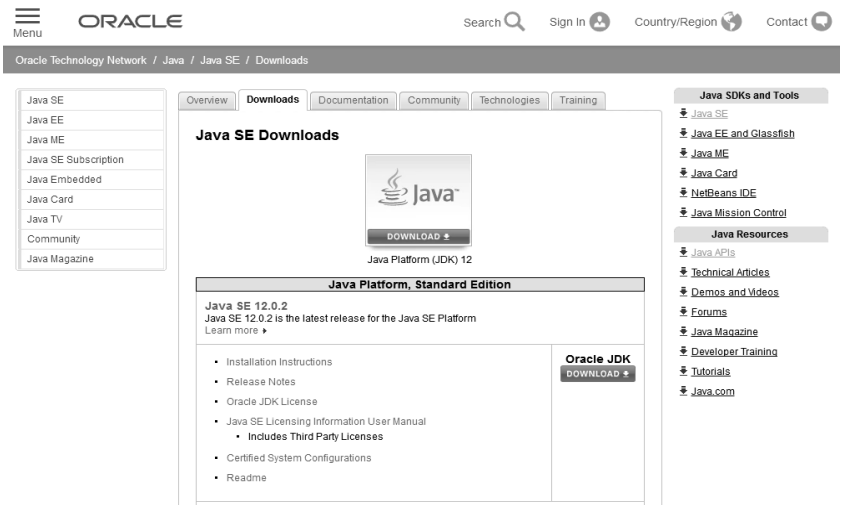


图 1-24　下载页面

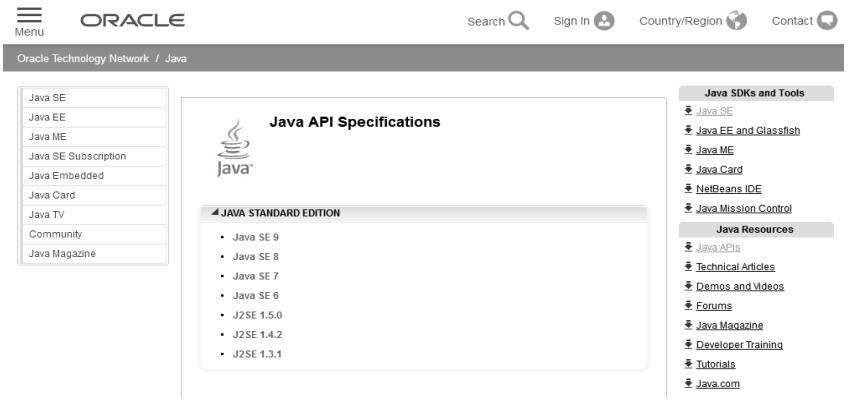


图 1-25　选择版本

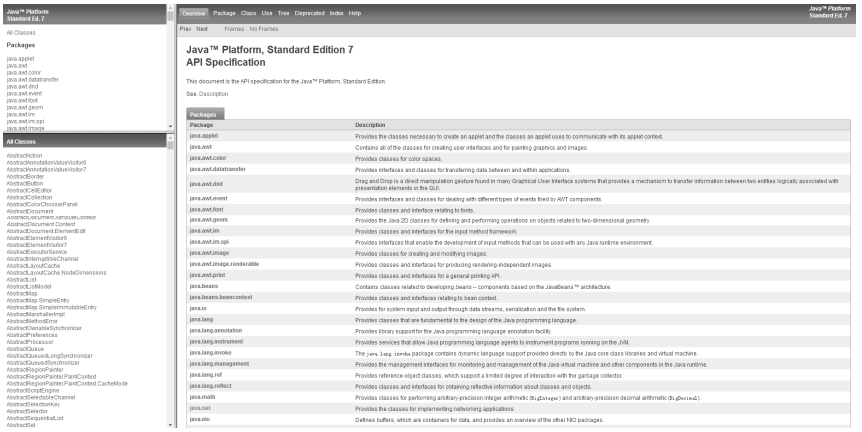


图 1-26　Java API 文档

## 1.6 本章习题

### 一、填空题

1．Java 程序的开发过程依次为编写代码、_____和解释运行。

2．JDK 用于开发 Java 程序，JRE 是 Java 运行环境，JVM 是_____的子集，JRE 是_____的子集。

3．JDK 的配置需要新建_____环境变量和修改_____环境变量。

4．Java 是_____的语言，所有代码必须位于类中，main()方法是 Java 应用程序的入口方法。

5．_____(Java Standard Edition)是 Java 标准版。

6．_____(Java Enterprise Edition)是 Java 企业版。

7．_____(Java Micro Edition)是 Java 微型版。

### 二、选择题

1．以下不是 Java 的特点的是(　　)。

A．与平台无关性　　B．高可靠性和安全性

C．指针运算　　D．分布式应用和多线程

2．下列选项中，对一个 Java 源文件进行编译，正确的语句是(　　)。

A．java Test　　B．javac Test

C．java Test.class　　D．javac Test.java

3．在 Java 语言中，(　　)是最基本的元素。

A．方法　　B．包　　C．对象　　D．接口

4．Java 语言属于(　　)。

A．面向机器的语言　　B．面向对象的语言

C．面向过程的语言　　D．面向操作系统的语言

5．阅读下列代码，该代码段正确的文件名为(　　)。

```
class Aa{
    void method1(){
        System.out.println("Method1 in class A");
    }
}
public class Bb{
    void method2(){
        System.out.println("Method2 in class B");
    }
    public static void main(String[] args){
        System.out.println("main() in class B");
    }
}
```

A．Aa.java　　B．Aa.class　　C．Bb.java　　D．Bb.class

## 三、编程题

1．编写 Java 程序，输出自己的专业、班级、姓名、年龄等信息。

2．编写 Java 程序，在屏幕上打印出以下内容：

```
**************************************
******** Java 语言程序设计教程 *******
**************************************
```

## 四、根据程序写结果

```
public class Test {
    public static void main(String[] args){
        System.out.println("This is my first Java Application!");
    }
}
```

程序运行结果：_____。