第1章 Java概述

目 录

contents


计 算 机 科 学 丛 书
PEARSON
原书第10版
Java语言程序设计
（基础篇）
[美] 梁勇（Y. Daniel Liang） 著
阿姆斯特朗亚特兰大州立大学
戴开宇 译
复旦大学
Introduction to Java Programming
Comprehensive Version Tenth Edition
INTRODUCTION TO
JAVA
PROGRAMMING
Y. Daniel Liang
机械工业出版社
China Machine Press


计 算 机 科 学 丛 书
Pearson
原书第10版
Java语言程序设计
（进阶篇）
[美] 梁勇（Y. Daniel Liang） 著
阿姆斯特朗亚特兰大州立大学
戴开宇 译
复旦大学
Introduction to Java Programming
Comprehensive Version Tenth Edition
INTRODUCTION TO
JAVA
PROGRAMMING
Y. Daniel Liang
机械工业出版社
China Machine Press

1.1 了解Java历史及特征

- Java语言是由Sun公司于1995年5月23日正式推出的面向对象的程序设计语言，集安全性、简单性、易用性和平台无关性于一身，适合网络环境下跨平台编程使用。
- 最大的优势就是跨平台运行，源代码被编译成.class文件后，把.class放到不同的平台上直接运行，代码的可移植性非常好。

1.1 了解Java历史及特征

◆Java语言具有以下特征

- 简单：Java源自于C++,但做了很多的简化。例如，取消了多重继承、指针、自动垃圾回收。
- 面向对象：纯粹的面对对象程序设计语言，没有面向过程语言的全局变量、全局函数等。
- 分布式：Java程序可以在多台计算机上协同计算，可以基于Java RMI，Java RPC编写分布式应用程序
- 解释性：Java的源程序被编译成字节码，在Java虚拟机上解释运行，因此效率不如C++。
- 健壮性： Java取消了指针，对数组下标越界进行检查，垃圾自动回收，具有运行时的异常处理功能。
- 安全性：从网络上下载的Applet程序，在Java的安全机制保护下，不会破坏本地系统。
- 与体系结构无关： Write once, run anywhere。
- 可移植性好： Write once, run anywhere。
- 高性能：基于Java的分布式计算环境能够应付高并发的服务请求
- 多线程： Java在语言级别（例如synchronized关键字）支持多线程编程，不需要额外的线程库。

1.2 JAVA语言规范，API，JDK，IDE

Java语言规范,API,JDK,IDE

◆Java语言规范(Java Language Specification, JLS)

Java语言的语法和语义技术性定义，对语言的语法、语义最权威的解释。

◆应用程序接口(Application Program Interface, API)

Java预定义类和接口。要熟练使用Java，必须熟悉API。

◆Java开发工具包(Java Development Kit, JDK)

包含软件库、基于命令行的编译器、解释器以及其它工具。

如javac（编译Java程序）、java（运行Java程序）、jdb（调试Java程序）

◆集成开发环境(Integrated Development Environment, IDE)

在一个图形界面中，完成工程管理、源代码编辑、编译、调试和在线帮助等功能。

1.2 JAVA语言规范，API，JDK，IDE

Java平台版本

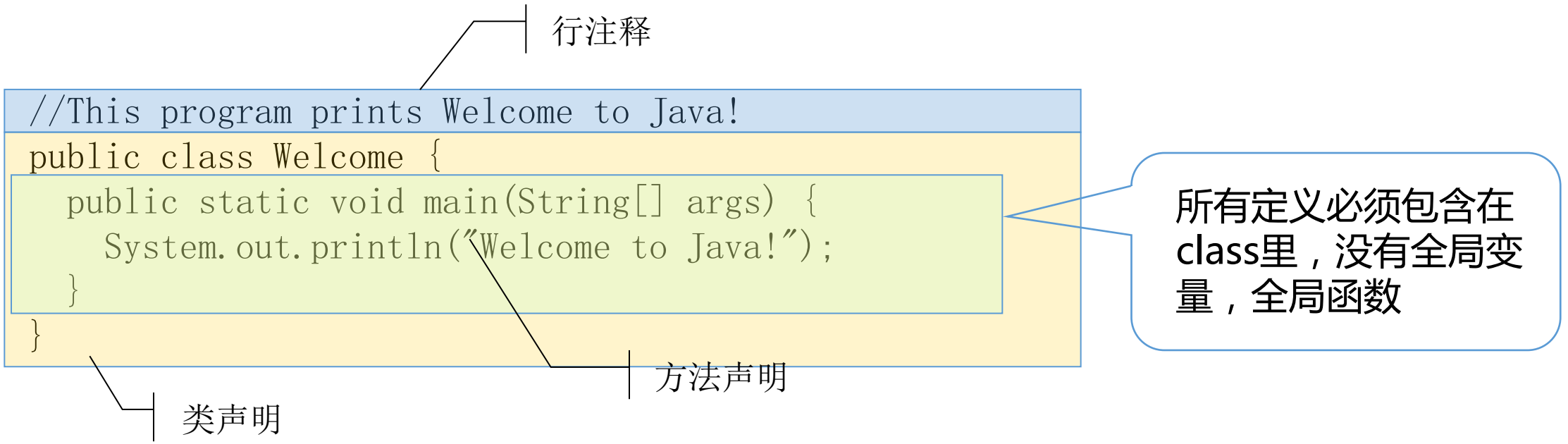
◆Java SE(Java Flatform, Standard Edition)
以前称为 J2SE。主要用于开发和部署在桌面使用的 Java 应用程序。

◆Java EE(Java Flatform, Enterprise Edition)
以前称为 J2EE。它帮助开发和部署可移植、健壮、可伸缩且安全的服务器端 Java 应用程序。Java EE 是在 Java SE 的基础上构建的，它提供 Web 服务、组件模型、管理和通信 API。

◆Java ME(Java Flatform, Micro Edition)
以前称为 J2ME。它为在移动设备和嵌入式设备（比如手机、PDA、电视机顶盒和打印机）上运行的应用程序提供一个健壮且灵活的环境。

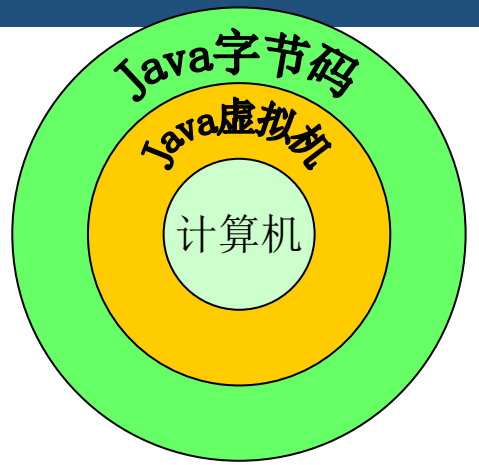
1.3 第一个简单的JAVA程序

在控制台中输出“Welcome to Java!”



一个类必须包含public static void main(String[ ])方法才能作为程序启动类
public static void main(String[ ])方法是程序的入口

1.4 JAVA运行环境：JVM



◆Java的目标代码可以在任何平台上运行。

◆Java的源代码编译之后生成.class文件，由字节码(bytecode)构成。

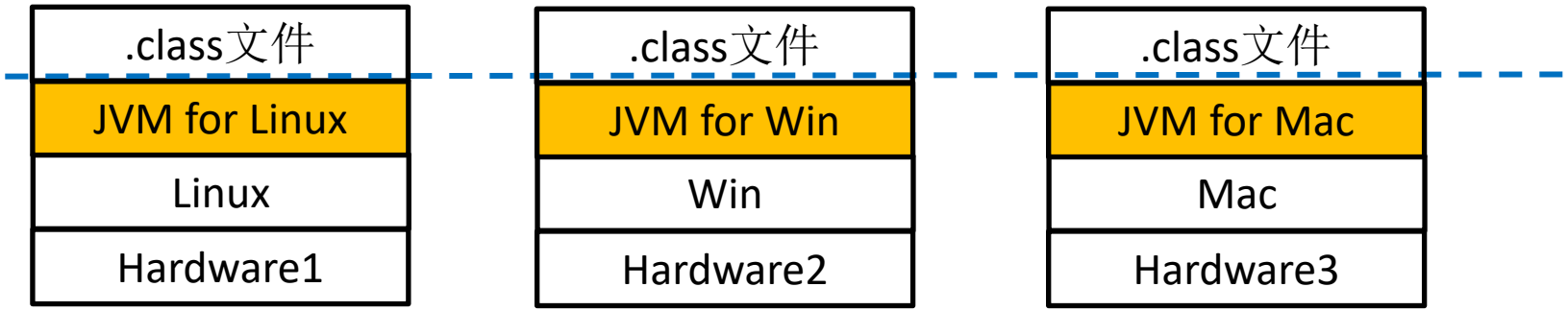
◆字节码可以在任何装有Java虚拟机(Java Virtual Machine)的计算机上运行。

◆Java虚拟机是一个用于解释字节码的软件，有一套虚拟的CPU指令集及汇编指令，,class文件包含了JVM的CPU指令集。

1.4 JAVA运行环境：JVM

从JVM层之上看，
三台机器是同构的
因此同一个.classe文件
可以在不同机器上到处
运行

Write Once
Run Anyware



1.5 安装JDK、创建、编译和执行JAVA程序

下载JDK

◆下载机器OS对应的JDK（官网: www.oracle.com，下载最新最稳定的Java SE版本）

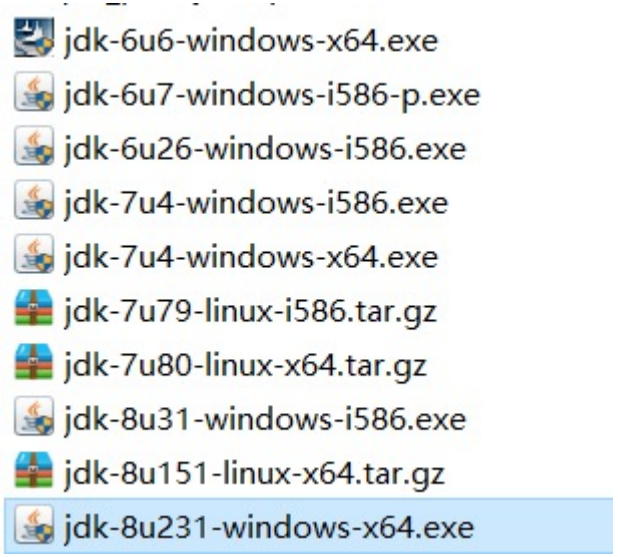
◆https://www.oracle.com/technetwork/java/javase/downloads/index.html，目前最新版本是Java SE 15.0.2

◆注意区分不同OS的JDK：JDK for Win、 JDK for Linux、JDK for MAC

◆除了下载JDK，建议下载该JDK对应的API帮助文档

◆一定要下载64位的JDK

◆不管机器实际物理内存多大，32位的JVM最大内存只能用3G



1.5 安装JDK、创建、编译和执行JAVA程序

安装JDK

◆安装JDK，记住安装目录，例如安装在D:\jdk1.8.0_231_64bit下

◆可以在机器上安装多个版本的JDK，不同版本的JDK安装在不同的目录下

◆通过环境变量JAVA_HOME可以切换不同JDK版本

◆JDK安装目录下内容为

bin目录包含命令行工具，如javac 、java、jdb

jre开头的目录包含Java Runtime Environment（JRE）如果不做开发，只是运行Java程序，则只需安装JRE即可

Java SE库的源代码

1.5 安装JDK、创建、编译和执行JAVA程序

配置JDK相关的环境变量

◆安装好JDK后，需要配置环境变量（以Windows为例，其它OS类似）

◆首先配置环境变量JAVA_HOME，指向JDK的安装目录，如D:\jdk1.8.0_231_64bit

◆将%JAVA_HOME%/bin路径加到环境变量PATH里，这样在命令行窗口里可以在任何位置调用JDK里面的命令行工具，如javac 、java等

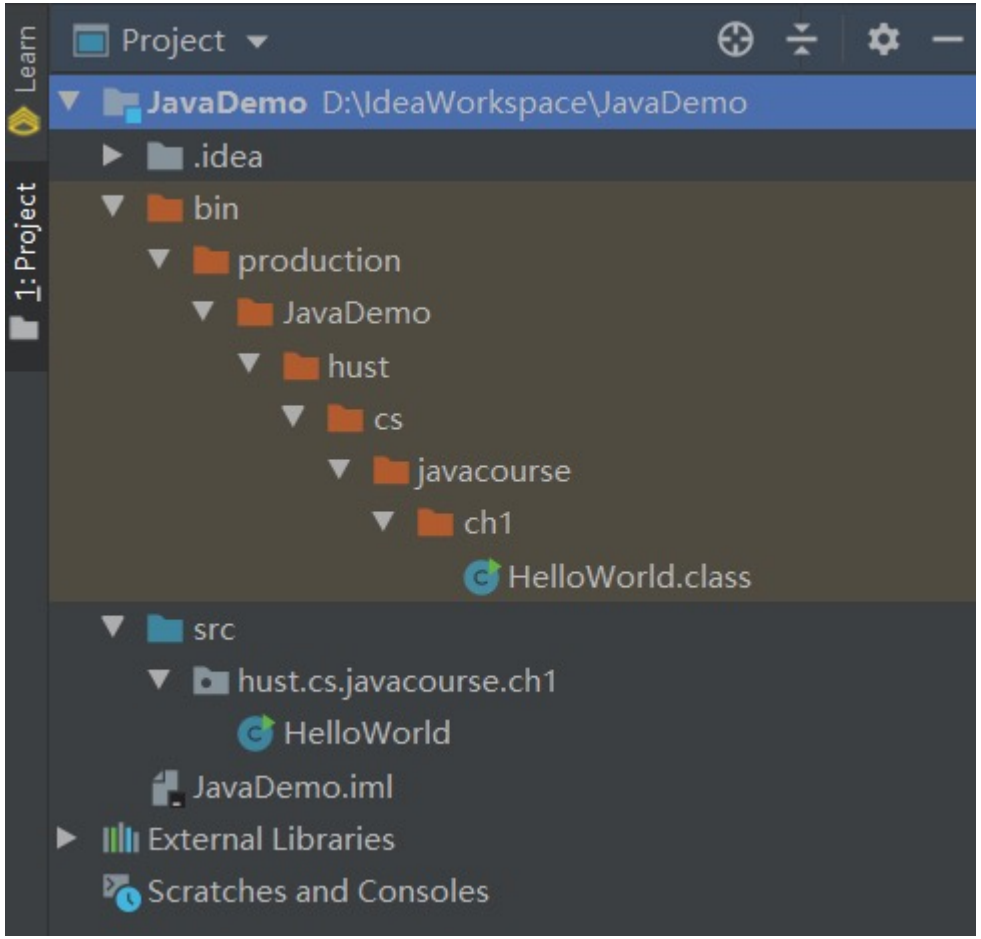
◆在通过命令行运行Java程序时，还需要配置环境变量CLASSPATH， CLASSPATH的作用是让JVM能够找到要运行的Java类。在设置CLASSPATH时，可以指定class文件所属包的顶级目录；如果多个class文件被打包到.jar文件，则一定要指定到jar文件（而不是jar文件所在的目录，因为.jar文件本质上就是一个目录）

1.5 安装JDK、创建、编译和执行JAVA程序

基于shell脚本配置环境变量示例

- ◆假设JDK安装在D:\jdk1.8.0_231_64bit
- ◆用IDEA创建的一个JAVA工程目录如右图所示
- ◆在src目录下创建package：hust.cs.javacourse.ch1
- ◆在package下创建类HelloWorld如下图所示
- ◆编译完成后，HelloWorld.class文件的位置如右图所示

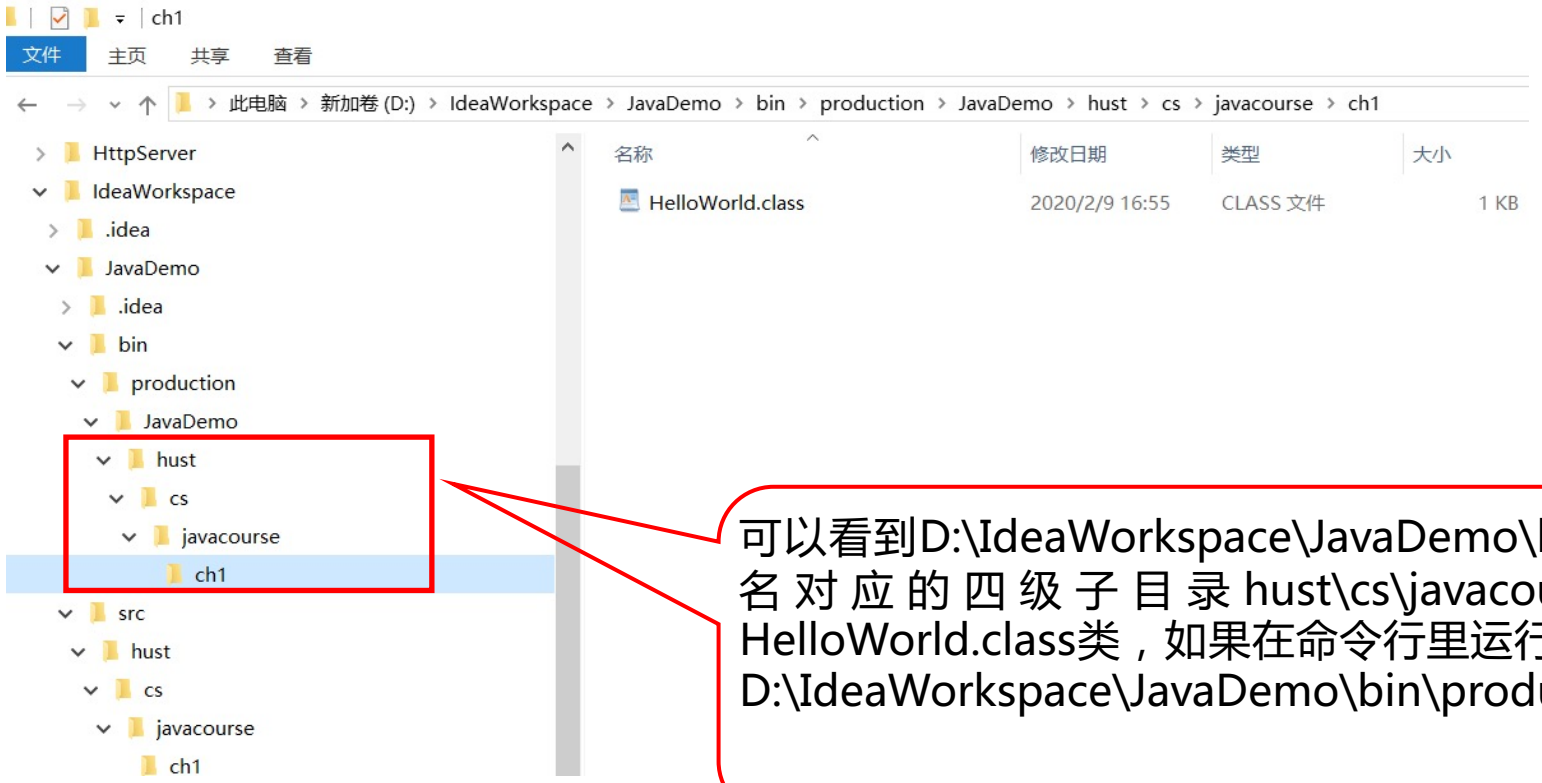
```
package hust.cs.javacourse.ch1;

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



1.5 安装JDK、创建、编译和执行JAVA程序

基于shell脚本配置环境变量示例

◆这个工程在资源管理器里的视图如图所示



可以看到D:\IdeaWorkspace\JavaDemo\bin\production\JavaDemo下面包含了和包名对应的四级子目录hust\cs\javacourse\ch1，在里面包含了编译得到的HelloWorld.class类，如果在命令行里运行这个类，只需要把这个类所在包的顶级目录D:\IdeaWorkspace\JavaDemo\bin\production\JavaDemo加到CLASSPATH里

1.5 安装JDK、创建、编译和执行JAVA程序

基于shell脚本配置环境变量示例

◆基于前面的工程目录，可以在直接在命令行运行的一个脚本（如run.bat文件）如下所示：

```
set JAVA_HOME=D:\jdk1.8.0_231_64bit
set PROJECT_HOME=D:\IdeaWorkspace\JavaDemo
set path=%path%;%JAVA_HOME%\bin
set classpath=%classpath%;%PROJECT_HOME%\bin\production\JavaDemo

java -classpath %classpath% hust.cs.javacourse.ch1.HelloWorld
```

设置JAVA_HOME环境变量

设置PROJECT_HOME环境变量

把JAVA_HOME目录的子目录bin加到环境变量PATH

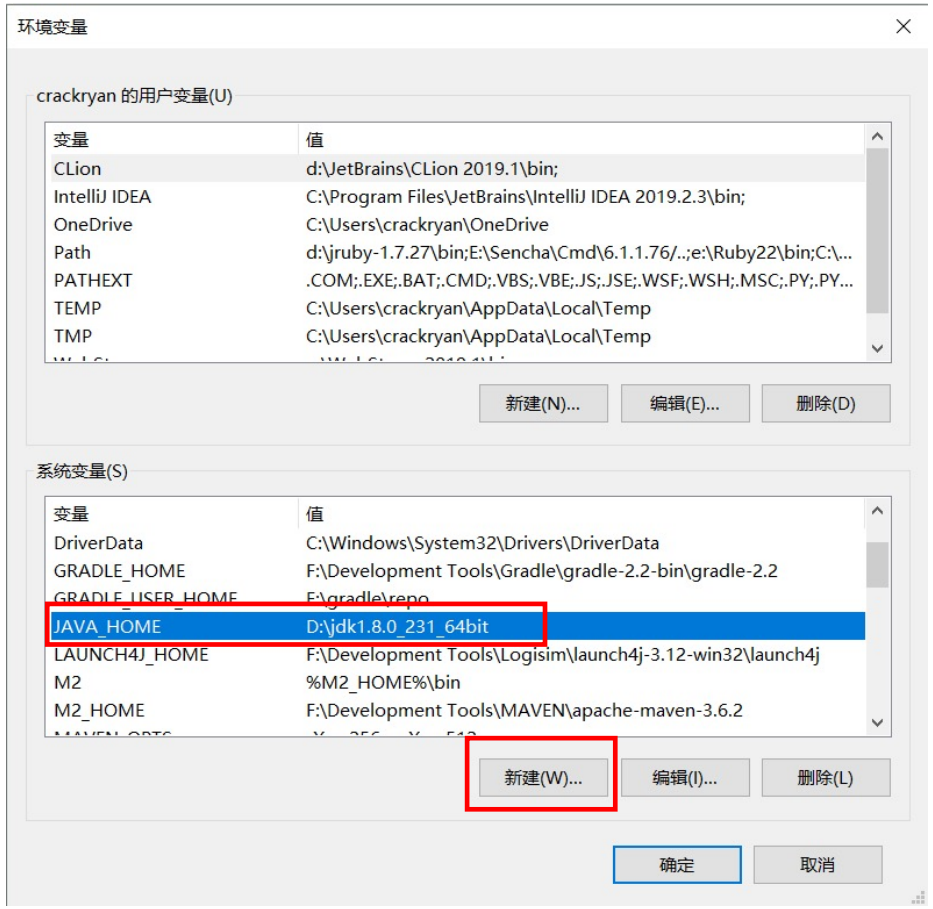
把PROJECT_HOME目录的子目录bin\production\JavaDemo加到环境变量CLASSPATH，这个目录是类HelloWorld所属包的顶级目录

启动类时，用类的完全限定名(带包名限定），并且带-classspath选项

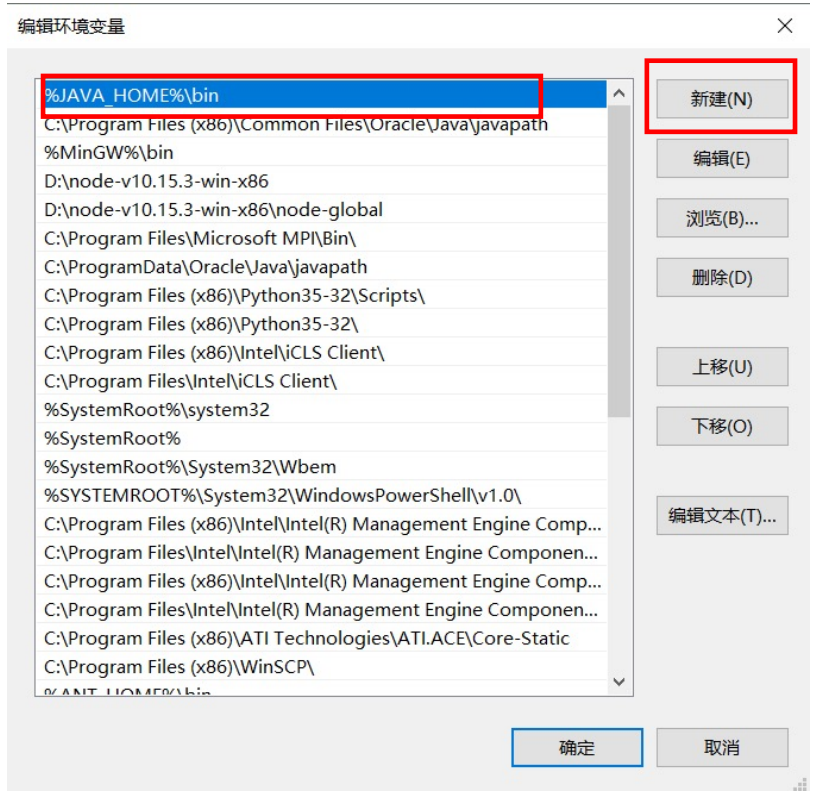
1.5 安装JDK、创建、编译和执行JAVA程序

在Windows里图形化方式设置环境变量

打开系统属性窗口，设置环境变量



新建JAVA_HOME环境变量



添加%JAVA_HOME%\bin到环境变量PATH

1.5 安装JDK、创建、编译和执行JAVA程序

◆创建和编辑源代码

可以使用任何文本编译器创建和编辑源代码。

文件名必须与公共的类名一致，文件后缀为.java

如果有多个类，且没有public类，文件名可与任一类名相同

一个JAVA源文件最多只能有一个public类

如果一个Java源文件包含多个class，那么编译后会产生多个.class文件

◆安装好JDK，配置好环境变量

◆编译

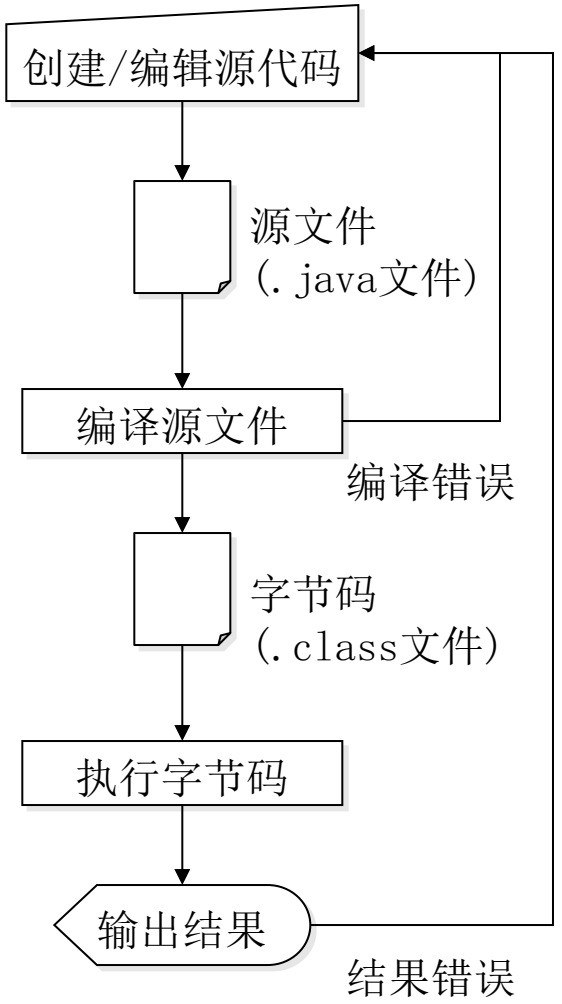
在控制台中执行：javac　源文件名（要保证OS能找到该文件）

生成.class字节码文件

◆执行

在控制台中执行： java　启动类完全限定类名（要保证JVM能找到这个类）

启动类必须有public static void main(String[] args)函数



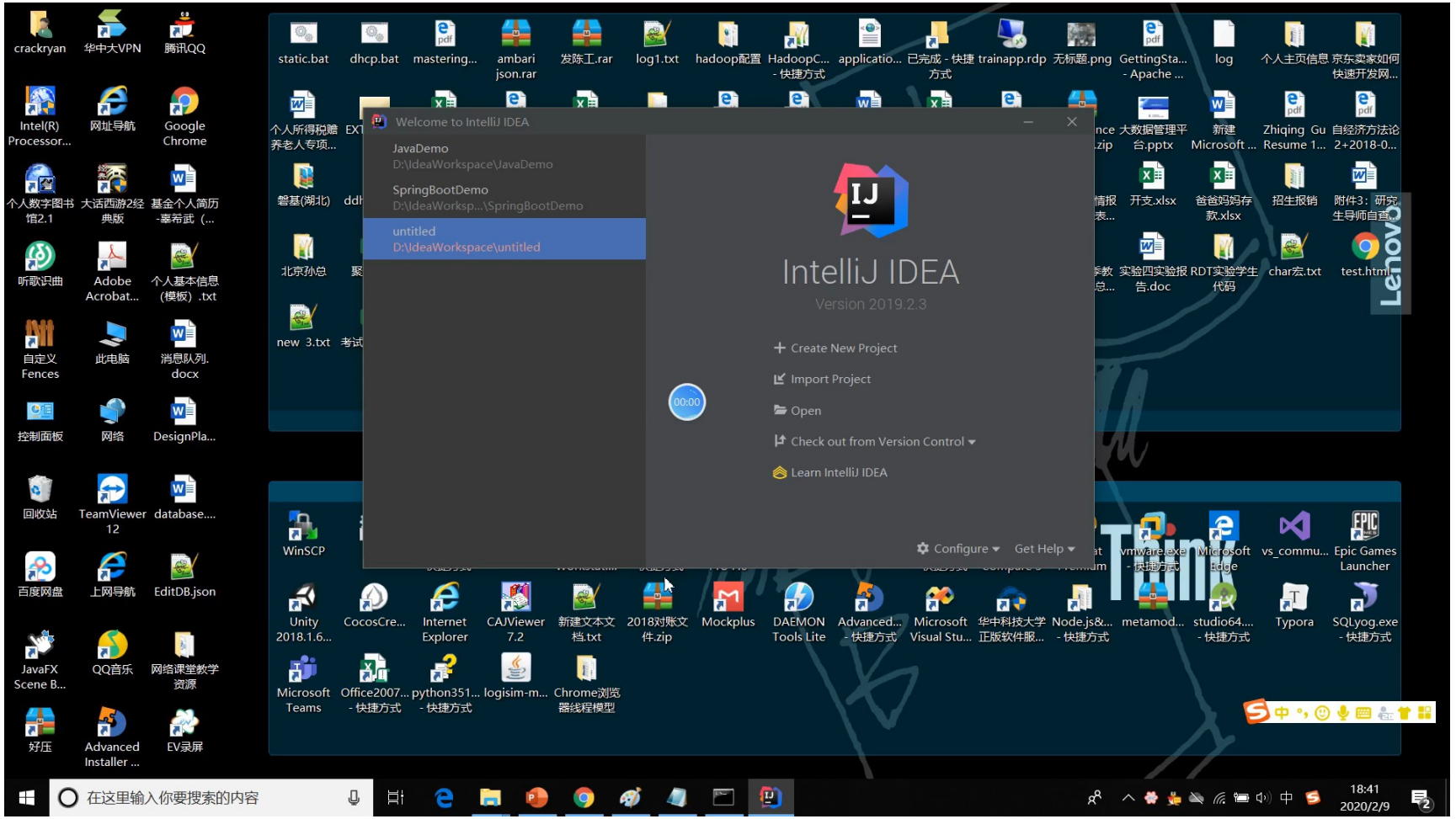
1.5 创建、编译和执行JAVA程序

在IntelliJ IDEA编写源码、编译、执行

JetBrain

1.5 创建、编译和执行JAVA程序

在IntelliJ IDEA创建工程

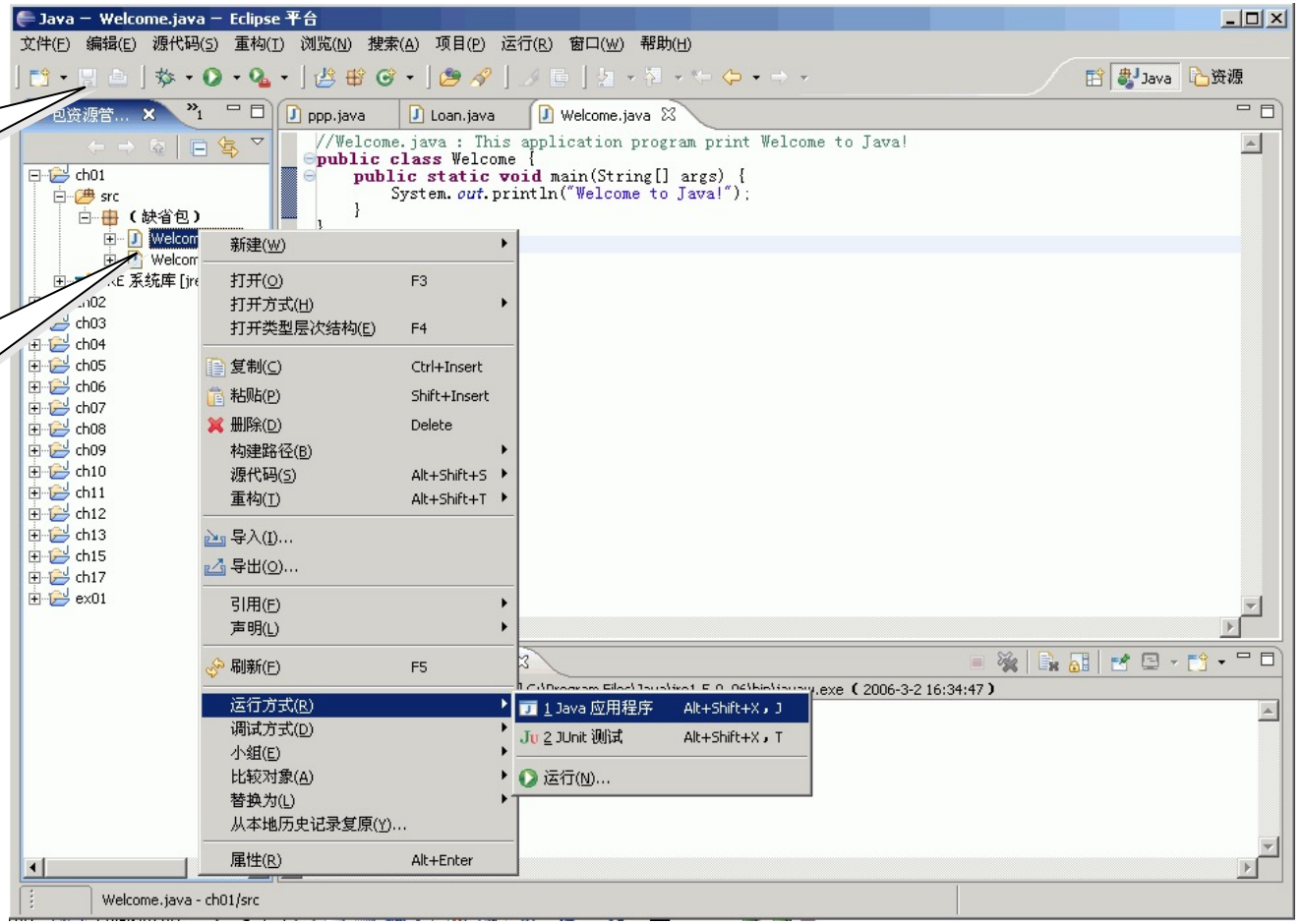


1.5 创建、编译和执行JAVA程序

在Eclipse中（开源）编译执行

保存源文件将自动执行编译

在鼠标右键菜单中选择*运行方式→Java应用程序*



1.6 JAVA程序剖析

- 注释
- 保留字
- 修饰符
- 语句
- 块
- 类
- 方法
- main方法

1.6 JAVA程序剖析

注释

◆Java程序包含三种注释

- 多行注释：用于某行的一部分、单行或多行注释

```
/* 注释内容
      注释内容
*/
```

- 单行注释：用于单行或不到一行的注释

```
// 注释内容
```

- 文档注释：可以使用javadoc提取注释，自动形成API文档

```
/** 注释内容
      注释内容
*/
```

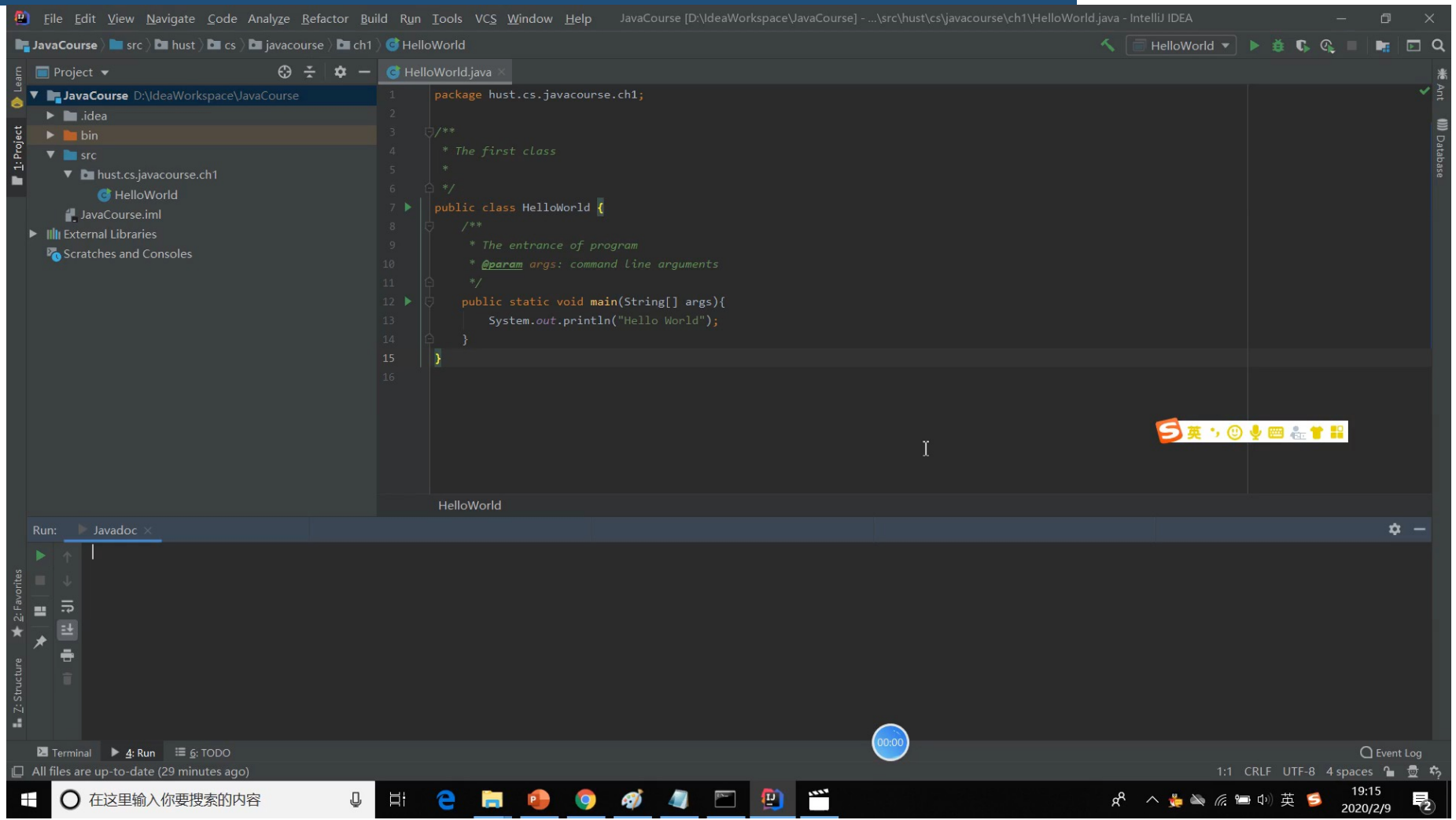
1.6 JAVA程序剖析

Javadoc注释的作用和范围

- Javadoc注释的作用是对代码的功能进行注释，同时自动形成API文档，注释的对象可以是
  - 类，特别是public 类
  - 类的方法，特别是类的公有方法（静态方法，实例方法）
  - 类的数据成员，特别是公有数据成员（静态成员、实例成员）

1.6 JAVA程序剖析

Javadoc自动生成



1.6 JAVA程序剖析

保留字

- 保留字或关键字(keyword)是对编译器有具体意义，不能在程序中用于其它目的的单词。
- 程序员定义的标识符不能是保留字。

```
//This program prints Welcome to Java!
public class Welcome {
  public static void main(String[] args) {
    System.out.println("Welcome to Java!");
  }
}
```

1.6 JAVA程序剖析

修饰符

◆Java中的某些关键字称为修饰符(modifier)，用于指定数据、方法、类的属性以及它们的用法。

◆常见修饰符：public, protected, private, static, abstract, final。

```
//This program prints Welcome to Java!
public class Welcome {
  public static void main(String[] args) {
    System.out.println("Welcome to Java!");
  }
}
```

1.6 JAVA程序剖析

语句

◆语句(statement)代表一个动作或一系列动作。

Java中的每个语句都以分号(;)结尾。

```
//This program prints Welcome to Java!
public class Welcome {
  public static void main(String[] args) {
    System.out.println("Welcome to Java!");
  }
}
```

◆要注意区分表达式（expression）和语句的区别

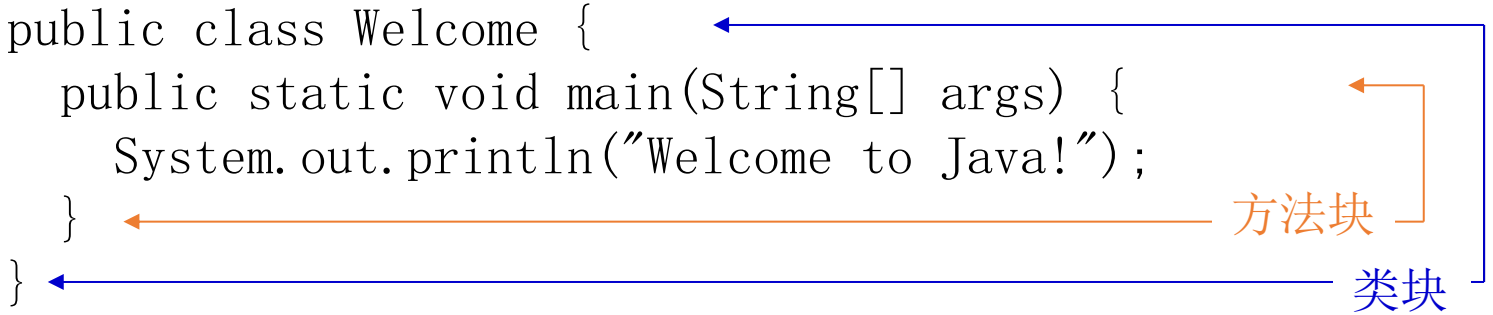
- ◆ i = 1 是赋值表达式
- ◆ i = 1; 是语句

1.6 JAVA程序剖析

块

◆程序中成对的大括号形成一个块(block)，是用于组织程序的重要部件。块是程序设计语言里非常重要的概念，它决定了变量的作用域(scope)

```
//This program prints Welcome to Java!
public class Welcome {
  public static void main(String[] args) {
    System.out.println("Welcome to Java!");
  }
}
```



1.6 JAVA程序剖析

类

◆类(class)是Java的基本结构，类是对象的模板或蓝图。一个程序可以包含一个或多个类。一个JAVA源文件里最多只有一个公有类。

```
//This program prints Welcome to Java!
public class Welcome {
  public static void main(String[] args) {
    System.out.println("Welcome to Java!");
  }
}
```

1.6 JAVA程序剖析

方法：必须通过对象或类调用

◆System.out是标准输出流对象，println是该对象的一个方法，该方法向标准输出流（显示屏）显示字符。括号中的字符串是方法的参数。

```
//This program prints Welcome to Java!
public class Welcome {
  public static void main(String[] args) {
    System.out.println("Welcome to Java!");
  }
}
```

1.6 JAVA程序剖析

main方法及init方法

◆Java解释器通过调用main方法执行应用程序。main方法是Java应用程序（Java Application）的入口。 Java有二种类型的程序。

- Application：由操作系统通过启动类的main方法启动。可以启动任意一个类，被启动的类必须有公有的静态的main方法
- Applet：只能嵌在网页里，在浏览器里运行。没有main方法，入口为init( )。

```
//This program prints Welcome to Java!
public class Welcome {
  public static void main(String[] args) {
    System.out.println("Welcome to Java!");
  }
}
```

1.6 JAVA程序剖析

在消息对话框中显示文本

◆可以使用JOptionPane的showMessageDialog方法显示消息文本。

```
import javax.swing.JOptionPane;

public class WelcomeInMessageDialogBox {
  public static void main(String[] args) {
    JOptionPane.showMessageDialog(null,
        “Welcome to Java!”,
        “Example 1.2 Output”,
       JOptionPane.INFORMATION_MESSAGE);
  }
}
```

1.6 JAVA程序剖析

在消息对话框中显示文本