第 1 章　第一个 Java 程序

且说孙悟空帮助唐僧到西天取到真经后，就在天上逍遥自在地当起了斗战胜佛。不知过了多少年头，有一次，他到人间游玩，发现大部分人无论上班或生活，都要摆弄一个名叫“电脑”的玩艺。悟空聪明好学，很快就掌握了玩电脑的本领。

如今，悟空会熟练地运行安装在 Windows 操作系统中的各种可执行程序，利用它们来完成特定任务。例如通过浏览器程序来上网，通过记事本程序来编辑文档，通过画图程序来画画，通过计算器程序来进行数学运算。

有一天，悟空正在网上东游西逛，花果山的小猴智多星跑过来，对悟空说：“孙爷爷，我看这电脑上的程序都是给人玩的，要是您也能编写点程序出来，专门给俺们猴儿耍耍，那该多好啊！”

智多星的想法正合悟空的心意。悟空想：要是自己学会了编程，就可以开发出符合猴子趣味的游戏给儿孙们耍耍，等到编程功底扎实了，还可以给花果山也开发个网站呢。

悟空因为特别喜欢本作者以前所写的《Java 面向对象编程》一书，就打算用 Java 语言来编程。在本章，悟空小试牛刀，编写了一个最简单的 Java 程序。通过这个程序，大家将跟随悟空一起来熟悉创建、编译和运行 Java 程序的过程，还将初步了解 Java 语言的面向对象（OO，Object Oriented）的基本思想，以及跨操作系统平台的特性。

1.1　程序的基本概念

悟空想让电脑模仿智多星说话。如图 1-1 所示，悟空先用猴语幽默地对电脑大声喊：“嘿，伙计，帮我在屏幕上打印一行字符串‘大家好，我是智多星’。”可是电脑置若罔闻，它又不是猴脑，哪懂得悟空的猴语啊。悟空宽宏大量地对电脑笑笑，经过西天取经的磨练，悟空已经改了动不动就亮出金箍棒唬人的毛病。他明白，要与对方交流，就首先要熟悉对方使用的语言。

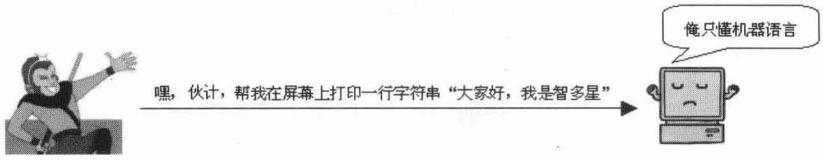

图 1-1　悟空试图用猴语与电脑对话

电脑作为硬件，只懂得由“1”和“0”排列组合成的机器语言。机器语言又复杂又枯燥，悟空可不想学呆板的机器语言。幸运的是，悟空可以不必直接用机器语言和电脑对话，而只需和电脑中的操作系统软件对话进行了。

图 1-2 演示了悟空运行 Windows 操作系统中的 IE 浏览器程序的过程。悟空只需双击 IE 浏览器程序图标，聪明的 Windows 操作系统就明白了悟空的意图，它就会与电脑交互，请求电脑执行 IE 浏览器程序。Windows 操作系统就像悟空与电脑之间的翻译员。

图 1-2　悟空与 Windows 操作系统对话，让其运行 IE 浏览器程序

悟空如果想让电脑模仿智多星说话，只需要编写一个会打印字符串的程序，接下来让操作系统来运行这个程序就行了。以 Windows 操作系统为例，它的可执行程序通常都是以“.exe”作为扩展名的文件。这些可执行程序中包含了二进制的操作指令，这些操作指令只有 Windows 操作系统才能看得懂。悟空显然没耐心去学习这些和机器语言一样枯燥乏味的操作指令。

幸运的是，悟空可以用高级编程语言来编程，高级编程语言与猴语在语法上更加接近，比较容易掌握。不过，操作系统可不懂高级编程语言，因此，还必须想办法把用高级编程语言编写出来的源程序转换为操作系统看得懂的可执行程序，这个转换的过程就叫做编译。

如图 1-3 所示，悟空用高级编程语言编写了一个源程序，接下来用现成的编译器程序把源程序编译为可执行程序，然后让操作系统来运行这个可执行程序。

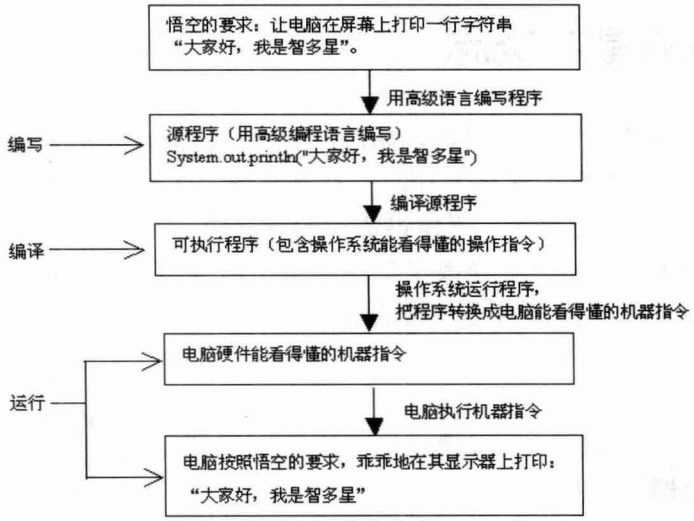

图 1-3　程序的编写、编译和运行过程

从图 1-3 可以看出，悟空要想让电脑能听从它的各种吩咐，主要的任务就是掌握一门高级编程语言，然后用它来编写源程序。

1.2　Java 程序的基本运行原理

在 1.1 节讲过，操作系统看不懂用高级编程语言编写出来的源程序，但是能看得懂编译生成的可执行程序。那么，对于同一个可执行程序文件，是不是所有的操作系统都能看得懂呢？答案是否定的。例如， IE 浏览器只能在 Windows 操作系统中运行，到了 Linux 操作系统中就无法运行，这是因为 IE 浏览器的可执行程序文件中包含了只有 Windows 操作系统才能看得懂的操作指令。

如图 1-4 所示，假如悟空想编写一个在 Windows 和 Linux 操作系统中都能运行的程序，那么就需要把源程序分别编译成适合这两种操作系统的可执行程序。

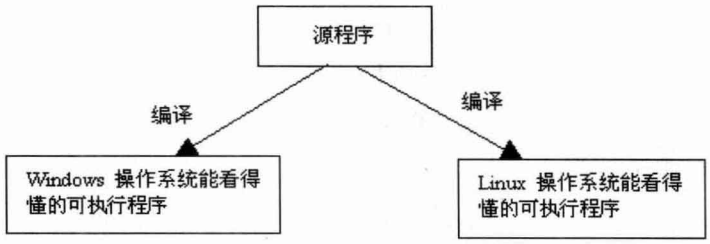

图 1-4　把源程序分别编译成适合 Windows 和 Linux 的可执行程序

悟空觉得这样做还是有点麻烦，要是有一种编程语言可以跨操作系统平台就好了，这意味着只需对用这种语言编写出来的源程序编译一次，编译出来的可执行程序能够在所有的操作系统中运行。刚好 Java 语言就是悟空所期望的跨操作系统平台的高级编程语言。

Java 语言为何会有跨操作系统平台的本领呢？这还得归功于 Java 虚拟机。Java 虚拟机这名字好玄乎！Java 虚拟机看不见摸不着，到底算何方神圣？它可不是工厂里的庞大无比的机器，其实它本身也不过是一个可执行程序，这个可执行程序的任务就是运行 Java 程序。

如图 1-5 所示，Java 虚拟机程序本身不是跨操作系统平台的，对于不同的操作系统，有着不同的 Java 虚拟机可执行程序。不过，不管是哪个操作系统中的 Java 虚拟机，它们的任务都是一样的，该任务就是请求底层操作系统运行 Java 程序。

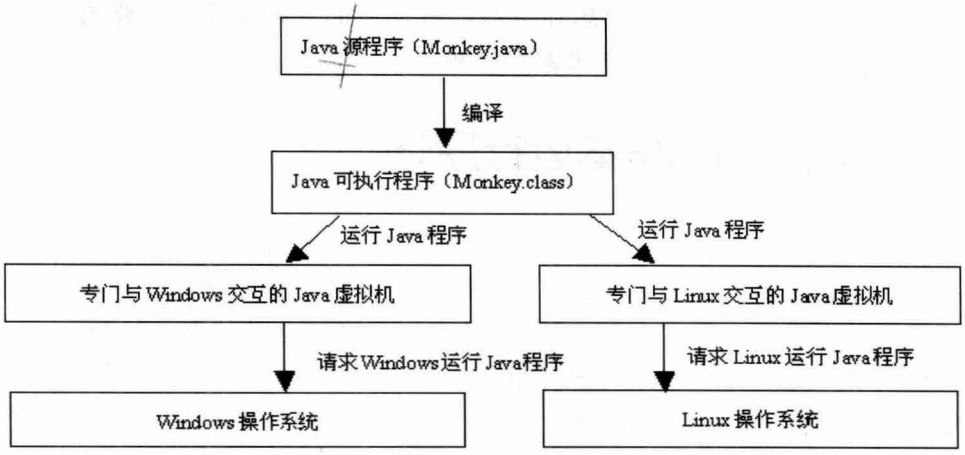

图 1-5　Java 程序的跨操作系统平台运行的过程

从图 1-5 可以看出，Java 源程序是以“.java”作为扩展名的文件，编译生成的可执行程序是以“.class”作为扩展名的文件。Java 可执行程序本身不能直接在操作系统中运行，它必须借助 Java 虚拟机才能运行。Java 可执行程序中包含了只有 Java 虚拟机才能看得懂的二进制字节码，而 Windows 和 Linux 操作系统都无法直接看得懂这些二进制字节码。

以下图 1-6 进一步展示了 Java 虚拟机执行 Java 程序的过程。

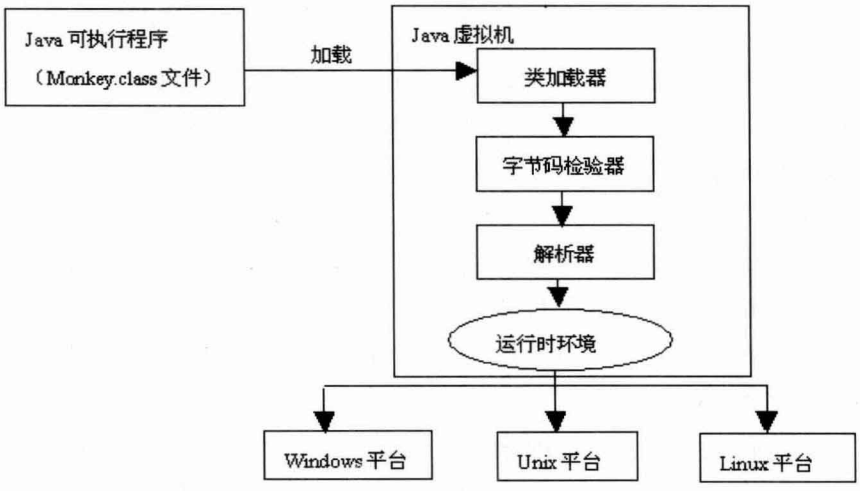

图 1-6　Java 虚拟机执行 Java 程序的过程

从图 1-6 可以看出，Java 虚拟机封装了底层操作系统的差异，不管是在哪种平台上，都按以下同样的步骤来运行程序。

（1）加载 Java 类，即把类的.class 文件中的二进制字节码加载到内存中。

（2）对类的二进制字节码进行验证。

（3）解析并请求操作系统执行指令。

1.3　创建面向对象的 Java 源程序

悟空经过一番认真学习，掌握了 Java 语言的基本语法，接着就猴急猴急地要在儿孙们面前演示 Java 语言的神奇妙用了。有一天，悟空把智多星叫过来，故作神秘地对智多星说："请你如实回答我的以下问题。"

悟空："你属于猫类、鸡类、狗类，还是猴类啊？"

智多星："当然是猴类啦。"

悟空："你叫什么名字？"

智多星："智多星呀。"

悟空："你能做哪些事？"

智多星："吃饭，说话，睡觉，爬树，杂耍……"

悟空接下来得意地对智多星说："我要把你装到电脑里。"智多星惊讶地瞪大了眼睛："孙爷爷啊，您神通广大，能把金箍棒变成针眼般大小，再把它塞到耳朵里。莫非您也要把孩儿变成芝麻般大小，再把孩儿塞到这电脑里？"悟空哈哈大笑："你想歪啦。我要在电脑里创建一个虚拟的智多星，它也能像你一样说话。"智多星兴奋得跳了起来："太好了，孩儿以前在电脑里看电视剧《西游记》，看到有您、唐僧和猪八戒等，还有白骨精呢，唯独没有我。这回终于也能过把进电脑的瘾了。"

1.3.1　定义 Monkey 类

悟空打开文本编辑器（如 Windows 中的记事本程序），编写了一个如例程 1-1 所示的 Java 源程序，文件名为 Monkey.java。

例程 1-1　Monkey.java

```
public class Monkey{                          //定义代表花果山猴类的 Monkey 类
  String name;                                //定义名字属性

  public Monkey(){}                           //不带参数的构造方法

  public Monkey(String name){                 //带参数的构造方法
    this.name=name;                           //设置 Monkey 对象的 name 属性
  }
```

```
    public void speak(){                    //定义模拟猴子说话的 speak()方法
      //猴子给大家打招呼
      System.out.println("大家好，我是"+name);
    }
}
```

悟空告诉智多星，在这个 Monkey.java 文件中定义了一个 Monkey 类，它代表咱们花果山上的猴类。Monkey.java 文件主要由以下内容构成：

（1）类的声明语句：

```
public class Monkey{……}
```

以上代码指明类的名字为“Monkey”，public 修饰符意味着这个类可以被公开访问。类的主体内容都位于大扩号“{}”内。

Tips

在本书中，“声明”和“定义”有着相同的含义。例如，“声明 Monkey 类”和“定义 Monkey 类”的意思相同；“声明 name 变量”和“定义 name 变量”的意思也相同。

（2）类的属性（也称为成员变量）的声明语句：

```
String name;
```

所有的猴子都有名字，用 Monkey 类的 name 属性来表示。name 属性为 String 字符串类型。

（3）方法的声明语句和方法主体：

```
public void speak(){
  System.out.println("大家好，我是"+name);
}
```

所有的猴子都具有说话的行为，Monkey 类的 speak()方法就用来模拟猴子说话的行为。speak()方法的大括号中的内容称为方法主体，以上方法主体中程序代码的作用是打印“大家好，我是 XXX”。

Monkey.java 文件中以“//”开头的文字代表注释，它不是程序代码，而是用于解释说明程序代码，从而便于编程人员理解程序代码。对于 Monkey.java 文件中 public 和 static 等修饰符，暂且不用细究它们的用途。到目前为止，读者只要大致了解 Monkey 类主要由 name 属性和 speak()方法构成就行了。

1.3.2 创建 Monkey 对象

在上一节，悟空已经定义了一个 Monkey 类。这个 Monkey 类是所有猴子的模板，每个猴子都是 Monkey 类的一个实例，例如猴子智多星就是 Monkey 类的一个

实例。Monkey 类具有 name 属性和 speak()方法，那么依照 Monkey 类模板生成的所有 Monkey 实例也会具有 name 属性和 speak()方法。例如，猴子智多星的 name 属性为“智多星”，speak()方法的作用是打印字符串“大家好，我是智多星。”

类的实例也叫做对象，对象是对现实世界中的各种实体的模拟。以下 Java 程序代码创建了一个代表智多星的 Monkey 对象：

```
//定义一个 Monkey 类型的引用变量 m
Monkey m;
//创建一个代表智多星的 Monkey 对象，并且使引用变量 m 引用这个对象
m=new Monkey("智多星");
```

以上程序通过 new 语句创建了一个 Monkey 对象，并且使引用变量 m 引用这个对象。本书第 4 章的 4.4.3 节（用 new 关键字创建对象）将进一步介绍 new 语句的作用。如图 1-7 所示，当程序运行时，这个用 new 语句创建的 Monkey 对象位于电脑的内存中，它占用了一定的内存空间。变量 m 引用这个 Monkey 对象：

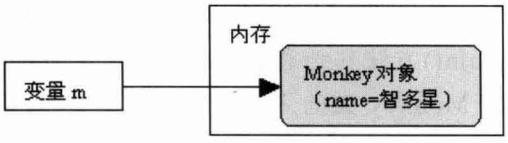

图 1-7　变量 m 引用内存中的 Monkey 对象

Tips

内存用来存放电脑在执行程序时所处理的数据。例如电脑执行“100+200”的数学运算时，数据 100、200，以及运算结果 300 都会先后存放在内存中。当程序运行结束，与该程序相关的数据就会被全部清除，从而及时释放这些数据占用的内存空间。Java 作为面向对象的编程语言，所处理的数据主要以对象及其属性等形式存在。当 Java 程序运行时，对象存在于内存中。

new 语句会调用 Monkey 类的构造方法，“new Monkey("智多星")”语句调用 Monkey 类的带参数的 Monkey(String name)构造方法，该构造方法把参数“智多星”赋值给 Monkey 对象的 name 属性：

```
public Monkey(String name){  //带参数的构造方法
  this.name=name;  //设置 Monkey 对象的 name 属性
}
```

Monkey 类还有一个不带参数的构造方法：

```
public Monkey(){}  //不带参数的构造方法
```

以下程序通过不带参数的构造方法创建 Monkey 对象：

```
m=new Monkey();  //通过不带参数的构造方法创建 Monkey 对象
m.name="智多星"; //设置 Monkey 对象的 name 属性
```

Monkey 对象创建好以后，就可以调用它的方法。以下程序代码调用 Monkey 对象的 speak()方法，来模拟智多星说话：

```
m.speak();
```

由于变量 m 引用代表智多星的 Monkey 对象，因此，“m.speak()”就会调用代表智多星的 Monkey 对象的 speak()方法。

以下程序代码创建了两个 Monkey 对象，分别代表猴子智多星和猴子小不点，然后再分别调用它们的 speak()方法：

```
Monkey m1=new Monkey("智多星");
Monkey m2=new Monkey("小不点");
m1.speak(); //智多星说话：大家好，我是智多星
m2.speak(); //小不点说话：大家好，我是小不点
```

1.3.3 程序入口 main()方法

Java 源程序中包含了许多代码，当程序运行时，到底从哪一行代码开始运行呢？Java 语言规定，以 main()方法作为程序的入口点。所有的 Java 程序都是从 main()方法开始运行的。悟空在 Monkey 类的末尾增加了一个作为程序入口的 main()方法：

```
public class Monkey{
  ……
  public static void main(String[] args) {
    Monkey m=new Monkey("智多星");  //创建代表智多星的 Monkey 对象
    m.speak();  //智多星说话
  }
}
```

以上 args 是 main()方法的参数，它属于 String 数组类型（String[]），第 17 章的 17.3 节（创建数组对象）将顺带介绍这个方法参数的用法。

作为程序入口的 main()方法必须同时符合以下 4 个条件：

- 必须使用 public 修饰符。
- 必须使用 static 修饰符。
- 必须有一个 String 数组类型的参数。
- 返回类型为 void。void 表示方法没有返回值。

在类中可以通过重载的方式提供多个不作为应用程序入口的 main()方法。关于方法重载的概念参见本书第 7 章的 7.2 节（方法重载）。例如，在以下例程 1-2 的 Tester 类中声明了多个 main()方法。

例程 1-2 Tester.java

```
public class Tester {
  /** 程序入口 main 方法 */
```

```
    public static void main(String args[]){……}

    /** 非程序入口 main 方法 */
    public static void main(String arg) {……}
    private int main(int arg) {……}
}
```

例程 1-2 的 Tester 类中包含 3 个 main()方法，第一个方法是作为程序入口的方法，其他两个方法是合法的普通方法，但不能作为程序入口。

Tips

在本书中，“合法”与“非法”是专有名词。“合法”是指程序代码正确，可以通过编译；“非法”是指程序代码有错误，无法通过编译。

1.4　编译和运行 Java 程序

现在，悟空已经创建了 Monkey.java 源程序。接下来，悟空要把它编译为 Monkey.class 类文件，然后就劳驾 Java 虚拟机来运行这个 Monkey 类。如图 1-8 所示，编译 Java 源程序需要有专门的 Java 编译器程序，运行 Java 程序需要有 Java 虚拟机程序。那么，Java 编译器程序和 Java 虚拟机程序在哪里呢？在 JDK 里面。

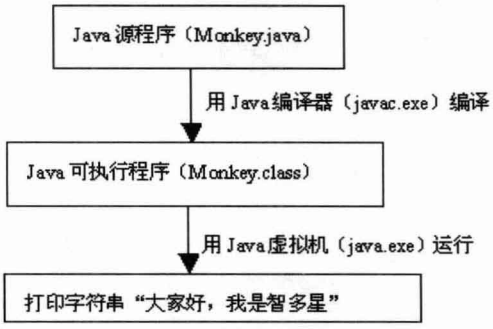

图 1-8　编译和运行 Java 程序

1.4.1　JDK 简介

JDK 是 Java Development Kit（Java 开发工具包）的缩写，由 SUN 公司提供。它为 Java 程序提供了基本的开发和运行环境。JDK 还可以称为 JavaSE（Java Standard Edition，Java 标准开发环境）。JDK 的官方下载地址为：http://java.sun.com。此外，在 JavaThinker.org 网站上也提供了 JDK 的下载，网址为：http://www.javathinker.org/download/software/jdk.rar。

JDK 主要包括以下内容：

- Java 虚拟机程序：负责解析和运行 Java 程序。在各种操作系统平台上都有相应的 Java 虚拟机程序。在 Windows 操作系统中，该程序的文件名为 java.exe。
- Java 编译器程序：负责编译 Java 源程序。在 Windows 操作系统中，该程序的文件名为 javac.exe。
- JDK 类库：提供了最基础的 Java 类及各种实用类。java.lang、java.io、java.util、java.awt 和 javax.swing 包中的类都位于 JDK 类库中。关于 Java 包的概念参见第 2 章的 2.5 节（包声明语句）。

假定 JDK 安装到本地后的根目录为 C:\jdk，在 C:\jdk\bin 目录下有一个 java.exe 和 javac.exe 文件，它们分别为 Java 虚拟机程序和 Java 编译器程序。

为了便于在 DOS 命令行下直接运行 Java 虚拟机程序和 Java 编译器程序，可以把 C:\jdk\bin 目录添加到操作系统的 PATH 系统环境变量中。在 Windows 操作系统中，选择【控制面板】→【系统】→【高级】→【环境变量】。接下来就可以编辑 PATH 系统环境变量了，参见图 1-9。

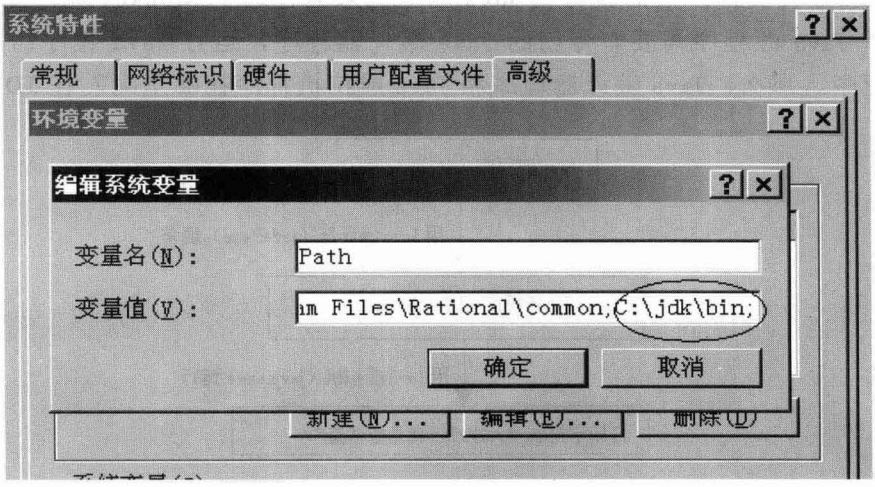

图 1-9　在操作系统的 PATH 系统变量中添加 C:\jdk\bin 目录

1.4.2　本范例的目录结构

本章范例位于 sourcecode\chapter01 目录下，读者可以把 chapter01 目录复制到本地硬盘的 C:\目录下。为了便于管理 Java 源文件及 Java 类文件，悟空决定把所有的 Java 源文件放在 src 子目录下，把所有的 Java 类文件放在 classes 目录下，参见图 1-10。

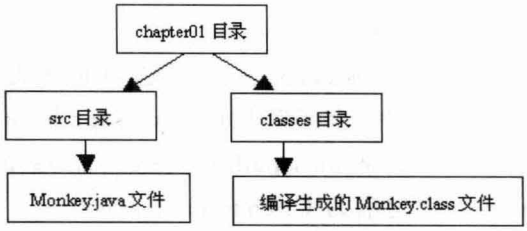

图 1-10　本范例的目录结构

1.4.3　编译 Java 源程序

JDK 中的 javac.exe 为 Java 编译器程序，可以在 DOS 控制台中运行该 Java 编译器程序。在 Windows 中选择【开始】→【程序】→【附件】→【命令提示符】，就会打开 DOS 控制台。在 DOS 命令行下，在同一行中输入以下用于运行 javac.exe 程序的命令：

```
javac -sourcepath C:\chapter01\src -d C:\chapter01\classes C:\chapter01\src\Monkey.java
```

以上 javac 命令包含以下内容：

- -sourcepath 选项：用于设定 Java 源文件所在的目录，此处为 C:\chapter01\src 目录。-sourcepath 选项的默认值为 DOS 命令行的当前目录。
- -d 选项：用于指定编译生成的 Java 类文件的存放目录，此处为 C:\chapter01\classes 目录。-d 选项的默认值为 DOS 命令行的当前目录。
- 待编译的 Java 源文件：此处为 C:\chapter01\src\Monkey.java 文件。

以上 javac 命令将对 C:\chapter01\src\ Monkey.java 源文件进行编译。先对该文件进行 Java 语法检查，如果发现错误，就停止编译，并返回错误信息。如果 Monkey.java 源文件中无语法错误，就会生成 Monkey.class 文件，并把它存放在 C:\chapter01\classes 目录下。

如果操作系统无法识别 javac 命令，那么说明事先没有把 C:\jdk\bin 目录添加到 PATH 系统环境变量中。此时，必须显式指定 javac 命令所在的目录：

```
C:\jdk\bin\javac -sourcepath C:\chapter01\src
-d C:\chapter01\classes  C:\chapter01\src\Monkey.java
```

1.4.4　运行 Java 程序

本章 1.2 节已经讲过，Java 程序必须借助 Java 虚拟机才能运行，而 JDK 中的 java.exe 程序就是 Java 虚拟机程序。在 DOS 命令行下，输入以下用于运行 java.exe 程序的命令：

```
java -classpath C:\chapter01\classes  Monkey
```

以上 java 命令包含以下内容：

- -classpath 选项：用来设置 classpath，该选项的默认值为当前路径。在运行 Java 程序时，很重要的一个环节是设置 classpath，classpath 代表 Java 类的根路径，Java 虚拟机会从 classpath 中寻找所需 Java 类的.class 文件。在本例中，classpath 选项的值为 C:\chapter01\classes。
- 待运行的 Java 类：此处为 Monkey 类。

以上 java 命令将会启动 Java 虚拟机，Java 虚拟机会从 C:\chapter01\classes 目录下找到 Monkey.class 类文件，然后运行 Monkey 类，执行它的如下 main()方法：

```
public static void main(String[] args) {
  //创建代表智多星的 Monkey 对象
  Monkey m=new Monkey("智多星");
  //智多星说话
  m.speak();
}
```

悟空把智多星叫过来，指着如图 1-11 所示的程序运行结果，对智多星说："你看，电脑里的那个虚拟智多星在学你说话呢。"智多星对悟空佩服得五体投地，赞叹道"孙爷爷，没想到这愣头电脑也能轻而易举地变化出我的化身，这和爷爷您身上的猴毛有异曲同工之妙啊。"

图 1-11　运行 Monkey 类的打印结果

不过，调皮的智多星很快就意识到，电脑里的虚拟智多星看不见摸不着，要是有个模样就更好玩了。在第 21 章的 21.6 节（创建动画），悟空将运用 GUI 编程，创建一个有模有样的虚拟智多星，来满足智多星的心愿。

悟空觉得每次用 java 命令运行 Monkey 类时，都要通过-classpath 选项来设置 classpath，有点麻烦。因此，他运用"set classpath"命令在当前 DOS 控制台先设置了 classpath，接下来再使用 java 命令时，就不用再设置 classpath 了：

```
C:\> set classpath=C:\chapter01\classes
C:\> java Monkey
```

1.4.5　创建用于编译和运行 Java 程序的批处理文件

每次编译或运行 Java 程序时，都要在 DOS 命令行中输入很长的 javac 或 java 命令，悟空觉得太麻烦，就编写了一个适用于 Windows 操作系统的批处理文件

build.bat，它的内容如下：

```
set currpath=.\
if "%OS%" == "Windows_NT" set currpath=%~dp0%

set src=%currpath%src
set dest=%currpath%classes
set classpath=%dest%

javac  -sourcepath %src%  -d %dest% %src%\Monkey.java
java -classpath %classpath% Monkey
```

以上 build.bat 批处理文件包含了 javac 命令和 java 命令。build.bat 批处理文件位于 chapter01 根目录下。只要运行这个批处理文件，就会编译并运行 Monkey 类。本书后面章节的范例中都提供了 build.bat 批处理文件。

1.5　小结

本章涉及的 Java 知识点总结如下：

- 程序的运行过程

编程人员用高级编程语言编写出源程序，把它编译为可执行程序，然后在操作系统中运行可执行程序。

- Java 语言的跨平台特性

同一个 Java 程序可以在多个操作系统中运行。例如，Monkey.class 类既可以在 Windows 中运行，也可以在 Linux 中运行。Java 语言的跨平台特性主要归功于 Java 虚拟机。

- Java 虚拟机的主要功能

Java 虚拟机的主要功能是运行 Java 程序。Java 虚拟机本身也是可执行程序，在不同的操作系统中，Java 虚拟机本身的实现方式是不一样的。不过，不管是在哪个操作系统中，Java 虚拟机都会按照同样的方式来解析和运行 Java 程序。

- JDK 的组成

JDK 主要由 Java 虚拟机程序（对应 java.exe）、Java 编译器程序（对应 javac.exe）和 JDK 类库组成。JDK 类库提供了最基础的 Java 类及各种实用类。

- Java 语言的面向对象的基本思想

程序是对现实世界的模拟，进行 Java 编程时，按照如下思维过程来模拟现实世界：

（1）首先识别出待模拟的实体，比如猴子智多星就是待模拟的实体。

（2）分析待模拟的实体所属的类，比如智多星属于猴类（Monkey）。

（3）根据应用需求，分析所属的类具有的共同属性和行为，比如猴类有名字属性和说话行为。

（4）创建一个 Monkey 类，它的 name 属性表示猴类的名字属性，它的 speak()方法代表猴类的说话行为。

（5）智多星属于 Monkey 类的一个实例，即一个 Monkey 对象。Java 语言用 new 语句来创建具体的 Monkey 对象。每一个 Monkey 对象都会拥有自己的 name 属性和 speak()方法。

- Java 程序的入口 main()方法

所有的 Java 程序都是从 main()方法开始运行的。作为程序入口的 main()方法采用 public 和 static 修饰符，必须有一个 String 数组类型的参数，返回类型为 void。

- Java 程序的创建、编译和运行过程

编程人员先用 Java 语言编写源程序，源程序是以“.java”为扩展名的 Java 源文件。接下来用 Java 编译器（对应 JDK 中的 javac.exe 程序）来编译 Java 源程序，编译生成以“.class”为扩展名的 Java 类文件。最后用 Java 虚拟机（对应 JDK 中的 java.exe 程序）来运行 Java 类，执行该 Java 类中作为程序入口的 main()方法。

第 2 章　Java 语言的基本语法

在第 1 章，悟空用 Java 语言编写了一个 Monkey 类，并且创建了一个代表智多星的 Monkey 对象。别看悟空编写 Monkey 类轻轻松松，可 Java 初学者如果想依葫芦画瓢地编写个阿猫阿狗类，到目前为止，肯定还无从下手。

Monkey 类在 Monkey.java 源文件中定义，本章将逐个字眼地把 Monkey.java 源文件从头到脚细凑一遍，看看它到底是如何编写出来的。本章会介绍 Monkey.java 源文件中涉及的种种 Java 语法，解释标识符和关键字的概念，介绍 Java 包、注释和 JavaDoc 文档的用法。学习了本章内容，读者就可以亲自动手，创建简单的 Java 程序了。

为了便于演示 Java 语言的一些特性，本章把 Monkey 类拆分为 Monkey 类（参见 2.6 节的例程 2-2）和 AppMain（参见 2.6 节的例程 2-3）类，它们分别位于 Monkey.java 和 AppMain.java 源文件中。原先 Monkey 类中的程序入口 main()方法被挪到了 AppMain 类中。

2.1　Java 源文件结构

一个 Java 程序可以包含一个或多个 Java 源文件，Java 源文件以“.java”作为扩展名。每个 Java 源文件只能包含下列内容（空格和注释忽略不计）：

- 零个或一个包声明语句（Package Statement）。
- 零个或多个包引入语句（Import Statement）。
- 零个或多个类的声明（Class Declaration）。

每个 Java 源文件可包含多个类的定义，但至多只有一个类是 public 的，而且 Java 源文件必须以其中 public 类型的类的名字命名。

例如，在 Monkey.java 文件中定义了 public 类型的 Monkey 类，因此，该文件以 Monkey 类的名字命名。同理，在 AppMain.java 文件中定义了 public 类型的 AppMain 类，因此，该文件以 AppMain 类的名字命名。

以下例程 2-1 的 Tester.java 文件中同时声明了 Tester 类、Sample1 类和 Sample2 类，只有 Tester 类为 public 类型，那么该文件应该以 Tester 类的名字命名。如果把 Tester.java 文件改名为 Sample1.java，那么会导致编译错误。

例程 2-1　Tester.java（源文件以 public 类型的 Tester 类的名字命名）

```
public class Tester{……}
```

```
class Sample1{……}
class Sample2{……}
```

2.2 关键字

Java 语言的关键字是程序代码中的特殊字符，它们有着特定的含义。例如在 Monkey.java 文件中，public、class、this 和 void 等都是关键字：

```
/** 以下粗体字部分都是关键字 */
public class Monkey{
  String name;
  public Monkey(){ }

  public Monkey(String name){
    this.name=name;
  }

  public void speak(){
    System.out.println("大家好，我是"+name);
  }
}
```

Java 语言的关键字包括：

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*

以上每个关键字都有特殊的作用，例如 package 关键字用于包的声明，import 关键字用于引入包，class 关键字用于类的声明，void 关键字表示方法没有返回值。现在，悟空对许多关键字都很陌生，等到完全掌握了 Java 语言，就会对这些关键字的用法如数家珍了。本书的后面章节会陆续介绍各个关键字的作用。

使用 Java 语言的关键字时，有以下值得注意的地方：

- 所有的关键字都是小写。
- 程序中的标识符不能以关键字命名。关于标识符的概念参见本章 2.3 节（标识符）。

例如，悟空有一次编写了一个名为“super”的类，结果编译出错，错误原因是不能把关键字“super”作为类的名字：

```
public class super{……}// 编译出错，super 是关键字，不能作为类的名字
```

2.3　标识符

标识符是指程序中包、类、变量或方法的名字。例如在 Monkey.java 文件中，Monkey、name 和 speak 等都是标识符：

```
/**  以下粗体字部分都是标识符 */
public class Monkey{
  String name;

  public Monkey(){}

  public Monkey(String name){
    this.name=name;
  }

  public void speak(){
    System.out.println("大家好，我是"+name);
  }
}
```

Java 关键字是 Java 语言固有的特殊字符，而标识符则是由编程人员自己随意命名的。例如，悟空可以把用于模拟猴子说话行为的方法命名为 speak()，也可以命名为 say()。不过，编程人员给标识符命名时，也不能太随心所欲，必须符合以下命名规则，否则就会导致编译错误：

- 标识符的首字符必须是字母、下画线_、符号$或者符号¥。
- 标识符由数字（0~9）、从 A~Z 的大写字母、a~z 的小写字母、下画线_，以及美元符$等组成。
- 不能把关键字做为标识符。
- 标识符是大小写敏感的，这意味着，hello、Hello 和 HELLO 是 3 个不同的标识符。

表 2-1 是一个正误对照表，列举了一些合法标识符和非法标识符。如果程序代码中包含非法标识符，会导致编译错误。

表 2-1　标识符正误对照表

合法标识符	非法标识符	说　明
monkey	monkey#	标识符中不能包含“#”
GROUP_7	7GROUP	标识符不能以数字符号开头
openDoor	open-door	标识符中不能包含“-”
boolean1	boolean	boolean 是关键字，不能用关键字做标识符