第1章　Java概述

所支撑的职业技能

本章以项目为导向，介绍了编写及运行Java程序的基本流程；通过本章的学习，读者能够了解Java语言的特点和工作原理，掌握JDK的安装和使用方法，掌握编写简单的Java应用程序的方法。

教学重点及难点

教学重点：Java语言的工作原理、JDK的用法和应用程序的开发过程。

教学难点：JDK的用法。

重难点学习建议

读者需了解Java的工作原理，搭建好Java开发所需环境即JDK和Eclipse，能独立写出简单的Java应用程序。

1.1　项目任务

(1)使用记事本编写第一个Java应用程序——“Hello World!”；

(2)使用集成开发环境(Eclipse)编写Java应用程序——“Hello World!”。

1.2　项目分析

1.项目完成思路

根据项目任务描述的功能需求，本项目需要先搭建Java运行的环境，在记事本中编写Java应用程序，然后在Eclipse集成开发环境中再次编写Java应用程序，具体可以按照如下过程实现：

(1)先安装JDK，配置环境变量；

(2)在记事本中编写Java应用程序的源程序；

(3)使用JDK编译和运行这个程序，理解Java运行的机制；

(4)Eclipse 中编写应用程序,控制台输出“Hello World!”,熟悉集成开发环境下开发 Java 程序的步骤及方法。

2. 需解决的问题

(1)如何搭建 Java 运行的环境?

具体须解决的问题包括:Java 程序要运行都需要什么环境? 什么是 JDK? 如何下载并安装 JDK? 如何配置环境变量?

(2)如何书写 Java 应用程序?

具体须解决的问题包括:Java 应用程序的结构是什么? 两种程序如何运行?

(3)如何在集成开发环境 Eclipse 中编写并运行 Java 程序?

具体须解决的问题包括:Eclipse 如何使用? Eclipse 中如何书写 Java 程序?

1.3 技术准备

要实现本章的项目任务,首先要了解 Java 运行的基本流程和 Java 运行的原理,掌握 JDK 的安装和使用方法,掌握编写 Java 应用程序的方法。以下逐一介绍实现本章项目所必备的相关技术。

1.3.1 Java 运行原理

Java 程序不必重新编译就能在各种平台上运行,具有很强的可移植性。网络上遍布着各种不同类型的主机和操作系统,为使 Java 程序能在网络的任何地方运行,Java 源程序被编译成一种在高层上与机器无关的 byte-code(字节码)。这种字节码被设计在虚拟机上运行,由机器相关的解释程序执行。只要在处理器和操作系统安装 Java 运行环境,字节码文件就可以在该计算机上运行。

运行 Java 字节码文件需要解释程序将字节码文件翻译成目标机上的机器语言,Java 字节码的执行原理如图 1-1 所示。

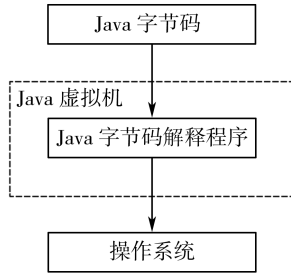

图 1-1 Java 字节码的执行原理

通过图 1-1 可以看出,字节码文件是在 Java 虚拟机上运行的,所以它的运行速度总是比 C 和 C++这类的编译语言要稍慢,但 Java 的速度已经能够满足大多数应用程序的要求,尤其是随着 CPU 速度的不断提高,这种运行速度上的差异变得不再重要,用户更看重 Java 语言具有的其他良好特性,而且 Java 的 JIT(Just In Time,即时编译技术),能在一定程度上加

速 Java 程序的执行速度。

1.3.2　JDK 简介

1. 什么是 JDK

JDK 是利用 Java 技术进行软件开发的基础，包括 Java 运行环境（Java Runtime Environment），是一组建立、测试 Java 程序的实用程序及 Java 基础类库。Java 运行环境是可以运行、测试 Java 程序的平台，它包括 Java 虚拟机、Java 平台核心类和支持文件。Java 类库包括语言结构类、基本图形类、网络类和文件 I/O 类。掌握 JDK 是学好 Java 的第一步。Oracle 官网可以下载 JDK 的各种版本，本教材所用到 JDK 的版本是 JDK 11，可以到官网去下载，然后安装到系统中，安装方法及配置见附录 A。

JDK 中最常用的工具如下：

javac：Java 语言编译器，能将 Java 源程序编译成 Java 字节码。

java：Java 字节码解释器，可以用来运行 Java 程序。

jar：可将多个文件合并为单个 jar 归档文件。将 applet 或应用程序的组件（.class 文件、图像和声音）使用 jar 合并成单个归档文件时，可以用浏览器在一次 HTTP 传输过程中对它们进行下载，而无需对每个组件都要求一个新连接。

javadoc：javadoc 是 Java API 文档生成器，可以从 Java 源文件生成帮助文档。javadoc 解析 Java 源文件中的声明和文档注释，并产生相应的 HTML 帮助页。

javah：javah 从 Java 类生成 C 语言头文件和 C 语言源文件，使 Java 和 C 代码可以进行交互。

javap：将字节码分解还原成源文件，显示类文件中的可访问功能和数据。

jdb：Java 调试器，可以逐行执行 Java 程序，设置断点和检查变量，是查找程序错误的有效工具。

JDK 一般有三种版本：

JavaSE（Java Platform Standard Edition，Java 平台标准版），是三个平台中最核心的部分，包括 Java 最核心的类库，主要是为开发普通桌面和商务应用程序提供解决方案。

JavaEE（Java Platform Enterprise Edition，Java 平台企业版），该平台用于开发、装配和部署企业级应用程序，主要包括 Servlet、JSP、JavaBean、EJB 等，主要为开发企业级应用程序提供解决方案。

JavaME（Java Platform Micro Edition，Java 平台微型版），该平台主要用于微型电子设备上软件程序的开发，比如家用电器智能化控制、开发手机游戏等，主要为开发电子消费类产品和嵌入式设备提供解决方案。

2. Java 程序的编辑、编译和执行

(1)编写代码

Java 源程序就是文本文件,所以可以用任何文本编辑工具进行编辑。最简单的是 Windows 操作系统自带的记事本,当然也可以使用像 UltraEdit 这样的编辑工具。

(2)Java 程序的编译

Java 程序的编译是由编译器 javac.exe 完成的。javac 命令将 Java 源程序编译成字节码,然后可以用 Java 命令来解释执行这些 Java 字节码。Java 源程序必须存放在扩展名为.java的文件中。对 Java 程序中的每一个类,javac 都将生成一个文件名与类名相同,扩展名为.class 的文件。默认编译器会把.class 文件放在 Java 文件的同一个目录下。

javac 命令的用法:javac [选项] Java 源文件名

例如,源文件为 Welcome.java,我们需要在命令提示符下输入如下命令:

```
javac Welcome.java
```

(3)Java 程序的执行

Java 程序的执行是由解释器 java.exe 完成的。

java 命令的用法:java [选项] classname [参数列表]

java 命令执行的是由 javac 命令编译生成的 Java 字节码文件,classname 是要执行的类名。在类名称后的参数都将传递给要执行类的 main 方法。

例如,我们要执行 Welcome.class 字节码文件,需要在命令提示符下输入如下命令:

```
java Welcome
```

Java 应用程序的编辑、编译和执行过程,如图 1-2 所示。

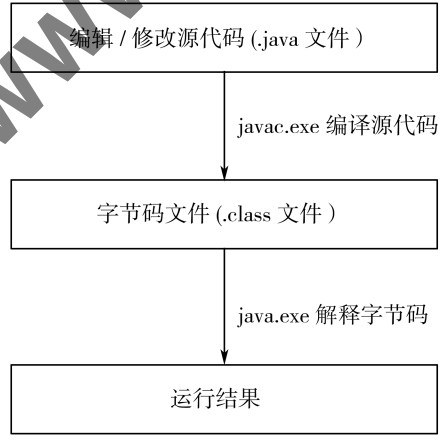

图 1-2 Java 应用程序的编辑、编译和执行过程

【注意】

classname 不包括扩展名,例如,要使用 Java 运行 Welcome.class,则在命令行输入 java Welcome 就可以了,如果输入 java Welcome.class 则会发生错误。

如果源程序有语法错误,在编译时会提示语法错误,必须修改程序,并重新编译,才能生成.class 文件。

Java 语言是区分大小写的，例如，关键字 public，如果写成 Public，在编译时就会提示语法错误，Java 的文件名也是区分大小写的。

3. Java 的集成开发环境——Eclipse

Eclipse 是一个基于 Java 的、开放源码的、可扩展的应用开发平台，它为编程人员提供集成开发环境(Integrated Development Environment，IDE)。Eclipse 具有强大的代码编排功能，可以帮助程序开发人员完成语法修正、代码补全、信息提示等工作，极大提高了程序开发的效率。Eclipse 的设计思想是“一切皆插件”。Eclipse 本身只是一个框架平台，通过安装不同的插件，Eclipse 可以支持不同的计算机语言，比如 C＋＋和 Python 等开发工具。许多软件开发商以 Eclipse 为框架开发自己的 IDE。本教材后续章节的 Java 代码编写及运行都将采用 Eclipse 作为开发工具，具体安装和使用请参考附录 A。

1.3.3　Java 程序的分类

1. 应用程序(Java Application)

Application 是独立程序，与其他高级语言编写的程序相同，可以独立运行。应用程序能够在任何具有 Java 解释器的计算机上运行。

2. 小程序(Java Applet)

Applet 是一种特殊的 Java 程序，它需要在兼容 Java 的 Web 浏览器中运行。Java Applet 嵌入 HTML 页面中，以网页形式发布到 Internet。

1.4　项目学做

1. 第一个 Java Application：向世界问好

(1)使用最简单的记事本编写程序，打开记事本，输入如下代码：

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

(2)书写完成后，保存文件，文件命名为 Welcome.java，保存类型为所有文件。

(3)编译源文件。使用 javac 命令对源文件进行编译。在控制台上输入 javac Welcome.java，如图 1-3 所示，编译后注意观察在源文件的同级目录下将生成一个 Welcome.class 的字节码文件。如果编译有错误，修改源文件后重新进行编译。

(4)运行程序。使用 java 命令执行 java 程序。在控制台输入 java Welcome，控制台上将显示“Hello World!”，效果如图 1-4 所示。

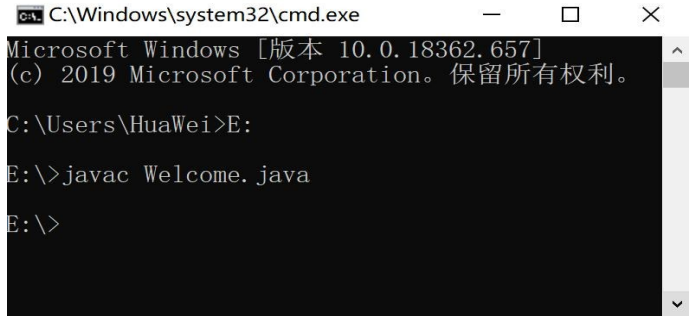

图 1-3 编译 Welcome.java

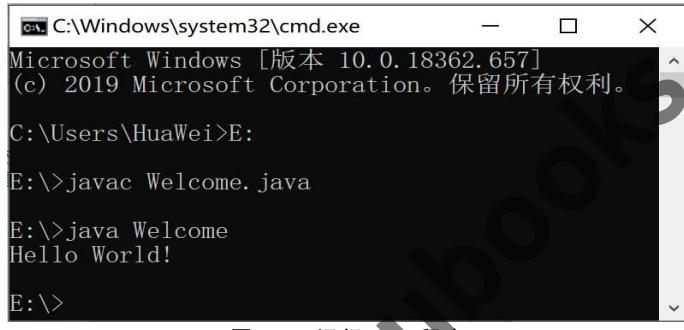

图 1-4 运行 Java 程序

【代码分析】

作为面向对象的语言，Java 要求所有的变量和方法都必须封装在类(class)或接口(interface)中。类是 Java 程序的基础，所有的 Java 程序都是由类组成的，Java 程序至少包含一个类，每个类从声明开始，定义自己的数据和方法。

整个类定义由大括号括起来。在该类中定义了一个 public static void main(String[] args)方法，其中 public 表示访问权限，表示所有的类都可以使用这一方法；static 指明该方法是一个类方法，它可以通过类名直接调用；void 则指明 main()方法不返回任何值。main 方法是一个特殊的方法，它是每一个应用程序所必需的，是应用程序解释执行的入口。Java 程序中可以定义多个类，每个类中可以定义多个方法，但是最多只能有一个被 public 修饰的类。main 方法的方法头格式是确定不变的，必须带有字符串数组类型的参数，但参数名可以任意。

main 方法的参数是一个字符串数组 args，虽然在本程序中没有用到，但是必须列出来。

main 方法中只有一行语句：System.out.println("Hello World!")，它的作用是向控制台输出字符串"Hello World!"。

Java 源程序文件都保存在以 java 为扩展名的文件当中。如果类被 public 修饰，这个源程序文件的名字必须和该类的类名一致(包括大小写在内)，如果类没有被 public 修饰，则无此限制。程序中公共类的名字是 Welcome，所以源程序文件的名字必须是 Welcome.java，否则，在使用 javac 进行编译时，就会发生错误。

2. 在 Eclipse 下用 Java 向世界问好

(1)在 Eclipse 中新建工程 Java01，新建类 Welcome.java，在编辑器区域编写代码，如图 1-5 所示。

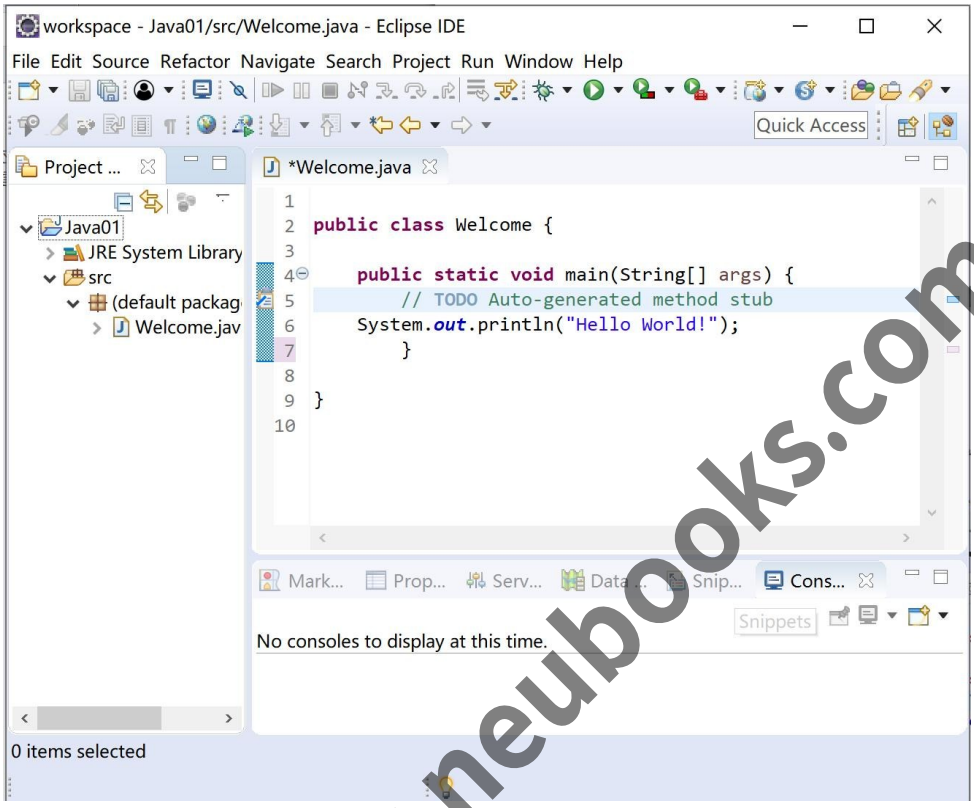

图 1-5　在 Eclipse 中编写代码

(2)点击“保存”,保存没有语法错误,Eclipse 会直接将源代码进行编译,运行时需要点击 Welcome.java 文件的右键,选择【Run As】下面的【Java Application】,如图 1-6 所示。

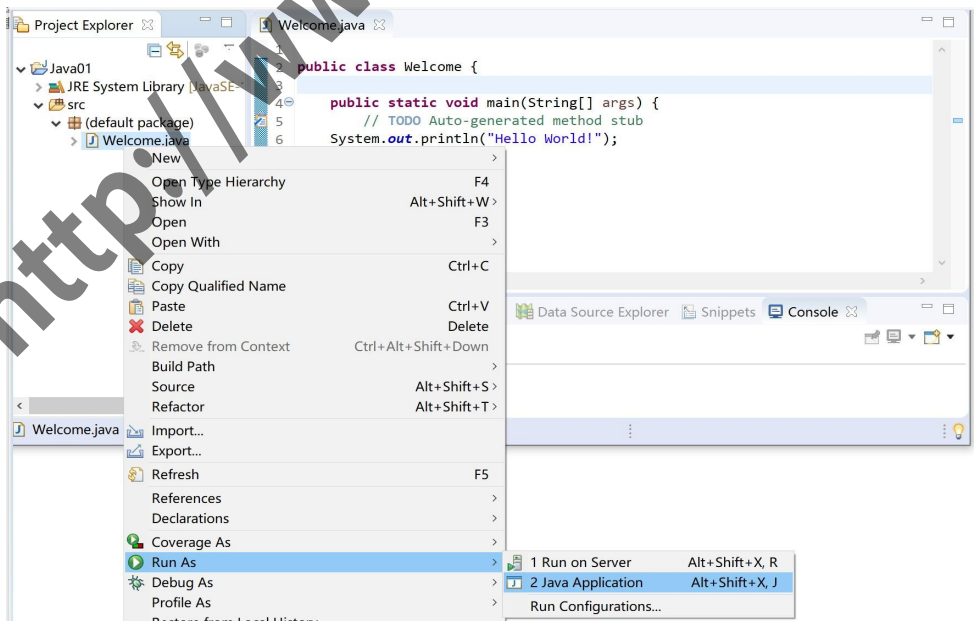

图 1-6　在 Eclipse 中运行 Java 程序

(3)在控制台中显示该程序的运行结果“Hello World!”,如图 1-7 所示。

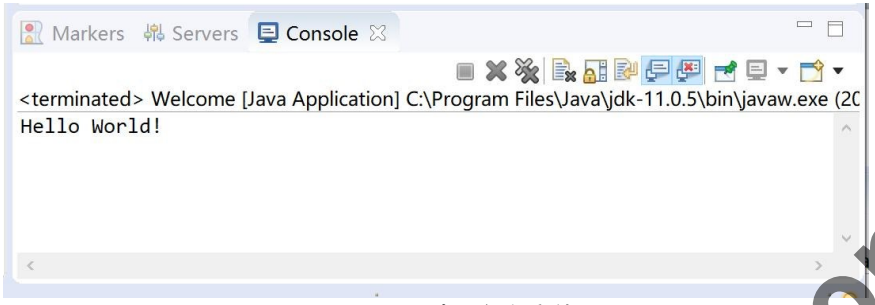

图 1-7 Eclipse 中运行程序结果

1.5 知识拓展

其他常用的 Java 集成开发环境:

1. IntelliJ IDEA

IntelliJ IDEA 是一款综合的 Java 编程环境,被许多开发人员和行业专家誉为市场上最好用的 IDE 之一,与 Eclipse 齐名。它提供了一系列最实用的工具组合、智能编码辅助和自动控制,支持 JavaEE、Ant、JUnit 和 CVS 集成,非平行的编码检查和创新的 GUI 设计器。IDEA 把 Java 开发人员从一些耗时的常规工作中解放出来,显著地提高了开发效率,具有运行更快速,生成更好的代码;持续的重新设计和日常编码变得更加简易,与其他工具的完美集成及高性价比等特点。

2. NetBeans

NetBeans IDE 是一个屡获殊荣的集成开发环境,可以方便地在 Windows、Mac、Linux 和 Solaris 中运行。NetBeans 包括开源的开发环境和应用平台,NetBeans IDE 可以使开发人员利用 Java 平台能够快速创建 Web、企业、桌面以及移动的应用程序,NetBeans IDE 目前支持 PHP、Ruby、JavaScript、Ajax、Groovy、Grails 和 C/C++等开发语言。NetBeans 项目由一个活跃的开发社区提供支持,NetBeans 开发环境提供了丰富的产品文档和培训资源以及大量的第三方插件。

3. JCreator

JCreator 是由 Xinox Software 公司开发的 Java 集成开发环境(IDE)。它的设计接近 Windows 界面风格。JCreator 为用户提供了相当强大的功能,例如个性化设置、语法高亮、行数、多功能编译器、向导功能以及完全可自定义的用户界面。JCreator 最大特点是与 JDK 的完美结合,JCreator 相对来说比较简单,对计算机的系统配置要求不高。

1.6 强化训练

编写程序输出以下信息:

```
* * * * * * * * * * * * * * * * * * *
*          Welcome To Java!           *
* * * * * * * * * * * * * * * * * * *
```

1.7　本章小结

本章利用记事本和 Eclipse 编写了简单的 Java 应用程序。通过实现本章的项目，应理解和掌握以下内容：

1. JDK 是利用 Java 技术进行软件开发的基础，包括 Java 运行环境以及建立、测试 Java 程序的实用程序和 Java 基础类库。

2. 编写好 Java 源文件后，编译器先将源文件编译成字节码文件，然后虚拟机解释并执行字节码文件。

3. 编写 Java 程序时注意如果类被 public 修饰，文件名要和 public 修饰的类名一致。

4. Java 语言是区分大小写的。

1.8　课后习题

一、课后自测

1. Java 区分大小写吗？

2. Java 源程序文件的扩展名是什么？字节码文件的扩展名呢？

3. 编译 Java 程序的命令是什么？

4. 运行 Java 程序的命令是什么？

5. 在控制台（显示器）显示字符串的语句是什么？

二、基础编程

1. 在控制台打印输出自己的学号、姓名、住所和电话。

2. 编写一个程序，在同一行上显示数字 1 到 4，相邻的数字用一个空格分开。按照如下要求编写该程序：

（1）使用一个 System.out 语句。

（2）使用四个 System.out 语句。

三、编程挑战

编写程序，输出如下图形。

```
        *
    *       *
*               *
    *       *
        *
```

第2章　基本符号

所支撑的职业技能

本章介绍Java语言的基本符号、基本数据类型、数据类型转换和运算符。通过本章的学习，读者能够掌握如何定义Java语言中的变量，掌握Java中的基本数据类型、数据类型转换及运算符的使用。

教学重点及难点

教学重点：标识符、Java基本数据类型。

教学难点：数据类型转换。

重难点学习建议

学习任何一门编程语言，都是从基础学起。本章作为Java语言的基础知识，需要掌握标识符应用的场合及命名规则，保存数据时能够选取合适的数据类型，把一种数据类型的值赋给另一种数据类型变量时，知道如何进行数据类型转换。

2.1　项目任务

编写程序来实现贷款计算器。要求从键盘输入年利率、贷款期限、贷款金额，计算出每月支付额和到期支付总额，并将结果在控制台打印出来。计算每月偿还额的公式如下：

$$\frac{\text{贷款金额} \times \text{月利率}}{1-\frac{1}{(1+\text{月利率})^{\text{贷款期限}\times 12}}}$$

2.2　项目分析

1. 项目完成思路

(1)从键盘输入年利率、贷款期限、贷款金额保存到变量中；

(2)根据年利率计算月利率；

(3)借助Math类中的pow方法实现数值的幂运算。

2. 需解决的问题

(1)项目中所需数据需要保存到变量中,变量该如何定义并使用?

(2)不同的数据需要保存到不同类型的变量中,Java 中都有哪些基本数据类型?

(3)数据的处理,计算过程中需要用到数学运算,Java 有哪些运算符?

2.3　技术准备

要实现本章的贷款计算器项目,需要掌握变量的使用,掌握 Java 语言中标识符的规定,掌握基本数据类型、数据类型转换和运算符,了解 Java 语言的基本符号。以下逐一介绍实现本章项目所必备的相关技术。

2.3.1　变量

变量主要用于保存输入、输出和程序运行过程中的中间数据。在 Java 中,每一个变量都属于某种类型。使用变量之前,先要对变量进行声明。

1. 声明变量

在声明变量时,变量所属的类型位于前面,随后是变量名,格式如下:

```
变量类型　变量名
```

下面是声明变量的一些例子:

```
int age;
double salary;
String name;
```

在以上例子中声明了 3 个变量,变量类型分别是 int、double 和 String,变量名分别为 age、salary 和 name。注意,在 Java 中,声明变量是一条完整的语句,每一个声明语句后面都要有分号。也可以在一行语句中同时声明多个变量,中间用逗号隔开:

```
int x,y;
```

为了提高程序的可读性,尽量使用逐行声明变量,这样,程序结构较清晰。

2. 初始化变量

声明一个变量后,要想在程序中使用该变量,必须经过赋值对其进行初始化。变量的数值必须与声明变量的数据类型相匹配,例如:

```
int age;
age=20;
System.out.println(age);
```

这样,在屏幕上将打印变量 age 的值 20。也可以将变量的声明和赋值写在一条语句中,如下:

```
int age=20;
System.out.println(age);
```

2.3.2　标识符

标识符是用户定义的,是用于表示变量名、类名、接口名、方法名、方法参数名等的符号。

标识符的命名应该符合一定的规则，标识符命名规则如下：

(1)由字母、数字、下划线或 $ 符号组成，对标识符的长度没有特别限制；

(2)必须以字母、下划线或 $ 符号开头；

(3)标识符区分大小写；

(4)标识符不能使用系统的保留字。

下面的标识符是合法的标识符：

test　字母组成。

test2　字母和数字组成。

test_3　字母、数字、下划线组成。

$ test　字母和美元符号组成。

下面的标识符是不合法的标识符：

2a　不能以数字开头。

test-3　“-”不是指定的字符。

class　是系统的保留字。

2.3.3　Java 的基本符号

和其他语言相同，Java 程序也是由多个文件组成的，每个文件又是由很多的代码行组成，每个代码行是由一些基本符号组成。本小节讨论组成 Java 程序的相关符号。

1. 数字

数字是由 0 到 9 这 10 个符号组成的数字序列，用于表示数字，可以使用负号“－”和数字一起表示负数，例如 123、35、－222 等。

在 Java 中不但可以用十进制来表示整数，还可以用八进制和十六进制来表示整数。如果以 0 为前缀，则表示八进制的整数，例如 015 表示 13，而不是 15。如果整数以 0X 或 0x 为前缀，则表示十六进制的整数，例如 0x15 表示 21，而不是 15。

如果表示小数可以使用“.”分隔符，例如 9.3、10.2 等。如果整数部分是 0，可以省略，例如“0.5”也可以写成“.5”。

2. 字符

Java 中的字符采用 Unicode 编码方式，Unicode 编码字符是用 16 位无符号整数表示的，可以表示目前世界上的大部分文字语言中的字符。

Java 中的字符是使用单引号括起来的单个字符，例如'a'。字符可以是数字，例如'0'，它不表示数字 0，而表示字符“0”。在 Java 中使用 Unicode 编码，所以字符可以用于表示一个汉字，例如'中'。

3. 字符串

字符串是使用双引号括起来的字符序列，例如"Hello，字符串"，即使双引号中只有一个字符也是字符串，例如"中"。

4. 布尔值

布尔类型的数值有两个：true 和 false。true 表示“真”，false 表示“假”。

2.3.4　数据类型

Java 有 8 种基本数据类型，各种基本数据类型在内存中占用的位数和表示的范围如表 2-1 所示。

表 2-1　　Java 基本数据类型

基本数据类型	位数	表示范围
byte	8 位	−128～127
short	16 位	$-2^{15}\sim 2^{15}-1$
int	32 位	$-2^{31}\sim 2^{31}-1$
long	64 位	$-2^{63}\sim 2^{63}-1$
float	32 位	$-3.4028235\times 10^{38}\sim 3.4028235\times 10^{38}$
double	64 位	正负 $1.79769313486231570\times 10^{308}$ 之间
char	16 位	采用 Unicode 编码，可以表示中文
boolean		值只能为 true 或者 false

1. byte 类型

byte 类型变量的定义和赋值：

```
byte b = 1;
```

2. short 类型

short 类型变量的定义和赋值：

```
short s = 2;
```

3. int 类型

int 类型变量的定义和赋值：

```
int i = 3;
```

通常情况下，int 型是最常用的。

4. long 类型

long 类型变量的定义和赋值：

```
long n = 10L;
```

程序中如果需要使用 long 型，则需要在数值中添加后缀“L”或“l”。

5. float 类型

float 类型变量的定义和赋值：

```
float f1 = 11.5f;
```

【注意】 float 类型的常量必须在数字后面使用“f”或“F”标识，如果浮点数不加后缀，它的默认类型为 double 类型；如果写成 float f1 = 11.5，编译的时候将会出错。

6. double 类型

double 类型变量的定义和赋值：

```
double dd = 28.5;
```

通常情况下，如果定义浮点数据类型变量，double 型是最常用的。

7. char 类型

字符类型变量的定义和赋值：

```
char c ='a';
```

字符常量使用单引号扩起来，另外下面的写法也是正确的：

```
char c=98;
```

实际上 98 是字符 b 的编码，和 c='b'的效果是完全相同的。在为字符类型变量赋值的时候，可以使用转义字符。

```
char c ='\\';
```

8. boolean 类型

布尔类型的常量只有 true 和 false，所以对布尔类型的变量的赋值只能是 true 或者 false。布尔类型的变量的定义和赋值如下：

```
boolean bb = true ;
boolean success = false;
```

变量的定义和赋值可以分开进行，例如：

```
int n1;
n1 = 33;
```

【注意】 变量的作用范围，可以把小作用范围值赋值给大作用范围的变量，但是不能把大作用范围值赋给小作用范围的变量，例如：

```
byte b2 = 43444;
```

这个赋值就会产生错误，因为 byte 表示的范围是－128 到 127，43444 超出了 byte 类型能够表示的范围。

2.3.5 数据类型转换

1. 自动转换

在 Java 中整型、浮点型、字符型被视为简单数据类型，这些类型由低级到高级为 byte—(short，char)—int—long—float—double。

自动转换是不用任何特殊说明的，系统会自动将其值转换为对应的类型。Java 中，低级的数值可以自动转换为高级的类型，转换实例如下：

```
byte b=27;
char c='a';
int i=b; //将 byte 转换为 int
short s=b; //将 byte 转换为 short
long m=c; //将 char 转换为 long
float f=50; //将 int 转换为 float
double d1=m; //将 long 转换为 double
double d2=f; //将 float 转换为 double
```

【注意】 如果低级类型为 char 型，向高级类型(整型)转换时，会转换为对应 Unicode 码值，例如：

```
char c='a';
int i=c;
System.out.println("output:"+i);
```

输出:output:97;

2. 强制类型转换

在 Java 中,有时需要将高级数据转换成低级的类型,这种转换可用强制类型转换完成。强制类型转换的语法格式:

目标变量=(转换的目标类型)待转换的变量或数值;

例如:

```
float f = (float)10.1;
int i = (int)f;
```

【注意】 boolean 类型不能和任何数据类型进行类型转换。

3. 运算过程中的类型转换

不同类型的数字进行运算的时候,系统会强制改变数据类型,例如下面的代码:

【例 2-1】 因运算过程中类型转换导致出错的程序。

```
// TypeConvert.java
public class TypeConvert {
    public static void main(String[] args) {
        byte b1 = 3;
        byte b2 = 4;
        byte b3=b1+b2;
    }
}
```

在编译的时候会报如下的错误:

```
Exception in thread "main" java.lang.Error: Unresolved compilation problem:
    Type mismatch: cannot convert from int to byte
    at TypeConvert.main(TypeConvert.java:6)
```

原因在于执行 b1+b2 的时候,系统会把 b1 和 b2 的类型都转换成 int 类型然后计算,计算的结果也是 int 类型,所以把 int 类型赋值给 byte 类型,这时候将产生错误。

类型转换的基本规则如下:

(1)操作数中如果有 double 类型,则都会转换成 double 类型。

(2)如果有 float 类型,则会转换成 float 类型。

(3)如果有 long 类型,则会转换成 long 类型。

(4)其他的都会转换成 int 类型。

如何解决上面的错误呢? 可以参考下面的代码:

【例 2-2】 例 2-1 经修改,可以正常编译和运行的程序。

```
// TypeConvert2.java
public class TypeConvert2 {
    public static void main(String[] args) {
        byte b1 = 3;
        byte b2 = 4;
        // 对计算结果进行强制转换
        byte b3 = (byte)(b1+b2);
    }
}
```

2.3.6 运算符

1. 算术运算符

标准的算术运算符有：+、-、*、/和%，分别表示加、减、乘、除和求余。另外“+”和“-”也可以作为单目运算符，表示“正”和“负”。

【例 2-3】 下面的程序演示了这几个运算符的用法。

```
// MathmeticsOperationTest.java
public class MathmeticsOperatorTest {
    public static void main(String[] args) {
        // 定义整型变量 a,b,分别赋值 20 和 7
        int a=20;
        int b=7;
        // 进行加、减、乘、除和求余运算
        int sum = a+b;
        int sub = a-b;
        int mul = a*b;
        int div = a/b;
        int res = a%b;
        // 输出运算的结果
        System.out.println("a="+a+"  b="+b);
        System.out.println("a+b="+sum);
        System.out.println("a-b="+sub);
        System.out.println("a*b="+mul);
        System.out.println("a/b="+div);
        System.out.println("a%b="+res);
    }
}
```

运行结果如下：

```
a=20  b=7
a+b=27
a-b=13
a*b=140
a/b=2
a%b=6
```

【注意】 整数运算的结果仍然是整数，例如：12/8 = 1，而不是 1.5。

2. 赋值运算符

(1)基本赋值运算符

“=”是赋值运算符，其实前面的很多例子都用到过。

用法：左边是变量，右边是数值或表达式。

作用：把右边的值或表达式的值赋给左边的变量。

例：

```
int a = 5;  //直接赋值
a = a+3;  //赋表达式
```

(2)复合赋值运算符

赋值运算符与其他运算符结合使用完成赋值的功能。看下面的示例：

【例 2-4】 复合赋值运算符使用的程序。

```
// CompoundOperator.java
public class CompoundOperator {
    public static void main(String[] args) {
        // 使用复合赋值表达式计算 a+3,并把结果赋值给 a
        int a = 3;
        a += 3;
        // 不使用复合赋值表达式计算 b+3,并把结果赋值给 b
        int b = 3;
        b = b+3;
        // 分别输出 a 和 b 的值
        System.out.println("a = "+a);
        System.out.println("b = "+b);
    }
}
```

运行的结果如下：

```
a = 6
b = 6
```

基本格式：a X= b;

“X”表示运算符，可以是各种运算符。

作用：使用左值与右值进行基本的“X”运算，然后把运算的结果赋值给左值，相当于下面的代码：

```
a = aXb;
```

大部分的运算符都可以和赋值运算符结合使用构成复合赋值运算符。

3. 自增、自减运算符

(1)自增运算符

自增运算符的基本功能是把自身增加 1，假设操作数是 x，自增运算符可以有两种格式：

```
x++;
++x;
```

两种格式的效果是一致的，都是把 x 的值增加了 1，相当于：

```
x = x+1;
```

但是，当自增运算符和赋值运算符一起使用的时候，两种格式的效果是不一样的，例如：

```
y = x++;
y = ++x;
```

前者相当于：

```
y = x;
x = x+1;
```

后者相当于：

```
x = x+1;
y = x;
```

【例 2-5】 自增运算符使用的程序。

```
// AddOne.java
public class AddOne {
    public static void main(String[] args) {
        int x1=3;
        int x2=3;
        // ++在后面
        int y1 = x1++;
        // ++在前面
        int y2 = ++x2;
        System.out.println("y1="+y1+" y2="+y2);
    }
}
```

运行结果如下：

```
y1=3 y2=4
```

【注意】 操作数在前面先赋值；操作数在后面后赋值。

(2)自减运算符

自减运算符的用法与自增运算符完全相同，进行自减操作。下面的实例是把例 2-5 中的“++”改为“--”而形成的：

```
public class SubOne {
    public static void main(String[] args) {
        int x1=3;
        int x2=3;
        // --在后面
        int y1 = x1--;
        // --在前面
        int y2 = --x2;
        System.out.println("y1="+y1+" y2="+y2);
    }
}
```

请读者自行分析程序的运行结果。

【注意】 操作数在前面先赋值；操作数在后面后赋值。

4. 比较运算符

比较运算符也称关系运算符，用于对两个值进行比较，其返回值为布尔类型。

比较运算符有：>、>=、<、<=、==、!=，分别表示大于、大于等于、小于、小于等于、等于、不等于。

基本用法：exp1 X exp2

其中，exp1 和 exp2 是两个操作数，可以是表达式，X 表示某种关系运算符，如果 exp1 和 exp2 满足“X”关系，结果为 true，否则结果为 false。例如：5>3，结果为 true，4!=6 结果为 true。

【例 2-6】 比较运算符使用的程序。

```
// CompareOperator.java
public class CompareOperator {
```

```
    public static void main(String[] args) {
        int a = 3;
        int b = 4;
        boolean bigger = a>b;
        boolean less = a<b;
        boolean biggerEqual = a>=b;
        boolean lessEqual = a<=b;
        boolean equal = a==b;
        boolean notEqual = a! =b;
        System.out.println("a="+a+" b="+b);
        System.out.println("a>b:"+bigger);
        System.out.println("a<b:"+less);
        System.out.println("a>=b:"+biggerEqual);
        System.out.println("a<=b:"+lessEqual);
        System.out.println("a==b:"+equal);
        System.out.println("a! =b:"+notEqual);
    }
}
```

运行结果如下：

```
a=3 b=4
a>b:false
a<b:true
a>=b:false
a<=b:true
a==b:false
a! =b:true
```

【注意】

(1)这些符号都是英文的，不能使用中文。

(2)“==”与“=”容易混淆，比较相等不能写成“=”。

5. 逻辑运算符

在 Java 中，逻辑运算符只能对布尔类型数据进行操作，其返回值同样为布尔类型的值。逻辑运算符有：&&、||、!、|、&、^。运算规则如下：

“&&”和“&”是逻辑与，只有当两个操作数都为 true 的时候，结果才为 true。

“||”和“|”是逻辑或，只有当两个操作数都为 false 的时候，结果才为 false。

“!”是逻辑非，如果操作数为 false，结果为 true；如果操作数为 true，结果为 false。

“^”是逻辑异或，如果两个操作数不同，结果为 true；如果两个操作数相同，结果为 false。

【例 2-7】 逻辑运算符使用的程序。

```
//LogicOperator.java
public class LogicOperator {
    public static void main(String[] args) {
        // 定义布尔类型的变量 b1 和 b2，并分别赋值
        boolean b1 = true;
        boolean b2 = false;
        // 进行各种布尔运算，并输出结果
        System.out.println("b1="+b1+" b2="+b2);
```

```
        System.out.println("b1&&b2="+(b1&&b2));
        System.out.println("b1&b2="+(b1&b2));
        System.out.println("b1||b2="+(b1||b2));
        System.out.println("b1|b2="+(b1|b2));
        System.out.println("! b1="+(! b1));
        System.out.println("b1^b2="+(b1^b2));
    }
}
```

运行结果为：

```
b1=true b2=false
b1&&b2=false
b1&b2=false
b1||b2=true
b1|b2=true
! b1=false
b1^b2=true
```

"&&"和"&"从运行结果来看是相同的，但是运行的过程不一样。看下面的例子：

【例 2-8】 逻辑运算符中"&&"和"&"使用的程序。

```
// FastLogicOperator.java
public class FastLogicOperator {
    public static void main(String[] args) {
        int a = 5 ;
        int b = 6;
        int c = 6;
        // 使用 && 进行逻辑运算
        System.out.println((a>b) && (a>(b--)) );
        // 使用 & 进行逻辑运算
        System.out.println((a>c) & (a>(c--)) );
        System.out.println("b="+b);
        System.out.println("c="+c);
    }
}
```

运行结果为：

```
false
false
b=6
c=5
```

从这个结果可以看出，"&&"和"&"的运算结果相同，但是 b 和 c 的值不同。使用"&&"的时候，后面的表达式没有计算，所以 b 的值没有发生变化；使用"&"的时候，后面的表达式进行计算了，所以 c 的值发生了变化。而实际上，进行与运算只要前面的表达式是 false，结果就是 false，所以后面就不用计算了，"&&"运算符正是使用了这个特性。

"||"和"|"的区别也是这样，只不过当||前面为真时，||后不做计算。

【注意】 "&&"和"||"是快速运算符，但是不能保证后面的表达式执行。

6. 位运算符

计算机中数字都是以二进制形式存储的，位运算符用来对二进制数位进行逻辑运算，操作数只能为整型或字符型数据，结果也是整型数。

位运算符有：&、|、～、^，分别表示按位与、按位或、按位非和按位异或。在运算的过程中对两个操作数的每位进行单独的运算，把运算后的每一位重新组合成数字。

例：15&3

15 表示成二进制为：

0000 0000 0000 1111

3 表示成二进制为：

0000 0000 0000 0011

进行按位与，得到：

0000 0000 0000 0011

结果就是 3。

下面的例子包含了 4 个位运算符。

【例 2-9】 位运算符使用的程序。

```
// BitOperator.java
public class BitOperator {
    public static void main(String[] args) {
        int a = 15;
        int b = 3;
        // 按位与，并输出结果
        System.out.println("a&b="+(a&b));
        // 按位或，并输出结果
        System.out.println("a|b="+(a|b));
        // 按位进行异或，并输出结果
        System.out.println("a^b="+(a^b));
        // 按位取反，并输出结果
        System.out.println("! a="+(~a));
    }
}
```

运行结果如下：

```
a&b=3
a|b=15
a^b=12
! a=-16
```

7. 移位运算符

移位运算符同样是对二进制位进行操作。

移位运算符有 3 个：

<<　　左移

>>　　右移

>>>　　无符号右移

(1)左移

基本格式:x<<y

其中 x 是要移位的数,y 是要移动的位数。结果相当于 x 乘以 2 的 y 次方,例如:5<<2 相当于 $5*2^2$,结果为 20。

(2)右移

基本格式:x>>y

其中 x 是要移位的数,y 是要移动的位数。移位之后,如果是正数,高位补 0;如果是负数,高位补 1。结果相当于 x 除以 2 的 y 次方,例如:5>>2 相当于 $5/2^2$,结果为 1。

(3)无符号右移

基本格式:x>>>y

其中 x 是要移位的数,y 是要移动的位数。移位之后,高位补 0。所以,如果是正数,结果与有符号右移的结果相同;如果是负数,移位之后会变成正数,因为高位补 0,高位 0 就是正数。

【例 2-10】 移位运算符使用的程序。

```
int a,b,c;
a = 15;
b = -15;
c = 2;
System.out.println("----------左移运算符----------");
System.out.println("a="+a+" b="+b+" c="+c);
System.out.println("a<<c= "+(a<<c));
System.out.println("b<<c= "+(b<<c));
a = 15;
b = -15;
System.out.println("----------右移运算符----------");
System.out.println("a="+a+" b="+b+" c="+c);
System.out.println("a>>c= "+(a>>c));
System.out.println("b>>c= "+(b>>c));
a = 15;
b = -15;
System.out.println("----------无符号右移运算符----------");
System.out.println("a="+a+" b="+b+" c="+c);
System.out.println("a>>>c= "+(a>>>c));
System.out.println("b>>>c= "+(b>>>c));
```

运行结果为:

```
----------左移运算符----------
a=15 b=-15 c=2
a<<c= 60
b<<c= -60
----------右移运算符----------
a=15 b=-15 c=2
a>>c= 3
b>>c= -4
----------无符号右移运算符----------
a=15 b=-15 c=2
a>>>c= 3
b>>>c= 1073741820
```

【注意】

无符号右移的结果都是正数，不管操作数是正数还是负数。

左移和右移，对于正数和负数移位结果可能是不相同的，但符号不变。示例中 15 右移 2 位得到的结果是 3，－15 右移 2 位得到的结果为－4，这样的结果与正数和负数在计算机中表示的方式相关。

8. 条件运算符

根据不同的逻辑结果，可以得到不同的值。

基本格式：op1? op2:op3;

op1 的结果应该为布尔类型，如果 op1 的值为 true，则表达式最终的结果为 op2；如果 op1 的值为 false，则表达式最后的结果是 op3。

【例 2-11】 下面的代码完成了求 a 和 b 的最大值的功能。

```
// TernaryOperator.java
public class TernaryOperator {
    public static void main(String[] args) {
        int a=10;
        int b=7;
        int c;
        // 如果 a>b,把 a 赋值给 c,如果不是 a>b,把 b 赋值给 c
        c = a>b? a:b;
        System.out.println(a+"和"+b+"的最大值为:"+c);
    }
}
```

运行结果为：

```
10 和 7 的最大值为:10
```

9. 字符串连接运算符

“＋”，用于连接字符串，实际上在前面的例子中已经使用了。

基本格式：op1＋op2

要求 op1 和 op2 中至少要有 1 个是字符串，另外一个可以是前面介绍的 8 种基本数据类型中的一种或是任何类的对象。

【例 2-12】 字符串连接运算符使用的程序。

```
// StringJoin.java
public class StringJoin {
    public static void main(String[] args) {
        byte b = 3;
        short s = 4;
        int i=10;
        long l = 11;
        float f = 3f;
        double d2 = 23.5;
        char c = 's';
        boolean bool = false;
        java.util.Date d = new java.util.Date();
        // 使用字符串与各种类型的数据进行连接
```

```
        System.out.println("byte 类型:"+b);
        System.out.println("short 类型:"+s);
        System.out.println("int 类型:"+i);
        System.out.println("long 类型:"+l);
        System.out.println("float 类型:"+f);
        System.out.println("double 类型:"+d2);
        System.out.println("char 类型:"+c);
        System.out.println("boolean 类型:"+bool);
        System.out.println("其他类的对象:"+d);
    }
}
```

运行结果为：

```
byte 类型:3
short 类型:4
int 类型:10
long 类型:11
float 类型:3.0
double 类型:23.5
char 类型:s
boolean 类型:false
其他类的对象:Sun Dec 10 09:19:34 CST 2006
```

2.4 项目学做

Java 中在 JDK 5.0 之后可以使用 Scanner 类完成键盘的输入操作。该类使用方法如下：

(1)首先，Scanner 类在 java.util 包中，通过 import java.util.Scanner 将类导入；

(2)创建该类的对象 Scanner sc=new Scanner(System.in)；

(3)通过使用 sc 调用 Scanner 类中的方法实现输入操作。如果输入整型数，通过语句 sc.nextInt()来实现；如果输入浮点数，通过语句 sc.nextDouble()来实现。该类其他方法的使用详见 2.5.7。

```
/*
 *问题描述:根据年利率、贷款期限、贷款金额来计算月支付额和支付总额。
 *程序输入:年利率、贷款期限、贷款金额。
 *程序输出:月支付额和支付总额。
 */
import java.util.Scanner;
public class ComputeLoan {
  public static void main(String[] args) {
    Scanner sc=new Scanner(System.in);
    System.out.println("请输入年利率(注:输入%前面的数字如 5.4):");
    double yearlyRate =sc.nextDouble();//从键盘输入浮点数
    double monthlyRate=yearlyRate/100/12;//由年利率计算月利率
    System.out.println("请输入贷款年限:");
    int year =sc.nextInt();//键盘输入整数
```

```
        System.out.println("请输入贷款金额");
        double amount =sc.nextDouble();
        //实现房贷公式
        double monthlyPay=amount * monthlyRate/1-1/Math.pow(1+monthlyRate,year * 12);
        double totalPay=monthlyPay * year * 12;//计算还款总额
        System.out.println("每月支付额为:" + monthlyPay) ;
        System.out.println("总还款额为:" + totalPay) ;
    }
}
```

2.5　知识拓展

2.5.1　保留字

保留字是系统定义有特殊意义的标识符,例如用于定义一个整型变量要使用保留字 int,要创建一个类需要使用 class 保留字,有些保留字现在没有被 Java 规范使用,等以后扩展的时候也许会用,例如 goto。Java 中所有的保留字都是小写的。

Java 规范中现在使用的保留字又称为关键字。用户在定义标识符的时候不能使用系统保留字。

Java 中的保留字有:

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

2.5.2　转义字符

在 Java 应用中有一些字符不能使用一个符号表示,例如,键盘上的回车键和后退键;还有一些字符如果直接使用会产生歧义,例如,单引号和双引号(通常使用单引号表示一个字符,而使用双引号表示一个字符串)。对于这些字符,通常需要使用转义字符表示。下面是常用的一些转义字符。

```
\ b          \u0008:后退键,键盘上的"←"
\ t          \u0009:Tab 键,用于生成多个空格
\ n          \u000a:换行符
\ f          \u000c:换页符
\ r          \u000d:回车键
\ "          \u0022:双引号 "
\ '          \u0027:单引号 '
```

\\	\u005c:反斜线 \
\0ddd	使用八进制表示的字符
\0xddd	使用十六进制表示的字符
\udddd	使用 Unicode 编码表示的字符

2.5.3 null 符号

null 符号表示一个空值。一个对象等于 null,说明这个对象不存在。关于对象的概念,在本教材后面章节中介绍。

2.5.4 void 符号

void 符号也表示一个空类型,当一个方法不需要返回值的时候,使用 void 表示没有返回值类型。

2.5.5 注释

注释是程序中比较特殊的语句,说它特殊在于编译程序的时候,不会编译注释的内容。

注释用来解释程序的某些部分,提高程序的可读性,方便程序的维护。在调试时,可以使用注释暂时屏蔽某些程序语句。

在 Java 中注释有三种格式:

格式一:

```
//注释的内容
```

格式二:

```
/*
    注释的内容 1
    注释的内容 2
    注释的内容 3
*/
```

格式三:

```
/**
    注释的内容 1
    注释的内容 2
    注释的内容 3
*/
```

格式一用于单行注释,如果注释的内容较少,可以使用单行注释。

格式二用于多行注释,如果注释较多,一行写不完,可以分多行写,以“/*”开始,以“*/”结束。

【注意】 在注释中不能出现“*/”,否则会被认为是注释的结束符号。

格式三被称为文档注释,当使用 javadoc 命令生成帮助文档的时候,文档注释的内容会生成在帮助文档中。前两种注释与 C++中的注释相同,文档注释是 Java 中新增的注释。

2.5.6 Math 类

Java 提供了 java.lang.Math 类,Math 类包含用于执行基本数学运算的方法,如计算指

数、对数、平方根和三角函数等。Math 类的静态属性及常用方法如下：

public static final double E：比任何其他值都更接近 e(即自然对数的底数)的 double 值；

public static final double PI：比任何其他值都更接近 pi(即圆的周长与直径之比)的 double 值；

public static double abs(double a)：返回 double 值的绝对值，该方法的重载方法还支持 float、int 和 long 类型的参数；

public static double max(double a,double b)：返回两个 double 值中较大的一个，该方法的重载方法还支持两个 float 值的比较、两个 int 值的比较、两个 long 值的比较；

public static double min(double a,double b)：返回两个 double 值中较小的一个，该方法的重载方法还支持两个 float 值的比较、两个 int 值的比较、两个 long 值的比较；

public static double pow(double a,double b)：返回第一个参数的第二个参数次幂的值，比如 pow(2,3)返回 2 的 3 次幂的值；

public static double random()：返回一个随机 double 值，该值大于等于 0.0 且小于 1.0；

public static long round(double a)：返回最接近参数的 long 值；

public static int round(float a)：返回最接近参数的 int 值；

public static double sqrt(double a)：返回 double 值的正平方根。

2.5.7　Scanner 类

Java 5 增加了 java.util.Scanner 类，这是一个用于扫描输入文本的新的实用程序。如果从键盘输入，可使用该类创建一个对象，如下：

```
Scanner sc=new Scanner(System.in);
```

然后，sc 对象调用方法，读取用户在命令行输入的各种数据类型。通过 Scanner 类获取用户输入时，控制台会一直等待用户的输入。下面将 Scanner 中常用方法介绍如下：

public byte nextByte()：获取一个 byte 类型的值。

public short nextShort()：获取一个 short 类型的值。

public int nextInt()：获取一个 int 类型的值。

public double nextDouble()：获取一个 double 类型的值。

public float nextFloat()：获取一个 float 类型的值。

public long nextLong()：获取一个 long 类型的值。

public String next()：获取一个 String 类型的值。读取输入直到空格，它不能读由空格隔开的单词。此外，next()在读取输入后将光标放在同一行中(next()只读空格之前的数据，并且光标指向本行)。

public String nextLine()：获取一个 String 类型的值。读取输入时包括单词之间的空格和除回车键以外的所有符号，即读到行尾。读取输入后，nextLine()将光标定位在下一行。

2.6　强化训练

1.定义一个变量保存秒数 500，编写程序计算以秒为单位的时间量所包含的分数和剩余

秒数。例如,500 秒就是 8 分钟 20 秒。

2. 编写程序实现一个数字分解器,目的是能将一个 1000 以内的整数分解,并将其各位数字之和赋值给一个整型变量并输出,比如 436,将会分解得到 4、3、6,计算各位数字之和为 13。

2.7 本章小结

本章利用 Java 中的变量、标识符、常用的数据类型及运算符实现了贷款计算器。通过实现本章的项目,应理解和掌握以下内容:

1. 变量在使用之前要先声明并且初始化。

2. 标识符是用户定义表示变量名、类名、接口名、方法名、方法参数名等所使用的符号。标识符命名由字母、数字、下划线或 $ 符号组成,而且必须以字母、下划线或 $ 符号开头,标识符区分大小写,标识符不能使用系统的保留字。

3. 会使用 Java 中的基本符号包括数字、字符、字符串、布尔值。

4. Java 中提供了 8 种基本数据类型,基本数据类型在内存中占用的位数和表示的范围不同。

5. Java 中精度低的数值可以自动转换为精度较高的类型,如果要将精度高的数据转换成精度较低的类型,这种转换可用强制类型转换完成。

6. Java 中常用运算符包括算术运算符,赋值运算符、自增自减运算符、比较运算符、条件运算符、字符串连接运算符。

2.8 课后习题

一、课后自测

1. 下列变量定义错误的是(　　)。

A. int a;　　B. double b=4.5;

C. boolean b=true;　　D. float f=9.8;

2. 下列数据类型的精度由高到低的顺序是(　　)。

A. float,double,int,long

B. double,float,int,byte

C. byte,long,double,float

D. double,int,float,long

3. 对于一个三位的正整数 n,取出它的十位数字 k(k 为整型)的表达式是(　　)。

A. k = n / 10 % 10　　B. k = (n−n / 100 * 100)%10

C. k = n % 10　　D. k = n / 10

4. 执行完下列代码后,

```
int a=3;
```

```
char b='5';
char c=(char)(a+b);
```

c 的值是(　　)。

A. '8'　　B. 53

C. 8　　D. 56

5. Unicode 是一种(　　)。

A. 数据类型　　B. java 包

C. 字符编码　　D. java 类

6. 6+5%3+2 的值是(　　)。

A. 2　　B. 1

C. 9　　D. 10

7. 下面的逻辑表达式中合法的是(　　)。

A. (7+8)&&(9-5)　　B. (9*5)||(9*7)

C. 9>6&&8<10　　D. (9%4)&&(8*3)

8. 假设 int a=3,b=2,c=1,以下语句正确的是(　　)。

A. c=c/float(a//b)　　B. c=c/((float a)/b)

C. c=(float)c/(a/b)　　D. c=c/(int)(a/(float)b)

9. 指出下列正确的语句(　　)。

A. byte i = 389;　　B. long lv = i*3+4.5;

C. int x = 87L;　　D. long l = 10;

10. 指出下列类型转换中正确的是(　　)。

A. int i='A'　　B. long L=8.4f

C. int i=(boolean)8.9　　D. int i=8.3

11. 以下的选项中能正确表示 Java 语言中的一个整型常量的是(　　)。

A. 12.　　B. -20

C. 1,000　　D. 4　5　6

12. 以下选项中,合法的赋值语句是(　　)。

A. a = = 1;　　B. ++ i;

C. a=a + 1= 5;　　D. y = int (i);

13. 若所用变量都已正确定义,以下选项中,非法的表达式是(　　)。

A. a ! = 4||b==1　　B. 'a' % 3

C. 'a' = 1/2　　D. 'A' + 32

14. 现有一变量声明为 boolean aa;下面赋值语句中正确的是(　　)。

A. aa=false;　　B. aa=False;

C. aa="true";　　D. aa=0;

15. 设有定义 int i = 6; 则执行以下语句后,i 的值为(　　)。

```
i+=i-1;
```

A. 10　　B. 121

C. 11　　D. 100

16. 设有定义 float x＝3.5f，y＝4.6f，z＝5.7f；则以下的表达式中，值为 true 的是(　　)。

A. x＞y || x＞z　　B. x！＝y

C. z＞(y＋x)　　D. x＜y & ！(x＜z)

17. 设有定义 int i=123；long j=456；下面赋值不正确的语句是(　　)。

A. j=i；　　B. j=(long)i；

C. i=(int)j；　　D. i=j；

18. 下列的变量定义中，错误的是(　　)。

A. int i；　　B. int i=10000；

C. int a1=100；　　D. int 123_$；

19. 以下的变量定义语句中，合法的是(　　)。

A. float $_*5＝3.4F；　　B. byte b1＝15678；

C. double a ＝10000；　　D. int _abc_＝3721L；

20. 以下字符常量中不合法的是(　　)。

A. '|'　　B. '\''

C. "\n"　　D. '我'

21. 若以下变量均已正确定义并赋值，下面符合 Java 语言语法的语句是(　　)。

A. b ＝ a！＝7；　　B. a ＝ 7 ＋ b ＋ c＝9；

C. i=12.3＊ ％ 4；　　D. a ＝ a ＋ 7 ＝ c ＋ b；

22. 执行下列程序段后，b、x、y 的值分别是(　　)。

```
int x=6,y=8;
boolean b;
b=x>y&&+x==--y;
```

A. true，6，8　　B. false，7，7

C. true，7，7　　D. false，6，8

23. 以下代码的输出结果是_____。

```
int i=9;
char c='a';
char d=(char)(c+i);
System.out.println(d);
```

24. 以下代码执行完后的输出是_____。

```
int x=3,y=4;
boolean b=true;
System.out.println("b is:"+(b==(y<x)));
```

25. int x=2，y=4，z=3，则 x>y&&z>y 的结果是_____。

26. 在 Java 语言中，逻辑常量只有 true 和_____两个值。

27. 表达式 1/2＊3 的计算结果是_____。

28. 执行以下程序段后：a ＝_____，b ＝_____。

```
int a = 5, b;  b = ++a * 3;
```

29. Java 中的字符使用的是 16 位的_____编码。

30. 当整型变量 n 的值不能被 13 除尽时，其值为 false 的 Java 语言表达式是_____。

31. 表达式 3/6 * 5 的计算结果是_____。

32. 若 a，b 为 int 型变量且已分别赋值为 2，4，表达式！（＋＋a！＝b－－）的值是_____。

33. 若 a，b 为 int 型变量且已分别赋值为 2，6，表达式(a＋＋)＋(＋＋b) ＋a * b 的值是_____。

34. Java 语言中的浮点型数据根据数据存储长度和数值精度的不同，进一步分为 float 和_____两种具体类型。

35. 所有的程序都可以用三种类型的控制结构编写：_____、_____、_____。

二、基础编程

1. 编写程序计算半径为 5 的圆的周长，计算公式为：周长＝2 * 半径 * 圆周率。

2. 编写程序将华氏温度 78 度转换为摄氏温度，转换成的摄氏温度在屏幕上显示出来。转换公式为：摄氏度＝(5/9) * (华氏度－32)。

三、编程挑战

1. 编写程序，获得汉字'我'在 Unicode 码中对应的十进制编码。

2. 编写程序获取一个正随机三位整数，将其以如下形式输出：

```
所取随机数为：* * *
它的百位数字为：*
它的十位数字为：*
它的个位数字为：*
```