

arena teaching system

Java 语言基础



Java企业应用及互联网
高级工程师培训课程

达内集团教学研发部 编著

目 录

Unit01	1
1. JAVA 开发环境	2
1.1. 认识 Linux 操作系统	2
1.1.1. Linux 的由来及发展	2
1.1.2. Linux 目录结构	2
1.1.3. pwd、cd、ls 命令	2
1.1.4. 相对路径和绝对路径	3
1.2. Java 开发环境	3
1.2.1. JAVA 编译运行过程	3
1.2.2. JDK、JRE、JVM 关系	4
1.2.3. 安装 JDK	4
1.2.4. 配置环境变量	5
1.3. Eclipse IDE	5
1.3.1. Eclipse 简介	5
经典案例	7
1. JDK 及 Eclipse 目录结构操作	7
2. JDK 的安装及配置	9
3. 控制台版的 JAVA HelloWorld	14
4. 使用 Eclipse 开发 Java 应用程序	17
课后作业	22
Unit02	24
1. JAVA 变量	27
1.1. 什么是变量	27
1.1.1. 什么是变量	27
1.2. 变量的声明	27
1.2.1. 变量的声明	27
1.2.2. 未经声明的变量不能使用	28
1.2.3. 一条语句中声明多个同类型变量	28
1.3. 变量的命名	28
1.3.1. 命名需要符合标识符语法要求	28
1.3.2. 命名需见名知意，且符合 Java 规范	29
1.4. 变量的初始化	30
1.4.1. 未经初始化的变量不能使用	30
1.4.2. 在变量声明时初始化	30
1.4.3. 在第一次使用变量前初始化	30

1.5. 变量的访问.....	31
1.5.1. 可以对变量中的值进行存取、操作.....	31
1.5.2. 变量的操作必须与类型匹配	31
2. JAVA 基本类型	31
2.1. 8 种基本数据类型	31
2.1.1. 8 种基本数据类型.....	31
2.2. int 类型	32
2.2.1. int 类型	32
2.2.2. 整数直接量是 int 类型.....	32
2.2.3. 整型数据的除法运算中的取整	33
2.2.4. 运算时要防止溢出的发生	33
2.3. long 类型	34
2.3.1. long 类型	34
2.3.2. 使用 long 类型进行较大整数的运算	34
2.3.3. 通过时间毫秒数来存储日期和时间.....	34
2.4. double 类型	35
2.4.1. 使用 double 进行浮点数的运算.....	35
2.4.2. 浮点数直接量是 double 类型.....	35
2.4.3. double 运算时会出现舍入误差	35
2.5. char 类型	36
2.5.1. char 类型	36
2.5.2. 对 char 型变量赋值	36
2.5.3. 使用转义字符	36
2.6. boolean 类型	37
2.6.1. 使用 boolean 变量进行关系运算.....	37
2.7. 类型间的转换.....	37
2.7.1. 基本类型间转换	37
2.7.2. 强制转换时的精度丧失和溢出	37
2.7.3. 数值运算时的自动转换.....	38
2.7.4. byte、char、short 转换为 int	38
3. 运算符和表达式 -1	38
3.1. 算术运算	38
3.1.1. 使用%运算符	38
3.1.2. 使用“++”和“--”运算符.....	39
3.2. 关系运算	39
3.2.1. 使用关系运算符	39
3.3. 逻辑运算	39
3.3.1. 逻辑运算.....	39

3.3.2. 使用 “&&” 运算符	40
3.3.3. 使用 “ ” 运算符	40
3.3.4. 使用 “!” 运算符	40
3.3.5. 关于 “短路逻辑” 的问题	41
经典案例	42
1. 变量使用常用错误汇总	42
2. 整数类型 (int、long) 使用常见问题汇总	44
3. 浮点类型 (float、double) 使用常见问题汇总	46
4. 对 char 类型变量的各种赋值方式汇总	47
5. 类型转换常见问题汇总	48
6. 年龄判断程序	50
课后作业	52
Unit03	54
1. 运算符和表达式 -2	56
1.1. 赋值运算	56
1.1.1. 使用 “=” 进行赋值运算	56
1.1.2. 使用扩展赋值表达式	56
1.2. 字符串连接运算	56
1.2.1. 使用 “+” 进行字符串连接	56
1.3. 条件 (三目) 运算	57
1.3.1. 使用条件 (三目) 运算符	57
1.3.2. 条件 (三目) 运算符的嵌套	57
2. 分支结构	57
2.1. 什么是分支结构	57
2.1.1. 什么是分支结构	57
2.2. if 语句	58
2.2.1. if 语句的执行逻辑	58
2.2.2. if 语句流程图	58
2.2.3. if 语句用于处理分支逻辑	59
2.2.4. if 语句不要省略 “ {} ”	59
2.3. if-else 语句	60
2.3.1. if-else 语句的执行逻辑	60
2.3.2. if-else 语句流程图	60
2.3.3. if-else 语句处理分支逻辑	60
2.4. else if 语句	61
2.4.1. if-else 语句的嵌套	61
2.4.2. else if 语句执行逻辑	61
2.5. switch-case 语句	62

2.5.1. switch-case 语句执行逻辑.....	62
2.5.2. switch-case 和 break 联合使用	62
2.5.3. switch-case 语句用于分支.....	63
2.5.4. switch-case 的优势	63
经典案例	64
1. 闰年判断程序	64
2. 完成收银柜台收款程序 V2.0	66
3. 完成收银柜台收款程序 V3.0	69
4. 完成成绩等级输出程序.....	72
5. 完成命令解析程序	74
课后作业	78
Unit04	81
1. 循环结构	83
1.1. 什么是循环.....	83
1.1.1. 什么是循环结构	83
1.2. while 语句	83
1.2.1. while 语句的执行逻辑.....	83
1.2.2. while 语句的流程图.....	83
1.2.3. while 语句用于处理循环逻辑	84
1.2.4. 使用 break 语句跳出循环.....	84
1.3. do-while 语句	84
1.3.1. do-while 语句的执行逻辑.....	84
1.3.2. do-while 语句的流程图.....	85
1.3.3. do-while 语句用于处理循环逻辑	85
1.3.4. while 和 do-while 语句的区别.....	85
1.4. for 语句	86
1.4.1. 考虑如下循环问题的相同之处	86
1.4.2. for 语句的执行逻辑.....	86
1.4.3. for 语句的流程图.....	87
1.4.4. for 语句用于实现固定次数循环.....	87
1.4.5. for 语句三个表达式特殊用法	87
1.4.6. 循环中使用 break 语句	89
1.4.7. 循环中使用 continue 语句	89
经典案例	90
1. 猜数字游戏 V1.0	90
2. 猜数字游戏 V2.0	95
3. 随机加法运算器	99
课后作业	105

Unit05	107
1. 循环结构	109
1.1. 循环问题	109
1.1.1. 循环问题	109
1.1.2. 循环问题定义——“当”循环	109
1.1.3. 循环问题定义——“直到”循环	109
1.1.4. 循环问题定义——固定次数循环	110
2. 数组	110
2.1. 什么是数组	110
2.1.1. 什么是数组	110
2.2. 数组的定义	111
2.2.1. 定义基本类型数组	111
2.3. 数组的初始化	111
2.3.1. 初始化数组	111
2.4. 数组的访问	112
2.4.1. 获取数组的长度	112
2.4.2. 通过下标访问数组元素	112
2.4.3. 遍历数组元素	113
2.5. 数组的复制	113
2.5.1. System.arraycopy 方法用于数组复制	113
2.5.2. Arrays.copyOf 方法用于数组复制	114
2.5.3. 数组的扩容	114
2.6. 数组排序	115
2.6.1. 数组的排序	115
2.6.2. 数组冒泡排序算法	115
2.6.3. Arrays.sort 方法用于数组排序	116
经典案例	117
1. 九九乘法表	117
2. 求数组元素的最大值	120
3. 求数组元素的最大值放在最后一位	123
4. 冒泡排序算法实现	126
课后作业	130
Unit06	132
1. 方法	133
1.1. 方法（函数，过程）	133
1.1.1. 方法（函数，过程）	133
1.2. 方法的定义	133
1.2.1. 定义方法（函数，过程）的功能	133

1.2.2. 定义参数和返回值	133
1.3. 方法的调用.....	134
1.3.1. return 语句	134
1.3.2. 调用方法时的参数传递	134
经典案例	139
1. 猜字母游戏——设计数据结构	139
2. 猜字母游戏——设计程序结构	142
3. 猜字母游戏——实现字母生成方法	144
4. 猜字母游戏——实现字母检测方法	147
5. 猜字母游戏——实现主方法	150
课后作业	158

Java 语言基础

Unit01

知识体系.....Page 2

JAVA 开发环境	认识 Linux 操作系统	Linux 的由来及发展
		Linux 目录结构
		pwd、cd、ls 命令
		相对路径和绝对路径
	Java 开发环境	JAVA 编译运行过程
		JDK、JRE、JVM 关系
		安装 JDK
		配置环境变量
	Eclipse IDE	Eclipse 简介

经典案例.....Page 7

JDK 及 Eclipse 目录结构操作	pwd、cd、ls 命令
	相对路径和绝对路径
JDK 的安装及配置	安装 JDK
	配置环境变量
控制台版的 JAVA HelloWorld	控制台版的 JAVA HelloWorld
使用 Eclipse 开发 Java 应用程序	Eclipse 简介
	使用 Eclipse 开发 JAVA 应用程序

课后作业.....Page 22


1. JAVA 开发环境

1.1. 认识 Linux 操作系统

1.1.1. 【认识 Linux 操作系统】Linux 的由来及发展

Linux的由来及发展

- Linux起源于1991年，1995年随着互联网的发展而流行开来；
- 是一个开源的操作系统，是一个类Unix操作系统；
- 目前，Linux是主流的服务器操作系统，广泛应用于互联网、云计算、智能手机（Android）等领域；
- 由于Java主要用于服务器端的开发，因此Java应用的部署环境有很多为Linux。

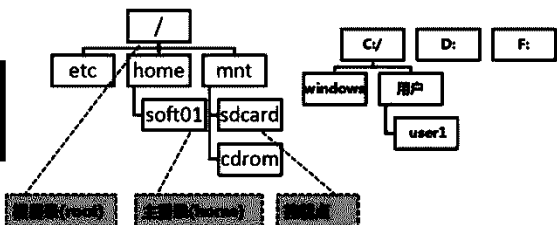


知识讲解

++

1.1.2. 【认识 Linux 操作系统】Linux 目录结构

Linux目录结构



知识讲解

++

1.1.3. 【认识 Linux 操作系统】pwd、cd、ls 命令

pwd、cd、ls命令

- 大多数用于服务器的Linux操作系统不提供图形界面，而是通过命令行的方式进行操作，这一点和Windows不同。Linux提供很多命令，其中经常用于操作目录的命令有：
 - pwd：用于显示当前工作目录；
 - ls：用于查看当前工作目录内容；
 - cd：用于改变当前工作目录；

知识讲解

++

1.1.4. 【认识 Linux 操作系统】相对路径和绝对路径

Tarena
达内科技

相对路径和绝对路径

- 路径用于指明一个文件（或目录）在文件系统的位置。路径有两种表示方式：相对路径和绝对路径。
- 相对路径：文件或目录相对于当前工作目录的位置，例如：路径“soft01/workspace”表示当前目录下的soft01目录下的workspace；
- 有两个特殊的相对路径：
 - “.”表示当前目录
 - “..”表示上一级目录
- 绝对路径：文件或目录相对于根目录的位置。绝对路径都从“/”开始，例如：“/home/soft01”，表示根目录下的home目录下的soft01；

知识讲解

1.2. Java 开发环境

1.2.1. 【Java 开发环境】JAVA 编译运行过程

Tarena
达内科技

JAVA编译运行过程

- 程序员编写的Java源文件（.java）首先要经过编译，生成所谓字节码文件（.class）；
- Java程序的运行需要JVM的支持。JVM是一个软件，安装在操作系统中，为字节码文件提供运行环境；

```

graph LR
    A[Java语言源文件] --> B[字节码文件]
    B --> C[操作系统]
    D[C语言源文件] --> E[可执行文件]
    E --> F[操作系统]
            
```

知识讲解

Tarena
达内科技

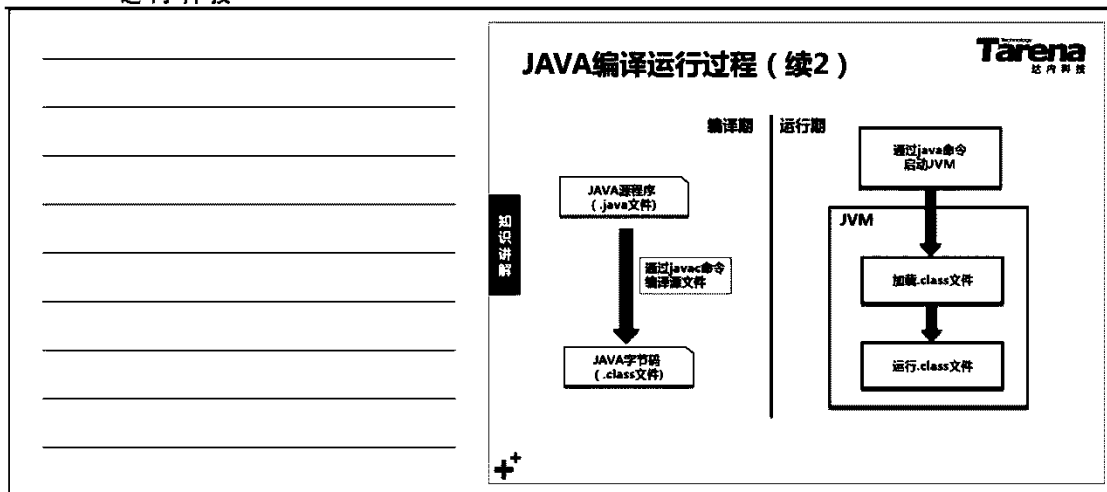
JAVA编译运行过程（续1）

- Java官方提供了针对不同平台的JVM软件，这些JVM遵循着相同的标准，只要是标准的.class文件，就可以在不同的JVM上运行，而且运行的效果相同，这样，就实现了所谓“一次编程到处使用”。

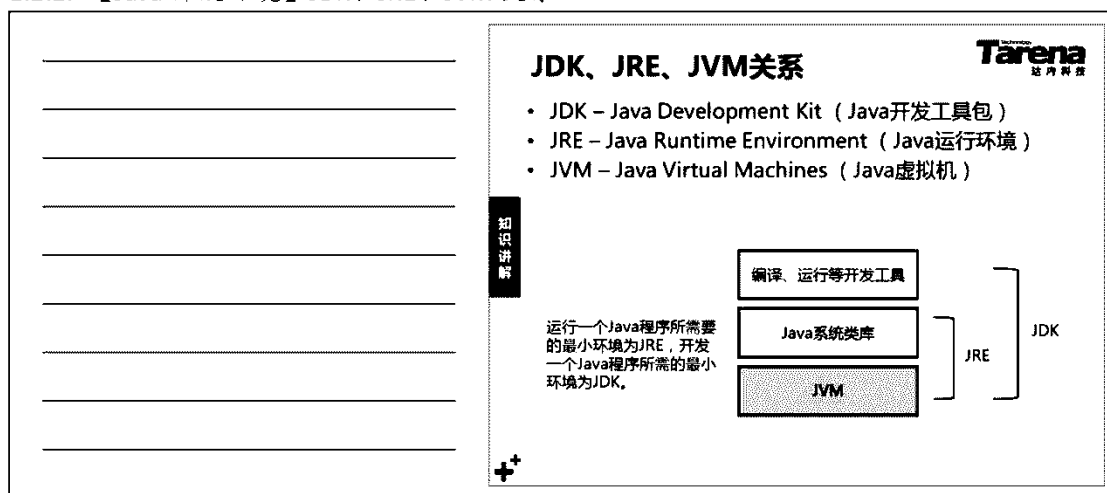
```

graph TD
    A["JAVA源代码 (.java文件)"] --> B["JAVA字节码 (.class文件)"]
    B --> C["JVM for Linux"]
    B --> D["JVM for Windows"]
    B --> E["JVM for XXX"]
            
```

知识讲解



1.2.2. 【Java 开发环境】JDK、JRE、JVM 关系



1.2.3. 【Java 开发环境】安装 JDK

安装JDK


- 可以从官方地址下载并安装JDK：
<http://www.oracle.com/technetwork/java/javase/downloads/index.html>；
- 下载安装时，要注意操作系统（OS）版本和JDK版本之间的对应关系：

Linux x86_64 JDK-7-linux-x64.tar.gz	5.7.0 MB	jdk-7u55-linux-armv6hf.tar.gz
Linux x86_64 JDK-7-linux-x64.tar.gz	57.54 MB	jdk-7u55-linux-armv6hf.tar.gz
Linux x86_64 JDK-7-linux-x64.tar.gz	107.55 MB	jdk-7u55-linux-armv6hf.tar.gz
Linux x86_64 JDK-7-linux-x64.tar.gz	122.94 MB	jdk-7u55-linux-armv6hf.tar.gz
Linux x86_64 JDK-7-linux-x64.tar.gz	155.00 MB	jdk-7u55-linux-armv6hf.tar.gz
Linux x86_64 JDK-7-linux-x64.tar.gz	171.54 MB	jdk-7u55-linux-armv6hf.tar.gz
Mac OS x JDK-7-macosx-x64.dmg	179.43 MB	jdk-7u55-macosx-x64.dmg
Solaris x86_64 JDK-7-solaris-x64.tar.gz	140.02 MB	jdk-7u55-solaris-x64.tar.gz
Solaris x86_64 JDK-7-solaris-x64.tar.gz	20.13 MB	jdk-7u55-solaris-x64.tar.gz
Solaris x86_64 JDK-7-solaris-x64.tar.gz	24.51 MB	jdk-7u55-solaris-x64.tar.gz
Solaris x86_64 JDK-7-solaris-x64.tar.gz	10.29 MB	jdk-7u55-solaris-x64.tar.gz
Solaris SPARC64 JDK-7-solaris-sparc64.tar.gz	139.23 MB	jdk-7u55-solaris-sparc64.tar.gz
Solaris SPARC64 JDK-7-solaris-sparc64.tar.gz	98.12 MB	jdk-7u55-solaris-sparc64.tar.gz
Solaris SPARC64 JDK-7-solaris-sparc64.tar.gz	21.84 MB	jdk-7u55-solaris-sparc64.tar.gz
Solaris SPARC64 JDK-7-solaris-sparc64.tar.gz	18.71 MB	jdk-7u55-solaris-sparc64.tar.gz
Windows x86 JDK-7-windows-x86.exe	123.42 MB	jdk-7u55-windows-x86.exe
Windows x86 JDK-7-windows-x86.exe	124.45 MB	jdk-7u55-windows-x86.exe

Tarena 达内科技


1.2.4. 【Java 开发环境】配置环境变量

<hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/>	<div style="text-align: right;">  </div> <h3 style="text-align: center;">配置环境变量</h3> <ul style="list-style-type: none"> • 如果希望用到JDK所提供的编译（javac）、运行（java）等命令，需要让操作系统可以找到这些命令文件所在的路径； • 可以通过配置PATH环境变量来实现； • PATH环境变量是一系列的目录，在执行命令时，操作系统会依次在PATH环境变量中的每一个目录中查找该命令； <div style="border: 1px solid black; padding: 5px; margin-top: 10px;"> <p>在Linux系统中，配置文件的路径为：/etc/profile； 可以使用cat命令查看配置文件内容： cat /etc/profile</p> </div> <div style="text-align: right;">+</div>
---	---

<hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/>	<div style="text-align: right;">  </div> <h3 style="text-align: center;">配置环境变量（续1）</h3> <ul style="list-style-type: none"> • profile文件中的配置信息： • export JAVA_HOME=/opt/jdk • export CLASSPATH=. • export PATH=/opt/jdk/bin:\$PATH • JAVA_HOME 指向Java JDK安装目录，通知某些软件如何找到JDK安装目录。 • CLASSPATH表示类的搜索路径，简单的可以使用点(.) • PATH 指向JDK的bin目录，javac、java等命令就安装在此目录中。 <div style="text-align: right;">+</div>
---	---

1.3. Eclipse IDE

1.3.1. 【Eclipse IDE】Eclipse 简介

<hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/>	<div style="text-align: right;">  </div> <h3 style="text-align: center;">Eclipse 简介</h3> <ul style="list-style-type: none"> • Eclipse 是目前主流的IDE开发环境； • 所谓集成开发环境（IDE，Integrated Development Environment）是专为程序员提供的应用软件，这些软件往往具备功能强大的图形界面。在IDE的辅助下，程序员可以更加高效的完成编译、调试、提交、重构等工作。作为一个合格的程序员应该对主流的IDE工具有较高的熟练度，但也要防止“过分依赖IDE”问题。 <div style="text-align: right;">+</div>
---	--

Eclipse 简介 (续1)

- 对于Java程序员有许多IDE环境可以选择，但最主流的要数大名鼎鼎的Eclipse了。Eclipse是IBM斥资数千万美元打造的开源项目，如今几乎统治了IDE市场。除了开源之外，Eclipse成功的最大原因在于它是基于“插件”的特性。Eclipse本身是一个平台框架，提供标准的服务，众第三方厂商可以通过开发插件扩展Eclipse的功能，相较于其他功能相对固定的IDE，Eclipse具有高度的灵活性。

打印并播放



Eclipse 简介 (续2)

- 可以从Eclipse的官方网站 (<http://www.eclipse.org/downloads/>) 下载免费的Eclipse。在Eclipse下载页面中选择下载“Eclipse Classic”，这个版本可以理解为是没有特殊安装插件“标准版”Eclipse，其他的版本，根据不同的需要预置了特定插件。

打印并播放



Eclipse 简介 (续3)

- Eclipse下载完成后，不需要特殊的安装，仅仅需要将下载的压缩包解压在某个文件夹中即可。由于Eclipse本身也是用Java语言编写的，它的运行需要有JRE环境，因此必须先安装JDK (或JRE)。

打印并播放



经典案例

1. JDK 及 Eclipse 目录结构操作

问题

为熟练掌握 Linux 下的目录操作，本案例需要完成如下操作：

- 在 Linux 系统下，浏览jdk 的目录结构。
- 在 Linux 系统下，浏览 eclipse 的目录结构。

方案

完成此案例，需要用到一些常用的 Linux 命令。这些命令如下所示：

- pwd ：显示当前工作路径。
- cd：改变当前工作目录。
- ls：浏览目录结构。

步骤

实现此案例需要按照如下步骤进行。

步骤一：浏览 JDK 的目录结构

首先，打开终端，效果如图-1 所示。



图- 1

然后，使用 pwd 命令查看当前所在目录位置。效果如图-2 所示：



图 - 2

由图 - 2 可以看出,当前所在的目录为/home/soft01,此目录为用户主目录。因为 JDK 目录在 /opt 目录下,因此,首先,需要使用 cd 命令进入 opt 目录,接着再次使用 cd 命令进入 opt 目录下的 jdk 目录。界面效果如图-3 所示:



图 - 3

进入 jdk 目录后,使用 ls 命令查看当前目录下的相关文件和目录,效果如图-4 所示:

```
[soft01@java3g jdk]$ ls
bin          lib          register.html
COPYRIGHT   LICENSE     register_ja.html
db          man         register_zh_CN.html
demo        README.html sample
include     README_ja.html src.zip
jre         README_zh_CN.html THIRDPARTYLICENSEREADME.txt
[soft01@java3g jdk]$
```

图 - 4

图 - 4 中,蓝色标识的为文件夹,黑色标识的为文件,绿色标识的为可执行文件,而红色标识的为压缩包。

步骤二：浏览 eclipse 的目录结构

eclipse 也在 opt 目录下,而我们目前在 /opt/jdk 目录下。因此,首先需要进入/opt 目录,然后才能进入 opt 下的 eclipse 目录。

首先,使用命令 cd .. 返回到上一层目录,即 /opt 目录,效果如图-5 所示:

```
[soft01@java3g jdk]$ cd ..
[soft01@java3g opt]$
```

图 - 5

然后，使用 cd 命令进入 eclipse 目录，效果图-6 所示：

```
[soft01@java3g jdk]$ cd ..
[soft01@java3g opt]$ cd eclipse
[soft01@java3g eclipse]$ █
```

图- 6

进入 eclipse 目录后，使用 ls 命令查看当前目录结构，效果如图-7 所示：

```
[soft01@java3g eclipse]$ ls
about_files    eclipse.ini    icon.xpm      plugins
about.html     eclipse.ini.backup  libcairo-swt.so  readme
configuration  epl-v10.html  links         startup.jar
eclipse        features      notice.html
[soft01@java3g eclipse]$
```

图- 7

完整代码

本案例中的代码均为命令代码，因此，没有完整代码呈现。

2. JDK 的安装及配置

问题

安装 JDK，并配置环境变量。

方案

请根据老师上课的讲解和本文档的步骤，完成 Java 开发环境的构建。

步骤

实现此案例需要按照如下步骤进行。

步骤一：下载并安装 JDK

学习 Java 语言要从 Java SE 平台开始。

Oracle 官方提供了两种针对 Java SE 平台的产品—JRE 和 JDK，可以从官方网站免费下载（<http://www.oracle.com/technetwork/java/index.html>）。JRE（Java SE Runtime Environment）称之为 Java SE 运行时环境，提供了运行 Java 应用程序所必须的软件环境，包含有 Java 虚拟机（JVM）和丰富的类库（Libraries）。无论是开发 Java 应用还是仅仅运行一个已经开发好的 Java 应用都必须安装 JRE。JDK（Java Development Kit）称为 Java 开发工具包，是 JRE 的超集，或者说 JDK 包含了 JRE。JDK 中除了包含有 JRE 的所有内容之外还提供了编写 Java 程序所必须的编译器和调试工具等。对于编写 Java

开发的人士一般需要下载 JDK，目前的主流版本为 JDK 6。

Java 官方提供了针对不同操作系统平台的 JDK 版本，如 Windows、Linux、Solaris 等，在下载 JDK 时，可以根据自己的需要来选择，如图-8 所示：

Java SE Development Kit 6u24

Provide Information, then Continue to Download

Select Platform and Language for your download:

Platform: Windows

Language: Multi-language

☒ I agree to the [Java SE Development Kit 6u24 License Agreement](#).

Continue »

选择适当的操作系统平台

图 - 8

以 Windows 版本的 JDK 为例，下载完成后需要进行安装。和安装其他 Windows 下的软件一样，根据安装界面的提示（如图-9 所示）选择好安装目录，然后“下一步”、再“下一步”，直到完成整个安装过程。效果如图 - 9 所示：

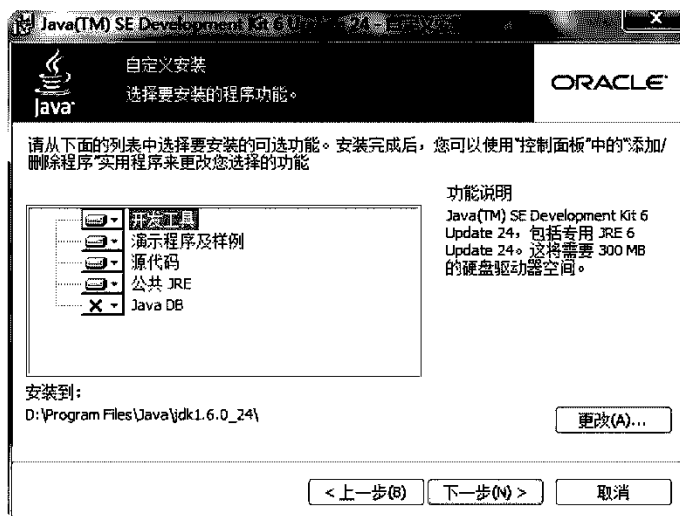


图 - 9

需要注意的是，在图-9 的 JDK 安装功能列表中，Java DB 是可以选择不安装的，这是一个 JDK 自带的纯 Java 语言实现的关系型数据库，通常的 Java 开发时用不到它的。另外，对于 Windows 版的 JDK 安装程序，除了安装一个已经包含有 JRE 的 JDK 之外，还要再安装一个独立的 JRE（所谓“公共的 JRE”）并在注册表中对其进行注册，其目的是对于一些需要用到 JRE 的应用程序可以通过注册信息自动的找到 JRE。

“公共 JRE”是一个可选项，可以选择不安装，毕竟 JDK 中已经包含有了一个完整的 JRE（只不过没有在注册表中注册）；如果选择了该项，在安装完 JDK 之后，安装程序还要提示用户选择公共 JRE 的安装目录。

除了下载 JDK 之外，开发人员一般还要下载 Java 官方文档，这是学习和开发 Java 语

言必备的资料。Java 官方文档的具体下载位置在 <http://www.oracle.com/technetwork/java/javase/downloads/index.html> 页面，如图-10 所示：

Additional Resources

Java SE Floating Point Updater Tool
The FPUUpdater tool allows you to update installed Java Development Kit (JDK) and Java Runtime Environment (JRE) software to address the hang that occurs when parsing certain strings to a binary floating point number.

JDK DST Timezone Update Tool - 1.3.39
The tzupdater tool is provided to allow the updating of installed JDK/JRE images with more recent timezone data in order to accommodate the latest timezone changes. [Learn more](#)

Java SE 6 Documentation

- [Java SE 6 Documentation](#)
- [Docs Installation Instructions](#)
- [Oracle License](#)

[Download](#)

[ReadMe](#)
[License](#)

[Download](#)

[ReadMe](#)

[Download](#)

图- 10

文档下载解压后将以 HTML 格式呈现，使用浏览器打开文档首页 (index.html)，可以看到如图-11 所示的“JDK 全貌”，通过单击相关链接可以了解其细节。

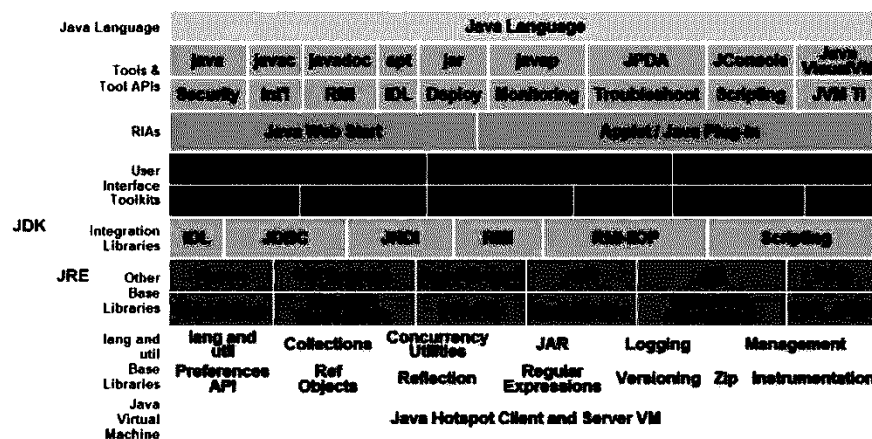


图- 11

由图-11 可以看出：JDK 包含 JRE 和开发工具包 (Tools & Tool APIs)；而 JRE 包含有 Java 虚拟机 (JVM) 和各种类库 (Libraries)。

步骤二：配置环境变量

JDK 安装完成后，在 JDK 安装路径下可以看到如下内容：

- bin 目录：用于存放 JDK 工具命令，比如用于编译 Java 程序的 javac 命令、用于启动 JVM 运行 Java 程序的 java 命令、用于生成文档的 javadoc 命令和

用于打包的 jar 命令等等；

- jre 目录：用于存放 JDK 所包含的 JRE，其中包含有 JVM 和核心类库；
- lib 目录：用于存放 JDK 工具命令所对应的工具包 (Tool APIs)；
- demo 目录：用于存放一些示例程序；
- src.zip 文件：用于存放核心类库的 Java 源代码。

其中，bin 中的 javac 命令和 java 命令是我们很快要用到的命令。如果想用到这些命令，需要让操作系统可以找到这些命令文件所在的路径。在 Windows 操作系统中，可以通过配置 Path 环境变量来实现。Path 环境变量是一串用分号 (;) 分隔开的目录，在通过控制台运行一个命令时，Windows 会依次在 Path 环境变量中的每一个目录中查找该命令，如果找到就可以执行，否则就会有如图-12 所示的错误提示：

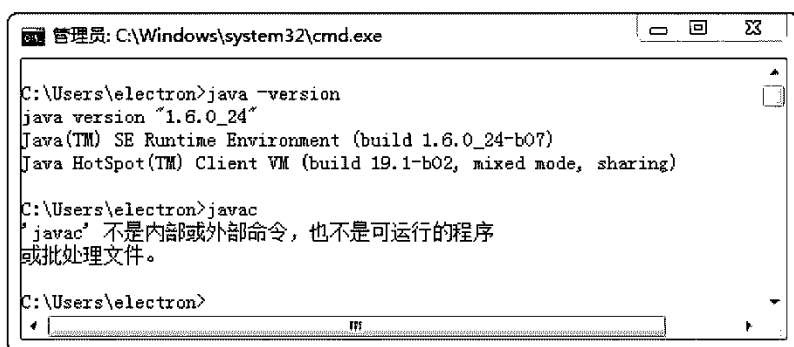


图- 12

在 Windows 系统中通过“控制面板”→“系统”→“高级系统设置”打开如图-13 所示的对话框：

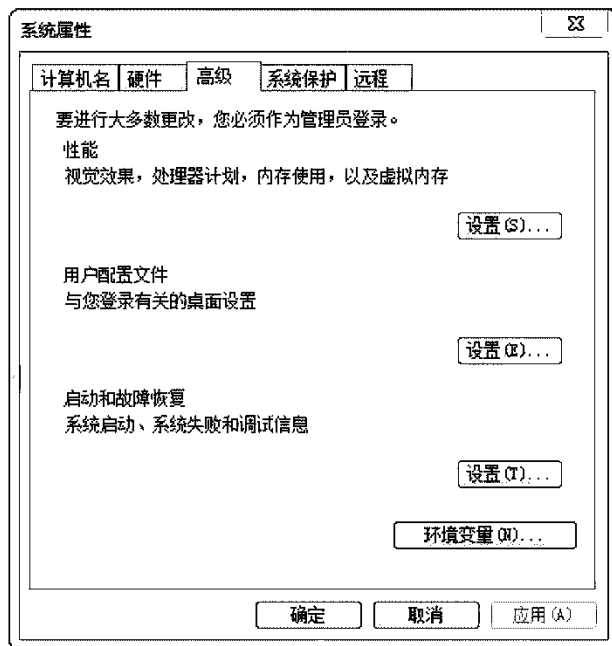


图- 13

单击图 - 13 中的“环境变量”按钮，并在弹出的对话框中双击系统变量 Path，并对其

进行编辑。在 Path 变量值的末尾追加 JDK 开发工具路径(“JDK 安装路径\bin”),假设 JDK 的安装路径为:“D:\Program Files\Java\jdk1.6.0_24”则需追加的 Path 路径为:“;D:\Program Files\Java\jdk1.6.0_24\bin”。注意,路径之间需要用分号隔开。效果如图-14 所示:

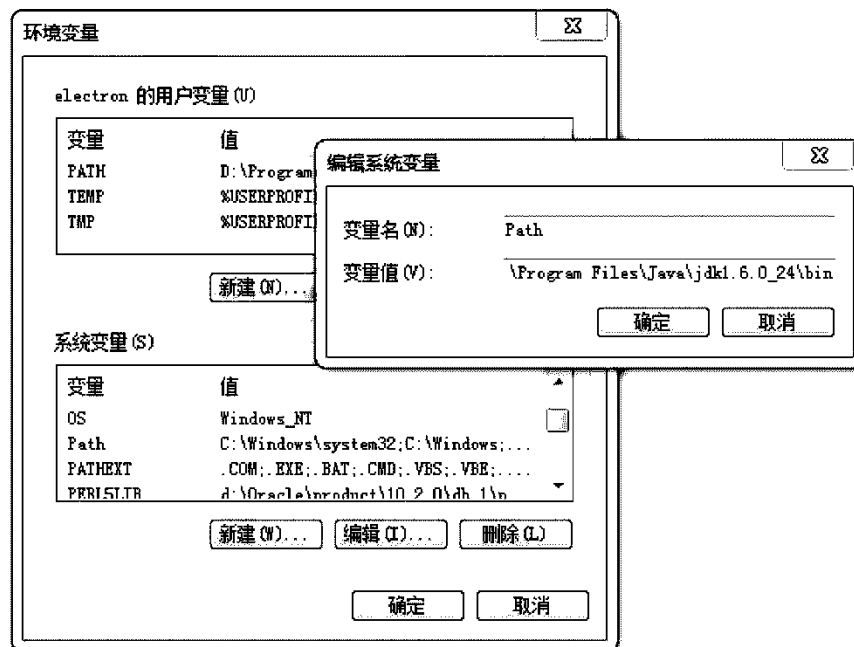


图- 14

环境变量设置完成后,通过“附件”→“命令提示符”(或者运行 cmd 命令)打开控制台,键入 java 或 javac 命令,看到输出正常的提示信息就表示环境变量配置成功了。效果如图-15 所示:

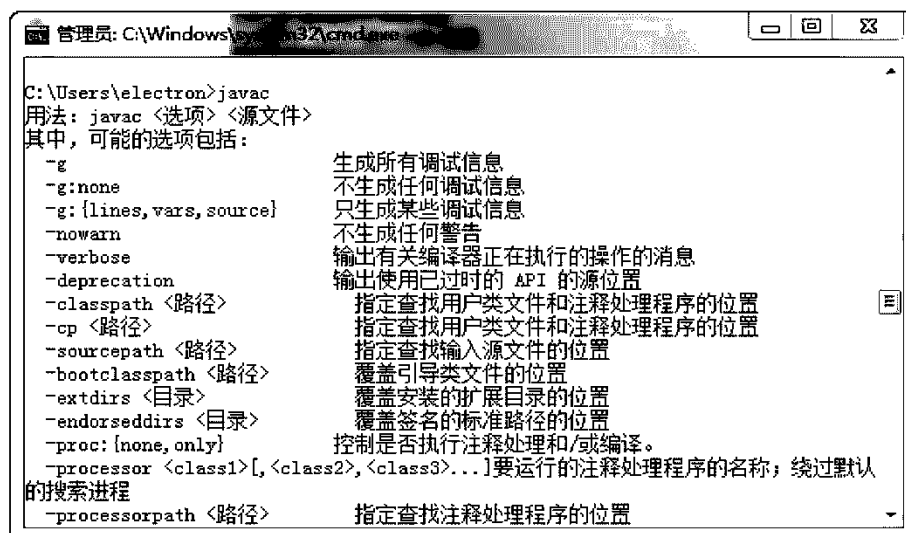


图- 15

需要注意的是,如果在安装 JDK 时选择安装了公共 JRE,则不需要配置 Path 环境变量也可以运行 java 命令,这是由于公共 JRE 路径写入了注册表的缘故。

完整代码

本案例均为实际操作，因此没有完整代码呈现。

3. 控制台版的 JAVA HelloWorld

问题

使用 vi 编写 HelloWorld.java 程序，运行后，在控制台输出 “Hello World”。

方案

请根据老师上课的讲解和本文档的步骤，慢慢体会 Java 的魅力。

步骤

实现此案例需要按照如下步骤进行。

步骤一：打开 vi，进入编辑模式

首先，打开终端；然后，在终端内输入 vi 及文件名称 HelloWorld.java 后，就进入 vi 全屏幕编辑画面，终端界面如图-16 所示。

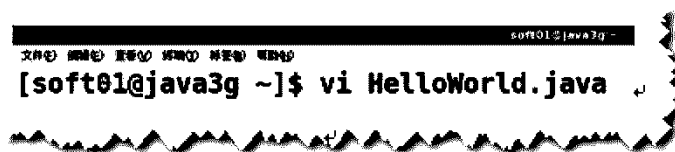


图- 16

vi 编辑界面如图-17 所示。



图- 17

步骤二：编写创建类的代码

首先，在 vi 编辑界面上，输入命令 i，使 vi 进入插入模式；然后，键入创建类的代码，

界面如图-18 所示。

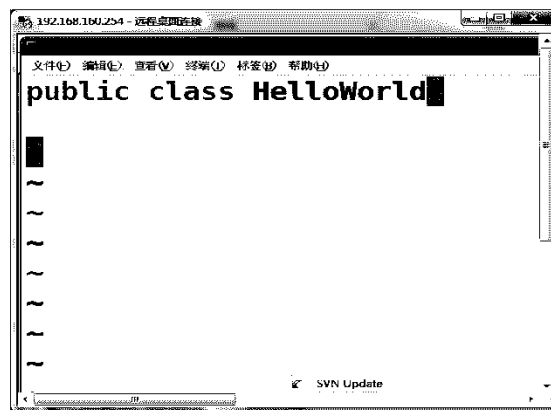


图- 18

Java 语言是纯粹的面向对象的语言，任何一段 Java 代码都需要从定义一个类开始。至于“类”的详细将在后续的课程内容中详解，这里可以暂做了解。public class 类名 { ... } 是定义类的语法，除了类名可以自己指定外，其他的 public 和 class 都是 Java 关键字（所谓 Java 关键字是指 Java 语言中预先定义的代表特定含义的字符），这里只要保证不要写错就可以了。

另外，在此需要注意，Java 语言是严格大小写区分的，“H”和“h”是两个不同的字符，编写时需要注意。

步骤三：定义 main 方法

接着，在 HelloWorld 类中，定义 Java 应用程序的入口方法 main，代码如图-19 所示：

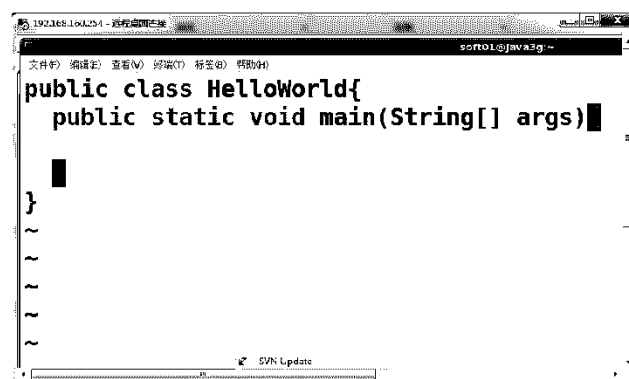


图- 19

这行代码的具体含义需要在后续课程内容中细述，这里只要记住两点：第一、如果一个类包含有这样一个方法，则该类就是一个可以被运行的类，而且该方法是程序的入口，也就是说程序从该方法的第一行代码开始逐行运行。第二、慢慢写，不要写错！

步骤四：输出信息到控制台

在 main 方法中，添加代码，以输出“Hello World”到控制台界面显示，代码如图-20 所示。

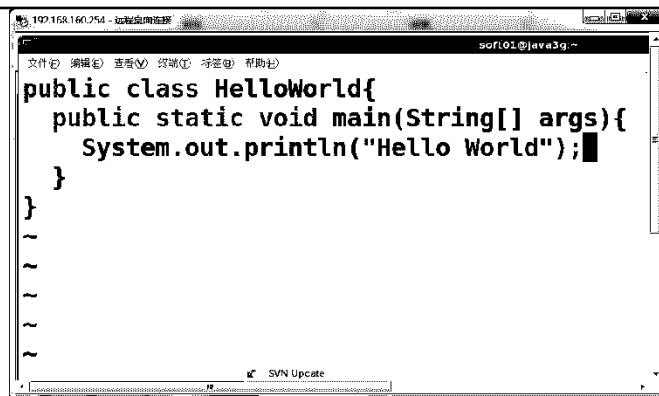


图- 20

步骤五：保存代码，退出 vi

首先，当前 vi 在插入模式下，按一下 Esc 键转到命令行模式；然后，按一下：冒号键进入最后一行模式；最后，输入命令 wq，存盘并退出 vi，回到终端界面，如图-21 所示。

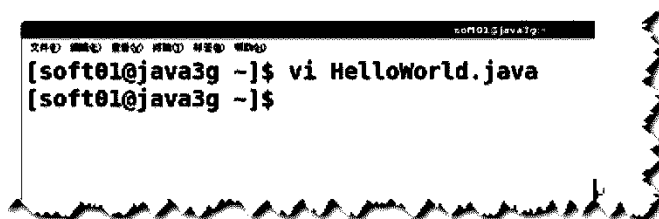


图- 21

此时，将 HelloWorld.java 文件保存在当前目录下，如图-22 所示。

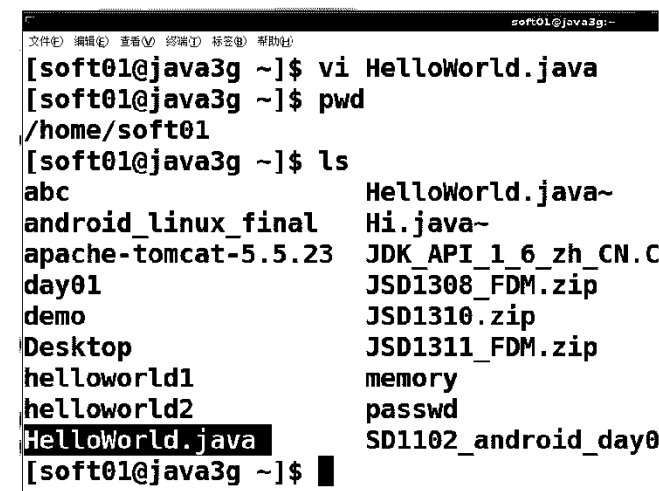


图- 22

步骤六：编译源文件

源文件编写完成后，需要进行编译，即转换为可以在 JVM 中运行的字节码文件。JDK 工具中的 javac 命令（在 JDK 安装目录的 bin 子目录下）可以实现这样的工作。

在终端，接着键入如下命令：

```
javac HelloWorld.java
```

命令运行成功后会在当前目录下生成 HelloWorld.class 文件。当然，对于初学者，可能会出现各种各样的错误。需要根据错误提示信息耐心的修改（大多数错误可能是由于字符书写错误造成的，比如大小写的问题）。

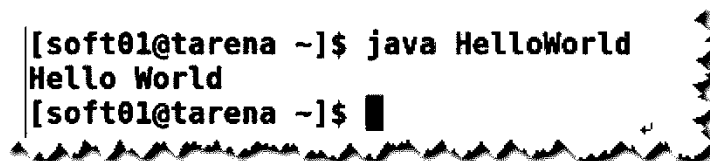
步骤七：启动 JVM，运行 Java 程序

编译成功之后，就可以使用 java 命令运行该字节码文件（更准确的说法是：启动 JVM 运行字节码文件）。

接着在终端，键入如下命令：

```
java HelloWorld
```

如果代码书写正确的，执行上述命令后，终端显示内容如图-23 所示。



```
[soft01@tarena ~]$ java HelloWorld
Hello World
[soft01@tarena ~]$
```

图- 23

从图-23 中可以看出，程序的输出结果为 “Hello World”。

完整代码

本案例的完整代码如下所示：

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

4. 使用 Eclipse 开发 Java 应用程序

问题

在上一案例“控制台版的 JAVA Hello World”中，我们使用最简单的编辑工具，编写源代码，并通过调用 JDK 工具命令体验了 Java 程序的编译、运行的全过程。理论上，我们可以使用这样的方式编写任何 Java 程序，但在真实的企业项目中，这种原始的开发方式势必会带来大量繁琐、重复、易错的操作，会极大的降低工作效益；同时也不利于项目的整体管理。在真实的场景中，开发人员总是会借助一些强大的“集成开发环境”进行代码的编写、调试、测试、提交、重构等操作，例如：Eclipse。

本案例要求使用开发工具 Eclipse 编写 HelloWorld.java 程序。在 Eclipse 的控制

台中，该程序的输出结果如图-24 所示：

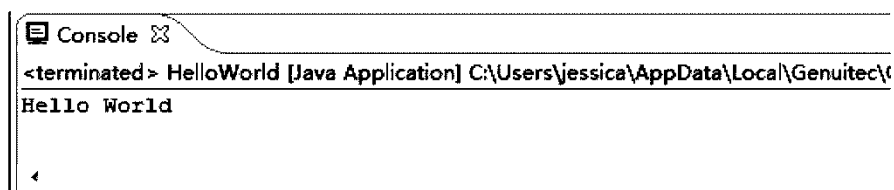


图- 24

方案

请根据老师上课的讲解和本文档的步骤，慢慢体会 Eclipse 工具开发 Java 程序的便利。

步骤

实现此案例需要按照如下步骤进行。

步骤一：开启开发工具 Eclipse

eclipse 启动后会弹出如下对话框，如图-25 所示。

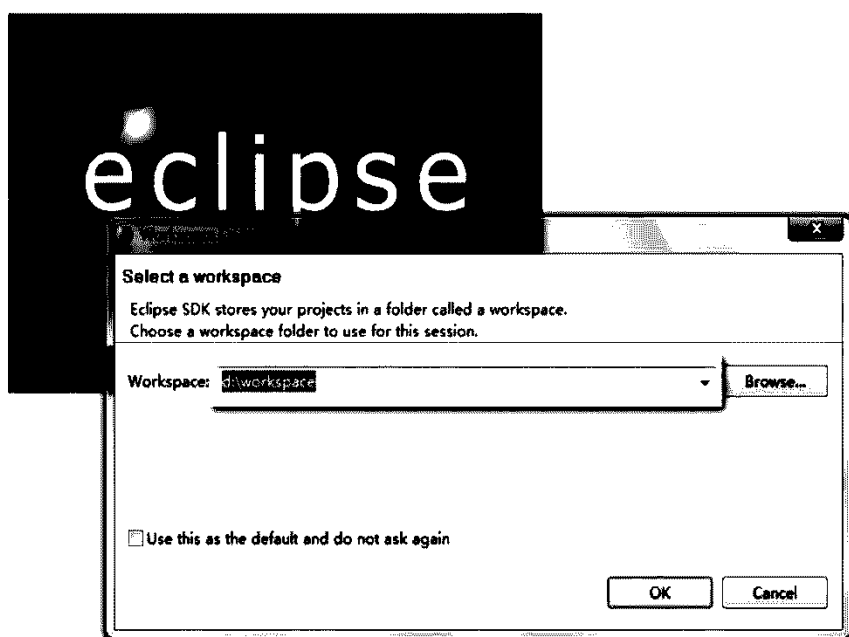


图- 25

弹出的对话框“Workspace Launcher”用于选择“工作区”(Workspace)。所谓“工作区”是指 Eclipse 用于存储工程的路径。Eclipse 通过“工程”(Project)来组织资料。程序员编写的源文件、编译生成类文件等以特定的目录结构存储在工程文件夹中。

步骤二：创建 Java 工程

选择菜单操作“File → New → Java Project”用于创建一个适合编写 Java 基本应用程序的工程。在弹出的对话框中填写工程的名称(Project Name)然后单击“Finish”

按钮。如图-26 所示：

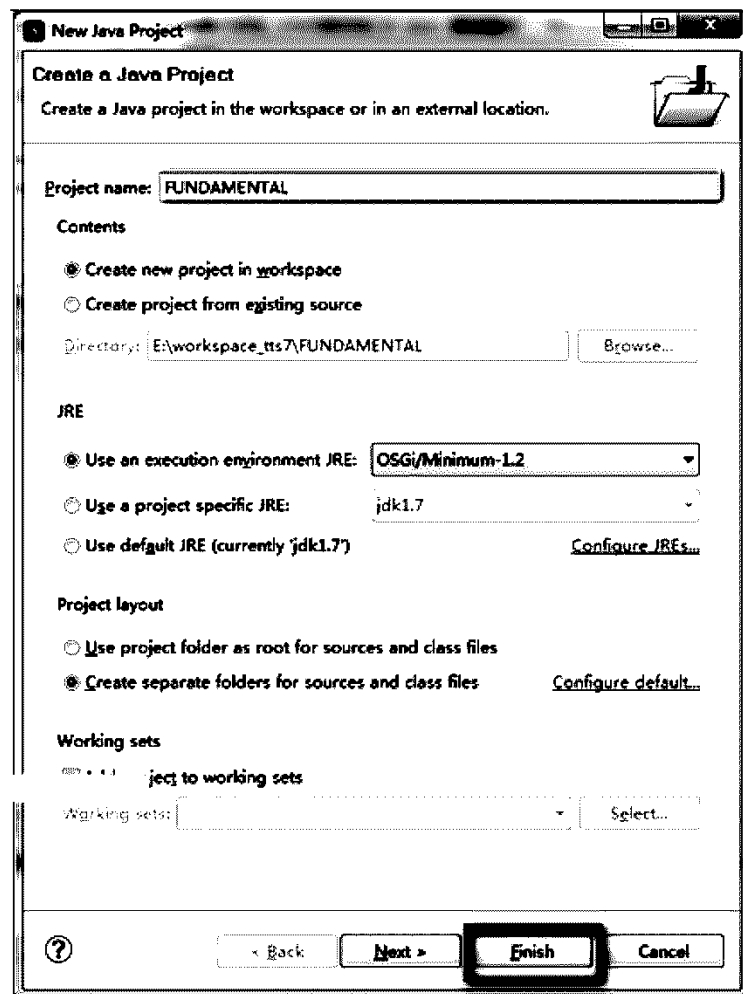


图- 26

工程创建完成后，会在工作区目录（Eclipse 启动时指定）生成一个与工程名称同名的文件夹。打开此文件夹，会看到如图-27 所示的目录结构：

名称	修改日期	类型	大小
.settings	2014/2/12 16:45	文件夹	
bin	2014/2/12 16:45	文件夹	
src	2014/2/12 16:45	文件夹	
.classpath	2014/2/12 16:45	CLASSPATH 文件	1 KB
.project	2014/2/12 16:45	PROJECT 文件	1 KB

图- 27

其中：“src”文件夹用于存放用户编写的 Java 源文件；“bin”文件夹用于存放 Eclipse 自动编译生成的 class 文件。Eclipse 具备自动编译的功能，当用户在编写 Java 源文件的同时，Eclipse 会自动的调用系统的 Java 编译器编译该文件，并将编译好的 class 文件存放在 bin 目录中。另外，“.classpath”和“.project”两个文件以及“.settings”文

文件夹中的内容是 Eclipse 用来维护工程信息的，一般可以不去理会。

步骤三：创建名为 HelloWorld 的类

工程创建完成以后，可以通过菜单操作“File → New → Class”创建 Java 源文件。在弹出的对话框中，填写要创建的 Java 类的类名（Name）和包名（Package），如图-28 所示。（关于包的含义将在后续课程中详述。）

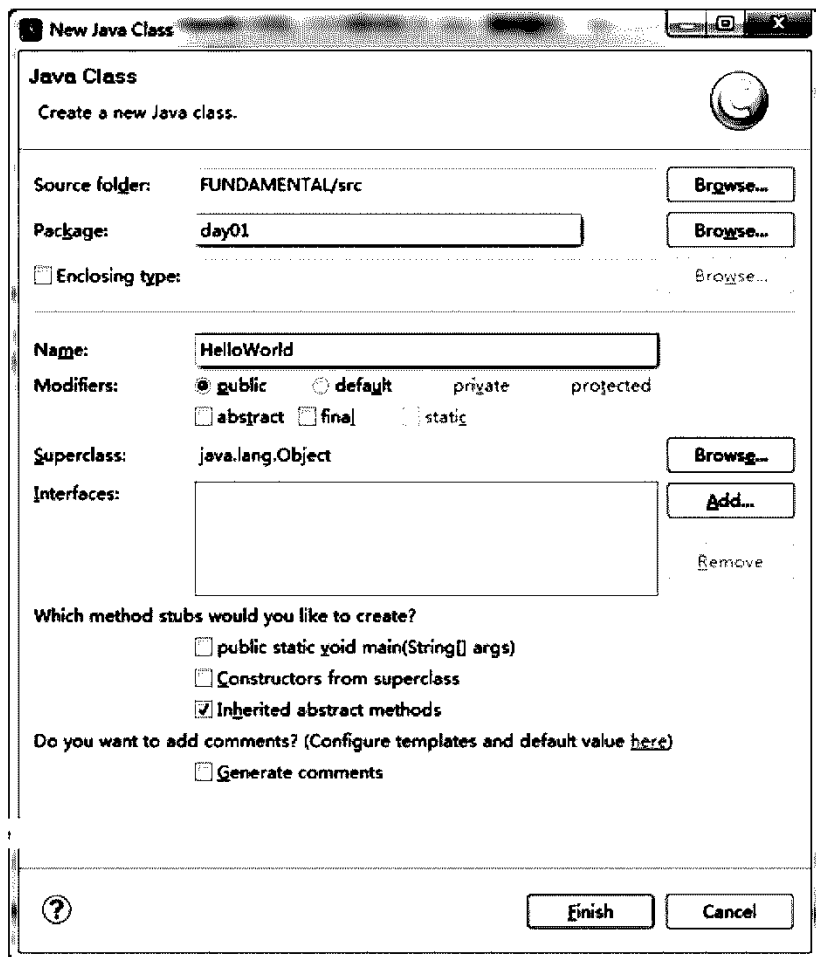


图- 28

填写完类名(本例中为“HelloWorld”)和包名(本例中为“day01”)后,单击“Finish”按钮。Eclipse 即创建了一个名为 HelloWorld.java 的源文件。

步骤四：定义类

在 Eclipse 编写 HelloWorld 程序，如图-29 所示。

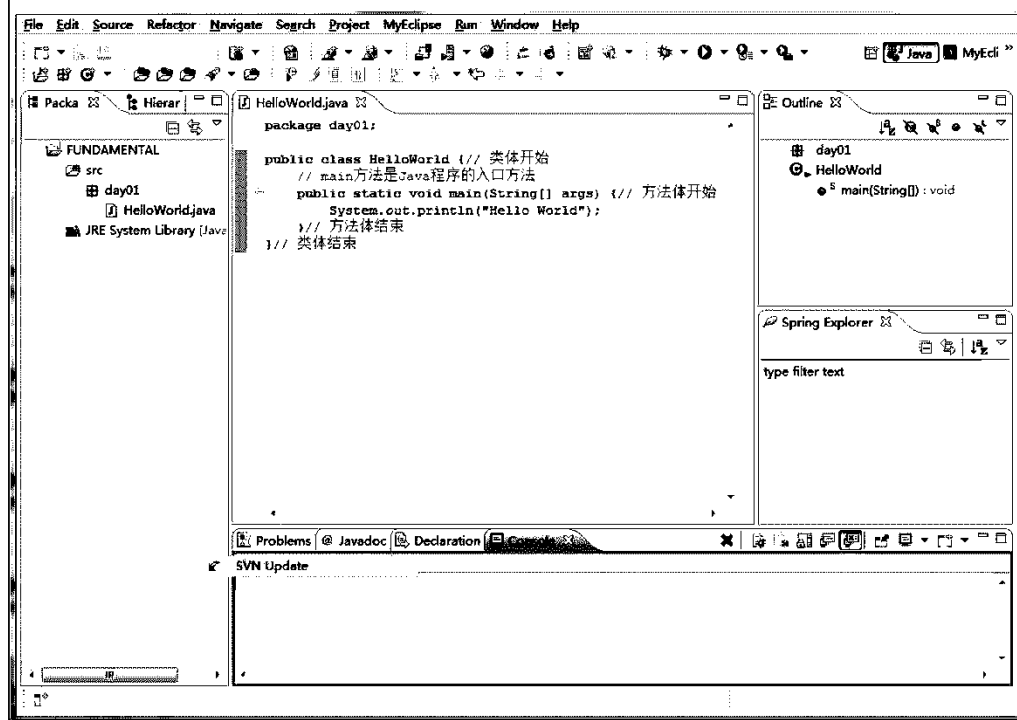


图- 29

程序编写完成，确认无编译错误后，可以使用菜单操作“Run → Run As → Java Application”来运行该程序。运行的结果会在代码下方的 Console 视图中显示。这里需要指出的是：这个操作在本质上与先前通过命令行方式运行 Java 程序并没有什么不同，可以理解为是 Eclipse 调用 JDK 的 java 命令，用更加友好的图形化界面方式实现运行 Java 程序的操作而已。

完整代码

本案例的完整代码如下所示：

```
package day01;

public class HelloWorld { // 类体开始
    // main 方法是 Java 程序的入口方法
    public static void main(String[] args) { // 方法体开始
        System.out.println("Hello World");
    } // 方法体结束
} // 类体结束
```

课后作业

1. 下面关于 Linux 说法正确的是?

- A . Linux 是计算机操作系统。
- B . Linux 系统下可以开发 Java 应用程序。
- C . Linux 系统和 Windows 系统使用的 JVM 相同。
- D . Linux 系统是开放源代码的。

2. 下面关于 Linux 目录结构说法正确的是?

- A . Linux 文件系统以树形目录的形式体现的，只有一个根目录。
- B . Linux 文件系统有两个根目录。
- C . Linux 文件系统和 Window 系统一样都有盘符，例如：c : /。
- D . Linux 下 U 盘的挂载点是盘符，而不是目录。

3. 用户在终端输入下列命令，最终显示的结果是?

```
[soft01@java3g ~]$ pwd
/home/soft01
[soft01@java3g ~]$ cd ..
[soft01@java3g home]$ pwd
```

- A ./home/soft01
- B ./soft01
- C ./home
- D ./home/soft01/java3g

4. 当前工作目录在/opt/jdk 目录下，需要转到/opt/eclipse 目录，下列命令正确的是?

- A . cd /opt/eclipse
- B . cd ../eclipse
- C . cd ../jdk
- D . cd opt/eclipse

5. 简述 Java 编译及运行过程

6. 名词解释 JVM、JRE、JDK

7. 根据 Cookbook 文档实现控制台版的 JAVA HelloWorld

8. 下列关于 Eclipse 说法正确的是？

A . 在源文件编写的同时 Eclipse 会自动的调用 Java 编译器编译该文件，如果出现任何编译错误，Eclipse 会立刻发现，并提示给用户。

B . Eclipse 中的 Workspace 是用于存储工程的路径。

C . 在 Linux 系统下，可以使用 ./eclipse 启动执行程序 eclipse。

D . Eclipse 是开放源代码的 Java 开发平台。

9. 根据 Cookbook 文档实现 JAVA HelloWorld (Eclipse)

Java 语言基础

Unit02

知识体系.....Page 27

JAVA 变量	什么是变量	什么是变量
	变量的声明	变量的声明
		未经声明的变量不能使用
		一条语句中声明多个同类型变量
	变量的命名	命名需要符合标识符语法要求
		命名需见名知意，且符合 Java 规范
	变量的初始化	未经初始化的变量不能使用
		在变量声明时初始化
		在第一次使用变量前初始化
	变量的访问	可以对变量中的值进行存取、操作
		变量的操作必须与类型匹配
JAVA 基本类型	8 种基本数据类型	8 种基本数据类型
	int 类型	int 类型
		整数直接量是 int 类型
		整型数据的除法运算中的取整
		运算时要防止溢出的发生
	long 类型	long 类型
		使用 long 类型进行较大整数的运算
		通过时间毫秒数来存储日期和时间
	double 类型	使用 double 进行浮点数的运算
		浮点数直接量是 double 类型
		double 运算时会出现舍入误差
	char 类型	char 类型
		对 char 型变量赋值
		使用转义字符
	boolean 类型	使用 boolean 变量进行关系运算
	类型间的转换	基本类型间转换
		强制转换时的精度丧失和溢出
		数值运算时的自动转换
		byte、char、short 转换为 int

运算符和表达式 -1	算术运算	使用%运算符
		使用 “++” 和 “--” 运算符
	关系运算	使用关系运算符
	逻辑运算	逻辑运算
		使用 “&&” 运算符
		使用 “ ” 运算符
		使用 “!” 运算符
		关于 “短路逻辑” 的问题

经典案例.....Page 42

变量使用常用错误汇总	什么是变量
	变量的声明
	未经声明的变量不能使用
	一条语句中声明多个同类型变量
	命名需要符合标识符语法要求
	命名需见名知意，且符合 Java 规范
	未经初始化的变量不能使用
	在变量声明时初始化
	在第一次使用变量前初始化
	可以对变量中的值进行存取，操作
	变量的操作必须与类型匹配
整数类型 (int、long) 使用常见问题汇总	int 类型
	整数直接量是 int 类型
	整型数据的除法运算中的取整
	运算时要防止溢出的发生
	long 类型
	使用 long 类型进行较大整数的运算
	通过时间毫秒数来存储日期和时间
浮点类型 (float、double) 使用常见问题汇总	使用 double 进行浮点数的运算
	double 运算时会出现舍入误差
	浮点数直接量是 double 类型
对 char 类型变量的各种赋值方式汇总	char 类型
	对 char 型变量赋值
	使用转义字符
类型转换常见问题汇总	基本类型间转换
	强制转换时的精度丧失和溢出
	数值运算时的自动转换
	byte、char、short 转换为 int
年龄判断程序	使用关系运算符
	逻辑运算
	使用 “&&” 运算
	使用 “ ” 运算

	使用"!"运算
	关于"短路逻辑"的问题

课后作业.....Page 52

1. JAVA 变量

1.1. 什么是变量

1.1.1. 【什么是变量】什么是变量

Technology
Tarena
达内科技

什么是变量

- 变量就是指代在内存中开辟的存储空间，用于存放运算过程中需要用到的数据。

```
int a = 5;
int b = 6;
int c = a + b;
```

变量a、b和c指代内存中三块用于存储整数的存储空间，分别用来存储两个整数以及这两个整数之和。

++

Technology
Tarena
达内科技

什么是变量（续1）

- 对于变量我们需要关注起如下几个方面：
 - 变量的声明：用特定的语法声明一个变量，让运行环境为其分配空间；
 - 变量的命名：变量需要有个见名知意的名字，而且要符合Java语言规范；
 - 变量的初始化：变量声明后，要为其赋一个确定的初值后再使用；
 - 变量的访问：可以对变量中的数据进行存取、操作，但必须和其类型匹配。

++

1.2. 变量的声明

1.2.1. 【变量的声明】变量的声明

Technology
Tarena
达内科技

变量的声明

- 当需要使用一个变量时，必须对该变量进行声明
- 变量的声明包含两点：变量名和变量类型。

变量必须指明其类型



int a

必须指明变量名称



JVM会为该变量在内存中开辟存储空间，不同的变量类型决定了存储空间的结构。

++

1.2.2. 【变量的声明】未经声明的变量不能使用



<hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/>	<div style="text-align: right;">  </div> <h4>未经声明的变量不能使用</h4> <ul style="list-style-type: none"> Java语言语法规则规定，变量使用之前必须声明，否则会有编译错误。 <div style="border: 1px solid black; padding: 5px; margin: 10px 0;"> <pre>public static void main(String[] args) { a = 1; // 编译错误，变量没有声明 int score = 0; scord = 100; // 编译错误 System.out.println(score); }</pre> </div> <div style="border: 1px solid black; padding: 5px; margin: 10px 0; text-align: center;"> 变量没有声明，有时候是因为拼写错误造成的 </div> <div style="text-align: right;">  </div>
---	---

1.2.3. 【变量的声明】一条语句中声明多个同类型变量

<hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/>	<div style="text-align: right;">  </div> <h4>一条语句中声明多个同类型变量</h4> <ul style="list-style-type: none"> 如果多个变量的类型一样，可以在一条语句中声明，中间用逗号分隔。 <pre>public static void main(String[] args) { int a=1, b=2; // 声明了两个整型变量，分别赋值为1和2。 int c, d=3; // 声明了两个整型变量，d赋初值为3，c没有赋初值。 }</pre> <div style="text-align: right;">  </div>
---	---

1.3. 变量的命名

1.3.1. 【变量的命名】命名需要符合标识符语法要求

<hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/>	<div style="text-align: right;">  </div> <h4>命名需要符合标识符语法要求</h4> <ul style="list-style-type: none"> 变量的命名必须符合Java标识符的规则： <ul style="list-style-type: none"> 可以由字母、数字、“_”和“\$”符组成； 首字符不能以数字开头； Java大小写敏感，命名变量时需要注意； 不能使用Java保留字（一些Java语言规定好的，有特殊含义的字符），如：int、if、for、break等； 中文可以作为变量名，但不提倡使用。 <div style="text-align: right;">  </div>
---	---

命名需要符合标识符语法要求 (续1)



- 下面的代码正确吗？


```
int 123go = 100 ;
int 成绩 = 60 ;
int break= 200 ;

int score = 80 ;
System.out.println(Score) ;
```

代码错误

+

命名需要符合标识符语法要求 (续2)



- 下面的代码体现了Java变量的命名规则


```
int 123go = 100 ; // 编译错误，不能以数字开头。
int 成绩 = 60 ; // 编译没错，但不建议使用。
int break= 200 ; // 编译错误，break是Java保留字。
int score = 80;
System.out.println(Score);
// 编译错误，Java大小写敏感，Score变量没有声明。
```

代码错误

+

1.3.2. 【变量的命名】命名需见名知意，且符合 Java 规范

命名需见名知意，且符合Java规范



- Java 变量名的定义应“见名知意”；
- 同时，Java编程规范要求：变量的命名需采用“驼峰命名法”，即如果变量的名字由多个单词组成，除第一个单词外，其他单词的首字母大写，其余的字母小写，
- 例如：salary、javaScore、studentName 等。


代码错误

+

1.4. 变量的初始化

1.4.1. 【变量的初始化】未经初始化的变量不能使用

未经初始化的变量不能使用




- Java语言规定变量在使用之前必须初始化，即必须给该变量赋予特定的值。
 - C语言中，变量使用之前可以不需要初始化，但是，其初始值不确定；Java语言的设计者为了避免因此而带来的错误，规定变量必须初始化之后才能使用。

```
public static void main(String[] args) {  
    int a, b = 10;  
    int c = a + b; // 编译错误，变量a没有初始化  
    System.out.println(c);  
}
```

1.4.2. 【变量的初始化】在变量声明时初始化

在变量声明时初始化




- 可以在变量声明时初始化：
变量类型 变量名称 = 初始值

```
public static void main(String[] args) {  
    int sum = 0;  
    int a = 5;  
    int b = 6;  
    sum = a + b;  
    System.out.println(sum);  
}
```

1.4.3. 【变量的初始化】在第一次使用变量前初始化

在第一次使用变量前初始化



- 可以在变量声明以后，通过赋值语句对变量进行初始化，但一定确保在第一次使用该变量之前。

```
public static void main(String[] args) {  
    int sum;  
    sum = 0; // 在使用sum变量之前对其进行初始化。  
    sum = sum + 100;  
    System.out.println(sum);  
}
```

1.5. 变量的访问

1.5.1. 【变量的访问】可以对变量中的值进行存取、操作

知识讲解

可以对变量中的值进行存取、操作 Tarena 达内科技

- 变量是存放数据的空间，可以对其赋值、更改和操作；要把对变量的操作理解为是对其所存储的数据的操作。

```
public static void main(String[] args) {
    int a = 100;
    a = a + 200; // 该条语句的含义为：
                // 将变量a中的值加上200所得结果再存入变量a
}
```

+ +

1.5.2. 【变量的访问】变量的操作必须与类型匹配

知识讲解

变量的操作必须与类型匹配 Tarena 达内科技

- 变量在声明时指定了类型，Java编译器会检测对该变量的操作是否与其类型匹配，如果对变量的赋值或者操作与其类型不匹配，会产生编译错误。

```
public static void main(String[] args) {
    int salary;
    salary = 15000.50; // 编译错误
                    // 整型变量不可以赋予浮点值（小数）。

    double d = 123.456;
    int n = d%2; // 编译错误
              // d%2为double型，不能赋值给整型n。
}
```

+ +

2. JAVA 基本类型

2.1. 8 种基本数据类型

2.1.1. 【8 种基本数据类型】8 种基本数据类型

知识讲解

8种基本数据类型 Tarena 达内科技


- Java语言有8种基本数据类型，分别用于存储整数、浮点数、字符数据和布尔类型数据。

```

graph LR
    A[Java基本数据类型] --> B[整数类型]
    A --> C[浮点类型]
    A --> D[char]
    A --> E[boolean]
    B --> F[byte]
    B --> G[short]
    B --> H[int]
    B --> I[long]
    C --> J[float]
    C --> K[double]
                    
```

+ +

8种基本数据类型（续1）



- Java 8 种基本类型的存储空间和使用场景如下表所示：

类型名称	字节空间	使用场景
byte	1字节（8位）	存储字节数据（较常用）
short	2字节（16位）	兼容性考虑（很少使用）
int	4字节（32位）	存储普通整数（常用）
long	8字节（64位）	存储长整数（常用）
float	4字节（32位）	存储浮点数（不常用）
double	8字节（64位）	存储双精度浮点数（常用）
char	2字节（16位）	存储一个字符（常用）
boolean	1字节（8位）	存储逻辑变量（true、false）（常用）

2.2. int 类型

2.2.1. 【int 类型】int 类型


int类型



- int是最常用的整数类型。一个int类型的变量占用4个字节（32位），最大表示范围为： $-2^{31} \sim 2^{31}-1$ ，即 -2147483648 ~ 2147483647。

2.2.2. 【int 类型】整数直接量是 int 类型

整数直接量是int类型



- 所谓整数直接量(literal)就是直接写出的整数。
- 例如：下面的语句中，100就是直接量。
`int a = 100;`
- 关于整数直接量，需要注意如下要点：
 - 整数的直接量的类型默认为int类型，如果直接写出的整数超过了int的表达范围，将会出现编译错误。
 - 除了通常的十进制书写形式，整数直接量也经常写16进制的形式（以0X或0x开头）或8进制的形式（以0开头）。

代码清单

整数直接量是int类型（续1）

Tarena
达内科技

```

int a = 100000; // 10进制
int b = 0x186a0; // 16进制
int c = 0303240; // 8进制

// 编译错误，因为整数直接量超过了整数的范围
int d = 10000000000;
                    
```

10000000000这个数值写出来就是错误的，因为Java认为所有直接写出的整数都是int类型，而这个数值超过了int的表达范围。

+

2.2.3. 【int 类型】整型数据的除法运算中的取整

代码清单

整型数据的除法运算中的取整

Tarena
达内科技

- 两个整数相除，会舍弃小数的部分（不是四舍五入），结果也是整数。

```

int c = 5/3;
System.out.println(c); // c的值为1

int total = 87;
int error = 23;
int percent = error / total * 100;
System.out.println(percent + "%");
// 结果为0%，23除以87整数部分为0
percent = 100 * error / total;
System.out.println(percent + "%");
// 结果为26%，230除以87整数部分为26
                    
```

+

2.2.4. 【int 类型】运算时要防止溢出的发生

代码清单

运算时要防止溢出的发生

Tarena
达内科技

- 整数运算的溢出：两个整数进行运算时，其结果可能会超过整数的范围而溢出。正数过大而产生的溢出，结果为负数；负数过大而产生的溢出，结果为正数。

```



int a = 2147483647;
int b = -2147483648;
a = a + 1;
b = b - 1;
System.out.println("a=" + a);
System.out.println("b=" + b);
                    
```

输出结果：
a=-2147483648
溢出，结果错误。
b=2147483647
溢出，结果错误。



+

2.3. long 类型



2.3.1. 【long 类型】long 类型

<div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div>	<div style="text-align: right;"></div> <h3>long类型</h3> <ul style="list-style-type: none">- 在表示整数时，如果int类型的范围不够，可以使用long型，一个long型的变量占用8个字节（64位），最大表示范围为：$-2^{63} \sim 2^{63}-1$，即 -9223372036854775808 ~ 9223372036854775807。- 如果要表示long直接量，需要以 L 或 l 结尾。 <pre>long a = 100000000000; // 会有编译错误 long b = 100000000000l;</pre> <div style="text-align: right;"></div>
---	---

2.3.2. 【long 类型】使用 long 类型进行较大整数的运算


<div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div>	<div style="text-align: right;"></div> <h3>使用long类型进行较大整数的运算</h3> <ul style="list-style-type: none">• 对于较大的整数运算（超过int的表达范围），可以使用long型。 <pre>long distance1 = 10000 * 365 * 24 * 60 * 60 * 299792458l; // 必须有一个long型数据参与的运算结果才是long型 System.out.println("distance1=" + distance1); // distance1=547836957965889536 结果正确 long distance2 = 10000 * 365 * 24 * 60 * 60 * 299792458; System.out.println("distance2=" + distance2); // distance2=-1973211136 溢出</pre> <div style="text-align: right;"></div>
---	---

2.3.3. 【long 类型】通过时间毫秒数来存储日期和时间


<div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div>	<div style="text-align: right;"></div> <h3>通过时间毫秒数来存储日期和时间</h3> <ul style="list-style-type: none">• JDK提供 System.currentTimeMillis() 方法，返回1970年1月1日零点到此时此刻所经历的毫秒数，其数据类型为long，该方法经常用于计时操作。 <pre>long time = System.currentTimeMillis(); System.out.println(time); // 输出的结果为：1383835712828</pre> <div style="text-align: right;"></div>
---	---

2.4. double 类型


2.4.1. 【double 类型】使用 double 进行浮点数的运算

<hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/>	<div style="text-align: right;">  </div> <h4>使用double进行浮点数的运算</h4> <ul style="list-style-type: none"> • 浮点数，就是小数，包括：float和double • double类型的精度值是float类型的两倍，这正是其名称（双精度）的来由。 • 大多数场合使用double表示浮点数。 <pre>double pi = 3.14; double r = 8; double s = pi * r * r; System.out.println("s=" + s); // 输出的结果为：s=200.96</pre> <div style="text-align: right;">+</div>
---	--

2.4.2. 【double 类型】浮点数直接量是 double 类型

<hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/>	<div style="text-align: right;">  </div> <h4>浮点数直接量是double类型</h4> <ul style="list-style-type: none"> • 浮点数的直接量有两种写法： <ul style="list-style-type: none"> – 通常写法，如：3.14、314、0.1、.5。 – 科学计数法，如：1.25E2、1.25e2、1.25E-2。其中，1.25E2表示1.25乘以10的2次方。 • 默认的浮点直接量为double型，如果需要表示float类型的直接量，需要加“f”或“F”后缀。例如： <pre>float f1 = 3.14</pre> 会有编译错误，应该写成3.14f <div style="text-align: right;">+</div>
---	---

2.4.3. 【double 类型】double 运算时会出现舍入误差


<hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/>	<div style="text-align: right;">  </div> <h4>double运算时会出现舍入误差</h4> <ul style="list-style-type: none"> • 2进制系统中无法精确的表示1/10，就好像十进制系统中无法精确的表示1/3一样。 • 所以，2进制表示10进制会有一些舍入误差，对于一些要求精确运算的场合会导致代码的缺陷。 <pre>double money = 3.0; double price = 2.9; System.out.println(money - price); // 输出的结果是：0.10000000000000009</pre> <div style="border: 1px solid black; padding: 5px; margin-top: 10px;"> 如果需要精确的运算可以考虑放弃使用double或float而采用BigDecimal 类来实现。关于这一点，将在后续的章节中介绍。 </div> <div style="text-align: right;">+</div>
---	---

2.5. char 类型

2.5.1. 【char 类型】char 类型

知识讲解

char 类型



- 字符类型事实上是一个16位无符号整数，这个值是对应字符的编码，
- Java字符类型采用Unicode字符集编码。Unicode是世界通用的定长字符集，所有的字符都是16位
- 字符直接量可以采用诸如：'中' 的形式，也可以采用16进制的表示形式，例如：'\u4e2d'


```
char c1 = '中'; //c1中存的是"中" 的编码
char c2 = '\u4e2d';
System.out.println(c1);
System.out.println(c2);
```

'4e2d' 为 '中' 所对应的16位Unicode编码的16进制表示形式。

2.5.2. 【char 类型】对 char 型变量赋值

知识讲解

对char型变量赋值




- 在对char型变量赋值时，可以采用如下三种方式：
 - 字符直接量：形如 'A'，变量中实际存储的是该字符的Unicode编码（无符号整数值），一个char型变量只能存储一个字符。
 - 整型直接量：范围在0~65535之间的整数，变量中实际存储的即该整数值，但表示的是该整数值所对应的Unicode字符。
 - Unicode形式：形如 '\u0041'，Unicode字符的16进制形式。

```
char c1 = 65;
char c2 = 'A';
char c3 = '\u0041';
```

2.5.3. 【char 类型】使用转义字符

知识讲解

使用转义字符



- 对于不方便输出的字符采用转义字符表示，例如：

转义字符	含义
'\n'	表示回车符
'\r'	表示换行符
'\\'	表示反斜杠 (\)
'\''	表示单引号 (')
'\"'	表示双引号 (")


```
char c = '\\';
System.out.println(c); //输出的结果为：\
```

36

2.6. boolean 类型

2.6.1. 【boolean 类型】使用 boolean 变量进行关系运算

使用boolean变量进行关系运算




- boolean类型适用于逻辑运算，表示某个条件是否成立。一般用于程序的流程控制。
- boolean类型只允许取值true或false，true表示条件成立而false表示条件不成立。
- boolean型变量经常用于存储关系运算的结果，所谓关系运算就是比较两个变量的大小相等关系。

```
int age = 18;
boolean isChild = age<16; // isChild的值为false
System.out.println(isChild);
boolean running = true;
boolean closed = false;
```

2.7. 类型间的转换

2.7.1. 【类型间的转换】基本类型间转换

基本类型间转换



- 不同的基本类型直接可以相互转换：
 - 自动类型转换（隐式类型转换）：从小类型到大类型可以自动完成。类型的大小关系如下图所示：
- 强制转换：从大类型到小类型需要强制转换符：
(需要转换成的类型) 变量
但这样转换有可能会造成精度损失或者溢出。

2.7.2. 【类型间的转换】强制转换时的精度丧失和溢出

强制转换时的精度丧失和溢出



- 基本类型转换示例，注意强制转换时可能会造成的精度丧失和溢出。


```
int a = 100;
int b = 200;
long c = a + b; // 自动将int转换为long

long l1 = 1024L;
int i = (int) l1; // 需要加强制转换符由于1024在int的范围内没有产生溢出


long l = 1024L * 1024 * 1024 * 4;
int j = (int) l; // 会产生溢出
System.out.println(j); // 结果为：0

double pi = 3.1415926535897932384;
float f = (float) pi; // 会造成精度的损失
System.out.println(f); // 结果为：3.1415927
```

2.7.3. 【类型间的转换】数值运算时的自动转换

<hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/>	<div style="text-align: right;">  </div> <h4 style="text-align: center;">数值运算时的自动转换</h4> <ul style="list-style-type: none"> 多种基本类型参与的表达式运算中，运算结果会自动的向较大的类型进行转换： <pre>// 由于有long型的直接量参与，整个表达式的结果为long long distance = 10000 * 365 * 24 * 60 * 60 * 299792458; // 由于有double型的直接量599.0参与， // 整个表达式的结果为double double change = 800 - 599.0; double percent1 = 80 / 100; // 结果为0.0，右边都是int型数据运算结果也为int类型，结果 // 为0，再赋值给double型，将0转换为0.0 double percent2 = 80.0 / 100; // 结果为0.8，右边表达式有double型直接量参与， // 运算结果为double型</pre> <div style="text-align: right;">+</div>
---	--


2.7.4. 【类型间的转换】byte、char、short 转换为 int

<hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/>	<div style="text-align: right;">  </div> <h4 style="text-align: center;">byte、char、short转换为int</h4> <ul style="list-style-type: none"> byte、char、short 三种类型实际存储的数据都是整数，在实际使用中遵循如下规则： <ul style="list-style-type: none"> int直接量可以直接赋值给byte、char和short，只要不超过其表示范围。 byte、char、short三种类型参与运算时，先一律转换成int类型再进行运算。 <div style="text-align: right;">+</div>
---	--

3. 运算符和表达式 -1

3.1. 算术运算


3.1.1. 【算术运算】使用%运算符

<hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/>	<div style="text-align: right;">  </div> <h4 style="text-align: center;">使用%运算符</h4> <ul style="list-style-type: none"> Java算术运算符除通常的加(+)、减(-)、乘(*)、除(/)之外，还包括取模运算(%) and 自增(++)及自减(--)运算。 取模运算(%)意为取余数，可适用于整数、char类型以及浮点数。 <pre>// 输出255除以8所得的余数。 int n = 225; System.out.println(n % 8); // 结果为1</pre> <div style="text-align: right;">+</div>
---	---

3.1.2. 【算术运算】使用“++”和“--”运算符

知识讲解


使用“++”和“--”运算符



- Java的自增运算符(++)和自减运算符(--)继承自C++，可以使变量的值加1或减1，但其写在变量前和变量后有不同的效果：
 - 如果写在变量前表示在使用这个变量之前加1或减1
 - 如果写在变量之后表示这个变量使用完之后再加1或减1

```
int a = 10, b = 20;
int c1 = a++; // 先将a的值赋给c1，然后a再自加
int c2 = ++b; // 先将b的值自加，然后再赋给c2

System.out.println("a=" + a + ", b=" + b +
    ", c1=" + c1 + ", c2=" + c2);
// 输出的结果为：a=11, b=21, c1=10, c2=21
```




3.2. 关系运算

3.2.1. 【关系运算】使用关系运算符


知识讲解

使用关系运算符



- 关系运算符用于判断数据之间的大小关系。包括大于(>)、小于(<)、大于等于(>=)、小于等于(<=)、等于(==)、不等于(!=)六个运算符。关系运算的结果为boolean类型，如果关系成立为true，否则为false。

```
int max = 10;
int num = 9;
boolean b1 = max > 15;
boolean b2 = num % 2 == 1;
System.out.println(b1); // 结果为false
System.out.println(b2); // 结果为true
```




3.3. 逻辑运算

3.3.1. 【逻辑运算】逻辑运算


知识讲解

逻辑运算




- 逻辑运算建立在关系运算的基础之上，逻辑运算符包括：与(&&)、或(||)和非(!)。
- 参与逻辑运算的变量或表达式都是boolean类型，运算结果也为boolean类型。逻辑运算规则如下表所示：


变量b1	变量b2	b1&b2	b1 b2	!b1
false	false	false	false	true
false	true	false	true	true
true	false	false	true	false
true	true	true	true	false



3.3.2. 【逻辑运算】使用“&&”运算符

<hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/>	<div style="text-align: right;">  </div> <h4>使用“&&”运算符</h4> <ul style="list-style-type: none"> 两个boolean变量参与“&&”运算时，只有当两个变量均为true时，运算结果才为true，否则结果为false； <pre>int score = 80; boolean b = score >= 60 && score < 90; System.out.println(b); // 结果为true</pre> <p>上面的代码中boolean变量b的结果为true，因为score的值同时满足大于等于60和小于90这两个条件，逻辑表达式“score >= 60”和“score < 90”的结果均为true，&&运算的结果即为true。</p> <div style="text-align: right;">+</div>
---	---



3.3.3. 【逻辑运算】使用“||”运算符


<hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/>	<div style="text-align: right;">  </div> <h4>使用“ ”运算符</h4> <ul style="list-style-type: none"> 两个boolean变量参与“ ”运算时，当两个变量有一个为true时，结果即为true，只有当两个变量均为false时结果为false。 <pre>boolean flag = true; int n = 200; boolean b1 = flag (n >= 0 && n < 100); System.out.println(b1); // 结果为true</pre> <p>上面这段代码中，表达式“flag (n >= 0 && n < 100)”的含义是：当flag为true或者n在0到100之间（n大于等于0且小于100）时，结果为true，否则为false。根据flag和n的值，最后的运算结果为true。</p> <div style="text-align: right;">+</div>
---	---

3.3.4. 【逻辑运算】使用“!”运算符

<hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/>	<div style="text-align: right;">  </div> <h4>使用“!”运算符</h4> <ul style="list-style-type: none"> “!”运算相对简单，只会有一个boolean变量参与运算，运算的值与该变量相反，变量为true时结果为false，变量为false时结果为true。 <pre>boolean flag = true; int n = 200; boolean b = !flag (n >= 0 && n < 100); System.out.println(b); // 结果为false</pre> <p>上面这段代码中，表达式“!flag (n >= 0 && n < 100)”的含义是：当flag为false或者n在0到100之间（n大于等于0且小于100）时，结果为true，否则为false。根据flag和n的值，最后的运算结果为false。</p> <div style="text-align: right;">+</div>
---	--

3.3.5. 【逻辑运算】关于“短路逻辑”的问题

<hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/>	<div style="text-align: right;">  </div> <h4 style="text-align: center;">关于“短路逻辑”的问题</h4> <ul style="list-style-type: none"> Java逻辑运算遵循“短路逻辑”的原则： <ul style="list-style-type: none"> 对于“&&”，当第一个操作数为false时，将不会判断第二个操作数，因为此时无论第二个操作数为何，最后的运算结果一定是false； 对于“ ”，当第一个操作数为true时，将不会判断第二个操作数，因为此时无论第二个操作数为何，最后的运算结果一定是true。 <div style="text-align: right;">  </div> <div style="text-align: right;">+</div>
---	---

<hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/>	<div style="text-align: right;">  </div> <h4 style="text-align: center;">关于“短路逻辑”的问题（续1）</h4> <pre> int i = 100, j = 200; boolean b1 = (i > j) && (i++ > 100); System.out.println(b1); // 结果为：false System.out.println(i); // 结果为：100，i++不会被执行 boolean b2 = i > 0 j++ > 200; System.out.println(b2); // 结果为：true System.out.println(j); // 结果为：200，j++不会被执行 </pre> <div style="text-align: right;">+</div>
---	--

经典案例

1. 变量使用常用错误汇总

- **问题**

在我们使用变量的过程中，会遇到一些问题，在此将这些问题进行汇总，在今后使用的过程中，避免出错。即使出现错误也可以很快的找到问题所在。

- **方案**

变量在使用的过程中，常见问题总结为如下几点：

- 1) 使用未经声明的变量。
- 2) 使用不符合 Java 标识符命名规则的变量。
- 3) 使用未经初始化的变量。
- 4) 变量的赋值与变量的类型不匹配

- **步骤**

实现此案例需要按照如下步骤进行。

步骤一：使用未经声明的变量

Java 语言语法规则规定，变量使用之前必须声明，否则会有编译错误。大多数时候我们都记得这个规范，但是还是会出现变量未声明就使用的情况，请看如下代码：

```
public static void main(String[] args) {  
    a = 1; // 编译错误，变量没有声明  
    int score = 0;  
    scord = 100; // 编译错误  
    System.out.println(score);  
}
```

编译上述代码，会发现在代码“a=1;”处和代码“scord=100;”处出现编译错误。出现编译错误的原因是变量 a 和变量 scord 没有被声明。变量的声明包含两点：变量的类型和变量的名称。a 变量没有被声明的原因是没有变量的类型。scord 变量没有被声明的原因也是因为没有变量类型，但是查看代码 scord=100; 的上下行的代码会发现声明了 score 变量，分析这三行代码，scord=100 行出现编译错误的原因是变量没有被声明，没有声明的原因是变量 score 拼写成了 scord。

步骤二：使用不符合 Java 标识符命名规则的变量

Java 中的变量的命名必须符合 Java 标识符的规则：

- 1) 可以以字母、数字、“_”和“\$”符组成；
- 2) 首字符不能以数字开头；

- 3) 中文可以作为变量名,但不提倡使用;
- 4) Java 大小写敏感,命名变量时需要注意;
- 5) 不能使用 Java 保留字(一些 Java 语言规定好的,有特殊含义的字符),如: int、if、for、break 等。

下面的代码体现了 Java 变量的命名规则:

```
int 123go = 100; // 编译错误,不能以数字开头。
int 成绩 = 60; // 编译没错,但不建议使用。
int break= 200; // 编译错误,break 是 Java 保留字。
int score = 80;
System.out.println(Score);
// 编译错误,Java 大小写敏感,Score 变量没有声明。
```

上述代码中,分别有如下错误:

- 1) 变量“123go”不符合 Java 的命名规范,原因是变量名不能以数字开头;
- 2) 变量“成绩”编译时是正确的,但是这种使用汉字进行命名的方式不建议使用;
- 3) 变量“break”处会出现编译错误,原因是 break 是 Java 的保留字,不能作为变量名;

4) 在输出变量“Score”处会出现编译错误,原因是变量名是大小写敏感的。int score=80;处声明的变量和下一行代码中输出的 Score 是两个变量,所以变量 Score 没有声明。Java 语言语法规则规定,变量使用之前必须声明,否则会有编译错误。

另外,Java 变量名的定义应“见名知意”;同时,Java 编程规范要求:变量的命名需采用“驼峰命名法”,即如果变量的名字由多个单词组成,除第一个单词外,其他单词的首字母大写,其余的字母小写,例如:salary、empNo、studentName 等。

步骤三:使用未经初始化的变量

Java 语言规定变量在使用之前必须初始化,即必须给该变量赋予特定的值。请看下列代码:

```
public static void main(String[] args) {
    int a, b = 10;
    int c = a + b; // 编译错误,变量 a 没有初始化
    System.out.println(c);
}
```

在上述代码中,代码行 int c = a + b;处会出现编译错误,因为此行代码使用到了变量 a,但是该变量却没有被初始化。

另外,有些语句结构(如 if、for 等)需要条件满足时才会执行;Java 编译器不认为在这些语句块中的赋值语句可以实现初始化操作。查看如下代码:

```
int sum;
int a = 20;
int b = 10;
if(a>0) {
```

```
        sum = 0; // 当 a 大于 0 的时候, 该语句才会执行。
        sum = a + b;
    }
    System.out.println(sum); // 编译错误, 编译器认为 sum 没有初始化。
```

上述代码中, 语句 `System.out.println(sum);` 处会出现编译错误, Java 编译器不认为放在 `if` 语句块中的 `sum=0`; 可以实现初始化操作。

步骤四：变量的赋值与变量的类型不匹配

变量在声明时指定了类型, Java 编译器会检测对该变量的操作是否与其类型匹配, 如果对变量的赋值或者操作与其类型不匹配, 会产生编译错误。

```
public static void main(String[] args) {
    int salary;
    salary = 15000.50; // 编译错误, 整型变量不可以赋予浮点值 ( 小数 )
}
```

上述代码中, 变量 `salary` 声明时的类型为 `int`, 后续赋值为 `15000.50`, 而 `15000.50` 是浮点类型, 因此导致编译错误。整数类型变量不可以赋予浮点类型的值。

- **完整代码**

本案例是总结性的知识, 没有完整的代码。

2. 整数类型 (int、long) 使用常见问题汇总

- **问题**

在我们使用整数类型的过程中, 会遇到一些问题, 在此将这些问题进行汇总, 在今后使用的过程中, 避免出错。即使出现错误也可以很快的找到问题所在。

- **方案**

整数类型在使用的过程中, 常见的问题有以下几点:

- 1) 整数直接量超出了整数的范围。
- 2) 关于整数的除法: 两个整数相除, 会舍弃小数的部分, 结果也是整数。
- 3) 整数运算的溢出: 两个整数进行运算时, 其结果可能会超过整数的范围而溢出。
- 4) 表示 `long` 直接量, 需要以 `L` 或 `l` 结尾。

- **步骤**

实现此案例需要按照如下步骤进行。

步骤一：整数直接量超出了整数的范围

`int` 是最常用的整数类型。一个 `int` 类型的变量占用 4 个字节 (32 位), 最大表示范围

为： $-2^{31} \sim 2^{31}-1$ ，即 $-2147483648 \sim 2147483647$ 。

整数直接量(literal)，即直接写出的整数。整数的直接量的类型默认为 int 类型，如果直接写出的整数超过了 int 的表达范围，将会出现编译错误。请看如下代码：

```
int d = 100000000000;
```

以上代码中，100000000000 不属于 int 类型的直接量，因为 Java 认为所有直接写出的整数都是 int 类型，而这个数值超过了 int 的表达范围。

步骤二：关于整数的除法

在 Java 中，两个整数相除，会舍弃小数的部分，结果也是整数。请看如下代码：

```
int c = 5/3;
System.out.println(c); // c 的值为 1
```

在上述代码中，运行后，c 的值为 1。说明两个整数相除，舍弃了小数部分，只保留了整数部分。

步骤三：整数运算的溢出

两个整数进行运算时，其结果可能会超过整数的范围而溢出，请看如下代码：

```
int a = 2147483647;
int b = -2147483648;
a = a + 1;
b = b - 1;
System.out.println("a=" + a);
System.out.println("b=" + b);
```

上述代码运行后的输出结果为：

```
a=-2147483648
b=2147483647
```

变量 a 最初的值为 2147483647，是 int 类型的最大值，加 1 以后出现了溢出现象，a 的值变成了 int 类型的最小值。而 b 变量最初赋的值为 -2147483648，是 int 类型的最小值，减 1 以后出现了溢出现象，b 的值变成了 int 类型的最大值。这显然不符合加法和减法的规则，所以，在今后使用的时候要注意类似的问题。

步骤四：表示 long 直接量，需要以 L 或 l 结尾

在表示整数时，如果 int 类型的范围不够，可以使用 long 型，一个 long 型的变量占用 8 个字节 (64 位)，最大表示范围为： $-2^{63} \sim 2^{63}-1$ ，即 $-9223372036854775808 \sim 9223372036854775807$ 。当一个直接量超过 int 类型的最大值时，那要用 long 类型来表示，如果要表示 long 直接量，需要以 L 或 l 结尾。请看下列代码：

```
long a = 1000000000000; // 会有编译错误
long b = 1000000000000L;
```

上述代码中，10000000000 超过了 int 类型的最大值，把它直接赋值给 long 类型会出现编译错误。需要像变量 b 那样在 10000000000 后边加 L。

- **完整代码**

本案例是总结性的知识，没有完整的代码。

3. 浮点类型 (float、double) 使用常见问题汇总

- **问题**

在我们使用浮点类型的过程中，会遇到一些问题，在此将这些问题进行汇总，在今后使用的过程中，避免出错。即使出现错误也可以很快的找到问题所在。

- **方案**

浮点类型在使用的过程中，常见的问题有以下几点：

- 1) 浮点数的直接量为 double 类型。
- 2) 浮点数存在舍入误差问题。

- **步骤**

实现此案例需要按照如下步骤进行。

步骤一：浮点数的直接量为 double 类型

浮点数，就是小数，包括：float 和 double。默认的浮点直接量为 double 型，如果需要表示 float 类型的直接量，需要加 “f” 或 “F” 后缀。请看如下代码：

```
float f1 = 3.14 ;
```

以上代码，会出现编译错误。3.14 是 double 类型的直接量，如果表示 float 类型的直接量应该写成 3.14f。

步骤二：浮点数存在舍入误差问题

由于浮点数内部用二进制的方式来表示十进制，会存在舍入误差。二进制系统中无法精确的表示 1/10，就好像十进制系统中无法精确的表示 1/3 一样。对于一些要求精确运算的场合会导致代码的缺陷。请看如下代码：

```
double money = 3.0;  
double price = 2.9;  
System.out.println(money - price);
```

上述代码的输出结果为：

0.100000000000000009

查看上述结果，并没有如我们想象的为 0.1。如果需要精确的运算可以考虑放弃使用 double 或 float 而采用 BigDecimal 类来实现。关于这一点，将在后续的章节中介绍。

完整代码

本案例是总结性的知识，没有完整的代码。

4. 对 char 类型变量的各种赋值方式汇总

• 问题

在我们使用 char 类型的过程中，会遇到一些问题，在此将这些问题进行汇总，使今后使用的过程中，不出错。即使出现错误也可以很快的找到问题所在。

• 方案

char 类型在使用的过程中，常见的问题有以下几点：

- 1) 字符类型存储中文。
- 2) char 类型的值可以作为整数类型直接使用。

• 步骤

实现此案例需要按照如下步骤进行。

步骤一：字符类型存储中文

char 类型是否可以存储中文？答案是肯定的。字符类型事实上是一个 16 位无符号整数，这个值是对应字符的编码，Java 字符类型采用 Unicode 字符集编码。Unicode 是世界通用的定长字符集，所有的字符都是 16 位字符直接量。对于中文，可以采用诸如：‘中’的形式，也可以采用其对应的 16 进制的表示形式，例如：‘\u4e2d’。

步骤二：整数类型和 char 类型的关系

char 类型的值可以直接作为整数类型的值来使用，字符类型事实上是一个 16 位无符号整数，即全部是正数，表示范围是 0~65535。请看如下代码：

```
char zhong='疯';
int zhongValue=zhong;
System.out.println(zhongValue);
```

上述代码的输出结果为：

30127

上述输出结果为 0 ~ 65535 范围的。

另外，如果把 0~65535 范围内的一个 int 整数赋给 char 类型变量，系统会自动把这个 int 类型整数当成 char 类型来处理。请看如下代码：

```
char c=97;  
System.out.println(c);
```

上述代码的输出结果为 a。这说明系统自动把整数类型 97 当成 char 类型来处理，处理的结果为 a，即，97 为字母 a 的 unicode 码。

- **完整代码**

本案例是总结性的知识，没有完整的代码。

5. 类型转换常见问题汇总

- **问题**

在我们数据类型转换的过程中，会遇到一些问题，在此将这些问题进行汇总，使今后使用的过程中，不出错。即使出现错误也可以很快的找到问题所在。

- **方案**

数据类型转换在使用的过程中，常见的问题有以下几点：

- 1) 强制转换时的精度丧失和溢出。
- 2) 数值运算时的自动转换。
- 3) byte、char、short 转换为 int 的问题。

- **步骤**

实现此案例需要按照如下步骤进行。

步骤一：强制转换时的精度丧失和溢出

基本类型之间的相互转换，需要注意的是，强制转换时可能会造成精度的丧失和溢出，请看如下代码：

```
long l = 1024L * 1024 * 1024 * 4;  
int j = (int) l; // 会产生溢出  
System.out.println(j); // 结果为：0
```

上述代码输出的结果为 0，是因为在将 long 类型变量 l 转换为 int 类型变量 j 的时候产生了溢出。

另外，请看如下精度丧失的例子：

```
double pi = 3.141592653589793;
float f = (float) pi; // 会造出精度的损失
System.out.println(f); // 结果为：3.1415927
```

上述代码的输出结果为 3.1415927, 值保留了 7 位小数, 这是因为将 double 类型的变量 pi 转换为 float 类型的变量 f 时造成了精度的损失。

步骤二：数值运算时的自动转换

多种基本类型参与的表达式运算中, 运算结果会自动的向较大的类型进行转化。请看如下示例：

```
long distance = 10000 * 365 * 24 * 60 * 60 * 299792458L;
```

上述代码中, 有 int 类型数据和 long 类型数据, 由于有 long 型的直接量 299792458L 参与, 整个表达式的结果为 long。

```
double change = 800 - 599.0;
```

上述代码中, 由于有 double 型的直接量 599.0 参与, 整个表达式的结果为 double。

```
double percent1 = 80 / 100;
```

上述代码中, 结果为 0.0。右边都是 int 型数据, 语法运算后的结果也为 int 类型, 结果为 0, 再赋值给 double 型, 将 0 转化为 0.0。请对比下面的代码：

```
double percent2 = 80.0 / 100;
```

上述代码中, 结果为 0.8, 右边表达式有 double 型直接量参与, 运算结果为 double 型。

步骤三：byte、char、short 转换为 int 的问题

byte、char、short 三种类型实际存储的数据都是整数, 在实际使用中遵循如下规则：

- 1) int 直接量可以直接赋值给 byte、char 和 short, 只要不超过其表示范围。
- 2) byte、char、short 三种类型参与运算时, 先一律转换成 int 类型再进行运算。

请看如下示例代码：

```
byte b1=28;
byte b2=20;
byte b3=b1+b2;
```

上述代码在第三行会出现编译错误, 原因是 b1+b2 的结果为 int 类型。改变上述代码如下：

```
byte b1=28;
byte b2=20;
int b3=b1+b2;
```


查看上述代码，会发现不会再出现编译错误。char 类型、short 类型和 byte 类型是相似的。

- **完整代码**

本案例是总结性的知识，没有完整的代码。

6. 年龄判断程序

- **问题**

本案例需要使用交互的方式判断年龄的范围：用户从控制台输入一个年龄，由程序判断该年龄是否在 18~50 之间。程序交互过程如图 - 1 所示：

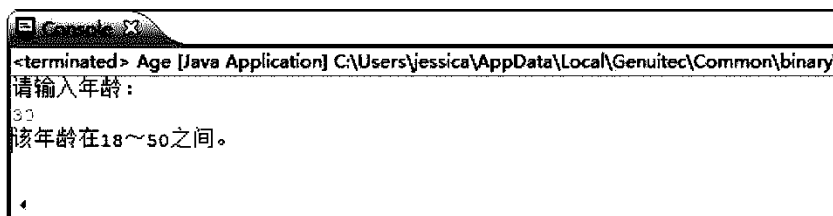


图- 1

- **步骤**

实现此案例需要按照如下步骤进行。

步骤一：定义类及 main 方法

首先定义一个名为 Age 的类，并在类中添加 Java 应用程序的主方法 main，代码如下所示：

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

步骤二：读取控制台输入

在 main 方法中，实例化 Scanner 类，并调用 Scanner 类的 nextInt() 方法接收用户从控制台输入的年龄，使用完毕后将 scanner 对象关闭，以释放资源。代码如下所示：

```
import java.util.Scanner;  
  
public class Age {  
    public static void main(String[] args) {  
  
        Scanner scanner = new Scanner(System.in);  
        System.out.println("请输入年龄：");  

```

```
int age = scanner.nextInt();
scanner.close();

}
}
```

在此需要注意导入 Scanner 类所在的包。

步骤三：判断年龄所在的范围

接收到年龄后，判断年龄是否在 18~50 之间。如果输出结果为 true，则说明年龄在 18~50 之间，否则，年龄不在 18~50 之间，代码如下所示：

```
import java.util.Scanner;

public class Age {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("请输入年龄：");
        int age = scanner.nextInt();

        System.out.println(age >= 18 && age <= 50);

    }
}
```

在上述代码中，使用了“&&”逻辑运算符来连接两个条件。年龄在 18~50 之间，即，年龄大于等于 18 且年龄小于等于 50，因此需要使用“&&”运算符。

- **完整代码**

本案例的完整代码如下所示：

```
import java.util.Scanner;

public class Age {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("请输入年龄：");
        int age = scanner.nextInt();
        scanner.close();
        System.out.println(age >= 18 && age <= 50);

    }
}
```

课后作业

1. 指出下面程序中的编译错误，并更正

```
int lenght=10;  
System.out.println(length);
```

2. 指出下面程序中的编译错误，并更正

```
int &size=20;  
System.out.println(&size);
```

3. 指出下面程序中的编译错误，并更正

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

4. 运行下面程序，指出变量的输出结果

```
int count=30;  
count=60;  
System.out.println(count);
```

5. 指出下面程序中的编译错误，并更正

```
int balance;  
balance =218.50;
```

6. 指出下面程序的编译或运行结果，并解释原因

```
int i=128;  
i = 10000000008;  
System.out.println(i);
```

7. 通过代码计算一段程序运行的时间

8. 指出下面程序的运行输出结果

```
double width = 6.0;  
double length = 4.9;  
System.out.println(width - length);
```

9. 指出下面程序的输出结果

```
char ascii=98;
System.out.println(ascii);
```

10. 指出下面程序中的编译错误，并更正

```
byte b1=10;
byte b2=20;
byte b3=b1+b2;
```

11. 指出下面程序的运行输出结果

```
int a = 1, b = 10;
int c1 = a++;
int c2 = ++b;
System.out.println("a=" + a + ", b=" + b + ", c1=" + c1 + ", c2=" + c2);
```

12. 指出下面程序的运行输出结果

```
int i = 100, j = 200;
boolean b1 = (i > j) && (i++ > 100);
System.out.println(b1);
System.out.println(i);
```

13. 完成收银柜台收款程序 V1.0

编写一个收银柜台收款程序。根据商品单价、购买数量以及收款金额计算并输出应收金额和找零，控制台交互情况如图-1 所示。

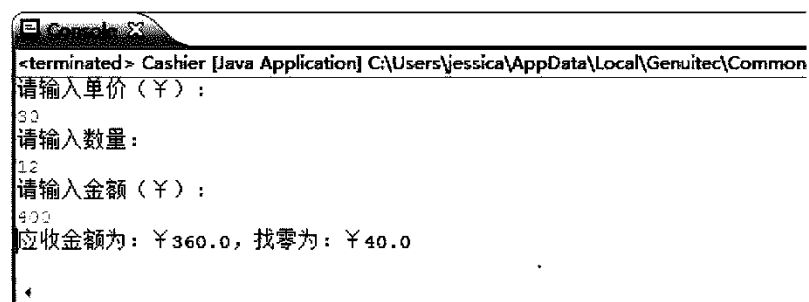


图- 1

Java 语言基础

Unit03

知识体系.....Page 56

运算符和表达式 -2	赋值运算	使用 “=” 进行赋值运算
		使用扩展赋值表达式
	字符串连接运算	使用 “+” 进行字符串连接
	条件（三目）运算	使用条件（三目）运算符
		条件（三目）运算符的嵌套
分支结构	什么是分支结构	什么是分支结构
	if 语句	if 语句的执行逻辑
		if 语句流程图
		if 语句用于处理分支逻辑
		if 语句不要省略 “{}”
	if-else 语句	if-else 语句的执行逻辑
		if-else 语句流程图
		if-else 语句处理分支逻辑
	else if 语句	if-else 语句的嵌套
		else if 语句执行逻辑
	switch-case 语句	switch-case 语句执行逻辑
		switch-case 和 break 联合使用
		switch-case 语句用于分支
		switch-case 的优势

经典案例.....Page 64

闰年判断程序	使用条件（三目）运算符
	条件（三目）运算符的嵌套
完成收银柜台收款程序 V2.0	if 语句的执行逻辑
	if 语句流程图
	if 语句用于处理分支逻辑
	if 语句块不要省略 “{}”
完成收银柜台收款程序 V3.0	if-else 语句的执行逻辑
	if-else 语句流程图

	if-else 语句用于处理分支逻辑
完成成绩等级输出程序	if-else 语句的嵌套
	else if 语句执行逻辑
完成命令解析程序	switch-case 语句执行逻辑
	switch-case 和 break 联合使用
	switch-case 语句用于分支
	switch-case 的优势

课后作业.....Page 78

1. 运算符和表达式 -2

1.1. 赋值运算

1.1.1. 【赋值运算】使用“=”进行赋值运算

使用“=”进行赋值运算

- “=”称为赋值运算符，用于对变量赋值。关于赋值运算符，除了将右边的表达式计算出来赋给左边以外还具备如下特点：**赋值表达式本身也有值，其本身之值即为所赋之值。**

```
int num = 18, index;

System.out.println(index = num % 5); // 结果为：3
System.out.println(index); // 结果为：3

int a, b, c;
a = b = c = 100; // c=100 整个表达式的值为100，将
// 其赋值给b，同样b= ( c=100 ) 整个表达式的值也为
// 100，然后又将这个值赋给了a
```

1.1.2. 【赋值运算】使用扩展赋值表达式

使用扩展赋值表达式

- 在赋值运算符“=”前加上其它运算符，即为扩展赋值运算符。

扩展赋值运算符			
运算符	表达式	计算	结果 (假设 X=10)
+=	X += 5	X = X + 5	15
-=	X -= 5	X = X - 5	5
*=	X *= 5	X = X * 5	50
/=	X /= 5	X = X / 5	2
%=	X %= 5	X = X % 5	0

1.2. 字符串连接运算

1.2.1. 【字符串连接运算】使用“+”进行字符串连接

使用“+”进行字符串连接

- “+”可以实现字符串的连接，同时可以实现字符串与其他数据类型的“相连”

```
int a = 100;
String msg = "a=" + a;
System.out.println(msg); // a=100

msg = "" + 100 + 200;
System.out.println(msg); // 结果为：100200

msg = 100 + 200 + "";
System.out.println(msg); // 结果为：300
```

1.3. 条件（三目）运算

1.3.1. 【条件（三目）运算】使用条件（三目）运算符

<div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div>	<p>使用条件（三目）运算符</p> <ul style="list-style-type: none"> 条件运算符又称“三目”运算符，其结构为： boolean表达式？表达式1：表达式2 条件运算符的规则如下： <ul style="list-style-type: none"> 先计算boolean表达式； 如果boolean表达式的值为true，整个表达式的值为表达式1的值； 如果boolean表达式的值为false，整个表达式的值为表达式2的值。 <pre>int a = 100, b = 200; int flag = a > b ? 1 : -1; // flag的值为-1</pre> <p style="text-align: right;">+</p>
---	---

1.3.2. 【条件（三目）运算】条件（三目）运算符的嵌套

<div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div>	<p>条件（三目）运算符的嵌套</p> <ul style="list-style-type: none"> 条件运算符可以嵌套使用，所谓嵌套是指在条件（三目）表达式：“boolean表达式？表达式1：表达式2”中的表达式1或表达式2也是条件（三目）表达式。 <pre>int a = -3; String r = a > 0 ? "正数" : (a == 0 ? "0" : "负数"); System.out.println(r); //结果为负数</pre> <p>上述代码将输出“负数”。这是因为a的值小于0，即boolean表达式的值为false，则取问号后第二个表达式的值作为表达式的结果。而问号后的第二个表达式也是一个三目运算符所构成的表达式。因为a==0表达式的值为false，则取“负数”为表达式的结果。</p> <p style="text-align: right;">+</p>
---	--

2. 分支结构

2.1. 什么是分支结构

2.1.1. 【什么是分支结构】什么是分支结构

<div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div>	<p>什么是分支结构</p> <ul style="list-style-type: none"> 任何复杂的程序逻辑都可以通过“顺序”，“分支”，“循环”三种基本的程序结构实现。 <div style="text-align: center;"> </div> <p style="text-align: right;">+</p>
---	---

什么是分支结构（续1）

- 程序可以在运行过程中，根据不同的条件运行不同的语句。

当条件满足时运行某些语句；
当条件不满足时则不运行这些语句——if结构。

当条件满足时运行某些语句；
当条件不满足时运行另外一些语句——if-else结构。

2.2. if 语句

2.2.1. 【if 语句】if 语句的执行逻辑

if语句的执行逻辑

```

语句0;
if ( 逻辑表达式 ) {
    语句1;
    语句2;
}
语句3;
        
```

1. 执行语句0；
2. 判断逻辑表达式的值：
 - 若值为true，则执行if语句块中的语句；
 - 若值为false，则不执行if语句块中的语句；
3. 执行语句3。

2.2.2. 【if 语句】if 语句流程图

if语句流程图

- 当条件满足时，执行语句块，然后执行if语句下面的语句；
- 否则跳过语句块，直接执行if语句下面的语句。

2.2.3. 【if 语句】if 语句用于处理分支逻辑

if语句用于处理分支逻辑

```

graph TD
    A[输入商品单价] --> B[输入购买数量]
    B --> C[输入收款金额]
    C --> D[计算商品总价]
    D --> E{商品总价大于等于500}
    E -- F --> G[计算找零]
    E -- T --> H[计算折扣后的应收金额]
    H --> G
    G --> I[输出应收金额和找零]
    
```

if语句用于处理分支逻辑（续1）

- 如果商品总价大于等于500，则打8折。

```

.....
double totalPrice = ..... ;
if (totalPrice >= 500) {
    totalPrice = totalPrice * 0.8;
}
.....
    
```

2.2.4. 【if 语句】if 语句不要省略“{}”

if语句不要省略“{}”

```

int num = 5;
if(num<2)
    System.out.println(num);
    
```

当if语句块中只有一条语句时，“{}”可以省略。但当将来代码发生变更时很容易产生错误，因此，即便if块只有一条语句，也不要省略“{}”

```

int num = 5;
if(num<2)
    System.out.print(num);
    System.out.println( "小于2" );
    
```

2.3. if-else 语句

2.3.1. 【if-else 语句】if-else 语句的执行逻辑

if-else语句的执行逻辑

1. 执行语句0；
2. 判断if逻辑表达式的值：
 - 若值为true，则执行语句块1；
 - 若值为false，则执行语句块2；
3. 执行语句3；

```

语句0；
if (逻辑表达式){
    语句块1；
} else {
    语句块2；
}
语句3；
            
```

2.3.2. 【if-else 语句】if-else 语句流程图

if-else语句流程图

```

graph TD
    Start(( )) --> Condition{条件}
    Condition -- T --> Statement1[语句1]
    Condition -- F --> Statement2[语句2]
    Statement1 --> Exit(( ))
    Statement2 --> Exit
    
```

当条件满足时，执行语句块1，然后执行if-else语句下面的语句；否则执行语句块2，再执行if-else语句下面的语句。

2.3.3. 【if-else 语句】if-else 语句处理分支逻辑

if-else语句处理分支逻辑

```

.....
if( money >= totalPrice ) {
    double change = money - totalPrice;
    System.out.println("应收金额为:" + totalPrice + "，找零为:" + change);
} else {
    System.out.println( "Error! 收款金额小于应收金额" );
}
.....
            
```

如果收款金额大于等于应收金额，则找零后输出；
如果收款金额小于应收金额，则输出错误提示信息。

2.4. else if 语句

2.4.1. 【else if 语句】if-else 语句的嵌套

知识讲解

if-else语句的嵌套

```
graph TD; A[制作沙拉] --> B{有黄瓜吗?}; B -- 没有 --> C[制作萝卜沙拉]; B -- 有 --> D{有胡萝卜吗?}; D -- 没有 --> E[不能上菜]; D -- 有 --> F[制作胡萝卜沙拉];
```

if-else 语句的嵌套 (续1)

知识讲解

```
graph TD; A[输入分数] --> B{大于等于90}; B -- T --> C[输出 "A"]; B -- F --> D{大于等于80}; D -- T --> E[输出 "B"]; D -- F --> F{大于等于60}; F -- T --> G[输出 "C"]; F -- F --> H[输出 "D"];
```

The flowchart illustrates a nested if-else structure for determining a grade based on a score. It starts with an input box '输入分数'. The first decision diamond is '大于等于90'. If true (T), it outputs 'A'. If false (F), it proceeds to the second decision diamond '大于等于80'. If true (T), it outputs 'B'. If false (F), it proceeds to the third decision diamond '大于等于60'. If true (T), it outputs 'C'. If false (F), it outputs 'D'.

2.4.2. 【else if 语句】else if 语句执行逻辑

知识讲解

else if语句执行逻辑

- 事实上，else if结构就是if else嵌套的简便写法：

```

... ..
if ( score >= 90 ) {
    // 输出A
} else {
    if (score >= 80) {
        // 输出B
    } else {
        ... ..
    }
}
            
```

```

... ..
if ( score >= 90 ) {
    // 输出A
} else if (score >= 80) {
    // 输出B
} else {
    ... ..
}
            
```

else if语句执行逻辑 (续1)

```

... ..
if(score>=90){
    System.out.println("A"); 分数大于等于90
} else if (score>=80){
    System.out.println("B"); 分数小于90且大于等于80
} else if(score>=60){
    System.out.println("C"); 分数小于80且大于等于60
} else {
    System.out.println("D"); 分数小于60
}
        
```

知识讲解

+

2.5. switch-case 语句

2.5.1. 【switch-case 语句】switch-case 语句执行逻辑

switch-case语句执行逻辑

switch case语句是一种特殊的分支结构，可以根据一个整数表达式的不同取值，从不同的程序入口开始执行。

```

switch (整型表达式) {
    case 整型常量值1:
        语句1;
        语句2;
    case 整型常量值2:
        语句3;
    ... ..
    default:
        语句n;
}
        
```

计算整型表达式的值

知识讲解

+

2.5.2. 【switch-case 语句】switch-case 和 break 联合使用

switch-case和break联合使用

通常case1、case2、...、caseN 对应完全不同的操作，可以和break语句配合使用，执行完相应语句后即退出switch块，不继续执行下面的语句。

```

switch (整型表达式) {
    case 整型常量值1:
        语句1;
        语句2;
        break;
    case 整型常量值2:
        语句3;
        break;
    default:
        语句n;
}
        
```

知识讲解

+

2.5.3. 【switch-case 语句】switch-case 语句用于分支

<hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/>	<p>switch-case语句用于分支</p> <pre>int num = 2; switch(num){ case 1: System.out.println("呼叫教学部"); break; case 2: System.out.println("呼叫人事部"); break; default: System.out.println("人工服务"); }</pre> <p>知识讲解</p> <p>+</p>
---	--

2.5.4. 【switch-case 语句】switch-case 的优势

<hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/>	<p>switch-case的优势</p> <ul style="list-style-type: none"> • switch-case常常和break语句结合使用实现分支的功能。 • switch-case在实现分支功能时和if-else的主要区别在于switch-case结构的效率要高、结构更清晰。 • 从JDK 7.0开始，switch-case支持字符串表达式。 <p>知识讲解</p> <p>+</p>
---	---

经典案例

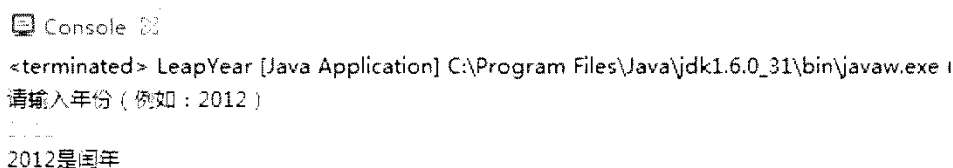
1. 闰年判断程序

- 问题

闰年(Leap Year)是为了弥补因人为历法规定造成的年度天数与地球实际公转周期的时间差而设立的。

地球绕太阳运行周期为 365 天 5 小时 48 分 46 秒 (合 365.24219 天) 即一回归年。公历的平年 (非闰年) 只有 365 日, 比回归年短约 0.2422 日, 所余下的时间约为四年累计一天, 故每四年则于 2 月加 1 天, 使当年的历年长度为 366 日, 这一年就为闰年。但是, 如果按照每四年一个闰年计算, 平均每年就要多算出 0.0078 天, 这样经过四百年就会多算出大约 3 天来, 因此, 每四百年中要减少三个闰年。所以规定, 公历年份是 100 的倍数的, 必须同时也是 400 的倍数, 才是闰年; 不是 400 的倍数的, 虽然是 100 的倍数, 也是平年。这就是通常所说的: 四年一闰, 百年不闰, 四百年再闰。 例如, 2000 年是闰年, 1900 年则是平年。

本案例需要使用交互的方式判断某年是否为闰年: 用户从控制台输入需要判断的年份值, 由程序判断该年是否为闰年, 并将判断结果输出到控制台。程序交互过程如图 - 1 所示:



```
Console
<terminated> LeapYear [Java Application] C:\Program Files\Java\jdk1.6.0_31\bin\javaw.exe
请输入年份 ( 例如: 2012 )
2012
2012是闰年
```

图 - 1

- 方案

首先, 此案例需要从控制台接收用户录入的年份值, 使用 Scanner 类的相应方法即可接收控制台的录入;

其次, 根据闰年的规则可以总结出, 如果年份可以被 400 整除, 则必然是闰年; 另外, 如果年份可以被 4 整除, 但是不能被 100 整除, 则也是闰年; 其他年份则是平年 (非闰年)。因此, 需要使用取余运算符 (%) 判断整除, 并需要使用逻辑运算符来构建判断表达式, 以进行判断。

- 步骤

实现此案例需要按照如下步骤进行。

步骤一: 定义类及 main 方法

首先定义一个名为 `LeapYear` 的类，并在类中添加 `main` 方法。代码如下所示：

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

步骤二：读取控制台的输入

在 `main` 方法中，实例化 `Scanner` 类，并调用 `Scanner` 类的 `nextInt()` 方法接收用户从控制台输入的年份数值，使用完毕后将 `scanner` 对象关闭。代码如下所示：

```
import java.util.Scanner;

public class LeapYear {
    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);
        System.out.println("请输入年份 (例如：2012)");
        int year = scanner.nextInt();
        scanner.close();

    }
}
```

步骤三：闰年判断

某年份只需要满足下列两个条件之一，即可认定是闰年：

- 年份可以被 4 整除且不能被 100 整除；
- 年份可以被 400 整除。

因此，需要使用 `%` 运算符、`==` 运算符、`!=` 运算符、`&&` 和 `||` 运算符，来综合判断某年份是否为闰年。代码如下所示：

```
import java.util.Scanner;
public class LeapYear {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("请输入年份 (例如：2012)");
        int year = scanner.nextInt();
        scanner.close();

        boolean isLeapYear = (year % 4 == 0 && year % 100 != 0)
            || year % 400 == 0;

    }
}
```

步骤四：输出结果

使用三目运算符，构建表示判断结果的 `String` 类型信息，并输出到控制台。代码如下所示：


```
import java.util.Scanner;
public class LeapYear {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("请输入年份 (例如: 2012)");
        int year = scanner.nextInt();
        scanner.close();
        boolean isLeapYear = (year % 4 == 0 && year % 100 != 0)
            || year % 400 == 0;

        String msg = isLeapYear ? year + "是闰年" : year + "不是闰年";
        System.out.println(msg);
    }
}
```

- **完整代码**

本案例的完整代码如下所示：

```
import java.util.Scanner;
public class LeapYear {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("请输入年份 (例如: 2012)");
        int year = scanner.nextInt();
        scanner.close();
        boolean isLeapYear = (year % 4 == 0 && year % 100 != 0)
            || year % 400 == 0;
        String msg = isLeapYear ? year + "是闰年" : year + "不是闰年";
        System.out.println(msg);
    }
}
```

2. 完成收银柜台收款程序 V2.0

- **问题**

编写一个收银柜台收款程序。根据商品单价、购买数量以及收款金额计算并输出应收金额和找零；当总价大于或等于 500 时，享受 8 折优惠。控制台交互情况如图-2 所示。

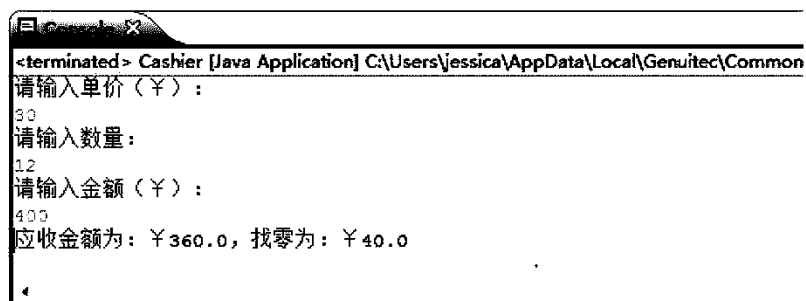


图- 2

- **方案**

本案例的实现方案如图-3 所示。图中 T 表示 true , F 表示 false。

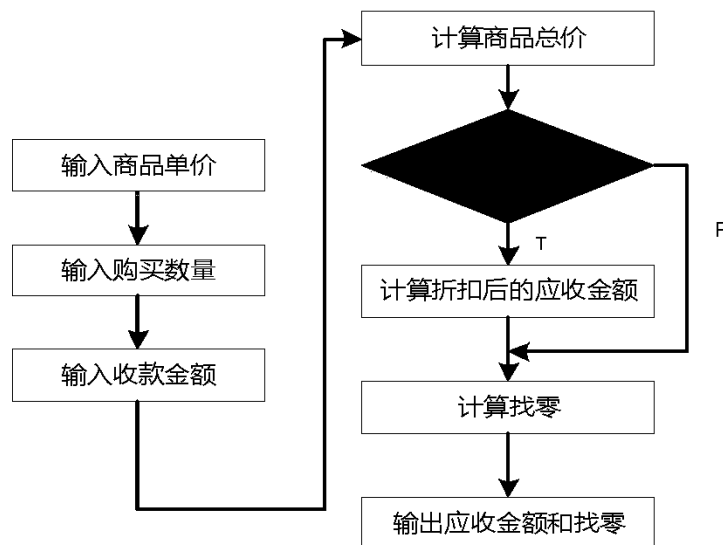


图- 3

• 步骤

实现此案例需要按照如下步骤进行。

步骤一：定义类及 main 方法

首先定义一个名为 Cashier 的类，并在类中添加 Java 应用程序的主方法 main，代码如下所示：

```
public class Cashier {
    public static void main(String[] args) {

    }
}
```

步骤二：读取控制台输入

在 main 方法中，实例化 Scanner 类，并调用 Scanner 类的 nextDouble() 方法接收用户从控制台输入的商品单价、购买数量、收款金额，使用完毕后将 scanner 对象关闭，以释放资源。代码如下所示：

```
import java.util.Scanner;

public class Cashier {
    public static void main(String[] args) {

        // 输入数据
        Scanner console = new Scanner(System.in);
        System.out.println("请输入单价 ( ¥ ): ");
        double unitPrice = console.nextDouble();
        System.out.println("请输入数量: ");
```

```
double amount = console.nextDouble();
System.out.println("请输入金额 ( ¥ ): ");
double money = console.nextDouble();
console.close();

}
}
```

在此需要注意导入 Scanner 类所在的包。

步骤三：计算所购商品总价并判断是否打折

首先，计算商品的总价；然后，使用 if 语句判断商品总价是否大于等于 500，如果大于等于 500，商品总价打八折；最后，计算打八折后的应收金额，代码如下所示：

```
import java.util.Scanner;

public class Cashier {
    public static void main(String[] args) {
        // 输入数据
        Scanner console = new Scanner(System.in);
        System.out.println("请输入单价 ( ¥ ): ");
        double unitPrice = console.nextDouble();
        System.out.println("请输入数量: ");
        double amount = console.nextDouble();
        System.out.println("请输入金额 ( ¥ ): ");
        double money = console.nextDouble();
        console.close();

        // 计算商品总价
        double totalPrice = 0.0;
        totalPrice = unitPrice * amount;
        if (totalPrice >= 500) {
            totalPrice = totalPrice * 0.8;
        }

    }
}
```

步骤四：计算找零并输出结果

首先，将收款金额减去应收金额，即为找零；然后，输出应收金额和找零金额，代码如下所示：

```
import java.util.Scanner;

public class Cashier {
    public static void main(String[] args) {
        // 输入数据
        Scanner console = new Scanner(System.in);
        System.out.println("请输入单价 ( ¥ ): ");
        double unitPrice = console.nextDouble();
        System.out.println("请输入数量: ");
        double amount = console.nextDouble();
```

```

System.out.println("请输入金额 ( ¥ ): ");
double money = console.nextDouble();
console.close();

// 计算商品总价
double totalPrice = 0.0;
totalPrice = unitPrice * amount;

if (totalPrice >= 500) {
    totalPrice = totalPrice * 0.8;
}

// 计算找零
double change = money - totalPrice;
System.out.println("应收金额为: ¥" + totalPrice + ", 找零为: ¥" + change);

}
}

```

- **完整代码**

本案例的完整代码如下所示：

```

import java.util.Scanner;

public class Cashier {
    public static void main(String[] args) {
        // 输入数据
        Scanner console = new Scanner(System.in);
        System.out.println("请输入单价 ( ¥ ): ");
        double unitPrice = console.nextDouble();
        System.out.println("请输入数量: ");
        double amount = console.nextDouble();
        System.out.println("请输入金额 ( ¥ ): ");
        double money = console.nextDouble();
        console.close();

        // 计算商品总价
        double totalPrice = 0.0;
        totalPrice = unitPrice * amount;

        if (totalPrice >= 500) {
            totalPrice = totalPrice * 0.8;
        }

        // 计算找零
        double change = money - totalPrice;
        System.out.println("应收金额为: ¥" + totalPrice + ", 找零为: ¥" + change);
    }
}

```

3. 完成收银柜台收款程序 V3.0

- **问题**

编写一个收银柜台收款程序，根据商品单价、购买数量以及收款金额计算并输出应收金额和找零；当总价大于或等于 500 时，享受 8 折优惠。考虑程序的异常情况：收款金额小

于应收金额。控制台交互情况如下：

当收款金额大于等于应收金额时，控制台交互情况如图-4 所示。

```
<terminated> Cashier [Java Application] C:\Users\jessica\AppData\Local\Genuitec\Common\binary'
请输入单价（¥）：
30
请输入数量：
12
请输入金额（¥）：|
400
应收金额为：¥360.0，找零为：¥40.0
```

图- 4

当收款金额小于应收金额时，控制台的交互情况如图-5 所示。

```
<terminated> Cashier [Java Application] C:\Users\jessica\AppData\Local\Genuitec\Common\binary'
请输入单价（¥）：
30
请输入数量：
12
请输入金额（¥）：
300
输入信息有误！
```

图- 5

• 方案

本案例的实现方案如图-6 所示。图中T表示 true，F表示 false。

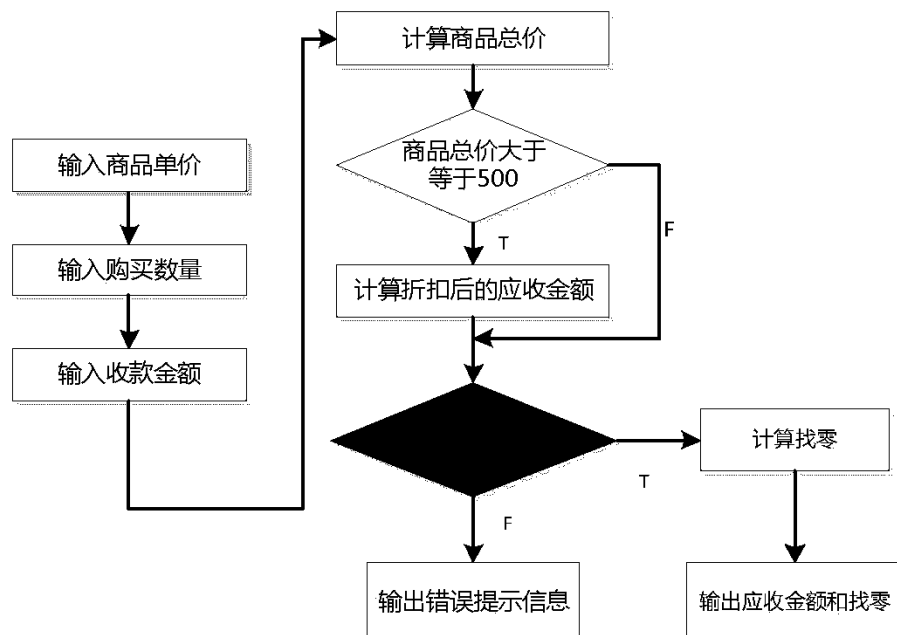


图- 6

• 步骤

实现此案例需要按照如下步骤进行。

步骤一：使用 if-else 修改上一案例

在本案例中，直到计算折扣之前与上一案例都是相同的。在此，只需要在上一案例的基础上修改为如下代码即可：

```
import java.util.Scanner;

public class Cashier {
    public static void main(String[] args) {
        // 输入数据
        Scanner console = new Scanner(System.in);
        System.out.println("请输入单价 ( ¥ ): ");
        double unitPrice = console.nextDouble();
        System.out.println("请输入数量: ");
        double amount = console.nextDouble();
        System.out.println("请输入金额 ( ¥ ): ");
        double money = console.nextDouble();
        console.close();

        // 计算商品总价
        double totalPrice = 0.0;
        totalPrice = unitPrice * amount;

        if (totalPrice >= 500) {
            totalPrice = totalPrice * 0.8;
        }

        if (money >= totalPrice) {
            double change = money - totalPrice;
            System.out.println("应收金额为: ¥" + totalPrice + ", 找零为: ¥" + change);
        } else {
            System.out.println("输入信息有误!");
        }
    }
}
```

从上述代码中，可以看出使用了 if-else 分支来判断收款金额和应收金额的大小。当收款金额大于等于应收金额时，计算找零，然后输出信息；当收款金额小于应收金额时，输出错误提示信息。

• 完整代码

本案例的完整代码如下所示：

```
import java.util.Scanner;

public class Cashier {
    public static void main(String[] args) {
```

```
// 输入数据
Scanner console = new Scanner(System.in);
System.out.println("请输入单价 (Y): ");
double unitPrice = console.nextDouble();
System.out.println("请输入数量: ");
double amount = console.nextDouble();
System.out.println("请输入金额 (Y): ");
double money = console.nextDouble();
console.close();

// 计算商品总价
double totalPrice = 0.0;
totalPrice = unitPrice * amount;

if (totalPrice >= 500) {
    totalPrice = totalPrice * 0.8;
}

if (money >= totalPrice) {
    double change = money - totalPrice;
    System.out.println("应收金额为: ¥" + totalPrice + ", 找零为: ¥" +
change);
} else {
    System.out.println("输入信息有误!");
}
}
```

4. 完成成绩等级输出程序

• 问题

学员成绩等级计算程序要求根据学员的分数计算该分数的所属等级并输出结果。首先，用户输入学员分数，该分数要求在 0-100 之间，如果录入错误，则提示错误信息，交互过程如图 - 7 所示。

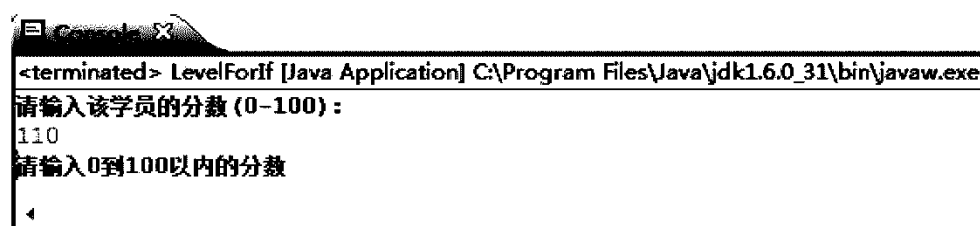


图- 7

如果用户录入的分数正确（在 0 到 100 之间），则根据表 - 1 中的规则计算该分数所对应的级别，并输出计算结果，交互过程如图 - 8 所示。

表 - 1 成绩和级别关系表

分数	等级
>=90	A
>=80	B
>=60	C
其它	D

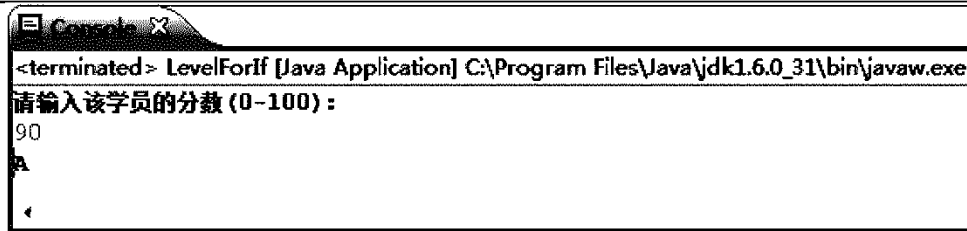


图- 8

另外，本案例要求使用 if-else 结构来实现。

• 步骤

实现此案例需要按照如下步骤进行。

步骤一：定义类及 main 方法

首先定义一个名为 LevelForIf 的类，并在类中定义 Java 应用程序的入口方法 main，代码如下所示：

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

步骤二：读取控制台的输入

在 main 方法中，实例化 Scanner 类，并调用 Scanner 类的 nextInt 方法接收用户所录入的学员分数，使用完毕后将 scanner 对象关闭。代码如下所示：

```
import java.util.Scanner;
public class LevelForIf {
    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);
        System.out.println("请输入该学员的分数(0-100):");
        int score = scanner.nextInt();
        scanner.close();

    }
}
```

步骤三：使用 if-else 语句进行判断

先使用 if 语句判断所录入的分数是否在 0 到 100 之间；如果不在正确的范围，则先输出提示信息，再使用 return 终止方法的运行。

如果录入的分数确实在 0 - 100 之间，则使用 if-else 结构判断不同的分数段并输出不同的级别。代码如下所示：

```
import java.util.Scanner;
public class LevelForIf {
    public static void main(String[] args) {
```



```
Scanner scanner = new Scanner(System.in);
System.out.println("请输入该学员的分数(0-100) : ");
int score = scanner.nextInt();
scanner.close();

if(score<0 || score>100){
    System.out.println("请输入 0 到 100 以内的分数");
}else if (score >= 90) {
    System.out.println("A");
} else if (score >= 80) {
    System.out.println("B");
} else if (score >= 60) {
    System.out.println("C");
} else {
    System.out.println("D");
}

}
}
```

- **完整代码**

本案例的完整代码如下所示：

```
import java.util.Scanner;
public class LevelForIf {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("请输入该学员的分数(0-100) : ");
        int score = scanner.nextInt();
        scanner.close();
        if(score<0 || score>100){
            System.out.println("请输入 0 到 100 以内的分数");
        }else if (score >= 90) {
            System.out.println("A");
        } else if (score >= 80) {
            System.out.println("B");
        } else if (score >= 60) {
            System.out.println("C");
        } else {
            System.out.println("D");
        }
    }
}
```

5. 完成命令解析程序

- **问题**

有命令解析程序，该程序提供三个功能选项供用户选择，用户选择某功能后，程序在界面上输出用户所选择的功能名称。程序的交互效果如图 - 9 所示。

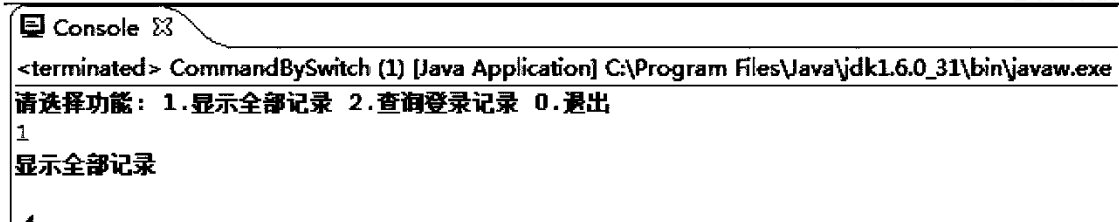


图 - 9

由图 - 9 可以看出，程序提供的功能有：显示全部记录、查询登录记录和退出。如果用户在控制台输入 1，则表示用户选择的功能为“显示全部记录”，此时，需要在界面上输出该功能的名称。

如果用户在控制台输入 2，则表示用户选择的功能为“查询登录记录”，此时，也需要在界面上输出该功能的名称，交互效果如图 - 10 所示。

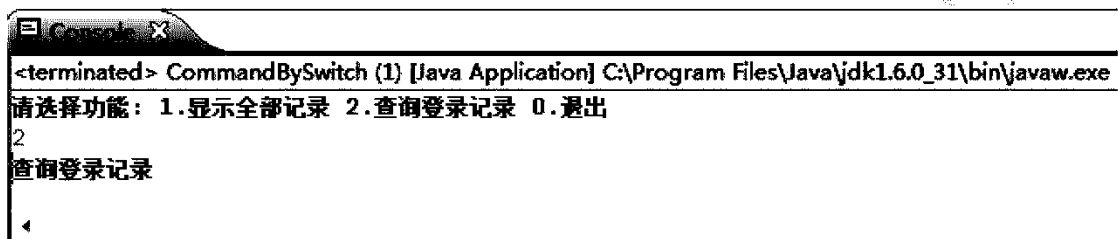


图 - 10

如果用户在控制台输入 0，则表示用户选择的功能为“退出”。此时，在界面上输出“欢迎使用”，表示程序结束。交互效果如图 - 11 所示。

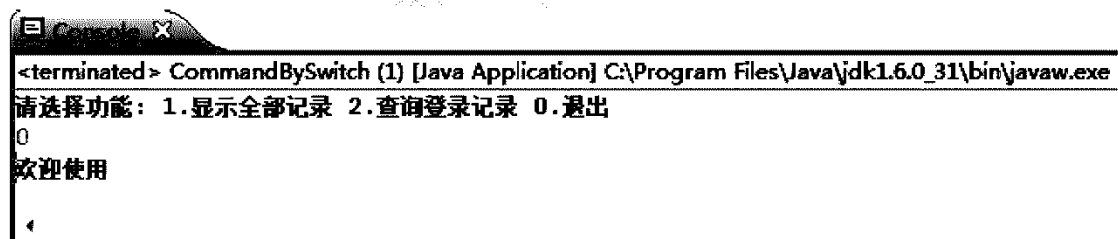


图 - 11

如果用户输入除 0, 1, 2 以外的其它数字，则表示选择错误，此时，在界面上输出“输入错误”。程序交互情况如图 - 12 所示。

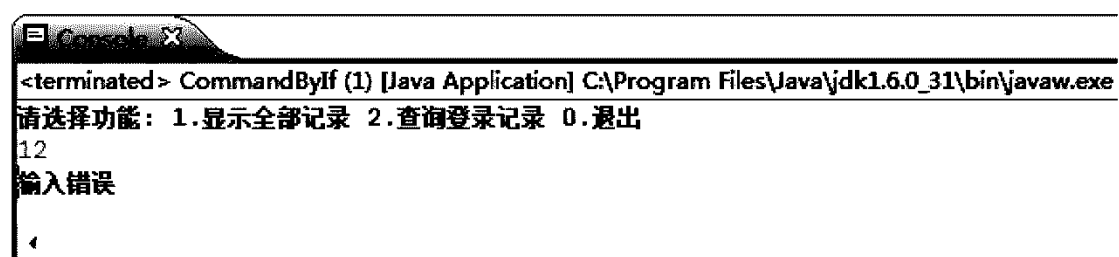


图 - 12

另外，本案例要求使用 switch-case 结构来实现。

- **步骤**

实现此案例需要按照如下步骤进行。

步骤一：定义类及 main 方法

首先定义一个名为 `CommandBySwitch` 的类，并在类中定义 Java 应用程序的入口方法 `main`，代码如下所示：

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

步骤二：读取控制台的输入

在 `main` 方法中，实例化 `Scanner` 类，并调用 `Scanner` 类的 `nextInt` 方法接收用户所选择的功能，使用完毕后将 `scanner` 对象关闭。代码如下所示：

```
import java.util.Scanner;  
  
public class CommandBySwitch {  
    public static void main(String[] args) {  
  
        Scanner scanner = new Scanner(System.in);  
        int command = 0;  
        System.out.println("请选择功能: 1.显示全部记录 2.查询登录记录 0.退出");  
        command = scanner.nextInt();  
        scanner.close();  
  
    }  
}
```

步骤三：使用 switch-case 解析命令

将 `switch-case` 结构配合 `break` 语句一起使用，判断用户所选择操作，并输出解析结果。代码如下所示：

```
import java.util.Scanner;  
public class CommandBySwitch {  
    public static void main(String[] args) {  
        Scanner scanner = new Scanner(System.in);  
        int command = 0;  
        System.out.println("请选择功能: 1.显示全部记录 2.查询登录记录 0.退出");  
        command = scanner.nextInt();  
        scanner.close();  
  
        switch (command) {  
            case 1:  
                System.out.println("显示全部记录");  
                break;  
            case 2:  
                System.out.println("查询登录记录");  
                break;  
            case 0:  

```

```

        System.out.println("欢迎使用");
        break;
    default:
        System.out.println("输入错误");
    }

}
}

```

- **完整代码**

本案例的完整代码如下所示：

```

import java.util.Scanner;
public class CommandBySwitch {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int command = 0;
        System.out.println("请选择功能: 1.显示全部记录 2.查询登录记录 0.退出");
        command = scanner.nextInt();
        scanner.close();
        switch (command) {
            case 1:
                System.out.println("显示全部记录");
                break;
            case 2:
                System.out.println("查询登录记录");
                break;
            case 0:
                System.out.println("欢迎使用");
                break;
            default:
                System.out.println("输入错误");
        }
    }
}

```

课后作业

1. 指出下面程序的运行输出结果

```
int a, b, c;  
a = b = c = 100;  
System.out.println("a="+a+",b="+b+",c="+c);
```

2. 指出下面程序的运行输出结果

```
System.out.println(5+6+" "+5+6);
```

3. 输出两个 int 数中的最大值

用户从控制台接收两个整数，通过程序找出两个数中的最大值。控制台的交互效果如图-1 所示。

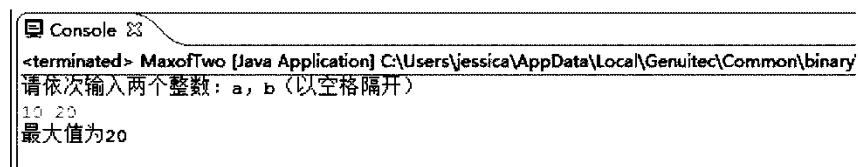


图- 2

4. 输出三个 int 数中的最大值（提高题，选做）

用户从控制台接收三个整数，通过程序找出三个数中的最大值。控制台的交互效果如图-2 所示。

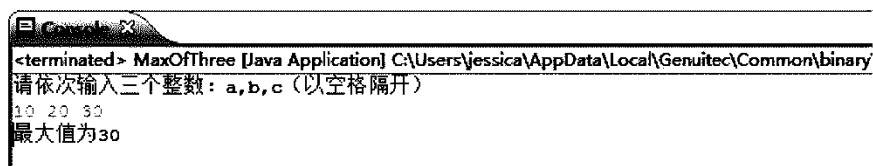


图- 3

5. 编写三个数值的排序程序

使用程序为用户所录入的 3 个数值进行升序排列，并将排序后的结果输出到控制台。程序交互过程如图 - 3 所示：

```

Console
<terminated> Swap [Java Application] C:\Program Files\Java\jdk1.6.0_31\bin\javaw.exe
请依次输入三个整数：a, b, c (以空格隔开)
20 5 10
您输入的是：
a=20, b=5, c=10
升序排列后，结果为：
a=5, b=10, c=20

```

图- 3

6. 编写程序判断某一个年份是否为闰年（使用 if-else）

本案例需要使用交互的方式判断某年是否为闰年：用户从控制台输入需要判断的年份值，由程序使用 if-else 判断该年是否为闰年，并将判断结果输出到控制台。程序交互过程如图 - 4 所示：

```

Console
<terminated> LeapYear [Java Application] C:\Program Files\Java\jdk1.6.0_31\bin\javaw.exe
请输入年份（例如：2012）
2012
2012是闰年

```

图- 4

7. 编写个人所得税计算程序

个人所得税是国家对本国公民、居住在本国境内的个人的所得和境外个人来源于本国的所得征收的一种所得税。目前，北京地区的个人所得税的计算公式为：应纳税额 = (工资薪金所得 - 扣除数) × 适用税率 - 速算扣除数。其中，扣除数为 3500 元，适用税率以及速算扣除数如下表-1 所示。

表 - 1 个人所得税缴纳标准

全月应纳税所得额	税率	速算扣除数(元)
全月应纳税额不超过 1500 元	3%	0
全月应纳税额超过 1500 元至 4500 元	10%	105
全月应纳税额超过 4500 元至 9000 元	20%	555
全月应纳税额超过 9000 元至 35000 元	25%	1005
全月应纳税额超过 35000 元至 55000 元	30%	2755
全月应纳税额超过 55000 元至 80000 元	35%	5505
全月应纳税额超过 80000 元	45%	13505

上表中的全月应纳税所得额=工资薪金所得 - 扣除数。

本案例要求计算个人所得税的缴纳额度：用户从控制台输入税前工资的金额，程序计算

所需要交纳的个人所得税的金额，并将计算结果输出到控制台。

程序的交互过程如图-5 所示：

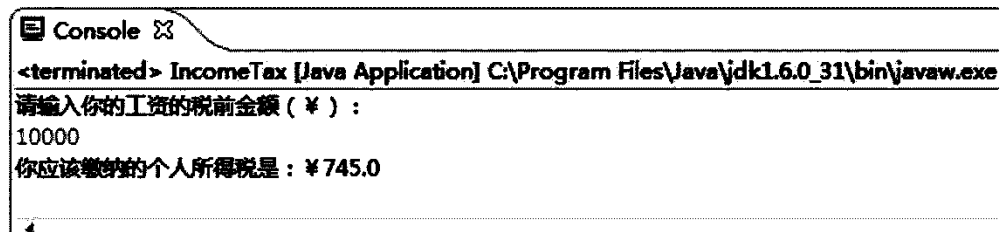


图- 5

8. 输入年份和月份，输出该月的天数（使用 switch-case）

一年有 12 个月，而每个月的天数是不一样的。其中，有 7 个月为 31 天，称为大月，分别为 1、3、5、7、8、10、12 月；有 4 个月为 30 天，称为小月，分别为 4、6、9、11 月；还有二月比较特殊，平年的二月只有 28 天，而闰年的二月有 29 天。

本案例需要使用交互的方式计算某年某月的天数：由用户在控制台输入年份和月份值，程序计算该年该月的天数，并将结果输出在控制台。程序交互情况如图 - 6 所示：

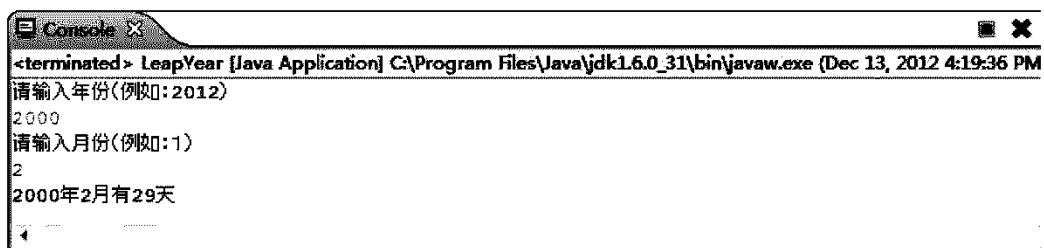


图- 6

Java 语言基础

Unit04

知识体系.....Page 83

循环结构	什么是循环	什么是循环结构
	while 语句	while 语句的执行逻辑
		while 语句的流程图
		while 语句用于处理循环逻辑
		使用 break 语句跳出循环
	do-while 语句	do-while 语句的执行逻辑
		do-while 语句的流程图
		do-while 语句用于处理循环逻辑
		while 和 do-while 语句的区别
	for 语句	考虑如下循环问题的相同之处
		for 语句的执行逻辑
		for 语句的流程图
		for 语句用于实现固定次数循环
		for 语句三个表达式特殊用法
		循环中使用 break 语句
		循环中使用 continue 语句

经典案例.....Page 90

猜数字游戏 V1.0	while 语句的执行逻辑
	while 语句的流程图
	while 语句用于处理循环逻辑
	使用 break 语句跳出循环
猜数字游戏 V2.0	do-while 语句的执行逻辑
	do-while 语句的流程图
	do-while 语句用于处理循环逻辑
	while 和 do-while 语句的区别
随机加法运算器	循环中使用 break 语句
	循环中使用 continue 语句

1. 循环结构

1.1. 什么是循环


1.1.1. 【什么是循环】什么是循环结构

知识讲解

什么是循环结构

Tarena
达内科技

- 循环是程序设计语言中反复执行某些代码的一种计算机处理过程，是一组相同或相似语句被有规律的重复性执行。
- 循环的要素：
 - 循环体（相同或相似的语句）
 - 循环条件（继续执行循环的条件，某些情况下循环条件以循环次数的方式体现）



1.2. while 语句

1.2.1. 【while 语句】while 语句的执行逻辑

知识讲解

while语句的执行逻辑

Tarena
达内科技

- 计算boolean表达式的值
- 如果值为true则执行语句块；语句块执行完后再次判断boolean表达式的值，如果为true则继续执行语句块；如此循环往复，直到boolean为false时退出while循环。

```

while ( boolean表达式 ) {
    语句块
}
                    
```

boolean表达式结果为真时，
执行语句块；否则退出

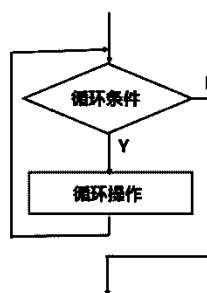
被重复执行的操作

1.2.2. 【while 语句】while 语句的流程图

知识讲解

while语句的流程图

Tarena
达内科技



需要注意：
一般情况下，循环操作中会存在使得循环条件不满足的可能性，否则将成为“死循环”。

83

1.2.3. 【while 语句】while 语句用于处理循环逻辑

代码编辑

while语句用于处理循环逻辑

Tarena
达内科技

```

int age = 1;

while (age <= 100) {
    System.out.println( "马上有钱" );
    age++;
}

System.out.println( "over" );
                    
```

+

1.2.4. 【while 语句】使用 break 语句跳出循环

代码编辑

使用break语句跳出循环

Tarena
达内科技

- break用在循环体中用于退出循环

```

int x=0;
while ( x < 10 ) {
    if ( x == 5 ) {
        break;
    }
    System.out.println(x);
    x ++ ;
}
                    
```

输出的结果是：0 1 2 3 4，当 x==5时，退出循环。

+

1.3. do-while 语句

1.3.1. 【do-while 语句】do-while 语句的执行逻辑

代码编辑

do-while语句的执行逻辑

Tarena
达内科技

1. 先执行语句块
2. 再计算boolean表达式的值，如果为true，再次执行语句块，如此循环往复，直到boolean表达式的值为false为止。

无论boolean表达式是否为true，都先执行一次语句块；

```

do{
    语句块
} while ( boolean表达式);
                    
```

+

84

1.3.2. 【do-while 语句】do-while 语句的流程图

Tarena
达内科技

do-while语句的流程图

```

graph TD
    subgraph while_flow [while]
        W_Start(( )) --> W_Cond{循环条件}
        W_Cond -- Y --> W_Oper[循环操作]
        W_Oper --> W_Cond
        W_Cond -- N --> W_End(( ))
    end
    subgraph do_while_flow [do-while]
        DW_Start(( )) --> DW_Oper[循环操作]
        DW_Oper --> DW_Cond{循环条件}
        DW_Cond -- Y --> DW_Oper
        DW_Cond -- N --> DW_End(( ))
    end
            
```

Tarena
达内科技

1.3.3. 【do-while 语句】do-while 语句用于处理循环逻辑

Tarena
达内科技

do-while语句用于处理循环逻辑

```

int pwd ;
do{
    System.out.print( "请输入密码" );
    pwd = scanner.nextInt();
} while ( 123 != pwd );
            
```

Tarena
达内科技

1.3.4. 【do-while 语句】while 和 do-while 语句的区别

Tarena
达内科技

while和do-while语句的区别

- while和do-while语句的区别：
 - while循环先判断再执行；
 - do-while循环先执行一次，再判断；
- 当初始情况不满足循环条件时，while循环一次都不会执行；do-while循环不管任何情况都至少执行一次。

while和do-while语句的不同仅仅会体现在第一次就不满足条件的循环中；如果不是这样的情况，while和do-while可以互换。

Tarena
达内科技

知识梳理

while和do-while语句的区别 (续1) Tarena 达内科技

```

int pwd ;
do{
    System.out.print( "密码" );
    pwd = scanner.nextInt();
} while ( 123 != pwd );

int pwd=0;
while ( 123 != pwd){
    System.out.print( "密码：" );
    pwd = scanner.nextInt();
}
                    
```

1.4. for 语句

1.4.1. 【for 语句】考虑如下循环问题的相同之处

知识梳理

考虑如下循环问题的相同之处 Tarena 达内科技

- 计算从1加到100的值；
- 计算 $1+1/3+1/5+\dots+1/999$ ；
- 找出从第1号学员到第500号学员中成绩大于90的学员。
- ...

- $i=1,2,\dots,100$ ，每次循环累加*i*的值。
- $i=1,3,5,\dots,999$ ，每次循环计算 $1/i$ ，并累加。
- $i=1,2,\dots,500$ ，每次循环判断第*i*号学员成绩是否大于90。

这样的变量*i*称之为循环变量；
在每次循环中，它规律地发生变化。同时，它作为判断是否继续循环的条件。

1.4.2. 【for 语句】for 语句的执行逻辑

知识梳理

for语句的执行逻辑 Tarena 达内科技

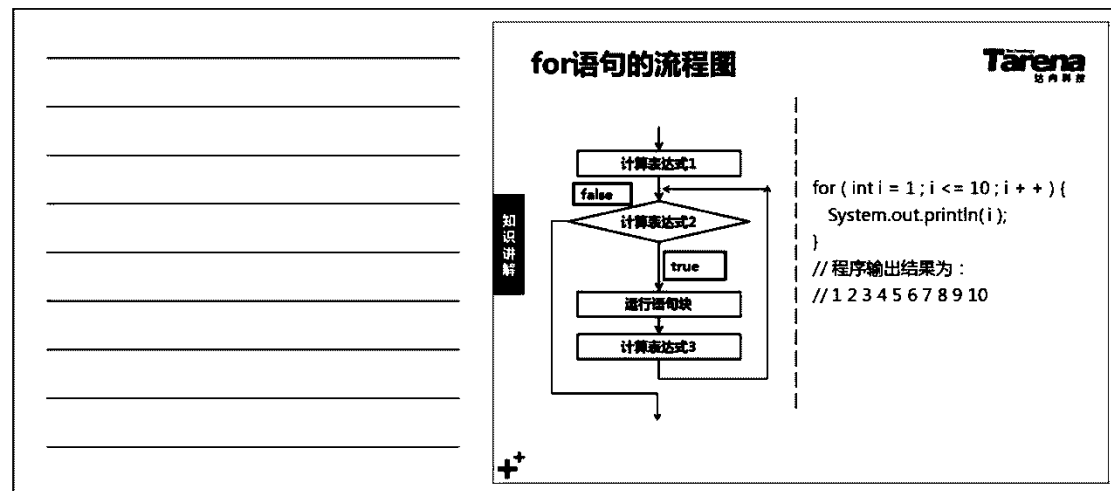
```

for (表达式1; 表达式2; 表达式3) {
    语句块 (循环体)
}
                    
```

1. 计算表达式1的值；
2. 计算表达式2 (表达式2为逻辑表达式) 的值，如果为true则执行循环体，否则退出循环；
3. 执行循环体；
4. 执行表达式3；
5. 计算表达式2，如果为true则执行循环体，否则退出循环；
6. 如此循环往复，直到表达式2的值为false。

86

1.4.3. 【for 语句】for 语句的流程图



1.4.4. 【for 语句】for 语句用于实现固定次数循环

Tarena
达内科技

for语句用于实现固定次数循环

知识讲解

- 累加
 - 求 $1 + 2 + 3 + \dots + 100 = ?$

```

int sum = 0;
for ( int i = 1; i <= 100; i ++ ) {
    sum += i;
}
System.out.println( "1到100的和为：" + sum );
                    
```
- 阶乘
 - 求 $1 * 2 * 3 * \dots * 10 = ?$

++

1.4.5. 【for 语句】for 语句三个表达式特殊用法

Tarena
达内科技

for语句三个表达式特殊用法

知识讲解

- 表达式1位置内容为空时：

```

int sum = 0;
int i = 1;
for ( ; i <= 10; i ++ ) {
    sum += i;
}
System.out.println( "1到10的和为：" + sum );
                    
```

++

for语句三个表达式特殊用法（续1） Tarena 达内科技

- 表达式3位置内容为空时：

```
int sum = 0;
for ( int i = 1; i <= 10; ) {
    sum += i;
    i ++;
}
System.out.println( "1到10的和为：" + sum );
```

代码清单

+

for语句三个表达式特殊用法（续2） Tarena 达内科技

- 表达式 1 , 2 , 3 位置内容均为空时：

```
for ( ; ; ) {
    System.out.println( "我要学习....." );
}
```

代码清单

死循环

代码清单

+

for语句三个表达式特殊用法（续3） Tarena 达内科技

- 表达式 1 和 3 位置内容的多样化：

```
for ( int i = 1, j = 6 ; i <= 6 ; i += 2, j -= 2 ) {
    System.out.println( " i, j = " + i + " , " + j );
}
```

代码清单

输出的结果是：



```
i, j = 1, 6
i, j = 3, 4
i, j = 5, 2
```

for语句中的三个表达式中表达式1和表达式3可以使用逗号表达式，逗号表达式就是通过“,”运算符隔开的多个表达式组成的表达式，从左向右计算。



代码清单

+

1.4.6. 【for 语句】循环中使用 break 语句

<hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/>	<div style="text-align: right;">  </div> <h3>循环中使用break语句</h3> <ul style="list-style-type: none"> • break可用于循环语句或switch语句中； • break用于循环，可使程序终止循环而执行循环后面的语句，常常与条件语句一起使用。 <div style="border: 1px solid black; padding: 5px; margin: 10px 0;"> <pre>int sum = 0; for(int i=1; i<=100; i++){ if(sum>=4000){ break; } sum += i; }</pre> </div> <div style="border: 1px solid black; padding: 2px; text-align: center; margin: 5px 0;"> 当总和大于等于4000时终止循环 </div> <div style="text-align: right;">  </div>
---	---

1.4.7. 【for 语句】循环中使用 continue 语句

<hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/>	<div style="text-align: right;">  </div> <h3>循环中使用continue语句</h3> <ul style="list-style-type: none"> • continue只能用于循环中 • 其作用为跳过循环体中剩余语句而执行下一次循环 <div style="border: 1px solid black; padding: 5px; margin: 10px 0;"> <pre>int sum = 0; for(int i=1; i<=100; i++){ if(i % 10 == 3){ continue; } sum += i; }</pre> </div> <div style="border: 1px solid black; padding: 2px; text-align: center; margin: 5px 0;"> 统计总和时，跳过所有个位为3的 </div> <div style="text-align: right;">  </div>
---	--

经典案例

1. 猜数字游戏 V1.0

- 问题

猜数字游戏，其游戏规则为：程序内置一个 1 到 1000 之间的数字作为猜测的结果，由用户猜测此数字。用户每猜测一次，由系统提示猜测结果：大了、小了或者猜对了；直到用户猜对结果，则提示游戏结束。用户可以提前退出游戏，即，游戏过程中，如果用户录入数字 0，则游戏终止。游戏的交互过程如下：

1. 游戏刚开始，即提示用户在控制台录入所猜测的数字，交互过程如图-1 所示：

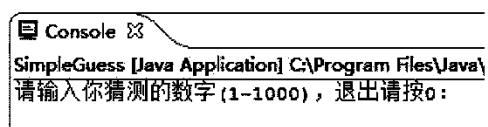


图 - 1

2. 用户录入所猜测的数字后，程序进行判断：如果用户所猜测的数字大于结果，则提示“太大了！”；如果用户所猜测的数字小于结果，则提示“太小了”。每次提示猜测结果后，并提醒用户继续猜测。交互过程如图 - 2 所示：

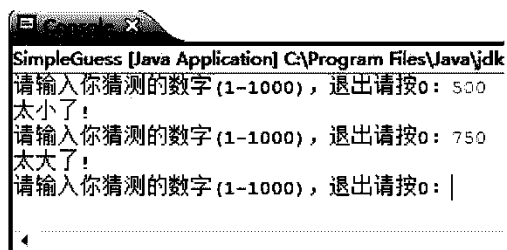


图 - 2

3. 如果用户猜测正确，则由系统提示“恭喜，你猜对了！”，游戏结束。交互过程如图 - 3 所示：

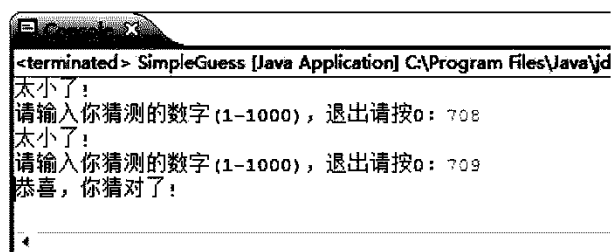


图 - 3

4. 如果用户希望提前退出游戏，则可以录入数字 0，游戏结束。交互过程如图 - 4 所示：

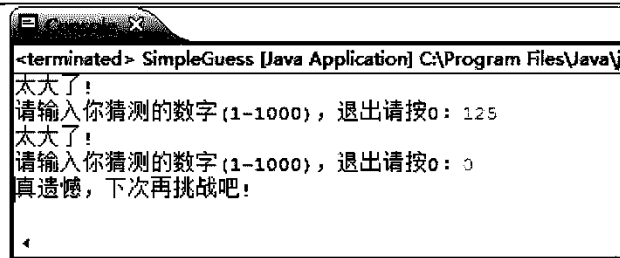


图 - 4

• 方案

首先，此案例中，需要产生一个 1 到 1000 之间的随机整数，该数值就是用户要猜测的结果；然后，提示用户进行第一次猜测，并得到用户从界面所录入的数字。

因为猜测的次数不确定，则使用 while 循环来构建猜测的过程：判断用户所猜测的数字是否与结果相同，只要猜测不正确，则循环继续。每次循环中，首先判断用户录入的数字是否为 0，如果是，则使用 break 退出循环；否则，根据比较结果输出提示信息（“太大了”或者“太小了”），并提示用户继续下一次猜测。

如果用户猜测正确或者录入数字 0，则 while 循环结束。循环结束后，需要判断用户最后一次所录入的数字，如果猜测正确，则提示用户“恭喜，你猜对了！”；如果录入的为数字 0，则提示用户“真遗憾，下次再挑战吧！”，然后程序结束。程序的流程如图 - 5 所示：

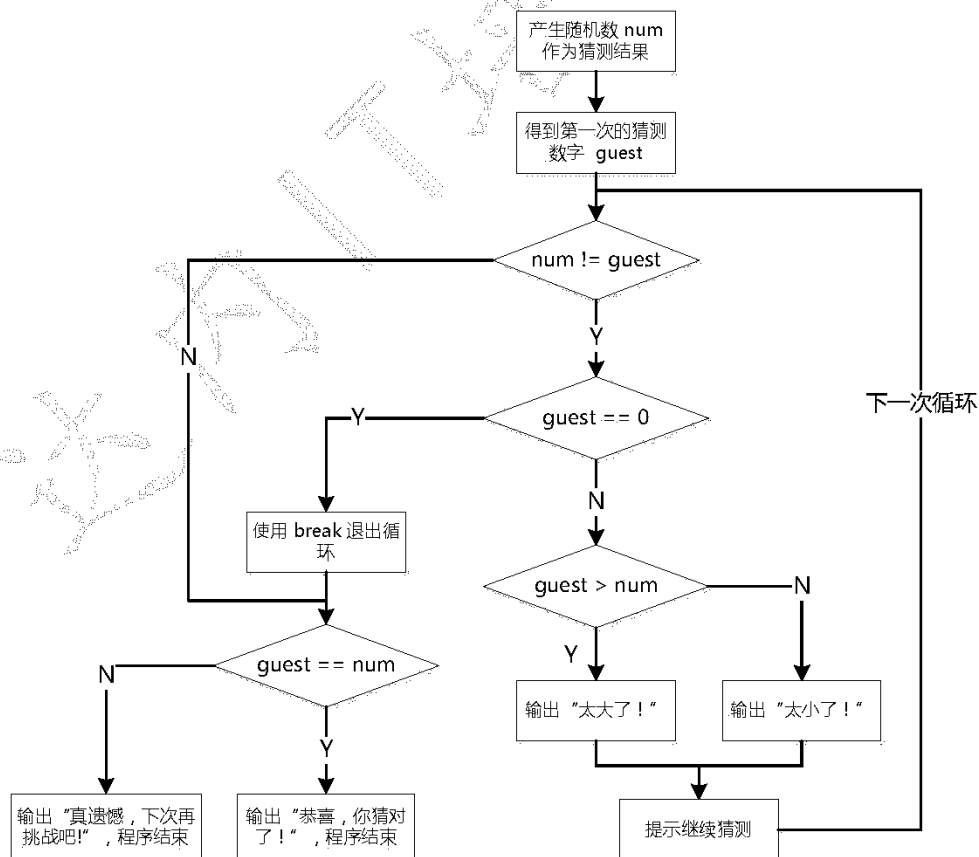


图 - 5

- **步骤**

实现此案例需要按照如下步骤进行。

步骤一：定义类及 main 方法

首先定义一个名为 NumberGuessV01 的类，并在类中添加 Java 应用程序的主方法 main，代码如下所示：

```
public class NumberGuessV01 {  
    public static void main(String[] args) {  
    }  
}
```

步骤二：读取第一次猜测结果

在 main 方法中，提示用户录入所猜测的数字，作为第一次猜测；然后，实例化 Scanner 类，并调用 Scanner 类的 nextInt 方法接收用户从控制台输入的数字。代码如下所示：

```
import java.util.Scanner;  
public class NumberGuessV01 {  
    public static void main(String[] args) {  
  
        //生成一个随机数作为猜测的结果  
        int num = (int) (Math.random() * 1000) + 1;  
  
        //第一次猜测  
        System.out.print("请输入你猜测的数字(1-1000)，退出请按 0：");  
        Scanner s = new Scanner(System.in);  
        int guest = s.nextInt();  
  
    }  
}
```

注意：此步骤中，需要导入 java.util 包下的 Scanner 类。

步骤三：构建循环

使用 while 循环，判断用户所猜测的数字是否与结果相同：只要猜测不正确，则循环继续。每次循环中，需要使用 if-else 结构判断用户所录入的数字。首先，判断录入的数字是否为 0，如果为 0，表示用户希望退出游戏，则使用 break 退出循环；然后比较猜测的数字和实际数字的大小，并根据比较结果输出提示信息（“太大了”或者“太小了”），然后提示用户继续下一次猜测，并调用 Scanner 类的 nextInt 方法接收用户录入的下一个数字。代码如下所示：

```
import java.util.Scanner;  
public class NumberGuessV01 {  
    public static void main(String[] args) {  
        //生成一个随机数作为猜测的结果  
        int num = (int) (Math.random() * 1000) + 1;  
  
        //第一次猜测
```

```

System.out.print("请输入你猜测的数字(1-1000), 退出请按 0:");
Scanner s = new Scanner(System.in);
int guest = s.nextInt();

//如果猜测错误, 则继续
while (guest!=num) {
    //输入为 0, 则退出循环; 否则判断数字
    if(guest == 0){
        break;
    } else if (guest > num) {
        System.out.println("太大了!");
    } else {
        System.out.println("太小了!");
    }
    System.out.print("请输入你猜测的数字(1-1000), 退出请按 0:");
    guest = s.nextInt();
}
}
}

```

步骤四：游戏结束

如果用户猜测正确或者录入数字 0 中止游戏, 则 while 循环结束。在 while 循环结束后, 需要判断用户最后一次所录入的数字, 如果猜测正确, 则提示用户“恭喜, 你猜对了!”; 如果录入的为数字 0, 则提示用户“真遗憾, 下次再挑战吧!”。最后将 scanner 对象关闭, 程序结束。代码如下所示:

```

import java.util.Scanner;
public class NumberGuessV01 {
    public static void main(String[] args) {
        //生成一个随机数作为猜测的结果
        int num = (int) (Math.random() * 1000) + 1;

        //第一次猜测
        System.out.print("请输入你猜测的数字(1-1000), 退出请按 0:");
        Scanner s = new Scanner(System.in);
        int guest = s.nextInt();

        //如果猜测错误, 则继续
        while (guest!=num) {
            //输入为 0, 则退出循环; 否则判断数字
            if(guest == 0){
                break;
            } else if (guest > num) {
                System.out.println("太大了!");
            } else {
                System.out.println("太小了!");
            }
            System.out.print("请输入你猜测的数字(1-1000), 退出请按 0:");
            guest = s.nextInt();
        }
    }
}

```

```
//提示用户最终结果
if(guess == num) {
    System.out.println("恭喜，你猜对了!");
}else{
    System.out.println("真遗憾，下次再挑战吧!");
}
s.close();

}
}
```

- **完整代码**

本案例的完整代码如下所示：

```
import java.util.Scanner;
public class NumberGuessV01 {
    public static void main(String[] args) {
        //生成一个随机数作为猜测的结果
        int num = (int) (Math.random() * 1000) + 1;

        //第一次猜测
        System.out.print("请输入你猜测的数字(1-1000)，退出请按 0：");
        Scanner s = new Scanner(System.in);
        int guest = s.nextInt();

        //如果猜测错误，则继续
        while (guest!=num) {
            //输入为 0，则退出循环；否则判断数字
            if(guess == 0){
                break;
            } else if (guest > num) {
                System.out.println("太大了!");
            } else {
                System.out.println("太小了!");
            }
            System.out.print("请输入你猜测的数字(1-1000)，退出请按 0：");
            guest = s.nextInt();
        }

        //提示用户最终结果
        if(guess == num) {
            System.out.println("恭喜，你猜对了!");
        }else{
            System.out.println("真遗憾，下次再挑战吧!");
        }
        s.close();
    }
}
```

```
}  
}
```

2. 猜数字游戏 V2.0

• 问题

使用 do - while 语句实现猜数字游戏，界面的交互过程和上一个案例相同。

• 方案

此案例的实现方案和之前的案例类似。

首先，此案例中，需要产生一个 1 到 1000 之间的随机整数，该数值就是用户要猜测的结果；然后，提示用户进行第一次猜测，并得到用户从界面所录入的数字。

因为猜测的次数不确定，则使用 do-while 循环来构建猜测的过程：判断用户所猜测的数字是否与结果相同，只要猜测不正确，则循环继续。每次循环中，首先判断用户录入的数字是否为 0，如果是，则使用 break 退出循环；否则，根据比较结果输出提示信息（“太大了”或者“太小了”），并提示用户继续下一次猜测。

如果用户猜测正确或者录入数字 0，则循环结束。循环结束后，需要判断用户最后一次所录入的数字，如果猜测正确，则提示用户“恭喜，你猜对了！”；如果录入的为数字 0，则提示用户“真遗憾，下次再挑战吧！”，然后程序结束。程序的流程如图 - 6 所示：

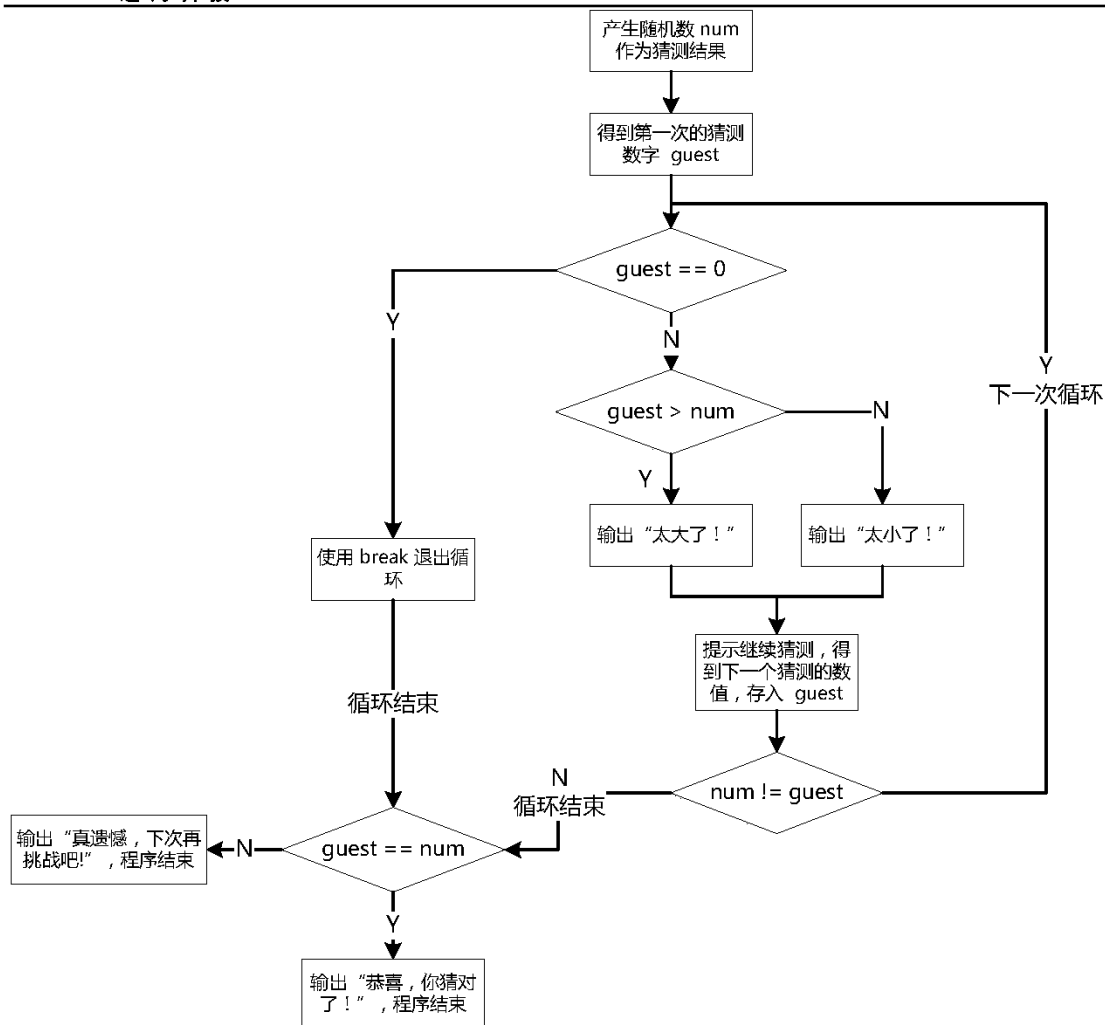


图 - 6

• 步骤

实现此案例需要按照如下步骤进行。

步骤一：定义类及 main 方法

首先定义一个名为 NumberGuessV03 的类，并在类中添加 Java 应用程序的主方法 main，代码如下所示：

```
public class NumberGuessV03 {
    public static void main(String[] args) {
    }
}
```

步骤二：准备录入

在 main 方法中，生成一个随机数作为猜测的结果，然后实例化 Scanner 类，用于接收用户从控制台输入的数字，还需要声明变量用于存储用户所录入的数字。代码如下所示：

```
import java.util.Scanner;
public class NumberGuessV03 {
```

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

    //生成一个随机数作为猜测的结果
    int num = (int) (Math.random() * 1000) + 1;

    //准备输入
    Scanner s = new Scanner(System.in);
    int guest = -1;

}
}
```

注意：此步骤中，需要导入 java.util 包下的 Scanner 类。

步骤三：构建循环

使用 do-while 循环，提示用户进行猜测，并调用 Scanner 类的 nextInt 方法接收用户录入的数字，然后判断用户所猜测的数字是否与结果相同：只要猜测不正确，则循环继续。

每次循环中，需要使用 if-else 结构判断用户所录入的数字。首先，判断录入的数字是否为 0，如果为 0，表示用户希望退出游戏，则使用 break 退出循环；然后比较猜测的数字和实际数字的大小，并根据比较结果输出提示信息（“太大了”或者“太小了”）。代码如下所示：

```
import java.util.Scanner;
public class NumberGuessV03 {
    public static void main(String[] args) {
        //生成一个随机数作为猜测的结果
        int num = (int) (Math.random() * 1000) + 1;

        //准备输入
        Scanner s = new Scanner(System.in);
        int guest = -1;

        do{
            System.out.print("请输入你猜测的数字(1-1000)，退出请按 0：");
            guest = s.nextInt();
            //判断：输入 0，则中止
            if(guest == 0){
                break;
            } else if(guest > num) {
                System.out.println("太大了!");
            } else if(guest < num){
                System.out.println("太小了!");
            }
        }while(guest!=num);

    }
}
```

步骤四：游戏结束

如果用户猜测正确或者录入数字 0 中止游戏，则循环结束。在循环结束后，需要判断

用户最后一次所录入的数字，如果猜测正确，则提示用户“恭喜，你猜对了！”；如果录入的为数字 0，则提示用户“真遗憾，下次再挑战吧！”。最后将 scanner 对象关闭，程序结束。代码如下所示：

```
import java.util.Scanner;
public class NumberGuessV03 {
    public static void main(String[] args) {
        //生成一个随机数作为猜测的结果
        int num = (int) (Math.random() * 1000) + 1;

        //准备输入
        Scanner s = new Scanner(System.in);
        int guest = -1;
        do{
            System.out.print("请输入你猜测的数字(1-1000)，退出请按 0：");
            guest = s.nextInt();
            //判断：输入 0，则中止
            if(guest == 0){
                break;
            } else if(guest > num) {
                System.out.println("太大了!");
            } else if(guest < num){
                System.out.println("太小了!");
            }
        }while(guest!=num);

        //提示用户最终结果
        if(guest == num) {
            System.out.println("恭喜，你猜对了!");
        }else{
            System.out.println("真遗憾，下次再挑战吧!");
        }
        s.close();
    }
}
```

- **完整代码**

本案例的完整代码如下所示：

```
import java.util.Scanner;
public class NumberGuessV03 {
    public static void main(String[] args) {
        //生成一个随机数作为猜测的结果
        int num = (int) (Math.random() * 1000) + 1;

        //准备输入
        Scanner s = new Scanner(System.in);
        int guest = -1;
        do{
```

```

        System.out.print("请输入你猜测的数字(1-1000)，退出请按 0：");
        guest = s.nextInt();
        //判断：输入 0，则中止
        if(guest == 0){
            break;
        } else if(guest > num){
            System.out.println("太大了!");
        } else if(guest < num){
            System.out.println("太小了!");
        }
    }while(guest!=num);

    //提示用户最终结果
    if(guest == num){
        System.out.println("恭喜，你猜对了!");
    }else{
        System.out.println("真遗憾，下次再挑战吧!");
    }
    s.close();
}
}

```

3. 随机加法运算器

• 问题

有加法运算器程序，其规则为：程序依次出 10 道加法题目，由用户输入题目的答案。用户每答完一道题，由系统提示结果：答错了或者答对了。10 道题目答完之后，系统计算得分并输出。如果用户希望提前结束，则可以输入 -1 提前退出。

本案例要求使用交互的方式实现此游戏，交互过程为：

1、程序开始，即出现一道加法题（两个加数均为 0 到 99 之间的随机整数），并提示用户输入该题目的答案（输入 -1 则会提前退出程序）。交互过程如图 - 7 所示：

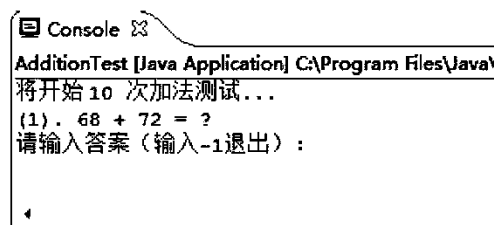
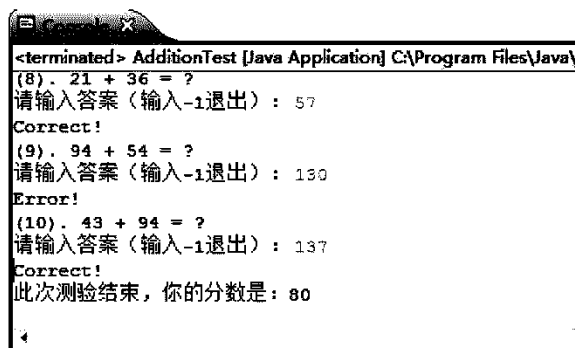


图 - 7

2、用户录入后，程序进行判断：如果用户录入的答案错误，则提示“Error !”；如果答案正确，则提示“Correct !”。然后给出下一道题目，并提醒用户继续答题。用户答完 10 道题目后，系统给出用户的得分并显示（每道题目 10 分），程序结束。交互过程如图 - 8

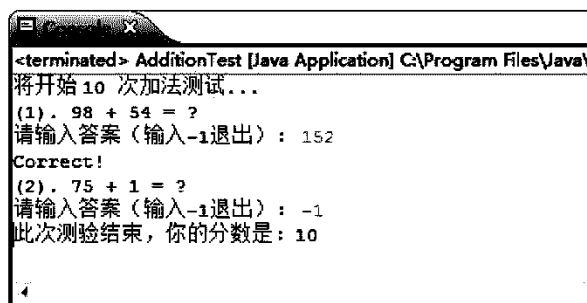
所示：



```
<terminated> AdditionTest [Java Application] C:\Program Files\Java\  
(8). 21 + 36 = ?  
请输入答案 (输入-1退出): 57  
Correct!  
(9). 94 + 54 = ?  
请输入答案 (输入-1退出): 130  
Error!  
(10). 43 + 94 = ?  
请输入答案 (输入-1退出): 137  
Correct!  
此次测验结束, 你的分数是: 80
```

图 - 8

3、如果用户录入 - 1，则表示希望提前退出程序，系统将提示用户的分数，且程序结束。交互过程如图 - 9 所示：



```
<terminated> AdditionTest [Java Application] C:\Program Files\Java\  
将开始 10 次加法测试...  
(1). 98 + 54 = ?  
请输入答案 (输入-1退出): 152  
Correct!  
(2). 75 + 1 = ?  
请输入答案 (输入-1退出): -1  
此次测验结束, 你的分数是: 10
```

图 - 9

• 方案

此案例中，需要使用 for 循环产生 10 道加法题目。

在每次循环中，需要产生两个 0 到 99 之间的随机整数，作为加法题目的两个加数，并计算出正确答案；然后，输出题目，并提示用户进行答题，从而得到用户从界面所录入的数字。

得到用户的录入后，进行判断：如果用户录入的为 - 1，则需要提前退出循环；如果用户录入的答案错误，则提示 “Error !”，并继续下一次答题；如果用户录入的答案正确，则提示 “Correct !”，继续下一次答题。

循环结束后，计算用户的得分并输出，程序结束。注：用户得分的规则是，每答对一道题，得 10 分，满分为 100 分。

程序的流程如图 - 10 所示：

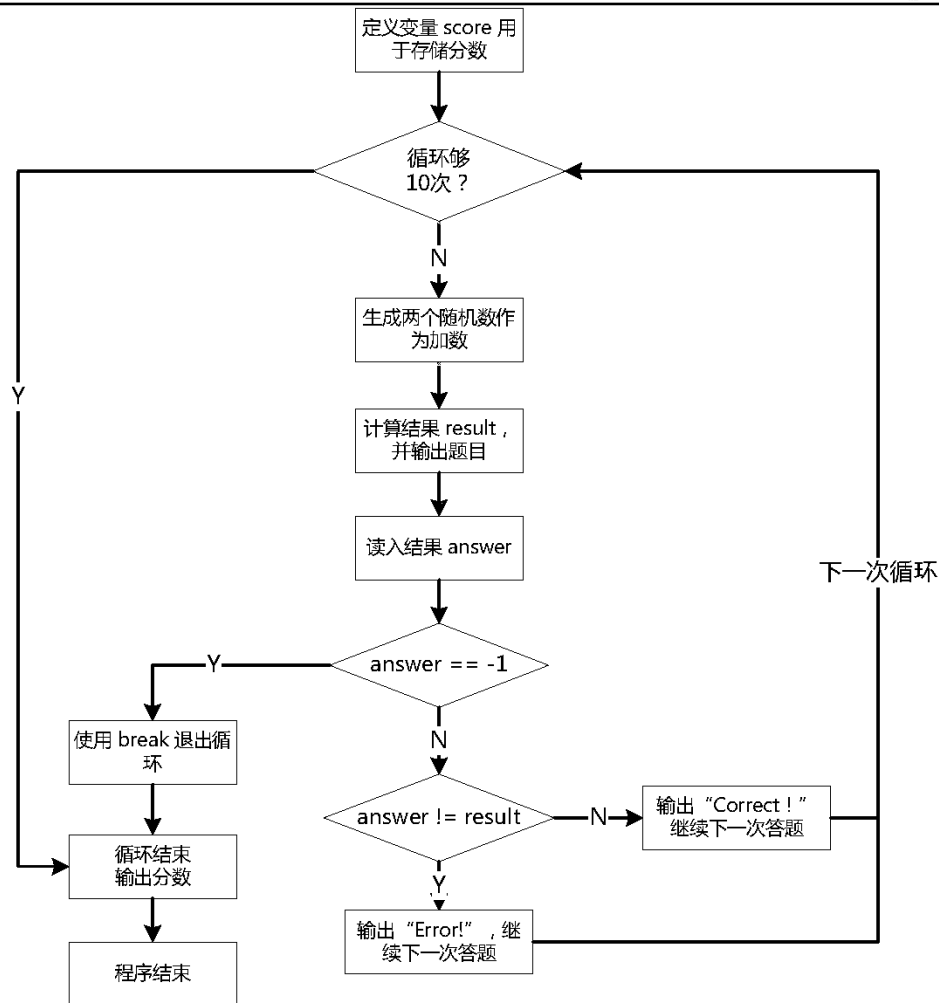


图 - 10

• 步骤

实现此案例需要按照如下步骤进行。

步骤一：定义类及 main 方法

首先定义一个名为 AdditionTest 的类 ,并在类中添加 Java 应用程序的主方法 main ,代码如下所示：

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

步骤二：准备

在 main 方法中，输出程序即将开始的提示信息，并实例化 Scanner 类，用于准备接收用户从控制台输入的数字，然后，声明变量 score 用于记载用户的得分。代码如下所示：

```
import java.util.Scanner;
public class AdditionTest {
```

```
public static void main(String[] args) {  
  
    System.out.println("将开始 10 次加法测试... ");  
    Scanner scanner = new Scanner(System.in);  
    //用于记载分数  
    int score = 0;  
  
}  
}
```

注意：此步骤中，需要导入 java.util 包下的 Scanner 类。

步骤三：构建循环

使用 for 语句，构建 10 次循环。

在循环中，使用 Math 类的 random 方法返回两个 0 到 1 之间的随机数（包含 0，不包含 1），然后将所产生的数值乘以 100，并取其整数部分，则可以生成两个 0 到 99 之间的随机整数（包含 0，也包含 99），表示加法题目中的两个加数。

计算加法运算的结果，并输出题目，代码如下：

```
import java.util.Scanner;  
public class AdditionTest {  
    public static void main(String[] args) {  
        System.out.println("将开始 10 次加法测试... ");  
        Scanner scanner = new Scanner(System.in);  
        //用于记载分数  
        int score = 0;  
  
        //构建 10 次循环  
        for (int i=1;i<=10;i++) {  
            //随机生成两个加数  
            int a = (int) (Math.random() * 100);  
            int b = (int) (Math.random() * 100);  
            int result = a + b;  
  
            //输出需要计算的加法表达式  
            System.out.println("(" + i + "). " + a + " + " + b + " = ?");  
            System.out.print("请输入答案 (输入-1退出): ");  
        }  
    }  
}
```

步骤四：判断录入

得到用户的录入后，进行判断：如果用户录入的为 -1，则需要提前退出循环；如果用户录入的答案错误，则提示“Error !”，并继续下一次答题；如果用户录入的答案正确，则提示“Correct !”，并计算用户的得分，然后继续下一轮的答题。用户得分的规则是，每答对一题，得 10 分（满分为 100 分）。

代码如下所示：

```
import java.util.Scanner;
public class AdditionTest {
    public static void main(String[] args) {
        System.out.println("将开始 10 次加法测试... ");
        Scanner scanner = new Scanner(System.in);
        //用于记载分数
        int score = 0;
        //构建 10 次循环
        for (int i=1;i<=10;i++) {
            //随机生成两个加数
            int a = (int) (Math.random() * 100);
            int b = (int) (Math.random() * 100);
            int result = a + b;

            //输出需要计算的加法表达式
            System.out.println("(" + i + "). " + a + " + " + b + " = ?");
            System.out.print("请输入答案 ( 输入-1 退出 ): ");

            //读入结果
            int answer = scanner.nextInt();
            //判断对错
            if(answer == -1){
                break;
            }else if (answer != result) {
                System.out.println("Error!");
                continue;
            } else {
                score += 10;
                System.out.println("Correct!");
            }
        }
    }
}
```

步骤五：输出结果

最后，需要将 scanner 对象关闭，并输出结果，代码如下所示：

```
import java.util.Scanner;
public class AdditionTest {
    public static void main(String[] args) {
        System.out.println("将开始 10 次加法测试... ");
        Scanner scanner = new Scanner(System.in);
        //用于记载分数
        int score = 0;
        //构建 10 次循环
        for (int i=1;i<=10;i++) {
            //随机生成两个加数
            int a = (int) (Math.random() * 100);
            int b = (int) (Math.random() * 100);
            int result = a + b;

            //输出需要计算的加法表达式
            System.out.println("(" + i + "). " + a + " + " + b + " = ?");
            System.out.print("请输入答案 ( 输入-1 退出 ): ");
        }
    }
}
```

```
//读入结果
int answer = scanner.nextInt();
//判断对错
if(answer == -1){
    break;
}else if (answer != result) {
    System.out.println("Error!");
    continue;
} else {
    score += 10;
    System.out.println("Correct!");
}

scanner.close();
System.out.println("此次测验结束，你的分数是：" + score);

}
}
```

- **完整代码**

本案例的完整代码如下所示：

```
import java.util.Scanner;
public class AdditionTest {
    public static void main(String[] args) {
        System.out.println("将开始 10 次加法测试... ");
        Scanner scanner = new Scanner(System.in);
        //用于记载分数
        int score = 0;
        //构建 10 次循环
        for (int i=1;i<=10;i++) {
            //随机生成两个加数
            int a = (int) (Math.random() * 100);
            int b = (int) (Math.random() * 100);
            int result = a + b;

            //输出需要计算的加法表达式
            System.out.println("(" + i + "). " + a + " + " + b + " = ?");
            System.out.print("请输入答案（输入-1 退出）： ");
            //读入结果
            int answer = scanner.nextInt();
            //判断对错
            if(answer == -1){
                break;
            }else if (answer != result) {
                System.out.println("Error!");
                continue;
            } else {
                score += 10;
                System.out.println("Correct!");
            }
        }
        scanner.close();
        System.out.println("此次测验结束，你的分数是：" + score);
    }
}
```

课后作业

1. 请描述 while 语句的执行流程

while 语句的语法格式如下：

```
while(表达式) {
    语句块
}
```

2. 指出下列程序运行后的情况

```
public static void main(String[] args) {
    int count=0;
    while(count<5);
    {
        System.out.print(count+" ");
        count++;
    }
}
```

3. 指出下列代码运行后的结果

```
public static void main(String[] args) {
    int count=10;
    do
    {
        System.out.print(count+" ");
        count++;
    }while(count<5);
}
```

4. 请简述 while 语句和 do-while 语句的区别

5. 请简述 for 语句的执行流程

for 语句的语法格式如下：

```
for(表达式1;表达式2;表达式3) {
    语句块
}
```

6. 指出下列代码中，for 循环的各个表达式是否正确

```
public static void main(String[] args) {
    for(int i=0,j=0,k=0;i<10&&j<4&&k<10;j+=2){
        System.out.println(i++);
        System.out.println(++k+i);
    }
```


7. 指出下列代码的运行结果

```
public static void main(String[] args) {
    for (int i = 0; i < 10; i++) {
        System.out.println("i=" + i);
        if (i == 2) {
            break;
        }
    }
}
```

8. 指出下列代码的运行结果

```
public static void main(String[] args) {
    for (int i = 0; i < 4; i++) {
        if (i == 2) {
            continue;
        }
        System.out.print("i=" + i + " ");
    }
}
```

9. 数列求和

有数列为：9，99，999，...，9999999999。要求使用程序计算此数列的和，并在控制台输出结果。交互效果如图 - 1 所示。

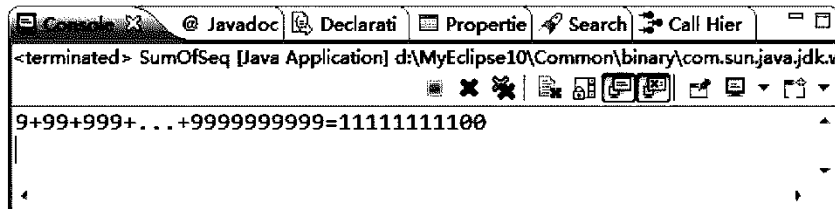


图- 1

另有数列：1+1/2+1/3...+1/n。要求使用交互的方式计算此数列的和：用户在控制台录入需要计算的整数 n 的值，程序计算此数列的和，并在控制台输出结果。程序的交互过程如图 - 2 所示。

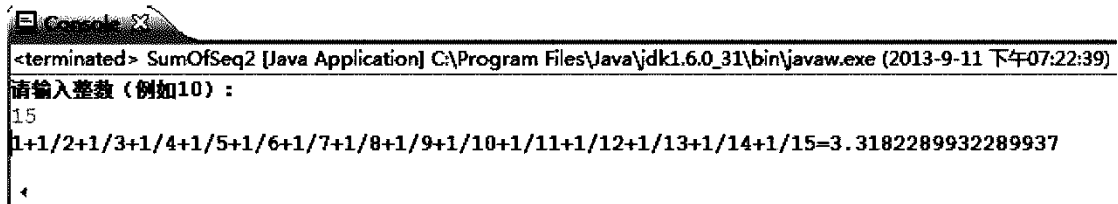


图- 2

Java 语言基础

Unit05

知识体系.....Page 109

循环结构	循环问题	循环问题
		循环问题定义——“当”循环
		循环问题定义——“直到”循环
		循环问题定义——固定次数循环
		逐步细化方式解决循环嵌套的问题
数组	什么是数组	什么是数组
	数组的定义	定义基本类型数组
	数组的初始化	初始化数组
	数组的访问	获取数组的长度
		通过下标访问数组元素
		遍历数组元素
	数组的复制	System.arraycopy 方法用于数组复制
		Arrays.copyOf 方法用于数组复制
		数组的扩容
	数组排序	数组的排序
		数组冒泡排序算法
		Arrays.sort 方法用于数组排序

经典案例.....Page 117

九九乘法表	逐步细化方式解决循环嵌套的问题
求数组元素的最大值	定义基本类型数组
	初始化数组
	获取数组的长度
	通过下标访问数组元素
	遍历数组元素
求数组元素的最大值放在最后一位	System.arraycopy 方法用于数组复制
	Arrays.copyOf 方法用于数组复制
	数组的“扩容”
冒泡排序算法实现	数组冒泡排序算法

1. 循环结构

1.1. 循环问题

1.1.1. 【循环问题】循环问题

<hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/>	<div style="text-align: right;">Tarena 达内科技</div> <h4>循环问题</h4> <ul style="list-style-type: none"> 需要多次重复执行一个或多个任务的问题考虑使用循环来解决； 一般情况下，for循环使用最多，对于for循环结构，一定要分析出需解决业务的3个部分： <ul style="list-style-type: none"> 循环变量初始状态 循环条件 循环变量的改变 <div style="text-align: center;"> </div> <div style="text-align: right;">+</div>
---	---


1.1.2. 【循环问题】循环问题定义——“当”循环

<hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/>	<div style="text-align: right;">Tarena 达内科技</div> <h4>循环问题定义——“当”循环</h4> <ul style="list-style-type: none"> 循环语句的选择 <ul style="list-style-type: none"> 如果业务可以转换为“当...”这样的句式时，优先选择while语句来实现 实例： <ul style="list-style-type: none"> 年存款利率为3%，本金为10000，存款总额超过12000时收益具体是多少？ 分析 <ul style="list-style-type: none"> 模板：“当”存款总额小于12000时，以3%利率增长 条件：total < 12000 写出代码：while(条件) { total += (total * 0.03); } <div style="text-align: right;">+</div>
---	---

1.1.3. 【循环问题】循环问题定义——“直到”循环


<hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/>	<div style="text-align: right;">Tarena 达内科技</div> <h4>循环问题定义——“直到”循环</h4> <ul style="list-style-type: none"> 循环语句的选择 <ul style="list-style-type: none"> 如果业务可以转换为“直到...”这样的句式时，优先选择do-while语句来实现 实例： <ul style="list-style-type: none"> 验证身份时必须提供密码并核对 分析 <ul style="list-style-type: none"> 模板：获取密码，“直到”输入的值为123456 条件：inputPwd != 123456 写出代码：do { } while(条件); <div style="text-align: right;">+</div>
---	--

1.1.4. 【循环问题】循环问题定义——固定次数循环



循环问题定义——固定次数循环


- 循环语句的选择
 - 如果业务中可以获取到一个确切的循环次数考虑for循环
- 例如：
 - 求1 - 100的和
 - 累加十次输入的数字的和
 - 录入三名学员的信息
 - 存储5门课程的成绩
 - ...



2. 数组


2.1. 什么是数组


2.1.1. 【什么是数组】什么是数组



什么是数组

- 程序 = 算法 + 数据结构；
- 前面学习的if、if-else、switch、循环解决的都是流程问题，即算法问题。
- 所谓数据结构，简单说就是把数据按照特定的某种结构来保存，设计合理的数据结构是解决问题的前提。
- 数组就是最基本的一种数据结构。





什么是数组（续1）

- 相同数据类型的元素组成的集合
- 元素按线性顺序排列。所谓线性顺序是指除第一个元素外，每一个元素都有唯一的前驱元素；除最后一个元素外，每一个元素都有唯一的后继元素（“一个跟一个”）
- 可以通过元素所在位置的顺序号（下标）做标识来访问每一个元素（下标从0开始，最大到元素个数-1）

a[0]	a[1]	a[2]	a[n-1]
10	23	30	

2.2. 数组的定义

2.2.1. 【数组的定义】定义基本类型数组

Tarena
达内科技

定义基本类型数组

- 声明数组的语法：
数据类型[] 数组名 = new 数据类型 [大小] ;
- 例： `int [] arr = new int [10] ;`

数组类型，
int[]表示数
组中的每一
个元素都是
int类型。

数组类型变
量（引用）。

数组的长度，即
数组中元素的个
数。

+

Tarena
达内科技

定义基本类型数组（续1）

- 定义基本类型数组的要点：
 - 确切的数据类型
 - 整体的数组名字
 - 不能缺少的 “[] ”
- 执行new语句才使得数组分配到了指定大小的空间
- `int [] arr` 与 `int arr []` 两种写法均可
- 声明数组时不规定数组长度，new关键字分配空间时需指定分配的空间大小

+

2.3. 数组的初始化

2.3.1. 【数组的初始化】初始化数组

Tarena
达内科技

初始化数组

- 基本类型的数组创建后，其元素的初始值：byte、short、char、int、long为0；float和double为0.0；boolean为false
- 可以在数组声明的同时对数组的元素进行初始化，例如：

```
int [ ] arr = { 10,23,30,-10,21 } ;
```

元素的个数即为数组的长度
- 此种写法只能用于声明时的初始化，不能用于赋值。如下面代码会有编译错误。

```
int [ ] arr;
arr = { 10,23,30,-10,21 } ;
```

+


代码编辑

初始化数组（续1）

• 可以通过下面的方式给已经声明的数组类型变量进行初始化：

```
int [] arr ;
arr = new int[] {10,23,30,-10,96} ;
```

注意：[]中不可以写长度，元素的个数就是数组的长度。



++

2.4. 数组的访问

2.4.1. 【数组的访问】获取数组的长度


代码编辑

获取数组的长度

• 调用数组的length属性可以获取数组的长度：

```
int[] arr = new int[] { 3,6,8,9 };
int len = arr . length ;
System.out.println( "数组长度为:" + len);
```

上述代码输出结果为：数组长度为:4



++

2.4.2. 【数组的访问】通过下标访问数组元素

代码编辑


通过下标访问数组元素

• 数组中的元素通过下标的方式进行访问：

注意：下标从0开始，最大到length-1



• 例如：



```
int[] arr = new int[] { 4,5,6,8};
int temp = arr [ 2 ]; // 获取第3个元素—6
// 交换数组下标为2和3的两个相邻元素的值
int temp = arr [ 2 ];
arr [ 2 ] = arr [ 3 ];
arr [ 3 ] = temp ;
// 交换后结果：4,5,8,6
```



++



2.4.3. 【数组的访问】遍历数组元素

<hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/>	<div style="text-align: right;">  </div> <h4>遍历数组元素</h4> <ul style="list-style-type: none"> 遍历数组元素，通常选择for循环语句，循环变量作为访问数组元素的下标，即可访问数组中的每一个元素 <pre>int[] arr = new int[10]; for (int i = 0; i < arr.length; i++){ arr[i] = 100; }</pre> <ul style="list-style-type: none"> 注意：循环的计数器的变化范围从0~length- 1 <div style="text-align: right;">  </div>
---	--

<hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/>	<div style="text-align: right;">  </div> <h4>遍历数组元素（续1）</h4> <ul style="list-style-type: none"> 遍历数组元素，正序输出 <pre>int [] arr = new int [] {10,20,30,40,50}; for (int i=0; i< arr.length; i++){ System.out.println (arr[i]); }</pre> <ul style="list-style-type: none"> 遍历数组元素，逆序输出 <pre>int [] arr = new int [] {10,20,30,40,50}; for (int i = (arr.length -1); i >= 0; i--){ System.out.println (arr[i]); }</pre> <div style="text-align: right;">  </div>
---	---

2.5. 数组的复制

2.5.1. 【数组的复制】System.arraycopy 方法用于数组复制

<hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/>	<div style="text-align: right;">  </div> <h4>System.arraycopy方法用于数组复制</h4> <ul style="list-style-type: none"> 使用System.arraycopy()方法可以实现数组的复制 <pre>public static void arraycopy(Object src, int srcPos, Object dest, int destPos, int length)</pre> <table border="0"> <tr> <td style="padding-right: 20px;">src</td> <td>– 源数组</td> </tr> <tr> <td>srcPos</td> <td>– 源数组中的起始位置</td> </tr> <tr> <td>dest</td> <td>– 目标数组</td> </tr> <tr> <td>destPos</td> <td>– 目标数组中的起始位置</td> </tr> <tr> <td>length</td> <td>– 要复制的数组元素的数量</td> </tr> </table> <div style="text-align: right;">  </div>	src	– 源数组	srcPos	– 源数组中的起始位置	dest	– 目标数组	destPos	– 目标数组中的起始位置	length	– 要复制的数组元素的数量
src	– 源数组										
srcPos	– 源数组中的起始位置										
dest	– 目标数组										
destPos	– 目标数组中的起始位置										
length	– 要复制的数组元素的数量										

System.arraycopy方法用于数组复制(续1)
Tarena
达内科技

```
int[] a = { 10,20,30,40,50};
int[] a1 = new int[ 6];
System.arraycopy( a, 1, a1, 0, 4 );
```

a[0]	a[1]	a[2]	a[3]	a[4]
10	20	30	40	50

a1[0]	a1[1]	a1[2]	a1[3]	a1[4]	a1[5]
20	30	40	50	0	0

2.5.2. 【数组的复制】Arrays.copyOf 方法用于数组复制

Arrays.copyOf方法用于数组复制
Tarena
达内科技

- 使用java.util.Arrays类的copyOf方法可实现数组的复制
 类型[] newArray = Arrays.copyOf (类型[] original, int newLength);
- 特点：生成的新数组是原始数组的副本
 newLength小于源数组，则进行截取
 newLength大于源数组，则用0或 null进行填充
- 所以产生的新数组可以大于源数组的长度

```
int [] a = { 10,20,30,40,50};
int [] a1 = Arrays . copyOf ( a, 6 );
```

a1数组元素为： 10 20 30 40 50 0

2.5.3. 【数组的复制】数组的扩容

数组的扩容
Tarena
达内科技

- 数组的长度在创建后是不可改变的，所谓扩容是指创建一个更大的新数组并将原有数组的内容复制到其中。
- 可以通过Arrays.copyOf()方法，简便实现数组的扩展。


```
int [] a = { 10,20,30,40,50};
a = Arrays . copyOf ( a, a.length+1 );
```

扩展后的数组
原数组
扩展后的长度

输出a数组元素：10,20,30,40,50,0

2.6. 数组排序

2.6.1. 【数组排序】数组的排序




数组的排序

- 排序是对数组施加的最常用的算法；
- 所谓排序，是指将数组元素按照从小到大或从大到小的顺序重新排列；
- 对于元素较多的数组，排序算法的优劣至关重要；
- 一般情况下，通过排序过程中数组元素的交换次数来衡量排序算法的优劣；
- 常用的排序算法有：插入排序、冒泡排序、快速排序等。

+


2.6.2. 【数组排序】数组冒泡排序算法



数组冒泡排序算法

- 冒泡排序的原则：比较相邻的元素，如果违反最后的顺序准则，则交换
- 可以简化理解为：
 - 第一次找到所有元素中最大的放在最后一个位置上，不再变动；
 - 第二次找到剩余所有元素中最大的放在倒数第二个位置上，不再变动；
 - 以此类推，直到排序完成。
- 比较时既可以采用“下沉”的方式，也可以使用“上浮”的方式实现。

+



数组冒泡排序算法（续1）

初始序列	89	50	84	57	61	20	86
		└───┘ └───┘ └───┘					
第1趟排序结果	50	84	57	61	20	86	89
		└───┘ └───┘ └───┘					
第2趟排序结果	50	57	61	20	84	86	89
		└───┘ └───┘					
第3趟排序结果	50	57	20	61	84	86	89
		└───┘					
第4趟排序结果	50	20	57	61	84	86	89
		└───┘					
第5趟排序结果	20	50	57	61	84	86	89

+

2.6.3. 【数组排序】Arrays.sort 方法用于数组排序

知识讲解

Arrays.sort方法用于数组排序

- JDK提供的Arrays.sort () 方法封装了数组的排序算法：

```
int[] arr = { 49, 81, 1, 64, 77, 50, 0, 54, 77, 18 };  
Arrays.sort ( arr );  
for( int i=0; i<arr.length; i++) {  
    System.out.println( arr[i] );  
}
```

输出结果 : 0 1 18 49 50 54 64 77 77 81

经典案例

1. 九九乘法表

• 问题

在界面打印九九乘法表，效果如图 - 1 所示：

```
<terminated> MultiplicationTable [Java Application] C:\Program Files\Java\jdk1.6.0_31\bin\java
1*1=1
1*2=2 2*2=4
1*3=3 2*3=6 3*3=9
1*4=4 2*4=8 3*4=12 4*4=16
1*5=5 2*5=10 3*5=15 4*5=20 5*5=25
1*6=6 2*6=12 3*6=18 4*6=24 5*6=30 6*6=36
1*7=7 2*7=14 3*7=21 4*7=28 5*7=35 6*7=42 7*7=49
1*8=8 2*8=16 3*8=24 4*8=32 5*8=40 6*8=48 7*8=56 8*8=64
1*9=9 2*9=18 3*9=27 4*9=36 5*9=45 6*9=54 7*9=63 8*9=72 9*9=81
```

图 - 1

• 方案

此案例需要使用嵌套循环来实现。

分析图 - 1 可以看出，九九乘法表一共需要输出九行数据，如图 - 2 所示：

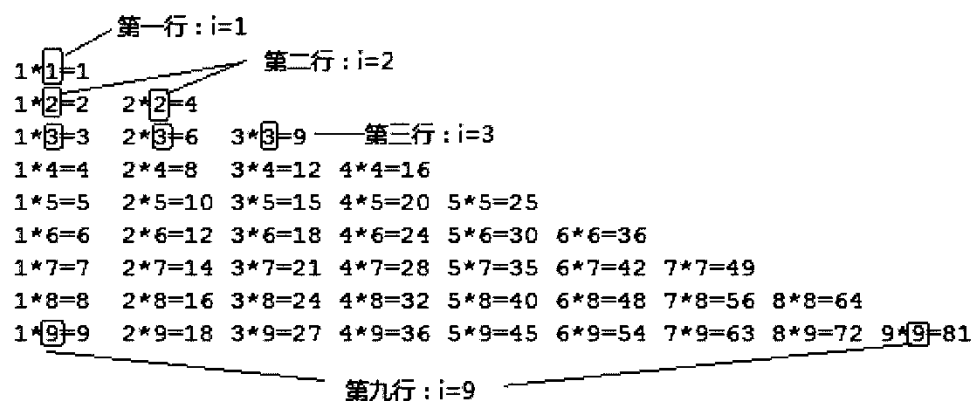


图 - 2

由图 - 2 可以看出，需要使用一个 for 循环来控制输出的行数。代码如下所示：

```
//i 变量用于控制行数
for (int i = 1; i <= 10; i++){
```

分析图 - 2 中的每行，可以看出，每行中的乘法表达式的个数正好和行数相同。每个乘法表达式中，第一个乘数从 1 开始，乘到最大值（当前行数），而另一个乘数正好是行数，如图 - 3 所示：

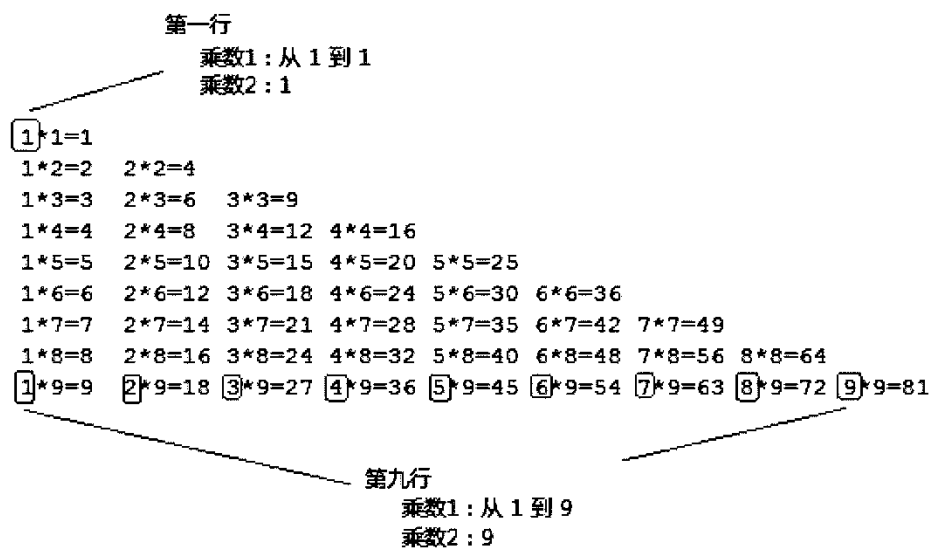


图 - 3

因此，在输出每行中的内容时，还需要使用一个 for 循环来控制每行中输出的表达式的个数。如果当前行为第 9 行，则循环的代码如下所示：

```
//假设当前行为第 9 行
int i = 9;
for (int j = 1; j <= i; j++) {
    System.out.print( j + "*" + i + "=" + j*i + "\t");
}
```

因为行数并不固定，而是从第一行到第九行，因此，需要将两个循环嵌套起来，代码如下所示：

```
//i 变量用于控制行数
for (int i = 1; i < 10; i++) {
    //j 变量用于控制每行中参与计算的最大数值：与行数相等
    for (int j = 1; j <= i; j++) {
        System.out.print( j + "*" + i + "=" + j*i + "\t");
    }
    //每行输出完毕后，需要换行
    System.out.println();
}
```

• 步骤

实现此案例需要按照如下步骤进行。

步骤一：定义类及 main 方法

首先定义一个名为 `MultiplicationTable` 的类,并在类中添加 Java 应用程序的主方法 `main`,代码如下所示:

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

步骤二：构建循环

在 `main` 方法中,构建两层嵌套的 `for` 循环:外层循环用于控制行,内层循环用于控制某行上的乘法表达式。需要注意的是,每行输出完毕后,需要换行。代码如下所示:

```
public class MultiplicationTable {
    public static void main(String[] args) {

        //i 变量用于控制行数
        for (int i = 1; i < 10; i++) {
            //j 变量用于控制每行中参与计算的最大数值:与行数相等
            for (int j = 1; j <= i; j++) {

            }
            //每行输出完毕后,需要换行
            System.out.println();
        }

    }
}
```

步骤三：输出乘法表

考虑到输出界面的美观性,使用 “\t” 进行排版对齐代码如下所示:

```
public class MultiplicationTable {
    public static void main(String[] args) {
        //i 变量用于控制行数
        for (int i = 1; i < 10; i++) {
            //j 变量用于控制每行中参与计算的最大数值:与行数相等
            for (int j = 1; j <= i; j++) {

                //设置输出的格式,控制排版对齐
                System.out.print( j + "*" + i + "=" + j*i + "\t");

            }
            //每行输出完毕后,需要换行
            System.out.println();
        }
    }
}
```

上述代码中的 “\t” 是水平制表符,其作用为从行首开始,每 8 字节算一个制表位,“\t” 会在当前内容结束后第一个空的制表位处连接上下文。

- **完整代码**

本案例的完整代码如下所示：

```
public class MultiplicationTable {
    public static void main(String[] args) {
        //i 变量用于控制行数
        for (int i = 1; i < 10; i++) {
            //i 变量用于控制每行中参与计算的最大数值：与行数相等
            for (int j = 1; j <= i; j++) {
                //设置输出的格式，使用"\t"控制排版对齐
                System.out.print(j + "*" + i + "=" + j*i + "\t");
            }
            //每行输出完毕后，需要换行
            System.out.println();
        }
    }
}
```

2. 求数组元素的最大值

- **问题**

创建程序，实现查询数组中最大值的功能，需求为：创建一个长度为 10 的数组，数组内放置 10 个 0 到 99 之间（包含 0，包含 99）的随机整数作为数组内容，要求查询出数组中的最大值，并打印显示在界面上，界面效果如图 - 4 所示：

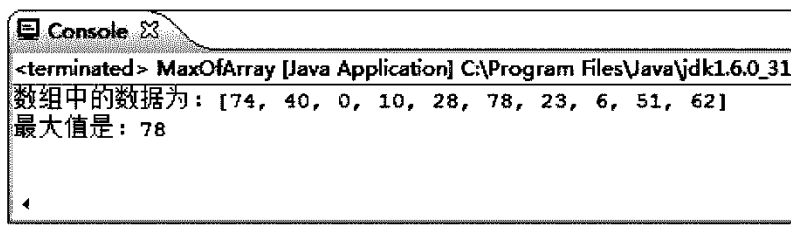


图 - 4

- **方案**

首先，此案例中，首先需要创建一个长度为 10 的整型数组，然后使用 for 循环来产生 10 个 0 到 99 之间的随机整数，并放入数组；然后查询数组中的最大值，并打印显示结果。

- **步骤**

实现此案例需要按照如下步骤进行。

步骤一：定义类及 main 方法

首先定义一个名为 MaxOfArray 的类，并在类中添加 Java 应用程序的主方法 main，代码如下所示：

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

步骤二：创建数组

在 main 方法中创建一个长度为 10 的数组，代码如下：

```
public class MaxOfArray {
    public static void main(String[] args) {

        //创建一个 10 个长度的数组
        int[] arr = new int[10];

    }
}
```

步骤三：数组的赋值

使用 for 语句构建一个 10 次的循环，在每次循环中，随机产生一个 0 到 99 之间的整数，并存入数组。

此案例中，使用 Random 类的 nextInt()方法产生随机数。代码如下所示：

```
import java.util.Random;

public class MaxOfArray {
    public static void main(String[] args) {
        //创建一个 10 个长度的数组
        int[] arr = new int[10];

        //随机生成 10 个 0 - 99 之间的数值，放入数组
        Random ran = new Random();
        for(int i=0;i<arr.length;i++){
            arr[i] = ran.nextInt(100);
        }

    }
}
```

注意：此步骤中，需要导入 java.util 包下的 Random 类。

步骤四：打印数组内容

数组赋值后，为方便用户查看数组中的数据内容，需要将数组中的数据打印在界面上。因此，需要使用 Arrays 类的 toString 方法来得到数组的内容。代码如下所示：

```
import java.util.Random;

import java.util.Arrays;
```



```
public class MaxOfArray {
    public static void main(String[] args) {
        //创建一个 10 个长度的数组
        int[] arr = new int[10];

        //随机生成 10 个 0 - 99 之间的数值，放入数组
        Random ran = new Random();
        for(int i=0;i<arr.length;i++){
            arr[i] = ran.nextInt(100);
        }

        //打印数组中的数据
        System.out.println("数组中的数据为：" + Arrays.toString(arr));

    }
}
```

注意：此步骤中，需要导入 java.util 包下的 Arrays 类。

步骤五：查询最大值

为找到数组中的最大值，依然需要使用循环来遍历数组。先定义一个变量 max 用于表示最大值，并赋初始值为数组中的第一个元素；然后依次把数组中的元素与变量 max 进行数值比较，如果数组中的某元素的数值大于 max，则将该元素的数值存入变量 max。代码如下所示：

```
import java.util.Random;
import java.util.Arrays;

public class MaxOfArray {
    public static void main(String[] args) {
        //创建一个 10 个长度的数组
        int[] arr = new int[10];

        //随机生成 10 个 0 - 99 之间的数值，放入数组
        Random ran = new Random();
        for(int i=0;i<arr.length;i++){
            arr[i] = ran.nextInt(100);
        }

        //打印数组中的数据
        System.out.println("数组中的数据为：" + Arrays.toString(arr));

        //查询最大值
        int max = arr[0];
        for(int i=1; i<arr.length; i++) {
            if(max < arr[i]) {
                max = arr[i];
            }
        }
        System.out.println("最大值是：" + max);

    }
}
```

• 完整代码

本案例的完整代码如下所示：

```
import java.util.Random;
import java.util.Arrays;

public class MaxOfArray {
    public static void main(String[] args) {
        //创建一个 10 个长度的数组
        int[] arr = new int[10];

        //随机生成 10 个 0—99 之间的数值，放入数组
        Random ran = new Random();
        for(int i=0;i<arr.length;i++){
            arr[i] = ran.nextInt(100);
        }
        //打印数组中的数据
        System.out.println("数组中的数据为: " + Arrays.toString(arr));

        //查询最大值
        int max = arr[0];
        for(int i=1; i<arr.length; i++) {
            if(max < arr[i]) {
                max = arr[i];
            }
        }
        System.out.println("最大值是: " + max);
    }
}
```

3. 求数组元素的最大值放在最后一位

• 问题

修改上一个案例中的程序，为程序添加功能：将数组的长度扩容为 11，然后将查询到的数组最大值作为数组的最后一个元素，并打印扩容后的数组的内容。界面效果如图 - 5 所示：

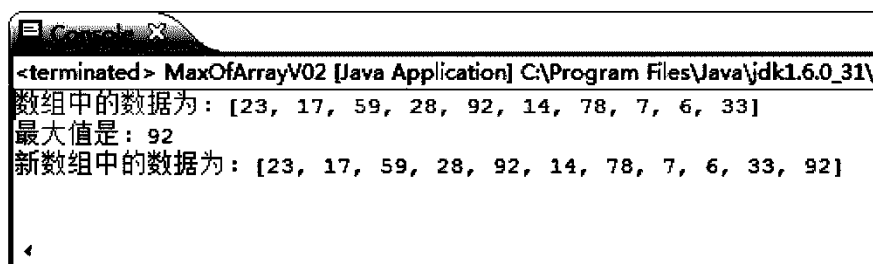


图 - 5

• 方案

实现此案例，只需要在上一个案例的基础上添加功能即可。

可以使用 Arrays 类的 copyOf() 方法实现数组的拷贝以及扩容，代码如下所示：

```
arr = Arrays.copyOf(arr, arr.length + 1);
```

如果原数组 `arr` 的长度为 10 ,那么 ,上述代码将产生一个长度为 11 的新数组 `arr` ,且新数组中的前 10 个数值和原数组 `arr` 中的 10 个数值相同。

然后 ,将查询到的最大值放入新数组的最后一个位置上 ,并打印新数组的内容即可。

• 步骤

实现此案例需要按照如下步骤进行。

步骤一：创建新数组

在上个案例中的 `main` 方法中继续添加代码 ,创建新数组 ,代码如下所示 :

```
import java.util.Random;
import java.util.Arrays;

public class MaxOfArray {
    public static void main(String[] args) {
        //创建一个 10 个长度的数组
        int[] arr = new int[10];

        //随机生成 10 个 0 - 99 之间的数值 ,放入数组
        Random ran = new Random();
        for(int i=0;i<arr.length;i++){
            arr[i] = ran.nextInt(100);
        }
        //打印数组中的数据
        System.out.println("数组中的数据为：" + Arrays.toString(arr));

        //查询最大值
        int max = arr[0];
        for(int i=1; i<arr.length; i++) {
            if(max < arr[i]) {
                max = arr[i];
            }
        }
        System.out.println("最大值是：" + max);

        //创建新数组
        arr = Arrays.copyOf(arr, arr.length + 1);

    }
}
```

步骤二：存储最大值并打印数组内容

将查询到的数组最大值放入新数组的最后一个位置 ,并打印数组内容显示。代码如下所示 :

```
import java.util.Random;
import java.util.Arrays;

public class MaxOfArray {
```

```
public static void main(String[] args) {
    //创建一个 10 个长度的数组
    int[] arr = new int[10];

    //随机生成 10 个 0 - 99 之间的数值，放入数组
    Random ran = new Random();
    for(int i=0;i<arr.length;i++){
        arr[i] = ran.nextInt(100);
    }
    //打印数组中的数据
    System.out.println("数组中的数据为：" + Arrays.toString(arr));

    //查询最大值
    int max = arr[0];
    for(int i=1; i<arr.length; i++) {
        if(max < arr[i]) {
            max = arr[i];
        }
    }
    System.out.println("最大值是：" + max);

    //创建新数组
    arr = Arrays.copyOf(arr, arr.length + 1);

    //最大值放入最后一个位置，并打印新数组内容
    arr[arr.length-1] = max;
    System.out.println("新数组中的数据为：" + Arrays.toString(arr));

}
}
```

• 完整代码

本案例的完整代码如下所示：

```
import java.util.Random;
import java.util.Arrays;

public class MaxOfArray {
    public static void main(String[] args) {
        //创建一个 10 个长度的数组
        int[] arr = new int[10];

        //随机生成 10 个 0—99 之间的数值，放入数组
        Random ran = new Random();
        for(int i=0;i<arr.length;i++){
            arr[i] = ran.nextInt(100);
        }
        //打印数组中的数据
        System.out.println("数组中的数据为：" + Arrays.toString(arr));

        //查询最大值
        int max = arr[0];
        for(int i=1; i<arr.length; i++) {
            if(max < arr[i]) {
                max = arr[i];
            }
        }
    }
}
```

```
System.out.println("最大值是: " + max);

//创建新数组
arr = Arrays.copyOf(arr, arr.length + 1);
//最大值放入最后一个位置,并打印新数组内容
arr[arr.length-1] = max;
System.out.println("新数组中的数据为: " + Arrays.toString(arr));
}
}
```

4. 冒泡排序算法实现

• 问题

冒泡排序 (Bubble Sort), 是一种较简单的排序算法。在冒泡排序算法中, 需要重复的走访要排序的数列, 一次比较两个元素, 如果它们的大小顺序错误就把它交换过来。走访数列的工作是重复地进行直到没有再需要交换的元素, 也就是说该数列已经排序完成。由于在排序过程中总是小数往前放, 大数往后放, 相当于气泡往上升, 所以称作冒泡排序。

本案例要求使用冒泡排序算法实现对数组的排序。有一个长度为 10 的整型数组, 使用冒泡排序算法将数组按照升序排列, 并输出排序的过程以及结果。程序交互情况如图-6 所示:

```
<terminated> BubbleSort [Java Application] C:\Program Files\
[8, 54, 17, 11, 97, 68, 72, 75, 22, 75]
-----冒泡排序开始-----
[8, 17, 11, 54, 68, 72, 75, 22, 75, 97]
[8, 11, 17, 54, 68, 72, 22, 75, 75, 97]
[8, 11, 17, 54, 68, 22, 72, 75, 75, 97]
[8, 11, 17, 54, 22, 68, 72, 75, 75, 97]
[8, 11, 17, 22, 54, 68, 72, 75, 75, 97]
[8, 11, 17, 22, 54, 68, 72, 75, 75, 97]
[8, 11, 17, 22, 54, 68, 72, 75, 75, 97]
[8, 11, 17, 22, 54, 68, 72, 75, 75, 97]
[8, 11, 17, 22, 54, 68, 72, 75, 75, 97]
-----冒泡排序结束-----
[8, 11, 17, 22, 54, 68, 72, 75, 75, 97]|
4
```

图 - 6

• 方案

为了理解冒泡排序算法, 我们先假设有一个长度为 7 的数组, 数组内容如图 - 7 所示:

初始顺序	89	50	84	57	61	20	70
------	----	----	----	----	----	----	----

图 - 7

图 - 7 中的数组元素为无序状态, 为实现升序排列, 可以先进行第一轮比较, 过程大致如下:

1. 先比较第一对相邻的元素 (第一个位置的 89 和第二个位置的 50): 如果第一个比

第二个大，就交换两个数值；

2. 继续比较第二对相邻的元素（第二个位置和第三个位置）：如果第二个位置上的数值大于第三个位置上的数值，则交换；

3. 继续下去，重复上述的比较工作，直到结尾的最后一对；此时，一轮比较交换完成后，最后的元素则是数列中最大的数。

第一轮比较的过程如图 - 8 所示：



图 - 8

由图 - 8 可以看出，第一轮比较后，已经将数组中的最大值通过位置交换，移动到数组的最后位置上。但是，其余的元素依然为无序状态，因此，依然进行第二轮比较：针对除最后一个元素外的其他元素重复以上步骤，以找到剩余元素中的最大数，且放置到倒数第二的位置上。第二轮比较的过程如图 - 9 所示：



图 - 9

由图 - 9 可以看出，第一个位置上的 50 比第二个位置上的 84 小，因此不发生交换；而其他位置将依次发生交换，从而将 84 交换到倒数第二的位置上。

然后，继续进行其他轮比较，持续进行，每次对越来越少的元素重复上面的步骤，直到没有任何一对数字需要比较，则排序完成。整个排序的过程如图 - 10 所示：



图 - 10

由图 - 10 可以看出，实现冒泡排序算法时，需要使用嵌套的两个循环来实现：外层循环用于控制排序的轮次，里层循环用于控制每轮排序中的两两交换。

- **步骤**

实现此案例需要按照如下步骤进行。

步骤一：定义类及 main 方法

首先定义一个名为 BubbleSort 的类，并在类中添加 Java 应用程序的主方法 main，代码如下所示：

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

步骤二：创建数组

在 main 方法中创建一个长度为 10 的数组，并使用 for 语句构建一个 10 次的循环，在每次循环中，随机产生一个 0 到 99 之间的整数，并存入数组，然后，打印数组的内容显示在界面上。

此案例中，使用 Random 类的 nextInt()方法产生随机数。代码如下所示：

```
import java.util.Random;
import java.util.Arrays;

public class BubbleSort {
    public static void main(String[] args) {

        //创建数组
        int[] arr = new int[10];
        Random ran = new Random();
        for (int i = 0; i < arr.length; i++) {
            arr[i] = ran.nextInt(100);
        }
        System.out.println(Arrays.toString(arr));

    }
}
```

注意：此步骤中，需要导入 java.util 包下的 Random 类和 Arrays 类。

步骤三：排序

使用冒泡排序算法对数组进行排序：每一轮比较中，两两相比，找到最大的数值移动到最后，直到都比较过一遍。为便于查看排序的过程，将每轮比较后的数组内容输出显示，并将最后的结果输出到控制台。代码如下所示：

```
import java.util.Random;
import java.util.Arrays;

public class BubbleSort {
    public static void main(String[] args) {
        //创建数组
        int[] arr = new int[10];
        Random ran = new Random();
```

```

    for (int i = 0; i < arr.length; i++) {
        arr[i] = ran.nextInt(100);
    }
    System.out.println(Arrays.toString(arr));

    // 冒泡排序
    System.out.println("-----冒泡排序 开始-----");
    for (int i = 0; i < arr.length - 1; i++) {
        for (int j = 0; j < arr.length - i - 1; j++) {
            if (arr[j] > arr[j + 1]) {
                int t = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = t;
            }
        }
        System.out.println(Arrays.toString(arr));
    }
    System.out.println("-----冒泡排序 结束-----");
    System.out.println(Arrays.toString(arr));

}
}

```

• 完整代码

本案例的完整代码如下所示：

```

import java.util.Random;
import java.util.Arrays;

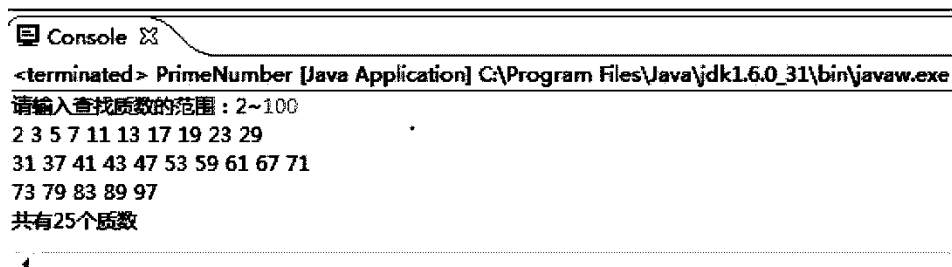
public class BubbleSort {
    public static void main(String[] args) {
        //创建数组
        int[] arr = new int[10];
        Random ran = new Random();
        for (int i = 0; i < arr.length; i++) {
            arr[i] = ran.nextInt(100);
        }
        System.out.println(Arrays.toString(arr));
        // 冒泡排序
        System.out.println("-----冒泡排序 开始-----");
        for (int i = 0; i < arr.length - 1; i++) {
            for (int j = 0; j < arr.length - i - 1; j++) {
                if (arr[j] > arr[j + 1]) {
                    int t = arr[j];
                    arr[j] = arr[j + 1];
                    arr[j + 1] = t;
                }
            }
            System.out.println(Arrays.toString(arr));
        }
        System.out.println("-----冒泡排序 结束-----");
        System.out.println(Arrays.toString(arr));
    }
}

```


课后作业

1. 质数问题

本案例要求使用交互的方式找出从 2 开始到某个数值范围内的所有质数,并输出结果。因为输出的质数可能较多,要求分行输出,每行最多输出 10 个质数。程序的交互过程如图-1 所示:



```
<terminated> PrimeNumber [Java Application] C:\Program Files\Java\jdk1.6.0_31\bin\javaw.exe
请输入查找质数的范围: 2~100
2 3 5 7 11 13 17 19 23 29
31 37 41 43 47 53 59 61 67 71
73 79 83 89 97
共有25个质数
```

图- 1

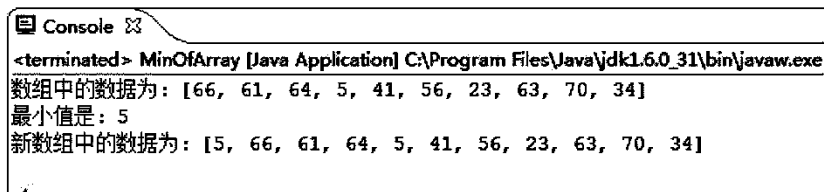
注意: 质数又称素数,指在大于 1 的自然数中,除了 1 和此整数自身外,不能被其它自然数整除的数。

2. int[]是一种数据类型吗?

3. 使用两种方式,初始化数组 arr 的长度为 4

4. 查询数组最小值,并将数组扩容形成新数组

创建程序,实现查询数组中最小值的功能,并将最小值放入数组的第一位。需求为:创建一个长度为 10 的数组,数组内放置 10 个 0 到 99 之间(包含 0,包含 99)的随机整数作为数组元素,要求查询出数组中的最小值,并打印显示在界面上。然后,将数组的长度扩容为 11,将查询到的数组最小值记载为数组的第一个元素,并打印扩容后的数组的内容。界面效果如图 - 2 所示:



```
<terminated> MinOfArray [Java Application] C:\Program Files\Java\jdk1.6.0_31\bin\javaw.exe
数组中的数据为: [66, 61, 64, 5, 41, 56, 23, 63, 70, 34]
最小值是: 5
新数组中的数据为: [5, 66, 61, 64, 5, 41, 56, 23, 63, 70, 34]
```

图- 2

5. 总结冒泡排序算法的原理

Java 语言基础

Unit06

知识体系.....Page 133

方法	方法（函数，过程）	方法（函数，过程）
	方法的定义	定义方法（函数，过程）的功能
		定义参数和返回值
	方法的调用	return 语句
		调用方法时的参数传递

经典案例.....Page 139

猜字母游戏——设计数据结构	
猜字母游戏——设计程序结构	
猜字母游戏——实现字母生成方法	
猜字母游戏——实现字母检测方法	
猜字母游戏——实现主方法	

课后作业.....Page 158

1. 方法

1.1. 方法（函数，过程）

1.1.1. 【方法（函数，过程）】方法（函数，过程）

Tarena
达内科技

方法（函数，过程）

- 各种语言都有方法的概念（有的语言称其为函数或过程）
- 方法用于封装一段特定的逻辑功能
如：执行计算或操作；
- 方法可以在程序中反复被调用；
- 方法可减少代码重复，便于程序的维护。

我需要 我也需要

排序算法

难道要我自己再写一次

+

1.2. 方法的定义

1.2.1. 【方法的定义】定义方法（函数，过程）的功能

Tarena
达内科技

定义方法（函数，过程）的功能

- 方法用于封装一个特定的功能。
- 定义方法的五个要素：修饰词、返回值类型、方法名、参数列表、方法体；

修饰词 返回值类型 方法名

```
public static int sum (int num1, int num2) {
    // 方法体
}
```

参数列表

+

1.2.2. 【方法的定义】定义参数和返回值


Tarena
达内科技

定义参数和返回值

- 方法的参数是指：在调用时传递给方法，需要被方法处理的数据；
- 方法可有参数也可以没有参数，有参可使方法处理更加灵活；
- 在方法定义时，需要声明该方法所需要的参数变量。
- 在方法调用时，会将实际的参数值传递给方法的参数变量。必须保证传递参数的类型和个数符合方法的声明。


```
void say() { }
void say( String name ) { }
int sum (int num1, int num2) { }
```

+


<hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/>	<div style="text-align: right;">  </div> <h3>定义参数和返回值（续1）</h3> <ul style="list-style-type: none"> • 方法调用结束后可以返回一个数据，称之为返回值。 • 方法在声明时必须指定返回值的类型。 <ul style="list-style-type: none"> - 若方法不需要返回数据，将返回值类型声明为void。 - 若方法需要返回数据，将返回值类型声明为特定数据类型。 <div style="text-align: right;">+</div>
---	---

1.3. 方法的调用

1.3.1. 【方法的调用】return 语句

<hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/>	<div style="text-align: right;">  </div> <h3>return语句</h3> <ul style="list-style-type: none"> • 可通过return语句返回，return语句的作用在于结束方法且将数据返回给调用方。 <div style="display: flex; justify-content: space-around; align-items: flex-start;"> <div style="border: 1px solid black; padding: 5px; text-align: center;"> return num1 + num2 ; </div> <div style="border: 1px solid black; padding: 5px; text-align: center;"> return ; </div> </div> <div style="display: flex; justify-content: space-around; margin-top: 10px;"> <div style="border: 1px solid black; padding: 5px; text-align: center; width: 40%;"> return语句返回该表达式的值。 </div> <div style="border: 1px solid black; padding: 5px; text-align: center; width: 40%;"> 对于返回值为void的方法也可以使用return语句，此时该语句的作用仅仅在于结束方法调用。 </div> </div> <div style="text-align: right;">+</div>
---	--

1.3.2. 【方法的调用】调用方法时的参数传递

<hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/>	<div style="text-align: right;">  </div> <h3>调用方法时的参数传递</h3> <p>定义方法：</p> <pre>public static int sum (int num1 , int num2){ }</pre> <p>main方法中调用：</p> <pre>int result = sum(5 , 6); int a = 50, b = 60; int result = sum (a , b);</pre> <p>定义方法：</p> <pre>public static void sayHi(String name) { }</pre> <p>main方法中调用：</p> <pre>sayHi ("wkj"); sayHi (" zhangsan");</pre> <div style="text-align: right;">+</div>
---	---

调用方法时的参数传递 (续1)




```
public static int max(int a, int b) { ... .. }
int a = 5; int b=6;
int myMax = max(a,b);
```

代码讲解




调用方法时的参数传递 (续2)




```
public static int max(int a, int b) { ... .. }
int a = 5; int b=6;
int myMax = max(a,b);
```

代码讲解

1) 为main方法中的变量a、b、myMax
分配空间并赋值。




调用方法时的参数传递 (续3)



```
public static int max(int a, int b) { ... .. }
int a = 5; int b=6;
int myMax = max(a,b);
```

代码讲解

1) 为main方法中的变量a、b、myMax
分配空间并赋值。



myMax	
b	6
a	5

}

Main
方法


图形标注

调用方法时的参数传递 (续4)

```

public static int max(int a, int b) { ... .. }
int a = 5; int b=6;
int myMax = max(a,b);
                    
```

- 1) 为main方法中的变量a、b、myMax 分配空间并赋值。
- 2) 调用方法max，为max方法的参数变量a，b分配空间。



myMax	
b	6
a	5

Main

方法

+


图形标注

调用方法时的参数传递 (续5)

```

public static int max(int a, int b) { ... .. }
int a = 5; int b=6;
int myMax = max(a,b);
                    
```

- 1) 为main方法中的变量a、b、myMax 分配空间并赋值。
- 2) 调用方法max，为max方法的参数变量a，b分配空间。



b	
a	

max

方法

myMax	
b	6
a	5

Main

方法

+


图形标注

调用方法时的参数传递 (续6)

```

public static int max(int a, int b) { ... .. }
int a = 5; int b=6;
int myMax = max(a,b);
                    
```

- 1) 为main方法中的变量a、b、myMax 分配空间并赋值。
- 2) 调用方法max，为max方法的参数变量a，b分配空间。
- 3) 将调用值传递到参数变量中。



b	
a	

max

方法

myMax	
b	6
a	5

Main

方法

+

代码讲解

调用方法时的参数传递 (续7)

```
public static int max(int a, int b) { ... .. }
int a = 5; int b=6;
int myMax = max(a,b);
```

- 1) 为main方法中的变量a、b、myMax 分配空间并赋值。
- 2) 调用方法max，为max方法的参数 变量a，b分配空间。
- 3) 将调用值传递到参数变量中。

b	6
a	5

myMax	
b	6
a	5

max
方法

Main
方法

++

代码讲解

调用方法时的参数传递 (续8)

```
public static int max(int a, int b) { ... .. }
int a = 5; int b=6;
int myMax = max(a,b);
```

- 1) 为main方法中的变量a、b、myMax 分配空间并赋值。
- 2) 调用方法max，为max方法的参数 变量a，b分配空间。
- 3) 将调用值传递到参数变量中。
- 4) max方法运行完返回，参数变量空间 释放。

b	6
a	5

myMax	
b	6
a	5

max
方法

Main
方法

++

代码讲解

调用方法时的参数传递 (续9)

```
public static int max(int a, int b) { ... .. }
int a = 5; int b=6;
int myMax = max(a,b);
```

- 1) 为main方法中的变量a、b、myMax 分配空间并赋值。
- 2) 调用方法max，为max方法的参数 变量a，b分配空间。
- 3) 将调用值传递到参数变量中。
- 4) max方法运行完返回，参数变量空间 释放。

myMax	
b	6
a	5

Main
方法

++

调用方法时的参数传递 (续10)

```
public static int max(int a, int b) { ... .. }
int a = 5; int b=6;
int myMax = max(a,b);
```

代码思路

- 1) 为main方法中的变量a、b、myMax分配空间并赋值。
- 2) 调用方法max，为max方法的参数变量a，b分配空间。
- 3) 将调用值传递到参数变量中。
- 4) max方法运行完返回，参数变量空间释放。
- 5) main方法中的myMax变量得到返回值。

myMax		
b	6	
a	5	

Main
方法



调用方法时的参数传递 (续11)

```
public static int max(int a, int b) { ... .. }
int a = 5; int b=6;
int myMax = max(a,b);
```

代码思路

- 1) 为main方法中的变量a、b、myMax分配空间并赋值。
- 2) 调用方法max，为max方法的参数变量a，b分配空间。
- 3) 将调用值传递到参数变量中。
- 4) max方法运行完返回，参数变量空间释放。
- 5) main方法中的myMax变量得到返回值。

myMax	6	
b	6	
a	5	

Main
方法



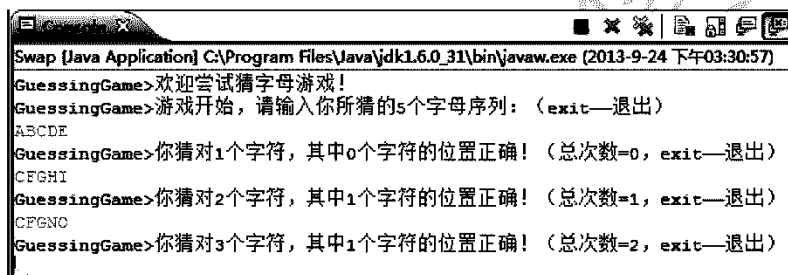
经典案例

1. 猜字母游戏——设计数据结构

• 问题

有猜字母游戏,其游戏规则为:程序随机产生 5 个按照一定顺序排列的字符作为猜测的结果,由玩家来猜测此字符串。玩家可以猜测多次,每猜测一次,则由系统提示结果。如果猜测的完全正确,则游戏结束,计算玩家的游戏得分并输出;如果没有猜对,则提示猜测的结果,如猜对了几个字符,以及猜对了几个字符的位置等信息,并提示玩家游戏继续。

本案例要求使用交互的方式实现此游戏:由玩家在控制台输入所猜测的字符串,如果所猜测的字符串与结果并不完全相同,则在界面输出比较后的结果,并提醒玩家继续猜测。交互过程如图 - 1 所示:

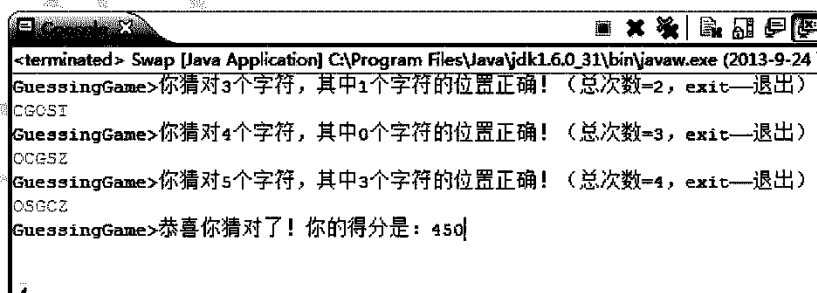


```
Swap [Java Application] C:\Program Files\Java\jdk1.6.0_31\bin\javaw.exe (2013-9-24 下午03:30:57)
GuessingGame>欢迎尝试猜字母游戏!
GuessingGame>游戏开始, 请输入你所猜的5个字母序列: (exit—退出)
ABCDE
GuessingGame>你猜对1个字符, 其中0个字符的位置正确! (总次数=0, exit—退出)
CFGHI
GuessingGame>你猜对2个字符, 其中1个字符的位置正确! (总次数=1, exit—退出)
CPGNC
GuessingGame>你猜对3个字符, 其中1个字符的位置正确! (总次数=2, exit—退出)
```

图 - 1

由图 - 1 可以看出,每次猜测后,程序将比较玩家所输入的字符串,比较字符以及字符的位置,然后提示结果:5 个字符中正确的字符个数,以及位置正确的字符个数,以便于玩家判断后续如何进行猜测。

玩家终于猜测正确后,游戏结束,并给出游戏得分,交互过程如图 - 2 所示:



```
<terminated> Swap [Java Application] C:\Program Files\Java\jdk1.6.0_31\bin\javaw.exe (2013-9-24
GuessingGame>你猜对3个字符, 其中1个字符的位置正确! (总次数=2, exit—退出)
CGGSI
GuessingGame>你猜对4个字符, 其中0个字符的位置正确! (总次数=3, exit—退出)
OCCSZ
GuessingGame>你猜对5个字符, 其中3个字符的位置正确! (总次数=4, exit—退出)
OSGCCZ
GuessingGame>恭喜你猜对了! 你的得分是: 450!
```

图 - 2

其中,游戏的得分规则为:字符的个数乘以 100 为总分,即此游戏的总分为 500 分。玩家如果第一次就猜对,则得满分(500 分);每多猜测一次,则扣 10 分。由图 - 2 可以看出,玩家共猜测了 5 次,因此,得分为 450。

最后,如果玩家在控制台录入 exit,则游戏中止,程序结束。交互过程如图 - 3 所示:

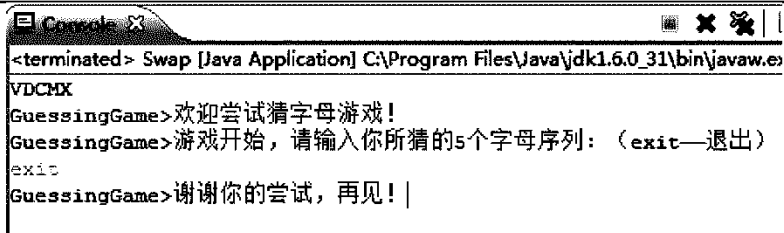


图- 3

本案例需要实现猜字母游戏中的数据结构设计,即,设计相关的数据结构,可以存储此程序中所用到的相关数据。

• 方案

分析猜字母游戏可以看出,此程序需要存储随机生成的字母个数、随机生成的字符串、玩家猜测的总次数、玩家录入的字符串,以及比较后的结果。因此,设计如下变量来存储此游戏中需要用到相关数据:

- int 类型变量 count:用于记录玩家猜字母的总次数;
- char 数组类型变量 input:用于保存用户猜测的数据。
- char 数组类型变量 chs:用于保存随机生成的多个字母所组成的字符串;
- int 类型数组变量 result:用于保存判断的结果。此数组有两个元素,第一个用于保存完全猜对的字母个数(字符正确且位置也正确),第二个元素用于保存猜对的字母个数(字符正确但位置不正确)。

• 步骤

实现此案例需要按照如下步骤进行。

步骤一：定义类及 main 方法

首先定义一个名为 GuessingGame 的类,并在类中添加 Java 应用程序的主方法 main,代码如下所示:

```
public class GuessingGame {
    public static void main(String[] args) {

    }
}
```

步骤二：存储猜测的次数

定义一个 int 类型变量 count,用于记录玩家猜字母的总次数,代码如下所示:

```
public class GuessingGame {
    public static void main(String[] args) {

        int count = 0;
    }
}
```

```
}  
}
```

步骤三：存储用户猜测的数据

char 数组类型变量 input：用于保存用户猜测的数据，代码如下所示：

```
public class GuessingGame {  
    public static void main(String[] args) {  
        int count = 0;  
  
        char[] input=null;  
  
    }  
}
```

步骤四：存储随机生成的多个字母

定义一个 char 数组类型 chs，用于保存随机生成的字母。代码如下所示：

```
public class GuessingGame {  
    public static void main(String[] args) {  
        int count = 0;  
        char[] input=null;  
  
        char[] chs =null;  
  
    }  
}
```

步骤五：存储比较结果

定义一个 int 数组类型变量 result，用于存储比较的结果。该数组有两个元素，第一个用于保存完全猜对的字母个数(字符和位置均正确)，第二个元素用于保存猜对的字母个数(字符正确，但是位置不正确)。代码如下所示：

```
public class GuessingGame {  
    public static void main(String[] args) {  
        int count = 0;  
        char[] input=null;  
        char[] chs =null;  
  
        int[] result =new int[2];  
  
    }  
}
```

• 完整代码

本案例的完整代码如下所示：

```
public class GuessingGame {  
    public static void main(String[] args) {  
        // 表示玩家猜测的次数
```

```
int count = 0;
//表示用户猜测的数据
char[] input=null;
// 表示猜测的字符串
char[] chs =null;
// 用于保存判断的结果
int[] result = new int[2];
}
}
```

2. 猜字母游戏——设计程序结构

- 问题

本案例需要实现猜字母游戏程序中的程序结构设计。

- 方案

分析猜字母游戏可以看出，程序首先需要随机产生 5 个不同的字母作为需要猜测的结果，因此，可以先定义一个方法，以实现此功能；其次，每当玩家猜测一次后，程序需要将玩家录入的字符串和正确答案进行比较，统计正确的字符个数以及正确的位置个数，因此，也可以先定义一个方法，专用于实现比较功能。这样，就可以在 main 方法中调用这两个方法。

- 步骤

实现此案例需要按照如下步骤进行。

步骤一：定义方法 generate()

在 GuessingGame 类中，定义方法 generate()，该方法用于随机生成五个不同的字母。代码如下所示：

```
import java.util.Scanner;

public class GuessingGame {
    public static void main(String[] args) {
        // 表示玩家猜测的次数
        int count = 0;
        //表示用户猜测的数据
        char[] input=null;
        // 表示猜测的字符串
        char[] chs =null;
        // 用于保存判断的结果
        int[] result = new int[2];
    }

    /**
     * 随机生成需要猜测的字母序列
     *
     * @return 存储随机字符的数组
     */
}
```

```

    */
    public static char[] generate() {
        char[] chs = new char[5];

        return chs;
    }

}

```

步骤二：定义方法 check()

在 `GuessingGame` 类中，定义方法 `check()`。该方法用于将玩家输入的多个字母（参数 `input`）和系统随机生成的多个字母（参数 `chs`）进行比较，统计正确的字符个数，以及位置正确的个数，并将结果存储到数组中，然后返回给调用方。代码如下所示：

```

import java.util.Scanner;

public class GuessingGame {
    public static void main(String[] args) {
        // 表示玩家猜测的次数
        int count = 0;
        //表示用户猜测的数据
        char[] input=null;
        // 表示猜测的字符串
        char[] chs =null;
        // 用于保存判断的结果
        int[] result = new int[2];
    }

    /**
     * 随机生成需要猜测的字母序列
     *
     * @return 存储随机字符的数组
     */
    public static char[] generate() {
        char[] chs = new char[5];

        return chs;
    }

    /**
     * 比较玩家输入的字母序列和程序所生成的字母序列，逐一比较字符及其位置，并记载比较结果
     *
     * @param chs
     *         程序生成的字符序列
     * @param input
     *         玩家输入的字符序列
     * @return 存储比较的结果。返回值 int 数组 的长度为 2，其中，索引为 0 的位置
     *         用于存放完全猜对的字母个数(字符和位置均正确)，索引为 1 的位置用于存放猜对的字
     *         母个数(字符正确，但是位置不正确)。
     */
    public static int[] check(char[] chs, char[] input) {
        int[] result = new int[2];

        return result;
    }
}

```

```
}
```

```
}
```

- **完整代码**

本案例的完整代码如下所示：

```
import java.util.Scanner;

public class GuessingGame {
    public static void main(String[] args) {
        // 表示玩家猜测的次数
        int count = 0;
        //表示用户猜测的数据
        char[] input=null;
        // 表示猜测的字符串
        char[] chs =null;
        // 用于保存判断的结果
        int[] result = new int[2];
    }

    /**
     * 随机生成需要猜测的字母序列
     *
     * @return 存储随机字符的数组
     */
    public static char[] generate() {
        char[] chs = new char[5];

        return chs;
    }

    /**
     * 比较玩家输入的字母序列和程序所生成的字母序列，逐一比较字符及其位置，并记载比较结果
     *
     * @param chs
     *         程序生成的字符序列
     * @param input
     *         玩家输入的字符序列
     * @return 存储比较的结果。返回值 int 数组 的长度为 2，其中，索引为 0 的位置
     *         用于存放完全猜对的字母个数(字符和位置均正确)，索引为 1 的位置用于存放猜对的字母个数(字符正确，但是位置不正确)。
     */
    public static int[] check(char[] chs, char[] input) {
        int[] result = new int[2];

        return result;
    }
}
```

3. 猜字母游戏——实现字母生成方法

- **问题**

实现猜字母游戏中的字母生成方法，即，随机生成 5 个不同的字母作为猜测的结果。

• 方案

实现 generate 方法，首先声明一个字符类型的数组，用于存储 26 个大写字母，然后声明一个 boolean 类型的数组，其长度也为 26。此数组中的初始值均为 false，意味着，程序起始，没有任何字母被选中。如果某个字母被选中，则同时设置该字母在 boolean 类型数组中对应位置上的值为 true，表示该字母被选中过。

然后，使用嵌套循环：外层循环用于控制所生成的字母个数，即，循环 5 次，以产生 5 个字母；而内层循环则用于判断所生成的字母是否重复。generate 方法的程序流程如图-4 所示。

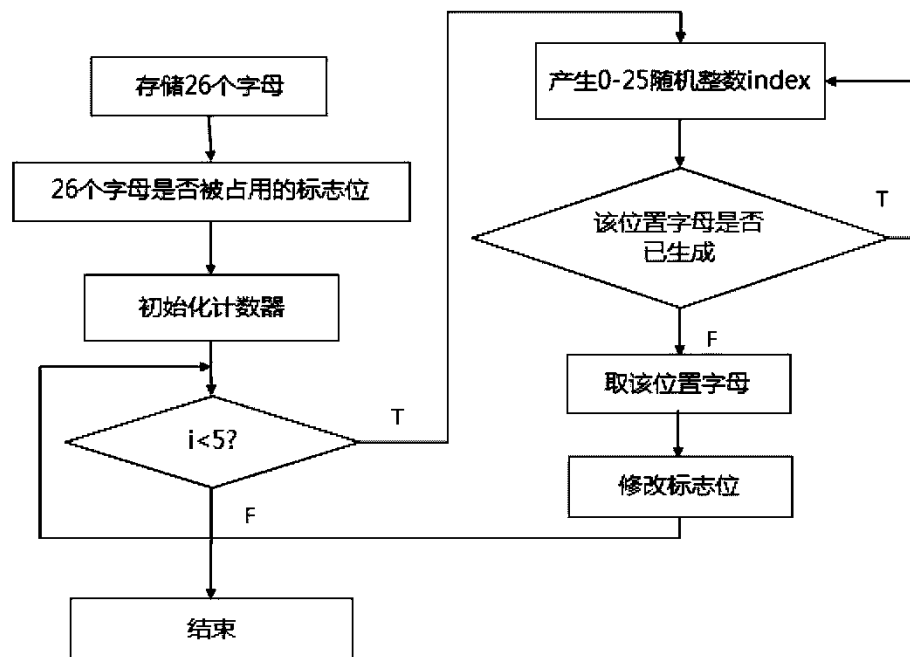


图- 4

• 步骤

实现此案例需要按照如下步骤进行。

步骤一：构建存储所有字母的数组

在 generate 方法中，首先定义 char 类型的数组变量 letters，用于存放 26 个大写字母，然后定义 boolean 类型的数组变量 flag，flag 数组的大小和 letters 数组的大小相同，用于记载某字母是否被选中，以便于判断字母是否重复。代码如下所示：

```

public static char[] generate() {
    char[] chs = new char[5];

    char[] letters = { 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J',
                       'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V',
                       'W', 'X', 'Y', 'Z' };
    boolean[] flags = new boolean[letters.length];

```



```
        return chs;
    }
}
```

步骤二：随机选择 5 个不同的字母

使用嵌套循环，随机选择 5 个不同的字母，并且这五个字母各不相同。代码如下所示：

```
public static char[] generate() {
    char[] chs = new char[5];
    char[] letters = { 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J',
        'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V',
        'W', 'X', 'Y', 'Z' };
    boolean[] flags = new boolean[letters.length];

    for (int i = 0; i < chs.length; i++) {
        int index;
        do {
            index = (int) (Math.random() * (letters.length));
        } while (flags[index]); // 判断生成的字符是否重复
        chs[i] = letters[index];
        flags[index] = true;
    }

    return chs;
}
```

• 完整代码

本案例的完整代码如下所示：

```
import java.util.Scanner;

public class GuessingGame {
    public static void main(String[] args) {
        // 表示玩家猜测的次数
        int count = 0;
        // 表示用户猜测的数据
        char[] input = null;
        // 表示猜测的字符串
        char[] chs = null;
        // 用于保存判断的结果
        int[] result = new int[2];
    }

    /**
     * 随机生成需要猜测的字母序列
     *
     * @return 存储随机字符的数组
     */
    public static char[] generate() {
        char[] chs = new char[5];
        char[] letters = { 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J',
            'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V',
            'W', 'X', 'Y', 'Z' };
        boolean[] flags = new boolean[letters.length];
        for (int i = 0; i < chs.length; i++) {
            int index;
            do {
                index = (int) (Math.random() * (letters.length));
```

```

        } while (flags[index]); // 判断生成的字符是否重复
        chs[i] = letters[index];
        flags[index] = true;
    }
    return chs;
}

/**
 * 比较玩家输入的字母序列和程序所生成的字母序列，逐一比较字符及其位置，并记载比较结果
 *
 * @param chs
 *         程序生成的字符序列
 * @param input
 *         玩家输入的字符序列
 * @return 存储比较的结果。返回值 int 数组 的长度为 2，其中，索引为 0 的位置
 *         用于存放完全猜对的字母个数(字符和位置均正确)，索引为 1 的位置用于存放猜对的字母个数(字符正确，但是位置不正确)。
 */
public static int[] check(char[] chs, char[] input) {
    int[] result = new int[2];

    return result;
}
}

```

4. 猜字母游戏——实现字母检测方法

- 问题

比较玩家输入的字母序列和程序所生成的字母序列，逐一比较字符及其位置，并记载比较结果。

- 方案

实现 `check` 方法，需要逐一取出玩家录入的每个字符，并和结果字符串——比较：比较字符本身以及字符所在的位置，并记载比较的结果。此案例需要使用嵌套循环来实现。`check` 方法的流程如图-5 所示，其中红色部分表示外层循环，蓝色部分表示内层循环。

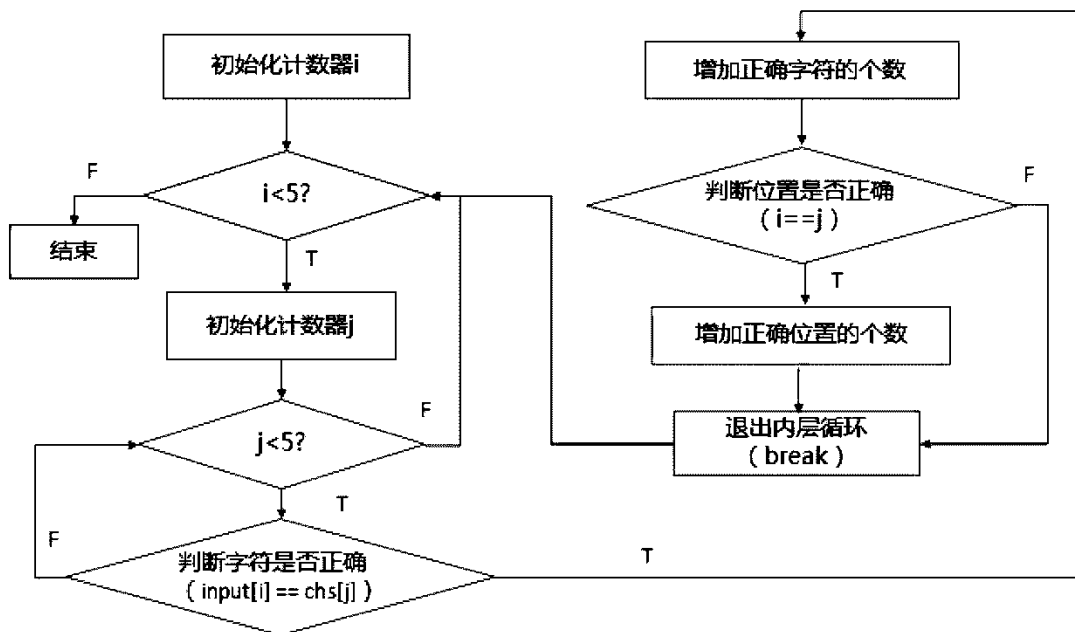


图- 5

• 步骤

实现此案例需要按照如下步骤进行。

步骤一：构建存储比较结果的数组

在 `check()` 方法中，首先定义 `int` 类型的数组类型变量 `result`，用于存储比较的结果。数组 `result` 的长度为 2，其中，`result[0]` 用于存放完全猜对的字母个数(字符和位置均正确)，`result[1]` 用于存放猜对的字母个数(字符正确，但是位置不正确)。代码如下所示：

```
public static int[] check(char[] chs, char[] input) {

    int[] result = new int[2];

    return result;

}
```

步骤二：比较

使用嵌套循环，统计完全猜对的字母个数，和猜对的字母个数(位置不对)，代码如下所示：

```
public static int[] check(char[] chs, char[] input) {
    int[] result = new int[2];

    for (int i = 0; i < input.length; i++) {
        for (int j = 0; j < chs.length; j++) {
            if (input[i] == chs[j]) {
```

```

        result[1]++;
        if (i == j) {
            result[0]++;
        }
        break;
    }
}

return result;
}

```

• 完整代码

本案例的完整代码如下所示：

```

import java.util.Scanner;

public class GuessingGame {
    public static void main(String[] args) {
        // 表示玩家猜测的次数
        int count = 0;
        //表示用户猜测的数据
        char[] input=null;
        // 表示猜测的字符串
        char[] chs =null;
        // 用于保存判断的结果
        int[] result = new int[2];
    }

    /**
     * 随机生成需要猜测的字母序列
     *
     * @return 存储随机字符的数组
     */
    public static char[] generate() {
        char[] chs = new char[5];
        char[] letters = { 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J',
            'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V',
            'W', 'X', 'Y', 'Z' };
        boolean[] flags = new boolean[letters.length];
        for (int i = 0; i < chs.length; i++) {
            int index;
            do {
                index = (int) (Math.random() * (letters.length));
            } while (flags[index]); // 判断生成的字符是否重复
            chs[i] = letters[index];
            flags[index] = true;
        }
        return chs;
    }

    /**
     * 比较玩家输入的字母序列和程序所生成的字母序列，逐一比较字符及其位置，并记载比较结果
     *
     * @param chs
     *         程序生成的字符序列
     * @param input
     *         玩家输入的字符序列
     * @return 存储比较的结果。返回值 int 数组 的长度为 2，其中，索引为 0 的位置
     *         用于存放完全猜对的字母个数(字符和位置均正确)，索引为 1 的位置用于存放猜对的字母个数(字符正确，但是位置不正确)。
     */
}

```

```
*/
public static int[] check(char[] chs, char[] input) {
    int[] result = new int[2];
    for (int i = 0; i < input.length; i++) {
        for (int j = 0; j < chs.length; j++) {
            if (input[i] == chs[j]) { // 判断字符是否正确
                result[1]++;
                if (i == j) { // 判断位置是否正确
                    result[0]++;
                }
                break;
            }
        }
    }
    return result;
}
}
```

5. 猜字母游戏——实现主方法

- **问题**

实现猜字母游戏的整体流程。

- **方案**

为实现猜字母游戏的整体过程，首先需要调用 `generate()` 方法，以生成五个字母；

其次，需要使用 `while(true)` 循环允许玩家进行多次猜测；

在循环中，接收玩家猜测的字母，然后调用 `check()` 方法，检查猜测的结果。如果猜测正确，则输出提示信息和分数，游戏结束；如果猜测错误，则输出猜测的判断结果并提示玩家游戏继续。main 程序流程如图-6 所示。

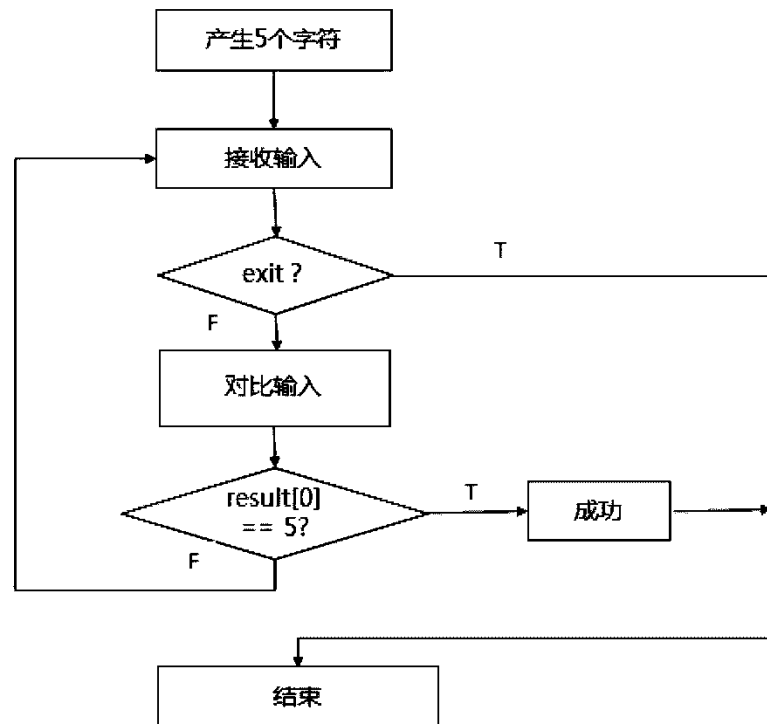


图- 6

• 步骤

实现此案例需要按照如下步骤进行。

步骤一：构建存储所有字母的数组

在 main 方法中，先输出提示信息表示游戏开始，然后调用 generate() 方法生成要猜测的五个字母，并提示玩家开始猜测，代码如下所示：

```

import java.util.Scanner;

public class GuessingGame {
    public static void main(String[] args) {
        // 表示玩家猜测的次数
        int count = 0;
        //表示用户猜测的数据
        char[] input=null;
        // 用于保存判断的结果
        int[] result = new int[2];

        Scanner scanner = new Scanner(System.in);
        System.out.println("GuessingGame>欢迎尝试猜字母游戏!");
        // 表示猜测的字符串
        char[] chs = generate();

        scanner.close();
    }
}

```

```
}

/**
 * 随机生成需要猜测的字母序列
 *
 * @return 存储随机字符的数组
 */
public static char[] generate() {

    char[] letters = { 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J',
        'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V',
        'W', 'X', 'Y', 'Z' };
    boolean[] flags = new boolean[letters.length];
    char[] chs = new char[5];
    for (int i = 0; i < chs.length; i++) {
        int index;
        do {
            index = (int) (Math.random() * (letters.length));
        } while (flags[index]); // 判断生成的字符是否重复
        chs[i] = letters[index];
        flags[index] = true;
    }
    return chs;
}

/**
 * 比较玩家输入的字母序列和程序所生成的字母序列，逐一比较字符及其位置，并记载比较结果
 *
 * @param chs
 *         程序生成的字符序列
 * @param input
 *         玩家输入的字符序列
 * @return 存储比较的结果。返回值 int 数组 的长度为 2，其中，索引为 0 的位置
 *         用于存放完全猜对的字母个数(字符和位置均正确)，索引为 1 的位置用于存放猜对的字母个数(字符正确，但是位置不正确)。
 */
public static int[] check(char[] chs, char[] input) {
    int[] result = new int[2];
    for (int i = 0; i < input.length; i++) {
        for (int j = 0; j < chs.length; j++) {
            if (input[i] == chs[j]) { // 判断字符是否正确
                result[1]++;
                if (i == j) { // 判断位置是否正确
                    result[0]++;
                }
                break;
            }
        }
    }
    return result;
}
}
```

步骤二：构建循环

使用 `while (true)` 循环，并在循环中调用 `Scanner` 类的 `next()` 方法接收玩家猜测的字符串。为方便字符串的比较，将玩家录入的字符串均转换为大写字母，然后先判断玩家录入的是否为“EXIT”，如果是，则循环中止，游戏结束。代码如下所示：

```
import java.util.Scanner;

public class GuessingGame {
    public static void main(String[] args) {
        // 表示玩家猜测的次数
        int count = 0;
        //表示用户猜测的数据
        char[] input=null;
        // 用于保存判断的结果
        int[] result = new int[2];
        Scanner scanner = new Scanner(System.in);
        System.out.println("GuessingGame>欢迎尝试猜字母游戏!");
        // 表示猜测的字符串
        char[] chs = generate();

        System.out.println("GuessingGame>游戏开始,请输入你所猜的 5 个字母序列:(exit—退出)");
        while (true) {
            String inputStr = scanner.next().trim().toUpperCase();
            if ("EXIT".equals(inputStr)) {
                System.out.println("GuessingGame>谢谢你的尝试,再见!");
                break;
            }
        }

        scanner.close();
    }

    /**
     * 随机生成需要猜测的字母序列
     *
     * @return 存储随机字符的数组
     */
    public static char[] generate() {

        char[] letters = { 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J',
            'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V',
            'W', 'X', 'Y', 'Z' };
        boolean[] flags = new boolean[letters.length];
        char[] chs = new char[5];
        for (int i = 0; i < chs.length; i++) {
            int index;
            do {
                index = (int) (Math.random() * (letters.length));
            } while (flags[index]); // 判断生成的字符是否重复
            chs[i] = letters[index];
            flags[index] = true;
        }
        return chs;
    }

    /**
     * 比较玩家输入的字母序列和程序所生成的字母序列,逐一比较字符及其位置,并记载比较结果
     *
     * @param chs
     *         程序生成的字符序列
     * @param input
     *         玩家输入的字符序列
     * @return 存储比较的结果。返回值 int 数组 的长度为 2,其中,索引为 0 的位置
```


* 用于存放完全猜对的字母个数(字符和位置均正确),索引为 1 的位置用于存放猜对的字母个数(字符正确,但是位置不正确)。

```
*/
public static int[] check(char[] chs, char[] input) {
    int[] result = new int[2];
    for (int i = 0; i < input.length; i++) {
        for (int j = 0; j < chs.length; j++) {
            if (input[i] == chs[j]) { // 判断字符是否正确
                result[1]++;
                if (i == j) { // 判断位置是否正确
                    result[0]++;
                }
                break;
            }
        }
    }
    return result;
}
}
```

步骤三：比较

如果玩家录入的不是“EXIT”，则调用 check() 方法与答案进行比较，并得到存储比较结果的数组 result。

比较完毕后，根据比较结果输出提示信息到界面，并计算分数。如果 5 个字符的位置均正确，则表示游戏结束，计算并输出玩家的得分；否则，将猜测的次数累加 1，并提示玩家所猜对的字符个数以及位置个数，游戏继续。代码如下所示：

```
import java.util.Scanner;

public class GuessingGame {
    public static void main(String[] args) {
        // 表示玩家猜测的次数
        int count = 0;
        //表示用户猜测的数据
        char[] input=null;
        // 用于保存判断的结果
        int[] result = new int[2];
        Scanner scanner = new Scanner(System.in);
        System.out.println("GuessingGame>欢迎尝试猜字母游戏！");
        // 表示猜测的字符串
        char[] chs = generate();

        System.out.println("GuessingGame>游戏开始，请输入你所猜的 5 个字母序列：(exit—退出)");

        while (true) {
            String inputStr = scanner.next().trim().toUpperCase();
            if ("EXIT".equals(inputStr)) {
                System.out.println("GuessingGame>谢谢你的尝试，再见！");
                break;
            }

            input = inputStr.toCharArray();
            result = check(chs, input);
        }
    }
}
```

```

        if (result[0] == chs.length) { // 完全猜对的情况
            int score = 100 * chs.length - count * 10;
            System.out.println("GuessingGame>恭喜你猜对了！你的得分是：" + score);
            break;
        } else {
            count++;
            System.out.println("GuessingGame>你猜对" + result[1] + "个字符，其中"
                + result[0] + "个字符的位置正确！(总次数=" + count + "，exit—
退出)");
        }

        scanner.close();
    }

    /**
     * 随机生成需要猜测的字母序列
     *
     * @return 存储随机字符的数组
     */
    public static char[] generate() {

        char[] letters = { 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J',
            'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V',
            'W', 'X', 'Y', 'Z' };
        boolean[] flags = new boolean[letters.length];
        char[] chs = new char[5];
        for (int i = 0; i < chs.length; i++) {
            int index;
            do {
                index = (int) (Math.random() * (letters.length));
            } while (flags[index]); // 判断生成的字符是否重复
            chs[i] = letters[index];
            flags[index] = true;
        }
        return chs;
    }

    /**
     * 比较玩家输入的字母序列和程序所生成的字母序列，逐一比较字符及其位置，并记载比较结果
     *
     * @param chs
     *         程序生成的字符序列
     * @param input
     *         玩家输入的字符序列
     * @return 存储比较的结果。返回值 int 数组 的长度为 2，其中，索引为 0 的位置
     *         用于存放完全猜对的字母个数(字符和位置均正确)，索引为 1 的位置用于存放猜对的字
     *         母个数(字符正确，但是位置不正确)。
     */
    public static int[] check(char[] chs, char[] input) {
        int[] result = new int[2];
        for (int i = 0; i < input.length; i++) {
            for (int j = 0; j < chs.length; j++) {
                if (input[i] == chs[j]) { // 判断字符是否正确
                    result[1]++;
                    if (i == j) { // 判断位置是否正确
                        result[0]++;
                    }
                    break;
                }
            }
        }
    }

```

```
    }
    }
    return result;
}
}
```

• 完整代码

本案例的完整代码如下所示：

```
import java.util.Scanner;

public class GuessingGame {
    public static void main(String[] args) {
        // 表示玩家猜测的次数
        int count = 0;
        //表示用户猜测的数据
        char[] input=null;
        // 用于保存判断的结果
        int[] result = new int[2];
        Scanner scanner = new Scanner(System.in);
        System.out.println("GuessingGame>欢迎尝试猜字母游戏!");
        // 表示猜测的字符串
        char[] chs = generate();
        System.out.println("GuessingGame>游戏开始, 请输入你所猜的 5 个字母序列:
(exit—退出)");
        while (true) {
            String inputStr = scanner.next().trim().toUpperCase();
            if ("EXIT".equals(inputStr)) {
                System.out.println("GuessingGame>谢谢你的尝试, 再见!");
                break;
            }

            input = inputStr.toCharArray();
            result = check(chs, input);
            if (result[0] == chs.length) { // 完全猜对的情况
                int score = 100 * chs.length - count * 10;
                System.out.println("GuessingGame>恭喜你猜对了! 你的得分是: " +
                                                                    score);
                break;
            } else {
                count++;
                System.out.println("GuessingGame>你猜对" + result[1] + "个字符,
                                                                    其中"
                                                                    + result[0] + "个字符的位置正确! (总次数=" + count + ", exit—
                                                                    退出)");
            }
        }
        scanner.close();
    }
}

/**
 * 随机生成需要猜测的字母序列
 *
 * @return 存储随机字符的数组
 */
public static char[] generate() {

    char[] letters = { 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J',
        'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V',
        'W', 'X', 'Y', 'Z' };
    boolean[] flags = new boolean[letters.length];
    char[] chs = new char[5];
```

```

        for (int i = 0; i < chs.length; i++) {
            int index;
            do {
                index = (int) (Math.random() * (letters.length));
            } while (flags[index]); // 判断生成的字符是否重复
            chs[i] = letters[index];
            flags[index] = true;
        }
        return chs;
    }

    /**
     * 比较玩家输入的字母序列和程序所生成的字母序列，逐一比较字符及其位置，并记载比较结果
     *
     * @param chs
     *         程序生成的字符序列
     * @param input
     *         玩家输入的字符序列
     * @return 存储比较的结果。返回值 int 数组 的长度为 2，其中，索引为 0 的位置
     *         用于存放完全猜对的字母个数(字符和位置均正确)，索引为 1 的位置用于存放猜对的字母个数(字符正确，但是位置不正确)。
     */
    public static int[] check(char[] chs, char[] input) {
        int[] result = new int[2];
        for (int i = 0; i < input.length; i++) {
            for (int j = 0; j < chs.length; j++) {
                if (input[i] == chs[j]) { // 判断字符是否正确
                    result[1]++;
                    if (i == j) { // 判断位置是否正确
                        result[0]++;
                    }
                    break;
                }
            }
        }
        return result;
    }
}

```

课后作业

1. 随机生成数组

封装一个方法 `generateArray`，该方法实现生成指定长度的 `int` 数组，该数组的元素为 0 到指定范围内的随机数，并将该数组返回。

2. 猜字母游戏——实现游戏等级

为猜字母游戏添加游戏等级。游戏等级设为三等：5、7 和 9，代表所需要猜测的字母个数。游戏开始时，由玩家选择游戏等级（5、7、9）。如果选择 7，则会随机产生 7 个字符，然后玩家输入一个字符串包含 7 个字符，看这 7 个字符和随机产生的 7 个字符比较，看是否正确，并统计分数。另外，如果输入其它，重新提示输入游戏等级。系统交互情况如图-1 所示：

```
GuessingGame (2) [Java Application] C:\Program Files\Java\jdk1.6.0_31\bin\javaw.exe (2013-12-12)
GuessingGame>欢迎尝试猜字母游戏!
GuessingGame>请输入游戏级别(5、7、9)? 8
GuessingGame>请输入游戏级别(5、7、9)? 7
GuessingGame>游戏开始, 请输入你所猜的7个字母序列: (exit—退出)
SDFREYI
GuessingGame>你猜对0个字符, 其中0个字符的位置正确! (总次数=0, exit—退出)
ASDFGHU
GuessingGame>你猜对3个字符, 其中1个字符的位置正确! (总次数=1, exit—退出)
```

图- 1