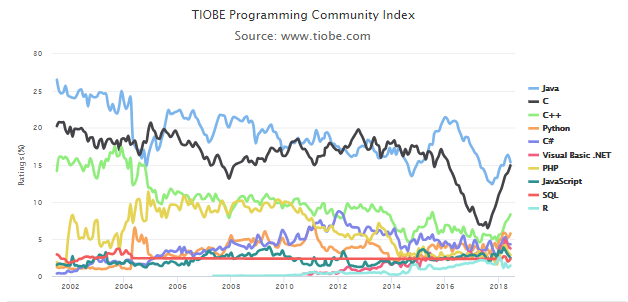
# 第一章 Java概述

## 一、导入

目前，Java语言可以说是最常用的编程语言之一，在应用软件领域，它唯一的竞争对手似乎只有微软的.NET。C/C++作为曾经的霸主，目前依然占据着系统软件和嵌入式系统绝对的市场份额，但正在逐步退出应用软件领域。



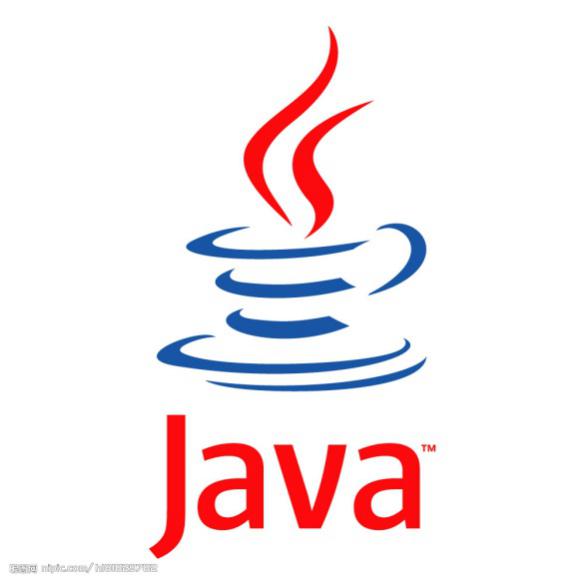
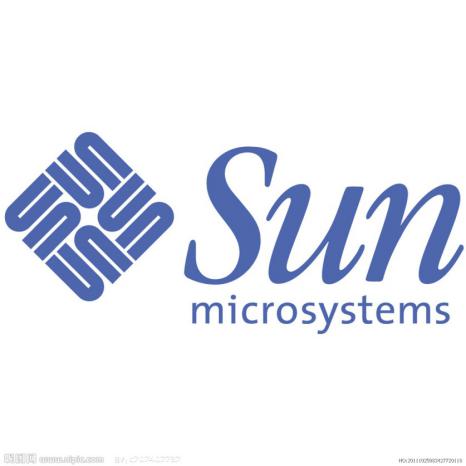
图 1-TIOBE Index for June 2018



TIOBE排行榜，每月更新一次，是根据互联网上有经验的程序员、课程和第三方厂商的数量，并使用搜索引擎（如Google、Bing、Yahoo!等）以及Wikipedia、Amazon、YouTube统计出排名数据，旨在反映某个编程语言的热门程度。该指数可以用来检阅开发者的编程技能能否跟上趋势，或是否有必要作出战略改变，以及什么编程语言是应该及时掌握的。该指数可以用来检阅开发者的编程技能能否跟上趋势，或是否有必要作出战略改变，以及什么编程语言是应该及时掌握的。

## 二、Java概述

### 1、Java语言发展史



Java语言诞生于SUN（Stanford University Network)公司，最初命名为Oak（橡树）Green Team小组成员James Gosling、Bill Joy、Patrick Naughton、Mike Sheridan，最初的目的：与家电一起使用，1994年，小组意识到 Oak 非常适合于互联网

1995年5月23日发布定名为Java

1996年1月，第一个JDK-JDK1.0正式版本诞生

1996年4月，10个最主要的操作系统供应商申明将在其产品中嵌入JAVA技术

1996年9月，约8.3万个网页应用了JAVA技术来制作

1997年2月18日，JDK1.1发布

1997年4月2日，JavaOne会议召开，参与者逾一万人，创当时全球同类会议规模之纪录

1997年9月，JavaDeveloperConnection社区成员超过十万

1998年2月，JDK1.1被下载超过2,000,000次

1998年12月8日，JAVA2企业平台J2EE发布

1999年6月，SUN公司发布Java的三个版本：标准版、企业版和微型版（J2SE、J2EE、J2ME）

2000年5月8日，JDK1.3发布

2000年5月29日，JDK1.4发布

2001年6月5日，NOKIA宣布，到2003年将出售1亿部支持Java的手机

2001年9月24日，J2EE1.3发布

2002年2月26日，J2SE1.4发布，自此Java的计算能力有了大幅提升。

2004年9月30日18:00PM，J2SE1.5发布，是Java语言的发展史上的又一里程碑事件。为了表示这个版本的重要性，J2SE1.5更名为J2SE5.0

2005年6月，JavaOne大会召开，SUN公司公开Java SE 6。此时，Java的各种版本已经更名以取消其中的数字“2”：J2EE更名为Java EE, J2SE更名为Java SE，J2ME更名为Java ME。

2006年11月13日，SUN公司宣布Java全线采纳GNU General Public License Version 2，从而公开了Java的源代码。

2009年4月20日甲骨文以现金收购Sun微系统公司，交易价格达74亿美元。

从2006年12月份Sun发布Java 6后，经过五年多的不懈努力在2011年7月底发布了Java 7正式版！这也是Sun被Oracle收购以来发行的第一个版本。而在三年后的今天，被冠名为“跳票王”的Oracle终于发布了Java 8的正式版，但对于很多开发者来说却比Java 7来的更漫长一些。主要原因还是因为Oracle原本计划在2013年发布正式版Java 8，却因受困于安全性的问题经过了两次跳票，经历9个里程碑版本。当然，我们更不愿意看到Oracle因如期发布而牺牲质量，把原先没有解决的一些缺陷的安全问题带到Java 8当中去。同时也很可能将放弃掉Lambda而导致广泛应用的可能性更小。不管怎么样，Java 8如今来了，全新“革命”而不再是“进化”的功能将会让无数开发者动容。

2014年3月18日Java8.0发布，这是继Java5.0以来变化最大的版本。一共有10大新特性。最主要的是Lambda表达式和强大的StreamAPI和新版的日期时间API，函数式接口和接口的默认方法和静态方法等。

2017年9月21日Java9的发布。Java 9的最主要目标是最大限度实现模块化以帮助人们实现积木式的应用编写。目的是帮助人们从JAR的束缚中解脱出来。该特性将贯穿整个Java库，并以单依赖图的方式重新整理依赖。Java 9会把所有三个Java开发平台统一起来，模块化特性会使得Java ME的可复用性得到增强，这将是反击Android和iOS的有力武器。

|  |  |  |  |
| --- | --- | --- | --- |
| 版本号 | 名称 | 中文名 | 发布日期 |
| JDK 1.1.4 | Sparkler | 宝石 | 1997-09-12 |
| JDK 1.1.5 | Pumpkin | 南瓜 | 1997-12-13 |
| JDK 1.1.6 | Abigail | 阿比盖尔--女子名 | 1998-04-24 |
| JDK 1.1.7 | Brutus | 布鲁图--古罗马政治家和将军 | 1998-09-28 |
| JDK 1.1.8 | Chelsea | 切尔西--城市名 | 1999-04-08 |
| J2SE 1.2 | Playground | 运动场 | 1998-12-04 |
| J2SE 1.2.1 | none | 无 | 1999-03-30 |
| J2SE 1.2.2 | Cricket | 蟋蟀 | 1999-07-08 |
| J2SE 1.3 | Kestrel | 美洲红隼 | 2000-05-08 |
| J2SE 1.3.1 | Ladybird | 瓢虫 | 2001-05-17 |
| J2SE 1.4.0 | Merlin | 灰背隼 | 2002-02-13 |
| J2SE 1.4.1 | grasshopper | 蚱蜢 | 2002-09-16 |
| J2SE 1.4.2 | Mantis | 螳螂 | 2003-06-26 |
| JavaSE 5.0 | Tiger | 老虎 | 2004-09-30 |
| JavaSE 6.0 | Mustang | 野马 | 2006-04 |
| JavaSE 7.0 | Dolphin | 海豚 | 2011-07-28 |
| Java SE 8.0 | Spider | 蜘蛛 | 2014-03-18 |
| JavaSE9.0 |  |  | 2017-9-21 |

### 2、Java语言主要特征

* **Java语言是易学的。**Java语言的语法与C语言和C++语言很接近，使得大多数程序员很容易学习和使用Java。
* **Java语言是强制面向对象的。**Java语言提供类、接口和继承等原语，为了简单起见，只支持类之间的单继承，但支持接口之间的多继承，并支持类与接口之间的实现机制（关键字为implements）。
* **Java语言是分布式的。**Java语言支持Internet应用的开发，在基本的Java应用编程接口中有一个网络应用编程接口（java net），它提供了用于网络应用编程的类库，包括URL、URLConnection、Socket、ServerSocket等。Java的RMI（远程方法激活）机制也是开发分布式应用的重要手段。
* **Java语言是健壮的。**Java的强类型机制、异常处理、垃圾的自动收集等是Java程序健壮性的重要保证。对指针的丢弃是Java的明智选择。
* **Java语言是安全的。**Java通常被用在网络环境中，为此，Java提供了一个安全机制以防恶意代码的攻击。如：安全防范机制（类ClassLoader），如分配不同的名字空间以防替代本地的同名类、字节代码检查。
* **Java语言是体系结构中立的。**Java程序（后缀为java的文件）在Java平台上被编译为体系结构中立的字节码格式（后缀为class的文件），然后可以在实现这个Java平台的任何系统中运行。
* **Java语言是解释型的。**如前所述，Java程序在Java平台上被编译为字节码格式，然后可以在实现这个Java平台的任何系统的解释器中运行。
* **Java是性能略高的。**与那些解释型的高级脚本语言相比，Java的性能还是较优的。
* **Java语言是原生支持多线程的。**在Java语言中，线程是一种特殊的对象，它必须由Thread类或其子（孙）类来创建。

### 3、Java技术体系平台

|  |
| --- |
| **Java SE(Java Standard Edition)Java标准版** |
| 支持面向桌面级应用（如Windows下的应用程序）的Java平台，提供了完整的Java核心API，此版本以前称为**J2SE** |
| **Java EE(Java Enterprise Edition)Java企业版** |
| 是为开发企业环境下的应用程序提供的一套解决方案。该技术体系中包含的技术如:Servlet 、Jsp等，主要针对于Web应用程序开发。版本以前称为**J2EE** |
| **Java ME(Java Micro Edition)Java小型版** |
| 支持Java程序运行在移动终端（手机、PDA）上的平台，对Java API有所精简，并加入了针对移动终端的支持，此版本以前称为**J2ME** |
| **Java Card** |
| 支持一些Java小程序（Applets）运行在小内存设备（如智能卡）上的平台 |

### 4、Java语言运行机制[☆]

#### （1）Java语言最主要的特点

* **特点一：面向对象**

面向对象的两个基本概念：类、对象

面向对象的三大特性：封装、继承、多态

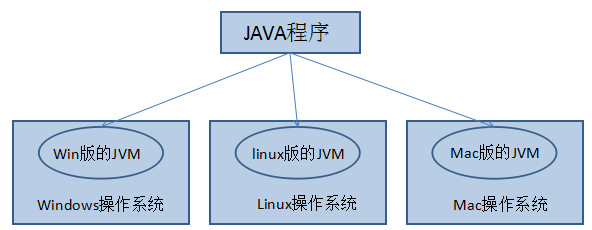
* 特点二：健壮性

吸收了C/C++语言的优点，但去掉了其影响程序健壮性的部分（如指针、内存的申请与释放等），提供了一个相对安全的内存管理和访问机制

* **特点三：跨平台性**

跨平台性：Java语言编写的应用程序在不同的操作系统平台上都可以运行。“Write once , Run Anywhere”一次编写，处处运行。

原理：只要在需要运行 java 应用程序的操作系统上，先安装一个Java虚拟机 (JVM Java Virtual Machine) 即可。由JVM来负责Java程序在该系统中的运行。



因为有了JVM，同一个Java 程序在三个不同的操作系统中都可以执行。这样就实现了Java 程序的跨平台性。

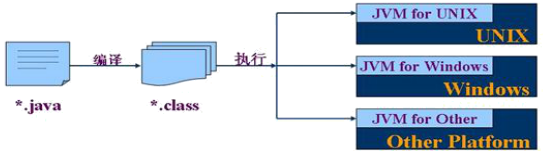
#### （2）Java两种核心机制

* **Java虚拟机（Java Virtal Machine）**

JVM是一个虚拟的计算机，具有指令集并使用不同的存储区域。负责执行指令，管理数据、内存、寄存器。

对于不同的平台，有不同的虚拟机。

Java虚拟机机制屏蔽了底层运行平台的差别，实现了“一次编译，到处运行”。



* **垃圾收集机制（Garbage Collection）**

1. 不再使用的内存空间应回收—— 垃圾回收。

在C/C++等语言中，由程序员负责回收无用内存。

Java 语言消除了程序员回收无用内存空间的责任：它提供一种系统级线程跟踪存储空间的分配情况，并在JVM空闲时，检查并释放那些可被释放的存储空间。

1. 垃圾回收在Java程序运行过程中自动进行，程序员无法精确控制和干预。

## 三、Windows操作系统常用的DOS命令

* dir：列出当前目录下的文件以及文件夹
* md：创建目录
* rd：删除目录
* cd：进入指定目录
* cd..：退回到上一级目录
* cd\：退回到根目录
* del：删除文件
* exit：退出 dos 命令行
* echo helloworld>1.txt

## 四、开发环境搭建

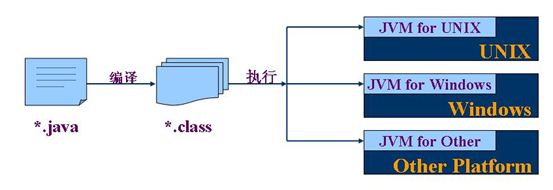
### 1、JVM、JRE、JDK介绍[☆]

1. JVM(JVM Java Virtual Machine)：核心机制，Java虚拟机

JVM是一个虚拟的计算机，具有指令集并使用不同的存储区域。负责执行指令，管理数据、内存、寄存器。

对于不同的平台，有不同的虚拟机。

Java虚拟机机制屏蔽了底层运行平台的差别，实现了“一次编译，到处运行”。



因为有了JVM，同一个Java 程序在三个不同的操作系统中都可以执行。这样就实现了Java 程序的跨平台性。

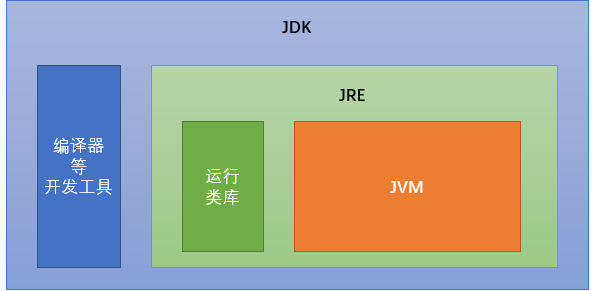
（2）JRE(Java Runtime Environment Java运行环境)

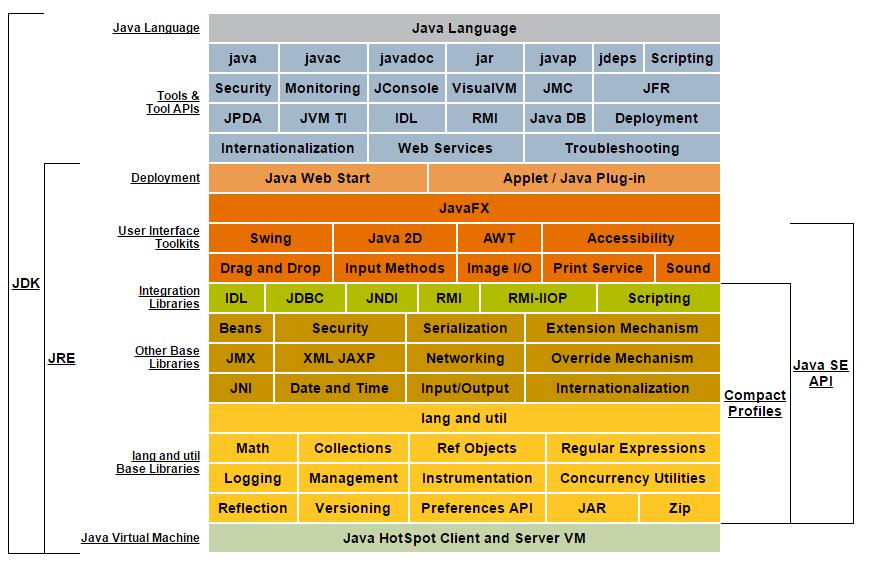
包括**Java虚拟机(JVM Java Virtual Machine)和Java程序所需的核心类库等**，如果想要运行一个开发好的Java程序，计算机中只需要安装JRE即可。

（3）JDK(Java Development Kit Java开发工具包)

**JDK是提供给Java开发人员使用的，包含了java开发工具+也包括了JRE。其中的开发工具：编译工具(javac.exe) 打包工具(jar.exe)等**

简单而言，使用JDK的开发工具完成的java程序，交给JRE去运行。

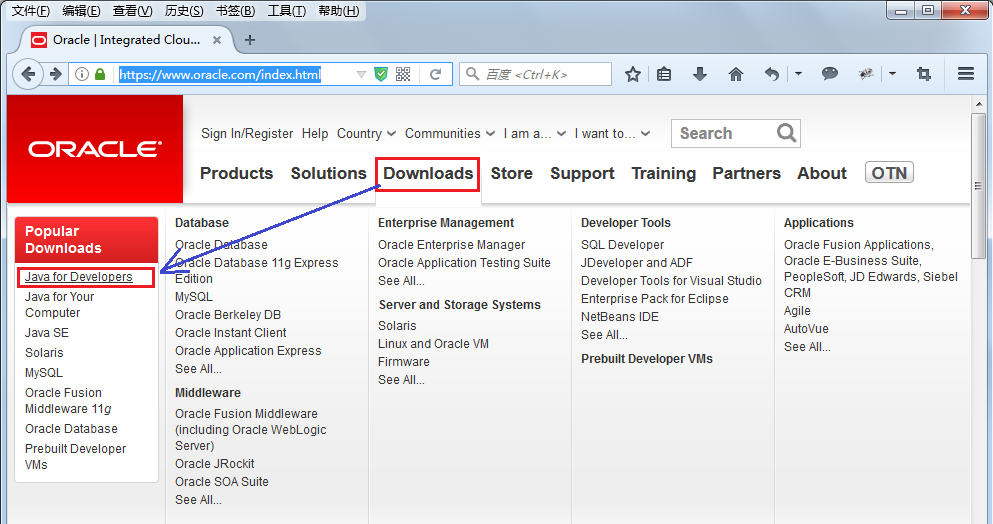




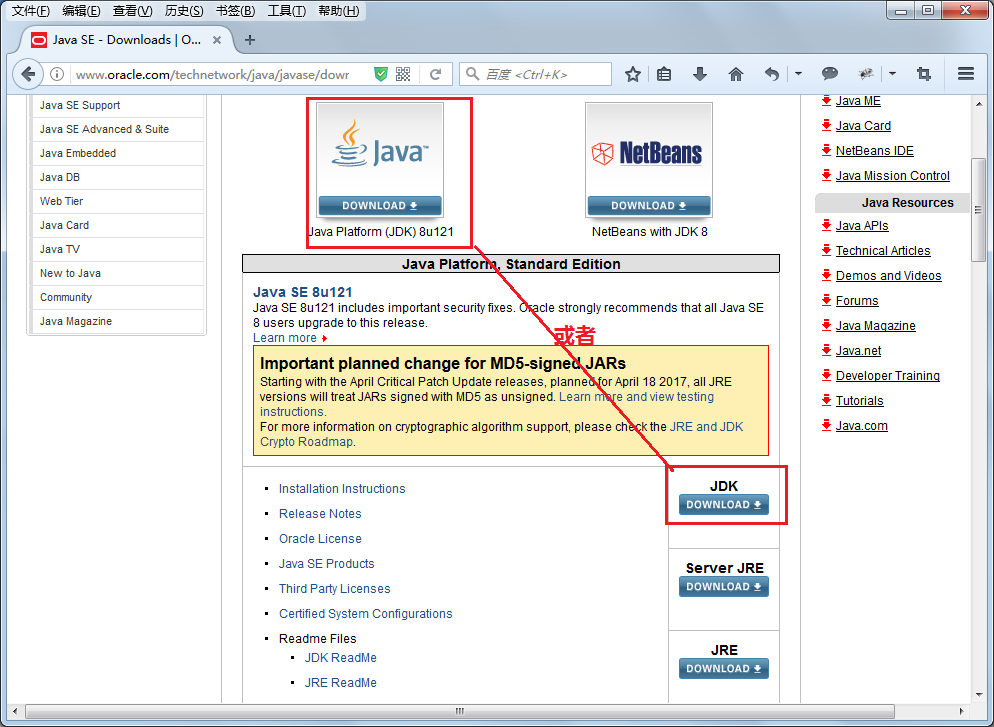
### JDK安装

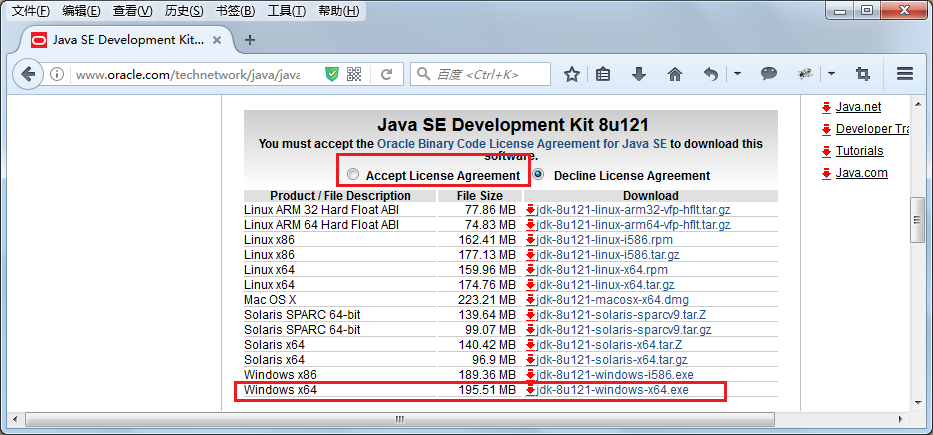
#### （1）下载 JDK

www.oracle.com



下载对应平台的JDK





双击下载：

Windows32位操作系统下载 如jdk-8u121-windows-i586.exe

Windows64位操作系统下载 如jdk-8u121-windows-x64.exe



#### （2）安装

* 安装 JDK
  + 傻瓜式安装，下一步即可。
  + 建议：安装路径不要有中文或者特殊符号如空格等。
  + 当提示安装 JRE 时，可以选择不安装。

双击安装包，进入安装向导





4、下一步，更改安装路径，选择安装所有组件



（1）取消安装独立的JRE

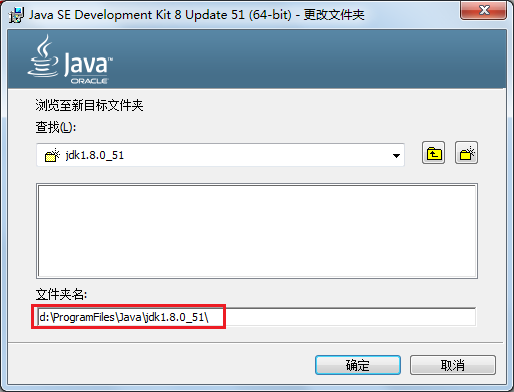
因为JDK中已经包含JRE，对于开发人员来说，可以不需要安装独立的JRE，如果想要取消安装JRE，





（2）更改JDK的安装路径

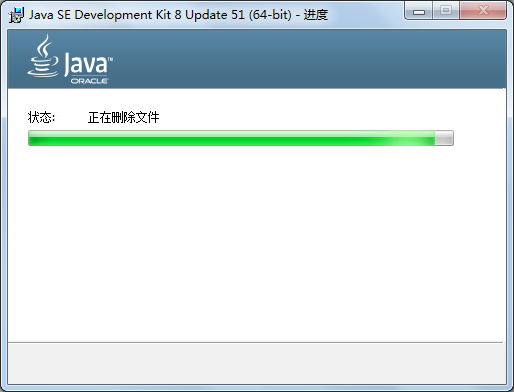




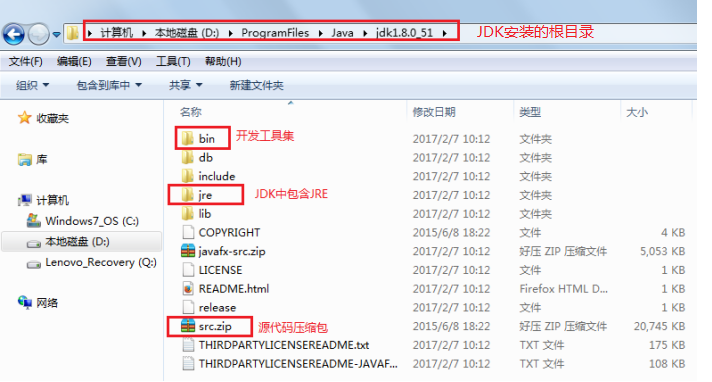
注意：这里安装目录中包含Java\JDK1.8.0\_51，这样下次就比较直观的能够清楚自己安装的是哪个版本的JDK



5、下一步，开始安装

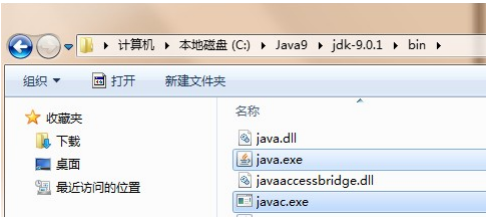


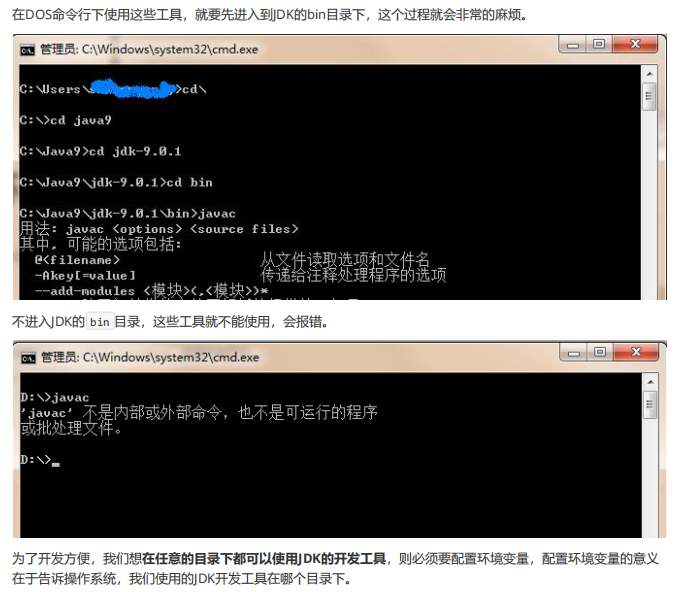




### 环境变量的配置





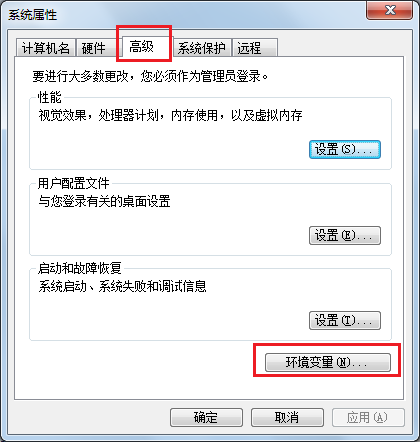


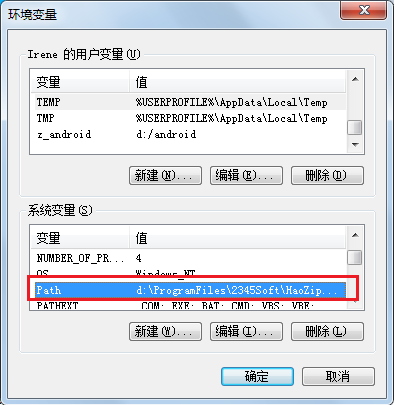
#### 方式一：直接配置path

我的电脑/计算机右键属性，选择高级系统设置



点击【环境变量】





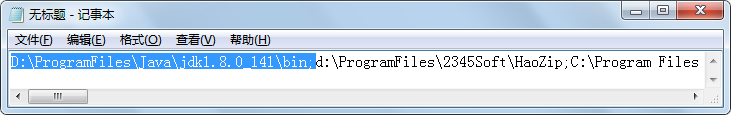


在path中新加环境变量时，建议把原来的值复制到记事本中再编辑，在原来的值前面增加

D:\ProgramFiles\Java\jdk1.8.0\_141\bin;

然后再复制回去，避免出错。

标点符号是英文半角输入方式。

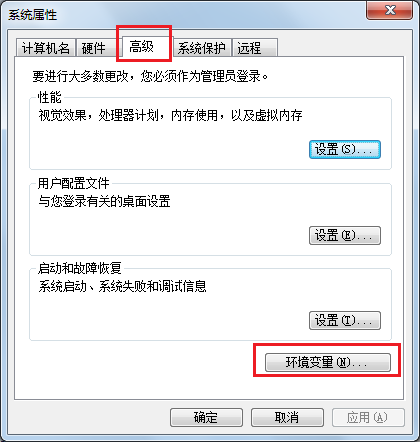


#### 方式二：配置JAVA\_HOME+path

我的电脑/计算机右键属性，选择高级系统设置



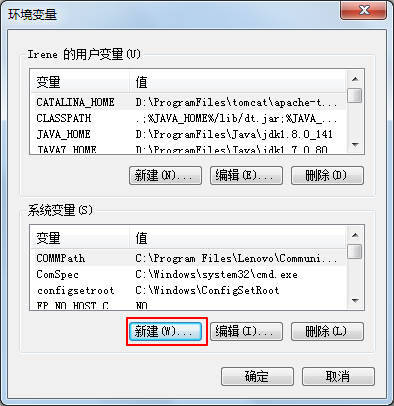
点击【环境变量】



新建变量“JAVA\_HOME”，值为JDK的安装路径，JDK安装的根目录：D:\ProgramFiles\Java\jdk1.8.0\_141

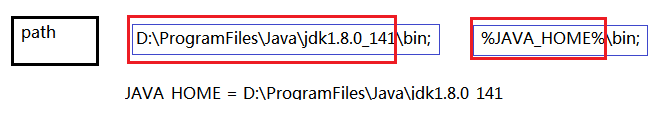
这里注意：

* 如果是以后都在该用户下开发使用JDK，那么可以选择新建用户变量；
* 如果是以后希望在所有用户下该变量都可见，那么选择新建系统变量；
* JAVA\_HOME是JDK安装的根目录，不是Java开发工具的bin目录





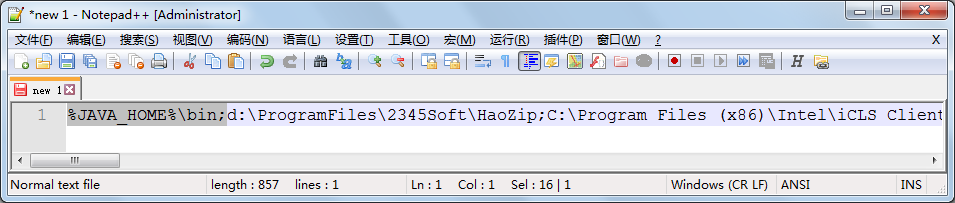
把JAVA\_HOME变量值添加到path变量中：%JAVA\_HOME%\bin;



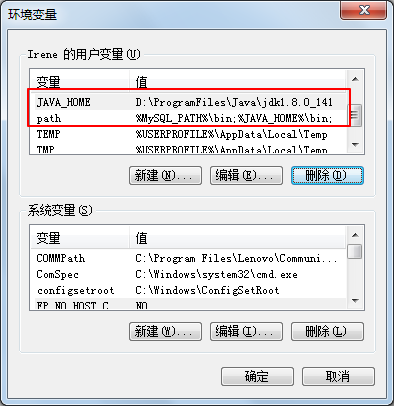
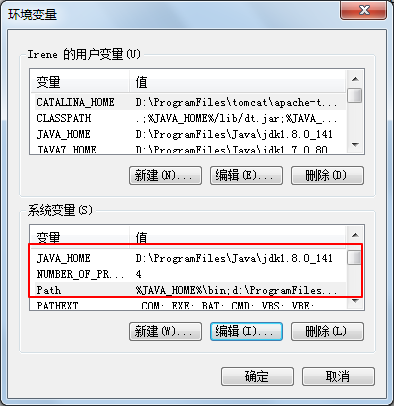


注意：

* 在path中新加环境变量时，建议把原来的值复制到记事本中，再编辑，在原来的值前面增加%JAVA\_HOME%\bin;然后再复制回去，避免出错

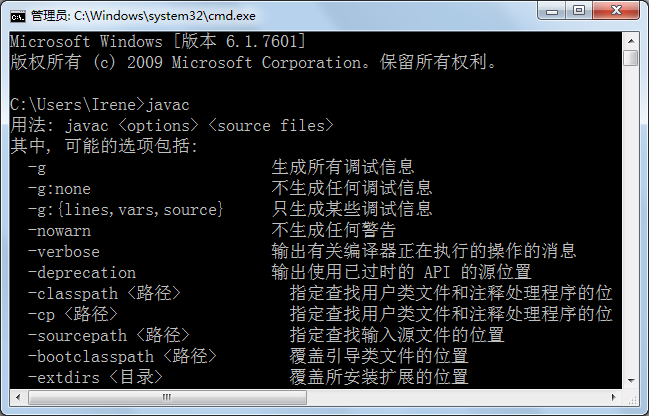


* 如果新建的是用户变量JAVA\_HOME，那么只能把JAVA\_HOME添加到用户变量的path中，否则可能出现找不到JAVA\_HOME的情况，如果用户变量没有path，可以再新建一个path变量（path变量是windows系统执行命令时要搜寻的路径），如果已经存在path，那么编辑即可，
* 如果新建的是系统变量JAVA\_HOME，那么只能把JAVA\_HOME添加到系统变量的path中，否则可能出现找不到JAVA\_HOME的情况

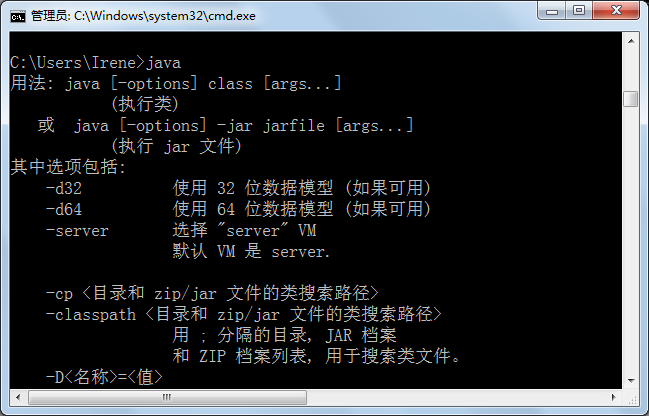
 **或**  

### 4、测试环境

检验javac命令

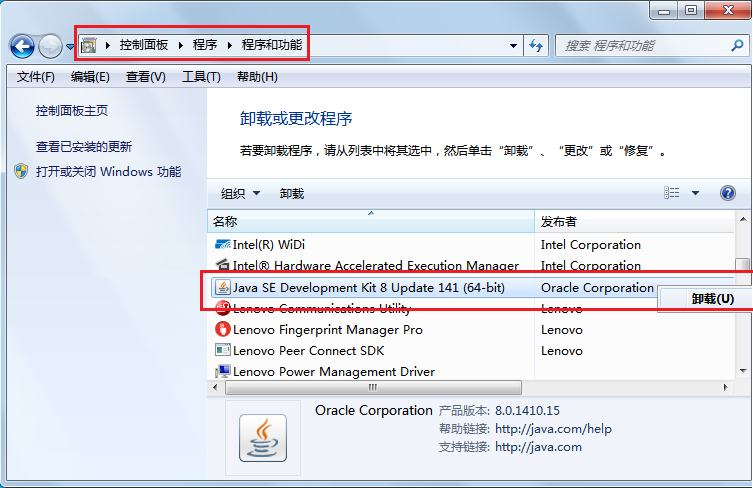


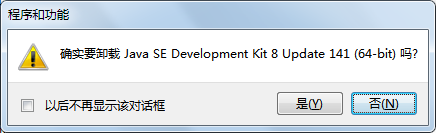
检验java命令

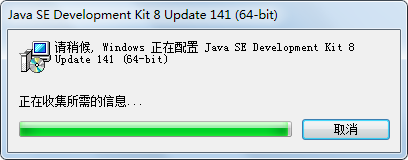


### 5、卸载

在控制面板或电脑管家卸载JDK







卸载后建议把相关的环境变量也删除了

### 6、选择合适的开发工具

选择合适的文本编辑器或 IDE 开发

* 文本编辑工具：

记事本、UltraEdit 、EditPlus、TextPad、NotePad++

* Java集成开发环境（IDE)：

IntelliJ IDEA、Eclipse、JBuilder 、MyEclipse、NetBeans

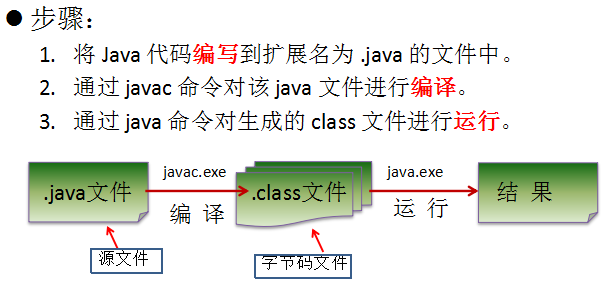
## 五、第一个Java应用程序

|  |
| --- |
| 类{  方法{  语句1;  语句2;  }  }  类=类签名+类体  类中包含 方法，  方法=方法签名+方法体 |

|  |
| --- |
| //其中//表示单行注释：表示该行的内容不属于代码  //这是我的第一个Java程序  //编写一个Java程序的步骤：  //1、编辑：文件的后缀名是.java，文件内容都是class 类名{}的结构  //2、编译：使用javac.exe编译工具把.java的源文件编译成一个或者多个.class的字节码文件  // 编译的命令格式：javac 文件名.java  //3、运行：使用java.exe工具运行，命令格式：java 类名  class HelloWorld{  //Java程序的入口是main()方法，它有固定的书写格式：public static void main(String[] args) {...}  public static void main(String[] args){  //System.out.println("");在控制台打印内容，""中的内容原样显示  //每一个语句以;结尾  System.out.println("hello world");  }  } |

讲解时，可以先不写main()然后运行，然后main()格式写错运行

### 1、步骤



### 2、常见问题

1、后缀名隐藏问题

2、当文件名与类名不一致时，javac与java命令的格式

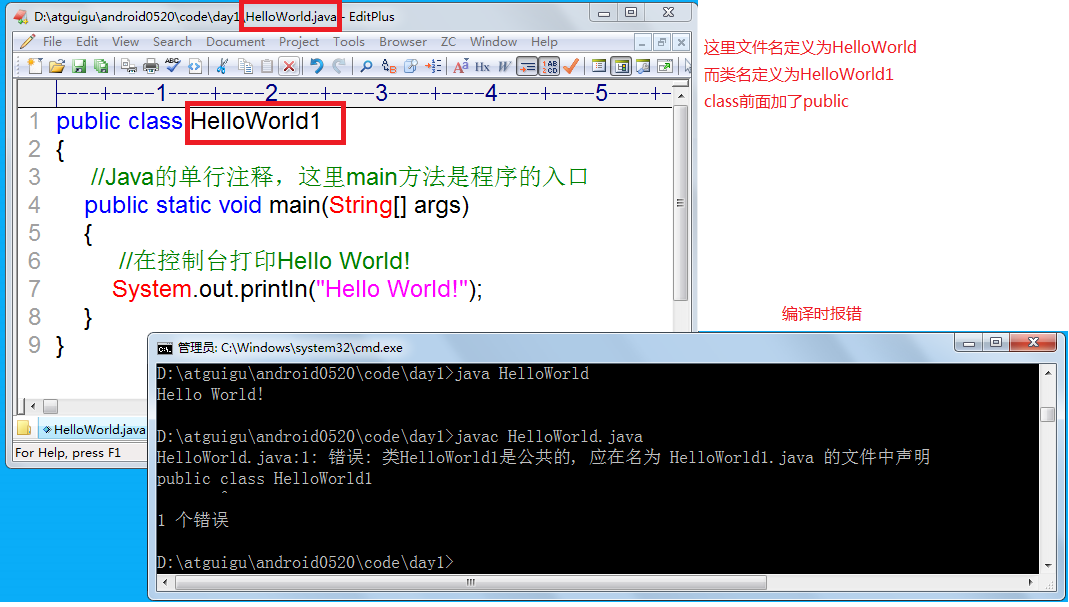
**3、一个源文件可以有多个class，编译后多个.class文件**

4、main()必须在public类中吗?

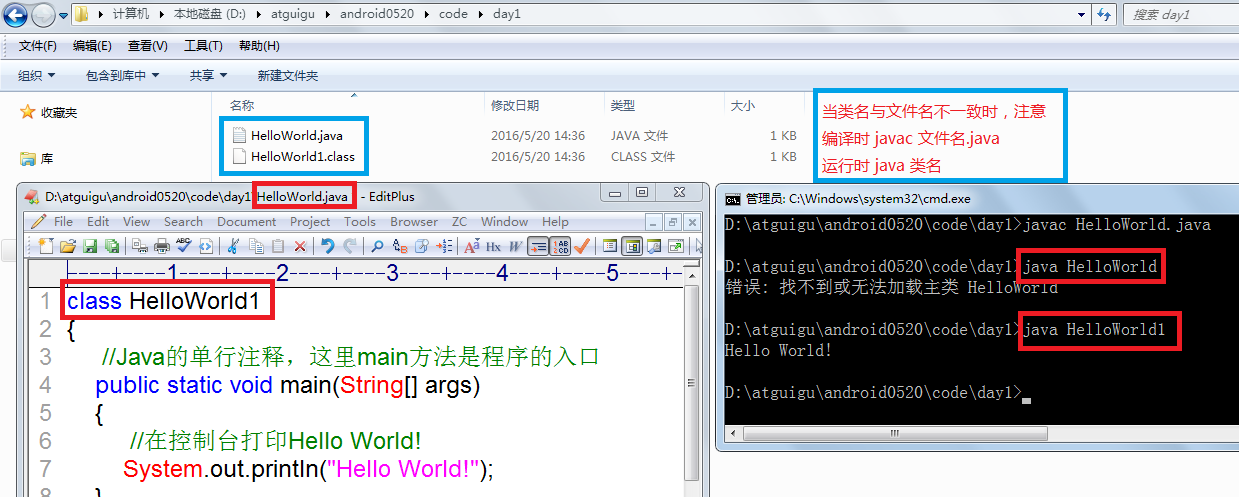
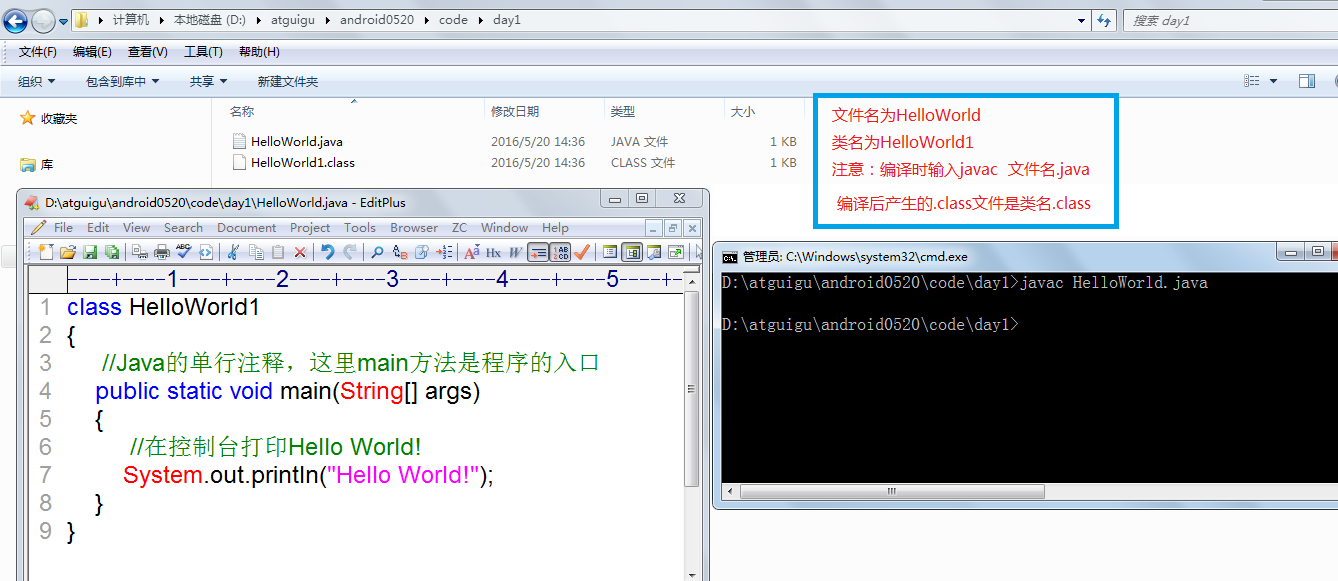
5、一个源文件只能有一个public类，而且类名与文件名一致？

答案：不是，如果类不是public修饰的，语法上不要求类名与文件名一致

但是如果这个类声明为public，那么必须要求类名与文件名一致

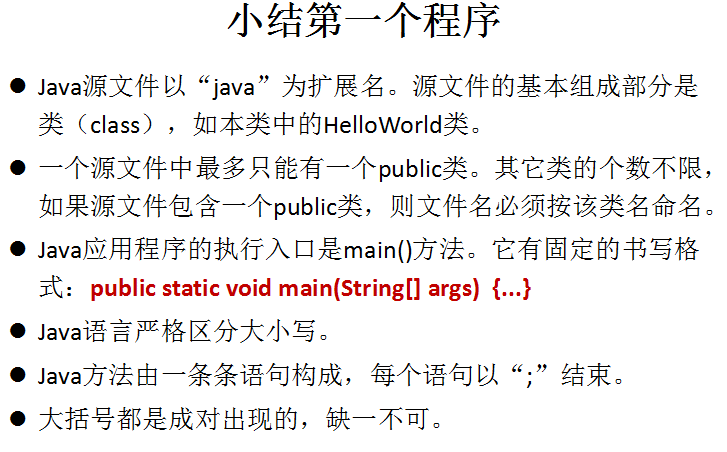


如果类不是public声明的，那么类名与文件名可以不一致，不过编译和运行时要注意区分文件名和类名



应该避免这种情况

### 3、小结



## 六、注释

用于说明解释程序的文字就是注释。

提高了代码的阅读性；调试程序的重要方法。

Java中的注释类型：

* 单行注释
* 多行注释
* 文档注释（java特有）

注释是一个程序员必须要具有的良好编程习惯。

将自己的思想通过注释先整理出来，再用代码去体现

### 单行注释和多行注释

单行注释： //注释文字

多行注释： /\* 注释文字 \*/

注：

对于单行和多行注释，被注释的文字，不会被JVM（java虚拟机）解释执行。

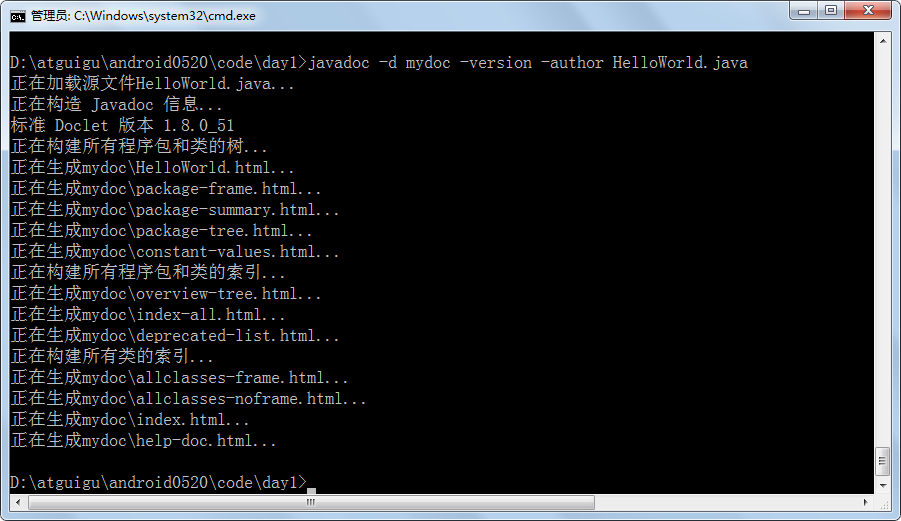
多行注释里面不允许有多行注释嵌套。

### 文档注释

文档注释：/\*\* 注释内容 \*/

注释内容可以被JDK提供的工具 javadoc 所解析，生成一套以网页文件形式体现的该程序的说明文档。

例如：javadoc -d mydoc -version -author HelloWorld.java





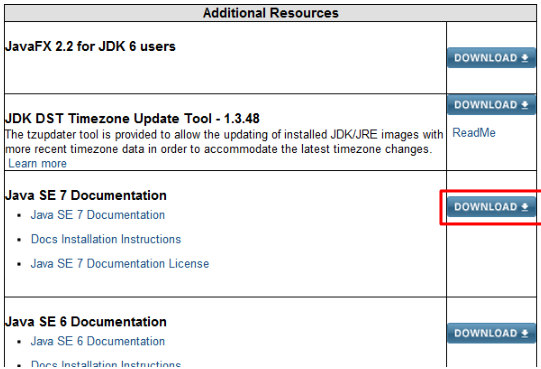
### Java API文档

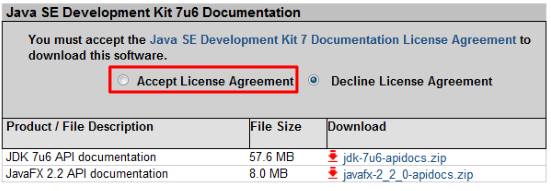
API （Application Programming Interface,应用程序编程接口）是 Java 提供的基本编程接口。

Java语言提供了大量的基础类，因此 SUN/Oracle 也为这些基础类提供了相应的API文档，用于告诉开发者如何使用这些类，以及这些类里包含的方法。

下载API：http://www.oracle.com/technetwork/java/javase/downloads/index.html

下拉滚动到Additional Resources-Java SE 7 Documentation下载。







## 七、练习

### 简答题

1、问题：System.out.println()和System.out.print()什么区别呢？

2、一个".java"源文件中是否可以包括多个类（不是内部类）？

3、源文件名是否必须与类名相同？如果不是，那么什么情况下，必须相同？

4、为什么要配置环境变量path

5、JDK,JRE和JVM的关系是什么？

### 编程题

1、将个人的基本信息（姓名、性别、年龄、学历、家庭住址、毕业学校）打印到控制台上输出。各条信息分别占一行。

2、编写一个Rectangle类，打印输出三角形如下：

\*

\*\*

\*\*\*

\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*\*

\*\*\*\*\*\*\*

3、结合\n(换行)，\t(制表符)，空格等在控制台打印出如下图所示的效果。



# 第二章 基础语法

## 一、标识符

Java 对各种变量、方法和类等要素命名时使用的字符序列称为标识符。

**凡是自己可以起名字的地方都叫标识符。**

### 1、标识符命名规则[☆]

定义合法标识符**规则（必死）**：

* **由26个英文字母大小写，0-9 ，\_或 $ 组成**
* **数字不可以开头，不能包含空格**
* 不可以使用关键字和保留字，但能包含关键字和保留字。
* **区分大小写，长度无限制。**

### 2、标识符命名规范

阅读性+有意义：“**见名知意**”

* **包名**：**xxxyyyzzz**
* **类名、接口名**：**XxxYyyZzz**
* **变量名、方法名**：**xxxYyyZzz**
* **常量名：XXX\_YYY\_ZZZ**

## 二、关键字、保留字、特殊值

关键字、保留字和特殊值：全部小写（50+3）

（1）关键字：在Java中被赋予的特定含义的单词。

基本数据类型相关（8个）：byte、short、int、long、float、double、char、boolean

流程控制语句相关（10个）：if、else、switch、case、default、break、for、while、do、continue

判断某个对象是否是某种类型的实例对象运算符：instanceof

定义类：class

创建类的对象：new

包相关：package、import

权限修饰符：public、protected、（缺省）、private

继承类：extends

定义接口：interface

实现接口：implements

当前对象：this

父类引用：super

表示无返回值：void

结束方法：return

定义枚举：enum

其他修饰符：abstract、static、final、native

异常处理：try、catch、finally、throws、throw

多线程同步和安全：synchronized、volatile

和IO序列化相关：transient

和单元测试相关：assert

其他：strictfp（strictfp 关键字可应用于类、接口或方法。如果你想让你的浮点运算更加精确，严格遵守FP-strict的限制,符合IEEE-754规范，而且不会因为不同的硬件平台所执行的结果不一致的话，可以用关键字strictfp.

）

1. 保留字：在Java中注册但还未使用

goto、const

（3）特殊值：true、false、null

## 三、变量

变量的作用：保存数据

变量的本质：内存的一块存储区域

变量的三要素：数据类型、变量名、变量值

数据类型：决定内存大小，可以存什么值

变量名：如何访问这块存储区域

变量值：里面存储的数据

声明变量：数据类型 变量名;

变量赋值：变量名 = 变量值;

声明变量和变量的赋值可以同时进行，也可以分开进行。**变量在被访问（读取）之前，必须有值。**

例如：数据类型 变量名 = 变量初始值;

变量的使用：通过变量名来访问这块存储区域

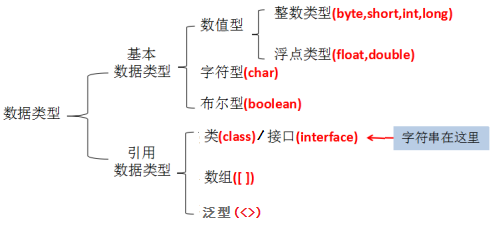
要求：

* **变量必须先声明后使用**
* **变量在使用之前必须有初始值**
* **变量有作用域**

|  |
| --- |
| /\*  变量：  某个值在程序中不同的地方都在使用，那么这个值就用一个变量保存。可以使用变量名来代表这个值。  变量本质对代表内存的一块存储区域。  变量的三要素：  1、变量名  比喻：房主，地址  2、数据类型（决定区域的大小，决定里面存的是什么类型的值）  比喻：面积，用途  3、变量值  比喻：里面的内容  如何使用变量？  1、变量必须先声明后使用  声明变量的格式：  数据类型 变量名;  例如：int age;  int num;  double score;  2、变量的使用之前，还必须初始化  变量赋值的格式：  变量名 = 变量值;  变量的声明和初始化可以在同一个语句完成。  例如：int age = 18;  **3、变量有作用域**  从声明处开始，到它所属的}结束  （1）同一个作用域中，一个变量声明一次  （2）同一个作用域变量不能重名  \*/  class BianLiang{  public static void main(String[] args){  //声明变量  int age;    //变量的赋值，第一次赋值叫做初始化  age = 18;  System.out.print("现在年龄：" + age);  System.out.print("明年的年龄：" + (age+1));  }    /\*  //age超过作用域  public static void test(){  System.out.print("现在年龄：" + age);  }  \*/  } |

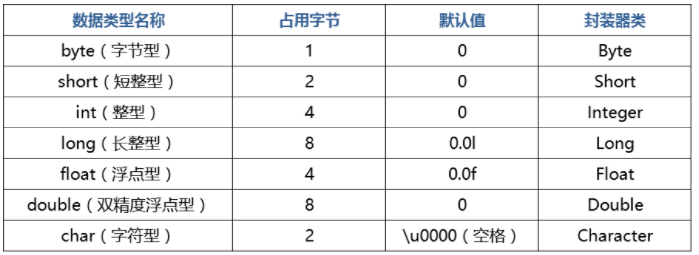
## 四、数据类型[☆]

哪些地方需要数据类型：变量（形参属于局部变量）、返回值



### 变量的数据类型

基本数据类型（8种）：



* byte：Java中最小的数据类型，在内存中占8位(bit)，即1个字节，取值范围-128~127，默认值0
* short：短整型，在内存中占16位，即2个字节，取值范围-32768~32717，默认值0
* int：整型，用于存储整数，在内在中占32位，即4个字节，取值范围-2147483648~2147483647，默认值0
* long：长整型，在内存中占64位，即8个字节-2^63~2^63-1，默认值0L
* float：浮点型，在内存中占32位，即4个字节，用于存储带小数点的数字（与double的区别在于float类型有效小数点只有6~7位），默认值0
* double：双精度浮点型，用于存储带有小数点的数字，在内存中占64位，即8个字节，默认值0
* char：字符型，用于存储单个字符，占16位，即2个字节，无负数，取值范围0~65535，默认值为空
* boolean：布尔类型，占1个字节，用于判断真或假（仅有两个值，即true、false），默认值false

32位操作系统：4字节来表示一个地址

这意味着32位操作系统，一个引用存储的对象地址占4字节

64位操作系统，一个引用存储的对象地址占8字节。

引用数据类型：

* 类（class）/接口（interface）----String字符串
* 数组（[]）
* 泛型（<>）

引用数据类型 为null 内存中也是0

注意：

* Java的整型常量默认为 int 型，如果想要表示某个整数值是**long型时，需要在值后面加L或l。**java程序中变量常声明为int型，除非不足以表示大数，才使用long
* Java 的浮点型常量默认为double型，**声明float型常量，须后加‘f’或‘F’**。float可以保证十进制科学计数法小数点后6位有效精度和第7位的部分精度。double可以保证十进制科学计数法小数点后15位有效精度和第16位的部分精度。
* **字符型的值必须使用单引号，只能而且必须存储单个字符，不能是空单引号**。例如：’a’、’\n’、'\u5c1a’。字符型使用Unicode编码（ASCII，特殊字符，中文、日文、韩文......）
* **String是引用数据类型，String的常量值必须使用双引号。单独的双引号也是一个字符串值。**

|  |
| --- |
| /\*  Java的数据类型：  （1）基本数据类型（8种）  （2）引用数据类型：类、数组、接口...  一、基本数据类型（8种）  bit：位，计算机中只有0和1，最小的存储单位  byte：字节，一个字节 = 8 位  1、数值型  （1）整型（整数）  byte：字节 1个字节 -128~127  short：短整型 2个字节 -32768~32767  int：整型（默认类型）4个字节  long：长整型 8个字节  特殊：long类型的数字后面需要加L或l  （2）浮点型（小数）  float：单精度浮点型（4个字节）  科学记数法：356.78 3.5678\*10^2  大约是：小数点后6~7位  特殊：float类型的数字后面需要加F或f  double：双精度浮点型（8个字节）默认类型  大约是：小数点后15~16位  2、字符型  char：字符型 2个字节 因为Java中所有的字符都使用Unicode字符集  Java中每一个字符，都有一个唯一的Unicode编码值。  可以存储一个字母，一个汉字，一个其他的字符  字符类型的值要使用单引号引起来''  字符串类型的值要使用双引号引起来""  char c = 'a';  char s = '女';  char line = '\n';//转义字符  char tab = '\t';  char d = '\b';  char b = '\'';  char b = '\"';  char c = '\u5c1a';//Unicode编码值的十六进制形式  3、布尔型，逻辑型  boolean：只有两个值，true和false  \*/  class LeiXing{  public static void main(String[] args){  byte b = 127;  System.out.println("b = " + b);//""中的内容原样输出    long num = 123456789012345678L;  System.out.println("num=" + num);    float f = 3.1F;  System.out.println("f=" + f);    double d = 3.1415926531234;  System.out.println("d=" + d);    //char gender = '女';  //String sex = "女";  //System.out.print("a\tb\nc\b");    char c = '尚';//char c = '\u5c1a';  System.out.print(c);    boolean beautiful = false;  if(beautiful){  System.out.println("柴老师是大美女");  }else{  System.out.println("柴老师是大大美女");  }  }  } |

|  |  |
| --- | --- |
| 文件保存的字符编码方式是GBK | D:\>javac Test.java -encoding GBK  通过  D:\>javac Test.java -encoding utf-8  报错 |
| 文件保存的字符编码方式是UTF-8 | D:\>javac Test.java -encoding utf-8  通过  D:\>javac Test.java -encoding GBK  报错 |
|  |  |

### 变量的运算之数据类型之间的转换

（1）自动类型转换（自动升级）：小容量转为大容量（系统自动完成）

**byte,short,char --> int -->long -->float -->double**

* + byte，short，char三者之间不进行运算，若运算都将自动升级为int再做运算
  + boolean不与其他任意数据类型转换
  + 任何数据类型与String之间使用“+”运算符都将自动“连接”成String类型

（2）强制类型转换：

大容量转小容量，需要使用**强制符()**，结果可能损失精度或溢出

|  |
| --- |
| /\*  基本类型的转换：  （1）自动类型转换  byte,short,char --> int -->long --> float --> double  1个字节，2个字节，2个字节 4个字节 8个字节 4个字节 8个字节  说明：  **byte,short,char之间是互不转换的，包括byte+byte等运算也会变成int**  boolean和谁都不转  String类型的数据与任意类型的数据“+”拼接后结果都是String字符串  ASCII码：  小写字母a：97  小写字母b：98  大写字母A：65  大写字母B：66  数字0：48  数字1：49  （2）强制类型转换  double --> float --> long -->int -->byte,short,char  需要强制类型转换符()  注意：强制类型转换可能损失精度或溢出  \*/  class TypeChange{  public static void main(String[] args){  byte b = 127; //把127赋值给b，存储到b变量中  int num = b; //把变量b中的值复制一份给num，也是说把变量b中的值赋值给num  System.out.println("b = " + b);  System.out.println("num = " + num);    char c = 'a';//Unicode编码值，兼容ASCII码  int letter = c;  System.out.println("c = " + c);//a  System.out.println("letter = " + letter);//97    byte b1 = 1;  byte b2 = 2;  byte b3 = (byte)(b1 + b2);  System.out.println("b3 = " + b3);    char c1 = 'a';  char c2 = (char)(c1 + 1);  System.out.println("c2 = " + c2);    double d = 10.56;  int a = (int)d;  System.out.println("a = " + a);//损失精度    int age = 130;  byte bb = (byte)age;//溢出  System.out.println("bb = " + bb);  }  } |

## 五、进制

### 1、进制

计算机中任何的计算和存储都是以二进制方式实现的

位（bit） — 是计算机中最小的存储单位

字节（byte） — 计算机中基本的存储单元

1byte = 8bits、1KB = 1024Byte、1MB = 1024KB、1GB = 1024MB、1T = 1024GB

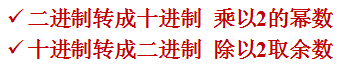
对于整数，有四种表示方式：

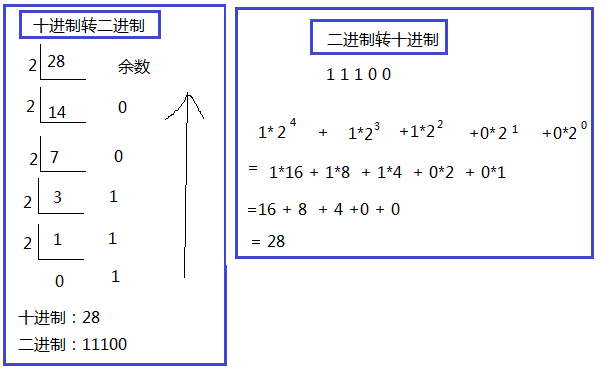
* 二进制：0,1 ，满2进1**.以0b或0B开头**（字母B大小写均可）。
* 十进制：0-9 ，满10进1.
* 八进制：0-7 ，满8进1.八进制数被冠以0来表示。
* 十六进制：0-9及A-F，满16进1，十六进制数被冠以0X来表示（字母X大小写均可）

如： 0x3f20（十六进制） 0732 （八进制）

|  |
| --- |
| public class JinZhi{  public static void main(String[] args){  //最终打印的结果是十进制  System.out.println("十进制：" + 10);  System.out.println("二进制：" + 0B10);  System.out.println("八进制：" + 010);  System.out.println("十六进制：" + 0X10);  }  } |

1、二进制与十进制之间的转换

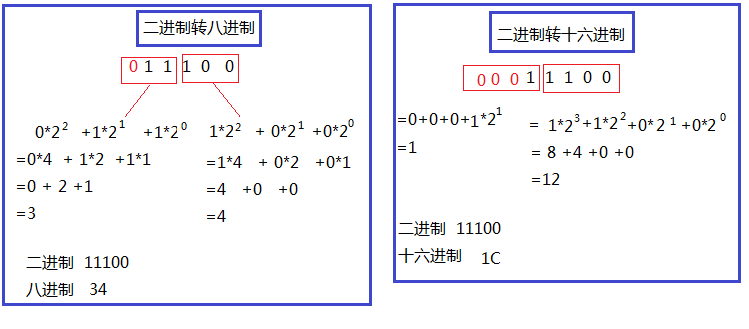




2、二进制 八进制 十六进制互转

二进制转八进制：**三位一组**，算出这三位对应的十进制值，然后挨个连接上即可

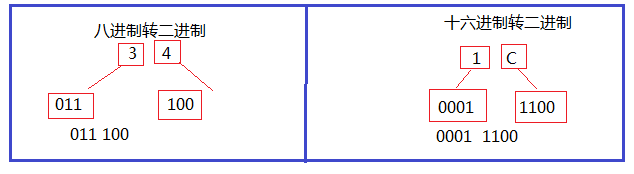
二进制转十六进制：**四位一组**，算出这四位对应的十进制值，然后挨个连接上即可



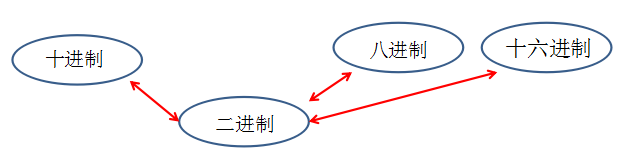
2、二进制 八进制 十六进制互转

二进制转八进制：**三位一组**，算出这三位对应的十进制值，然后挨个连接上即可

二进制转十六进制：**四位一组**，算出这四位对应的十进制值，然后挨个连接上即可



3、十进制与八进制或十进制与十六进制之间的转换，建议通过二进制过渡一下



### 2、原码、反码、补码

所有数字在计算机底层都以二进制形式存在。一个数在计算机中的二进制表示形式,  叫做这个数的**机器数**。机器数是带符号的，在计算机用一个数的最高位存放符号, 正数为0, 负数为1。

Java整数常量默认是int类型，当用二进制定义整数时，其第32位是符号位；当是long类型时，二进制默认占64位，第64位是符号位。

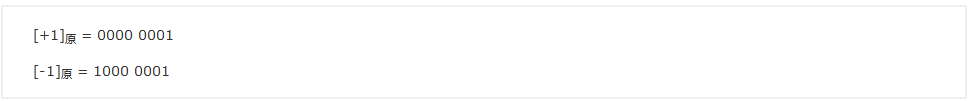
因为机器数是带符号的，因此机器数与一个数的“二进制值”还是有区别的。

正数的原码、反码、补码都相同

负数的补码是其反码加1。

1、原码

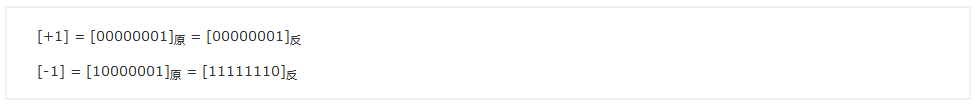
原码即用第一位表示符号, 其余位表示值. 比如如果是8位二进制:



2、反码

正数的反码是其本身

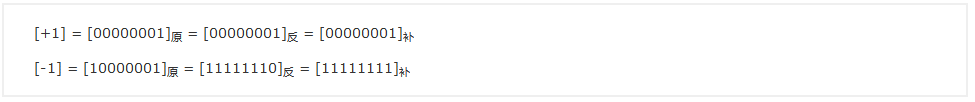
负数的反码是在其原码的基础上, 符号位不变，其余各个位取反.



3、补码

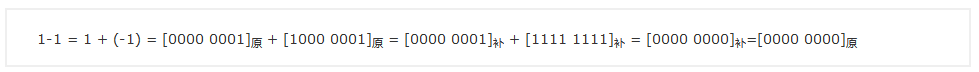
正数的补码就是其本身

负数的补码是在其原码的基础上, 符号位不变, 其余各位取反, 最后+1. (即在反码的基础上+1)



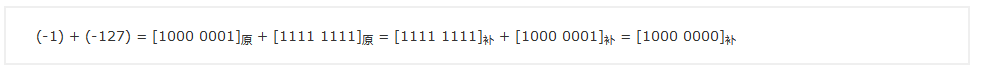
为什么要使用原码、反码、补码表示形式呢？

计算机辨别"符号位"显然会让计算机的基础电路设计变得十分复杂! 于是人们想出了将符号位也参与运算的方法. 我们知道, 根据运算法则减去一个正数等于加上一个负数, 即: 1-1 = 1 + (-1) = 0 , 所以机器可以只有加法而没有减法, 这样计算机运算的设计就更简单了. **实际运算都是补码在运算，运算完再转成原码**



|  |
| --- |
| 0000 0000：0  0000 0001 ~ 0111 1111 ： 1~ 127  1000 0001 ~ 1111 1111 ： -127 ~ -1  1000 0000：-128  （-127） - （1） = -128  （1000 0001 - （0000 0001）= （1000 0000） |

**特殊值：**

****

-1-127的结果应该是-128, 在用补码运算的结果中, **[1000 0000]补 就应该是-128**.

但是此时-128，如果用补码到反码，再到原码去计算是不对的，所以**-128是个特殊值，记住它**

## 六、运算符

运算符：

1、算术运算符

2、赋值运算符

3、比较运算符（关系运算符）

4、逻辑运算符

5、条件运算符

6、位运算符

表达式：操作数+运算符

### 1、算术运算符：

1. 加、减、乘、除：+、-、\*、/

除：当两个整数相除时，只保留整数部分

1. 取模（求余数）：%

**取模：被模数%模数，当模数有负号时，会被忽略**

（3）+：作为单目运算时表示正数

与其他基本数据类型计算时当做加法用

当与字符串String类型进行运算时表示连接，计算结果是字符串String类型

1. -：作为单目运算符时表示负数

与其他基本数据类型计算时当做减法用

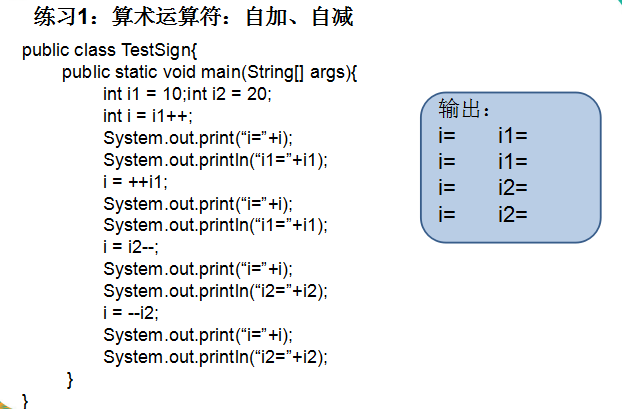
（5）++，-- 自增和自减

++在前，变量先自增，然后参与其他运算，在后，变量的值先参与其他运算，然后变量本身在自增

--类似

|  |
| --- |
| /\*  算术：arithmetic  一、算术运算符  （1）加：+  （2）减：-  （3）乘：\*  （4）除：/  特殊：如果是两个整数相除，结果只保留整数部分  （5）取余，取模：% 被模数%模数  特殊：模数的负号被忽略  （6)在JAVA中：+ 还表示拼接  只要+两边有一个是字符串，那么就是拼接，结果仍然是字符串  a + "+"：a变量的值 拼接上 “+”符号 23+  （7）正号 +  （8）负号 -  （9）自增 ++  对于自增变量本身来说，++在前或在后都一样，自增变量都要加1  i++或++i，i就是自增变量    对于表达式来说，i++和++i不一样的，++在前，先自增，再进行其他运算，++在后，先进行其他运算，然后再自增  （10）自减 --  同自增  \*/  class SuanShu{  public static void main(String[] args){  int a = 23;  int b = 12;  System.out.println(a + "+" + b + "=" + (a+b));  int sum = a + b;  System.out.println(a + "+" + b + "=" + sum);    int sub = a - b;  //System.out.println(a + "-" + b + "=" + a-b);//错误，原因是a + "-" + b + "=" + a的结果是字符串，字符串不能进行减法  System.out.println(a + "-" + b + "=" + (a-b));  System.out.println(a + "-" + b + "=" + sub);    int mul = a \* b;  System.out.println(a + "\*" + b + "=" + a\*b);  System.out.println(a + "\*" + b + "=" + mul);    //整数相除，只保留整数部分  int div = a / b;  System.out.println(a + "/" + b + "=" + a/b);  System.out.println(a + "/" + b + "=" + div);    double d = (double)a/b;//先把a的类型进行转换，转换成double类型，然后再和b相除  System.out.println(a + "/" + b + "=" + d);    int yu = a % b;  System.out.println(a + "%" + b + "=" + yu);    System.out.println("------------------特殊的取模----------------------");  System.out.println(5%2);//1  System.out.println(-5%-2);//-1  System.out.println(-5%2);//-1  System.out.println(5%-2);//1    System.out.println("------------------负号----------------------");  int num1 = 12;  int num2 = -num1;  System.out.println("num2=" + num2);    System.out.println("------------------自增----------------------");  int i = 0;  System.out.println("自增之前i=" + i);  i++;  System.out.println("自增第一次之后i=" + i);  ++i;  System.out.println("自增第二次之后i=" + i);  int j = ++i;//把i自增1，然后结果赋值给j，或者说，先算++i，然后再赋值  System.out.println("自增第三次之后i=" + i);  System.out.println("j="+j);  int k = i++;//先算赋值，把i的值赋值给k，i原来是3，把3赋值给k，然后i在自增1，i变成4  System.out.println("自增第四次之后i=" + i);  System.out.println("k="+k);    //面试题：陷阱题  i = i++;//先赋值，把i原来的值重新赋值给i，不变，然后i自增，但是这个自增后的值没有在放回变量i的位置  System.out.println("自增第五次之后i=" + i);  }  } |

#### 练习1



|  |
| --- |
| public class TestSign{  public static void main(String[] args){  int i1 = 10;  int i2 = 20;  int i = i1++;//把i1自增之前的值赋值给i，然后i1再自增1  System.out.print("i="+i);//10  System.out.println("，i1="+i1);//11    i = ++i1;//把i1自增之后的值赋值给i  System.out.print("i="+i);//12  System.out.println("，i1="+i1);//12    i = i2--;//把i2自减之前的值赋值给i，然后i2自减1  System.out.print("i="+i);//20  System.out.println("，i2="+i2);//19    i = --i2;//把i2自减之后的值赋值给i，先自减再赋值  System.out.print("i="+i);//18  System.out.println("，i2="+i2);//18  }  } |

#### 练习2

练习:

随意给出一个三位数，打印显示它的个位数，十位数，百位数的值。

格式如下：

数字xx的情况如下：

个位数：

十位数：

百位数：

例如：

数字153的情况如下：

个位数：3

十位数：5

百位数：1

|  |
| --- |
| class TestBaiShiGe{  public static void main(String[] args){  int num = 153;  int bai = num / 100;//因为两个整数相除，只保留整数部分  int ge = num % 10;  //int shi = num / 10 % 10;//num / 10得到15，然后15%10得到5  int shi = num % 100 / 10;//num%100得到53，然后53/10得到5    System.out.println("num的");  System.out.println("百位是：" + bai);  System.out.println("十位是：" + shi);  System.out.println("个位是：" + ge);  }  } |

### 2、赋值运算符：

基本的赋值运算符：=

扩展的赋值运算符：+= -= \*= /= %= &= ....

扩展的赋值运算符的说明：中间不能有空格，而且结果隐含数据类型转换

例如：short s1 = 1;

short s2 = 2;

s2 += s1;

|  |
| --- |
| /\*  赋值：assign  最基本的赋值运算符：=  Java中赋值，永远是把等号=右边的赋值给左边的变量。  右边如果是常量值，那么就把常量的值直接赋值给左边的变量；  右边如果是变量，那么就把变量的值直接赋值给左边的变量；  右边如果是表达式，那么就把表达式的运算结果直接赋值给左边的变量；  扩展的赋值运算符：  +=，-=，\*=，/=，%=  注意：  （1）+=等，中间是不能有空格的，即不能写成 + =  **（2）如果结果的类型与左边的变量不在一样时，隐含了强制类型转换**  \*/  class FuZhi{  public static void main(String[] args){  int a = 10;//把10赋值给a  int b = a; //把a的值赋值给b  int c = a + b;//把a+b的运算结果赋值给c    //20 = a;//错误的    System.out.println("-------------------------------");  byte b1 = 1;  byte b2 = 127;  //b2 = b1 + b2;//右边的结果是int类型，需要强制类型转换  b2 += b1;//把左边的变量的值 + 右边的变量的值，结果再赋值给左边的变量，如果计算的结果是升级后的结果，会强制类型转换  System.out.println("b1=" + b1);  System.out.println("b2=" + b2);    System.out.println("-------------------------------");  int i = 2;  int j = 3;  j \*= i + 1;//等价于 j = j \* (i+1);  System.out.println("i=" + i);  System.out.println("j=" + j);    System.out.println("-------------------------------");  int m = 2;  int n = 3;  n \*= m++;// (1)n = n \* m; (2)m++;  System.out.println("m=" + m);//对于自增变量来说，++在前还是在后都要自增1  System.out.println("n=" + n);    System.out.println("-------------------------------");  int x = 2;  int y = 3;  y \*= ++x;// (1)++x; (2)y = y \* x;  System.out.println("x=" + x);//对于自增变量来说，++在前还是在后都要自增1  System.out.println("y=" + y);    System.out.println("-------------------------------");  i = 1;  i \*= 0.1;//把double类型强制转换成int  System.out.println(i);//0  i++;  System.out.println(i);//1  }  } |

### 3、关系运算符

关系运算符，比较运算符：运算的结果只有true或false的布尔值

（1）> < >= <= !=

（2）== 判断是否相等，一定要与=赋值运算符区分开

（3）instanceof，引用数据类型的关系运算符后面再讲

|  |
| --- |
| /\*  比较：compare  关系：relation  比较运算符又称为关系运算符：  1、大于：>  2、小于：<  3、大于等于：>=(中间不能有空格)  4、小于等于：<=  5、是否等于：==  6、是否不等于：!=  条件判断时，常常使用比较运算符  比较运算符的表达式的结果只有两种：true或false，即结果是一个boolean值  \*/  class BiJiao{  public static void main(String[] args){  int age = 19;    System.out.println(age >= 18);  System.out.println(age < 18);    System.out.println("------------------------------");    System.out.println(age == 18);//比较运算符的表达式，结果是true  System.out.println(age = 18);//赋值表达式，结果是18    boolean result = false;  if(result = true){//变成了一个赋值表达式  System.out.println("条件成立");  }  System.out.println("result="+result);    /\*  编译错误(age=18)不是比较表达式，结果也不是true或false，条件只能是布尔值  if(age = 18){  System.out.println("刚刚成年");  }  \*/    /\*  if(age>=18){  System.out.println("欢迎进入成人世界！");  }else{  System.out.println("未成年不允许进入！");  }  \*/  }  } |

### 4、逻辑运算符 【左右两边是布尔值】

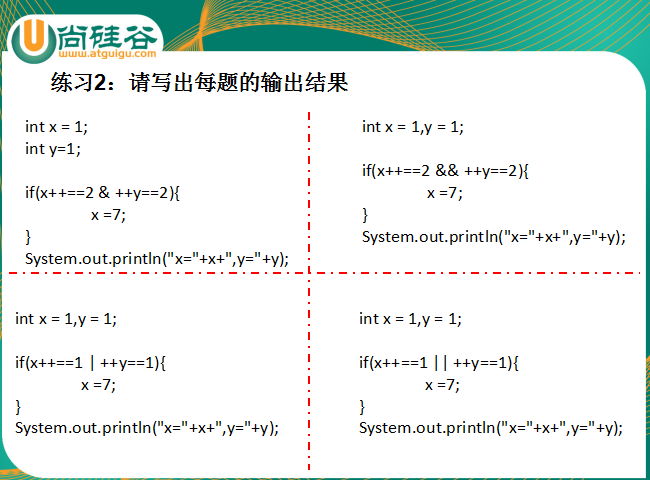
逻辑运算符：结果也是布尔型，而且**操作数也必须是布尔型**

|  |
| --- |
| 与， & 表示&符号两边的表达式/变量的值都必须是true，整个逻辑表达式的结果才为true |
| 或 | 表示|符号两边的表达式/变量的值只要有一个true，整个逻辑表达式的结果就为true |
| 非 ! 表示逻辑值取反 |
| 短路与 && 表示&&符号两边的表达式/变量的值都必须是true，整个逻辑表达式的结果才为true  但是，如果左边的表达式已经是false时，右边的就不再计算，不看了 |
| 短路或 || 表示||符号两边的表达式/变量的值只要有一个true，整个逻辑表达式的结果就为true  但是，如果左边的表达式已经为true时，右边的就不再计算 |
| 异或运算 ^ 求的就是个性，不同，两个操作数不同时，结果为真，如果相同就为假 |

|  |  |  |
| --- | --- | --- |
| true & true 结果才为true  true & false 结果为false  false & true 结果为false  false & false 结果为false | false | false 结果为false  false | true 结果为true  true | false 结果为true  true | true 结果为true | !false 结果为true  !true 结果为false |
| true ^ true 结果为false  true ^ false 结果为true  false ^ true 结果为true  false ^ false 结果为false | true && true 结果才为true  true && false 结果为false  false && true 结果为false  false && false 结果为false | false || false 结果为false  false || true 结果为true  true || false 结果为true  true || true 结果为true |

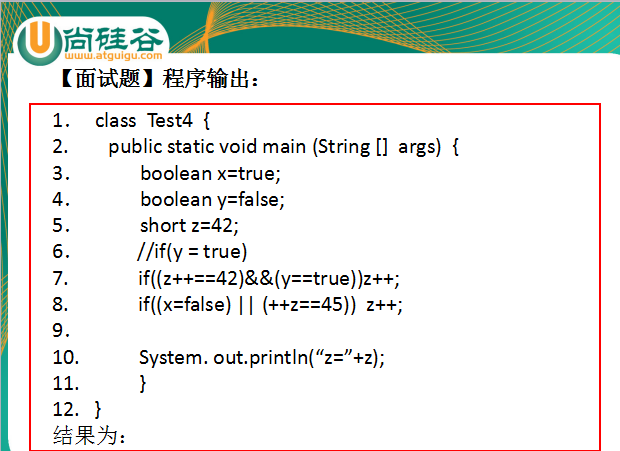
|  |
| --- |
| /\*  逻辑：logic,logical  逻辑运算符：  (1)逻辑与：&  表示&左右两边的条件要同时成立  true & true 结果才为true  true & false 结果为false  false & true 结果为false  false & false 结果为false  (2)逻辑或： |  表示|左右两边的条件只要有一个成立就可以  false | false 结果为false  false | true 结果为true  true | false 结果为true  true | true 结果为true  (3)逻辑非：!  !false 结果为true  !true 结果为false  (4) 逻辑异或：^  ^左右两边的条件的结果相反时为true  true ^ true 结果为false  true ^ false 结果为true  false ^ true 结果为true  false ^ false 结果为false  (5)短路与：&&  表示&&左右两边的条件要同时成立  true && true 结果才为true  true && false 结果为false  false && true 结果为false  false && false 结果为false  同逻辑与&有所不同：当&&左边的表达式为false时，右边的表达式不看了  (6)短路或：||  表示||左右两边的条件只要有一个成立就可以  false || false 结果为false  false || true 结果为true  true || false 结果为true  true || true 结果为true  和逻辑或|有所不同：当||左边的表达式为true时，右边的表达式不看了    二元运算符：运算符需要两个操作数  a + b， a \* b， a = b， a > b，...  一元运算符： 运算符需要一个操作数  ++，--，正号，负号  i++，+a  \*/  class Logic{  public static void main(String[] args){  //成绩  int score = -75;    //如果成绩在80~90之间，我们显示成绩良好  //80<=score<=90  /\*错误  if(80 <= score <= 90){  System.out.println("良好");  }  \*/    if(score>=80 & score<=90){  System.out.println("良好");  }    //如果成绩小于0或大于100，我们显示超过合理的成绩范围  if(score<0 | score>100){  System.out.println("超出成绩范围");  }  System.out.println("------------------------------------");  System.out.println(true & true);//true  System.out.println(true & false);//false  System.out.println(false & true);//false  System.out.println(false & false);//false    System.out.println("------------------------------------");  System.out.println(true | true);//true  System.out.println(true | false);//true  System.out.println(false | true);//true  System.out.println(false | false);//false    System.out.println("------------------------------------");  int num = 100;  boolean flag = num<100 | num>100;  System.out.println("flag = " + flag);  if(!flag){  System.out.println("num等于100");  }  System.out.println("------------------------------------");  System.out.println(true ^ true);//false  System.out.println(true ^ false);//true  System.out.println(false ^ true);//true  System.out.println(false ^ false);//false    System.out.println("------------------------------------");  int i = 0;  int j = 1;  if(i>0 && ++j > 1){  i = 20;  }  System.out.println("i=" + i);  System.out.println("j=" + j);  }  } |

#### 练习1



|  |  |
| --- | --- |
| public class LogicExer1{  public static void main(String[] args){  int x = 1;  int y = 1;  //x==2 ,x++ false x = 2 左边为false  //右边继续  //++y y==2 y=2 y==2成立 右边为true  //整个条件还是false  if(x++==2 & ++y==2){  x =7;  }  System.out.println("x="+x+",y="+y);//x=2,y=2  }  } | public class LogicExer2{  public static void main(String[] args){  int x = 1,y = 1;  //x==2,x++ 左边条件为false，x=2  //因为短路与，右边不算  //整个条件为false  if(x++==2 && ++y==2){  x =7;  }  System.out.println("x="+x+",y="+y);//x=2,y=1  }  } |
| public class LogicExer3{  public static void main(String[] args){  int x = 1,y = 1;  //x==1,x++ 左边为true，x=2  //因为是逻辑与,右边继续  //++y, y==1 y=2 右边为false  //条件true | false，最终为true  if(x++==1 | ++y==1){  x =7;  }  System.out.println("x="+x+",y="+y);//x=7,y=2  }  } | public class LogicExer4{  public static void main(String[] args){  int x = 1,y = 1;  //x==1,x++ 左边为true，x=2  //因为是短路或，左边为true，右边就不看了  //整个条件为true  if(x++==1 || ++y==1){  x =7;  }  System.out.println("x="+x+",y="+y);//x=7,y=1  }  } |

#### 练习2



|  |
| --- |
| public class LogicExer5{  public static void main (String [] args) {  boolean x = true;  boolean y = false;  short z = 42;    //如果if((z++==42)&&(y==true))条件成立，执行z++，不成立，就不执行z++  //左边的条件：z==42,z++ z==42成立,z++变成43  //中间是短路与，因为左边现在是true,右边还要看  //右边 y==true 不成立  //true && false 结果为false  if((z++==42)&&(y==true)) z++;    //左边为x=false,赋值 结果就为false  //中间为短路或，因为左边是false,右边继续看  //++z,z==45 ++z变成44，z==45是否成立，不成立  //false || false 结果为false  if((x=false) || (++z==45)) z++;  System. out.println("z="+z);//44  }  } |
| class Test4\_2 {  public static void main (String [] args) {  boolean x = true;  boolean y = false;  short z = 42;    //如果if(y=true)条件成立，接着判断if((z++==42)&&(y==true)) z++;  //如果不成立，if((z++==42)&&(y==true)) z++; 不看的  /\*  if(y = true)    if((z++==42)&&(y==true)) z++;    if((x=false) || (++z==45)) z++;  \*/    //标准  //y=true赋值，y就被修改为true,if(true)成立  if(y=true){  //左边：z==42,z++ 成立,z变成43  //&&短路与，不满足短路的情况，右边继续  //y==true 成立  //true && true，结果为true  if((z++==42)&&(y==true)){  //z++变成44  z++;  }  }  //左边：x=false不成立  //中间虽然是短路或，但是没满足短路的情况，右边继续  //++z,z==45 ++z变成45，z==45成立  if((x=false) || (++z==45)){  //z++，变成46  z++;  }  System. out.println("z="+z);//46  }  } |

### 5、三元条件运算符

三元运算符：格式 条件表达式?表达式1:表达式2

注意条件表达式结果必须是布尔类型

当条件表达式的结果为true,那么整个三元运算符的结果就取表达式1的结果，否则就取表达式2的结果

案例：求两个数最大值

求三个数最大值

|  |
| --- |
| //找出三个整数中的最大值  public class FindMax{  public static void main(String[] args){  int x = 3;  int y = 2;  int z = 5;    int max = x>=y ? x : y;  //运行到这里，max中存的是x,y中较大者    max = max >= z ? max : z;    System.out.println("max = " + max);  }  } |

### 6、位运算符[**操作数是整数**]

|  |
| --- |
| 左移 << ：右边补0 |
| 右移 >> ：左边补0或1，原数最高位是1，就补1，原数最高位是0，就补0 |
| 无符号右移 >>>：左边补0 |
| 按位与 & ：二进制对应位置取与 ，同时为1才为1，否则为0 |
| 按位或 | ：二进制对应位置取或 ，有一个为1就为1 |
| 按位异或运算 ^ ：二进制对应位置取异或 ，两者不同才为1 |
| 按位取反 ~ ：二进制对应位置取反 ，原来是1，变为0，原来是0变为1 |

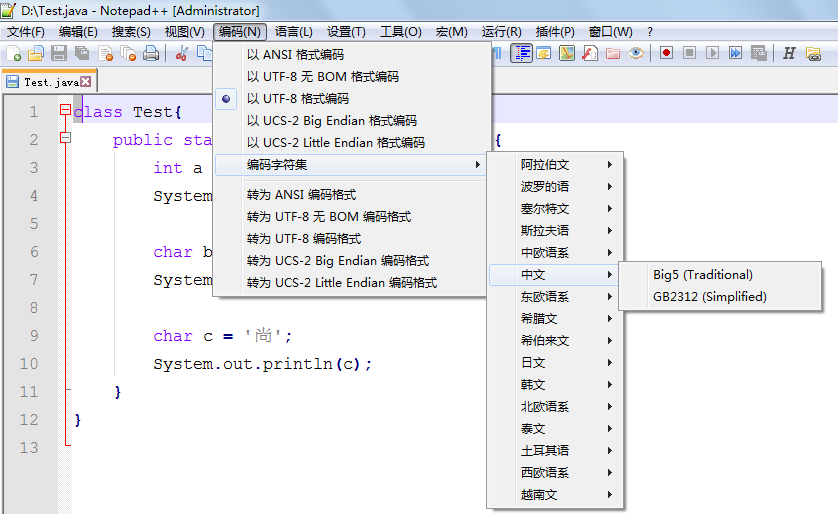
说明：位运算符都是机器数直接运算的

|  |  |  |  |
| --- | --- | --- | --- |
| <<：  左移几位，乘以2的几次方 | >>：  右移几位，除以2的几次方 | >>>：  对于正数来说，结果和>>一样，  对于负数来说，>>>后变成正数 | |
| 0 & 0 结果是0  0 & 1 结果是0  1 & 0 结果是0  1 & 1 结果是1 | 0 | 0 结果是0  0 | 1 结果是1  1 | 0 结果是1  1 | 1 结果是1 | 0 ^ 0 结果是0  0 ^ 1 结果是1  1 ^ 0 结果是1  1 ^ 1 结果是0 | ~1 结果为0  ~0 结果为1 |

代码示例：

|  |
| --- |
| public class BitOperator{  public static void main(String[] args){  int a = 8;    //a的二进制：24个零 0000 1000  //a<<2  //24个零 0010 0000  System.out.println(a << 2);    //a的二进制：24个零 0000 1000  //a>>2  //24个零 000000 10  System.out.println(a >> 2);    **int b = -8;**  **//b的二进制：**  **//原码 1 23个0 0000 1000**  **//反码：1 23个1 1111 0111**  **//补码：1 23个1 1111 1000**  **//左移b<<2，右边补0**  **//1 23个1 1110 0000 补码 【可以看出实际的运算是补码】**  **//反码：1 23个1 1101 1111**  **//原码：1 23个0 0010 0000 -32**  **System.out.println(b << 2);**    **//b的二进制：**  **//原码 1 23个0 0000 1000**  **//反码：1 23个1 1111 0111**  **//补码：1 23个1 1111 1000**  **//右移：b>>2，左边补什么，看最高位，**  **//补码：1 23个1 111111 10**  **//反码：1 23个1 111111 01**  **//原码：1 23个0 0000 0010 -2**  **System.out.println(b >> 2);**    //无符号右移  //b的二进制：  //原码 1 23个0 0000 1000  //反码：1 23个1 1111 0111  //补码：1 23个1 1111 1000  //无符号右移：b >>> 2,不看最高位，左边直接补0  //00 1 23个1 1111 10  System.out.println(b >>> 2);    System.out.println("-----------------------------");  //a的二进制：24个零 0000 1000  //b的二进制：  //原码 1 23个0 0000 1000  //反码：1 23个1 1111 0111  //补码：1 23个1 1111 1000  //24个零 0000 1000 & 1 23个1 1111 1000  //24个零 0000 1000 -》 8  System.out.println(a & b);    //24个零 0000 1000 | 1 23个1 1111 1000  //1 23个1 1111 1000 ->-8  System.out.println(a | b);    //24个零 0000 1000 | 1 23个1 1111 1000  //24个1 1111 0000补码  //反码：24个1 1110 1111  //原码：1 23个0 0001 0000 -》 -16  System.out.println(a ^ b);    //a的二进制：24个零 0000 1000  //~a  //24个1 1111 0111补码  //反码：24个1 1111 0110  //原码：1 23个0 0000 1001  System.out.println(~a);  }  } |

## 七、字符编码（了解）



首先来看一下常用的编码有哪些，截图自Notepad++。**其中ANSI在中国大陆即为GBK（以前是GB2312）**，最常用的是 GBK 和 UTF8无BOM 编码格式。

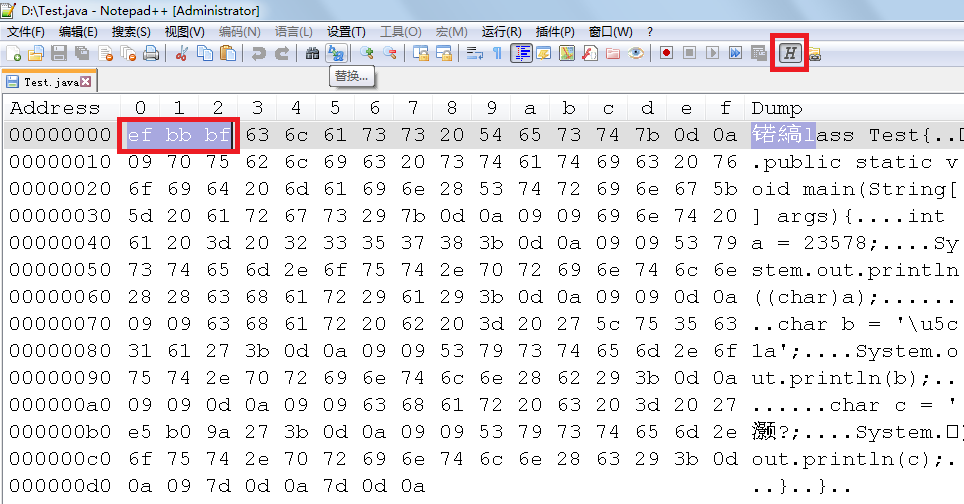
后面三个都是有BOM头的文本格式，UCS-2即为人们常说的Unicode编码，又分为大端、小端。

所谓BOM头（Byte Order Mark）就是文本文件中开始的几个并不表示任何字符的字节，用二进制编辑器（如bz.exe）就能看到了。

UTF8的BOM头为 0xEF 0xBB 0xBF

Unicode大端模式为 0xFE 0xFF

Unicode小端模式为 0xFF 0xFE



### ASCII码

计算机中**字符-数字的对应关系**必须得一致，否则就会造成同一段数字在不同计算机上显示出来的字符不一样。

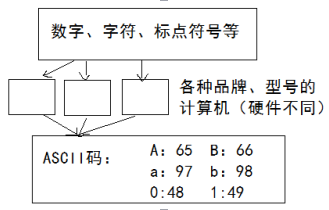
因此美国国家标准协会ANSI制定了一个标准，规定了常用字符的集合以及每个字符对应的编号，这就是ASCII字符集（Character Set），也称ASCII码。

**编码：字符序列编码为二进制流**

**解码: 二进制流解码成字符**

其中：

* 0～31及127(共33个)是控制字符或通信专用字符（其余为可显示字符），如控制符：LF（换行）、CR（回车）、FF（换页）、DEL（删除）、BS（退格)
* 32～126(共95个)是字符(32是空格），其中48～57为0到9十个阿拉伯数字。
* 65～90为26个大写英文字母，97～122号为26个小写英文字母，其余为一些标点符号、运算符号等。
* 后128个称为扩展ASCII码。许多基于x86的系统都支持使用扩展（或“高”）ASCII。扩展ASCII 码允许将每个字符的第8 位用于确定附加的128 个特殊符号字符、外来语字母和图形符号。



### 多字节字符集（MBCS）和中文字符集

**单字节编码**：一个字节翻译成一个字符。这对于拉丁语系国家来说可能没有什么问题，因为他们通过扩展第8个比特，就可以得到256个字符了，足够用了。

对于亚洲国家来说：**字节字符集（Muilti-Bytes Charecter Set**）

例如中国使用的就是双字节字符集编码。

例如目前最常用的中文字符集GB2312，涵盖了所有简体字符以及一部分其他字符；GBK（K代表扩展的意思）则在GB2312的基础上加入了对繁体字符等其他非简体字符。这两个字符集的字符都是使用1-2个字节来表示。Windows系统采用936代码页来实现对GBK字符集的编解码。在解析字节流的时候，如果遇到字节的最高位是0的话，那么就使用936代码页中的第1张码表进行解码，这就和单字节字符集的编解码方式一致了。如果遇到字节的最高位是1的话，那么就表示需要两个字节值才能对应一个字符。

假如你使用GB2312写了这么一句话：

我叫ABC

它的二进制编码是这样的：

11001110 11010010 10111101 11010000 01000001 01000002 01000003

全角？

全角是一种电脑字符，且每个全角字符占用两个标准字符（或半角字符）位置。通常的英文字母、数字键、符号键都是半角的，半角的显示内码都是一个字节。为了排列整齐，英文和其它拉丁文的字符和标点也提供了全角格式。在中文输入法中，切换全角和半角格式的快捷键为SHIFT+空格。

### ANSI标准、国家标准、ISO标准

美国ANSI组织制定了ANSI标准字符编码

我们现在通常说到ANSI编码，通常指的是平台的默认编码，例如英文操作系统中是ISO-8859-1，中文系统是GBK），

ISO组织制定的各种ISO标准字符编码，还有各国也会制定一些国家标准字符集，例如中国的GBK，GB2312和GB18030。

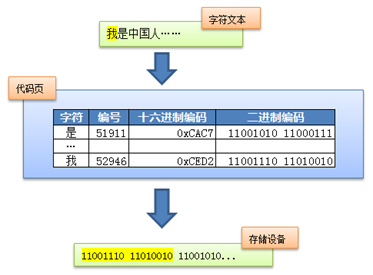
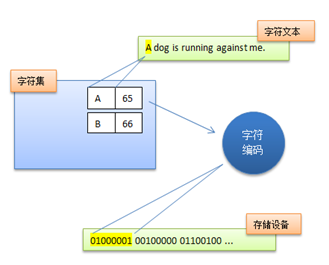
### Unicode的出现

虽然通过使用不同字符集，我们可以在一台机器上查阅不同语言的文档，但是我们仍然**无法解决一个问题：如果一份文档中含有不同国家的不同语言的字符，那么无法在一份文档中显示所有字符**。为了解决这个问题，我们需要一个全人类达成共识的巨大的字符集，这就是Unicode字符集。

Unicode字符集涵盖了目前人类使用的所有字符，并为每个字符进行统一编号，分配唯一的字符码（Code Point）。

#### 编码系统的变化

**在Unicode出现之前，所有的字符集都是和具体编码方案绑定在一起的（即字符集≈编码方式）**，都是直接将字符和最终字节流绑定死了，例如ASCII编码系统规定使用7比特来编码ASCII字符集；GB2312以及GBK字符集，限定了使用**最多**2个字节来编码所有字符，并且规定了字节序。这样的编码系统通常用简单的查表，也就是通过代码页就可以直接将字符映射为存储设备上的字节流了。例如下面这个例子：

Unicode同样也不完美，这里就有三个的问题，一个是，我们已经知道，英文字母只用一个字节表示就够了，第二个问题是如何才能区别Unicode和ASCII？计算机怎么知道两个字节表示一个符号，而不是分别表示两个符号呢？第三个，如果和GBK等双字节编码方式一样，用最高位是1或0表示两个字节和一个字节，就少了很多值无法用于表示字符，不够表示所有字符。Unicode在很长一段时间内无法推广，直到互联网的出现，为解决Unicode如何在网络上传输的问题，于是面向传输的众多 UTF（UCS Transfer Format）标准出现了，顾名思义，UTF-8就是每次8个位传输数据，而UTF-16就是每次16个位。UTF-8就是在互联网上使用最广的一种Unicode的实现方式，这是为传输而设计的编码，并使编码无国界，这样就可以显示全世界上所有文化的字符了。

**UTF-8最大的一个特点，就是它是一种变长的编码方式。**它可以使用1~4个字节表示一个符号。从unicode到uft-8并不是直接的对应，而是要过一些算法和规则来转换（**即Uncidoe字符集≠UTF-8编码方式**）。

Unicode符号范围 | UTF-8编码方式

(十六进制) | （二进制）

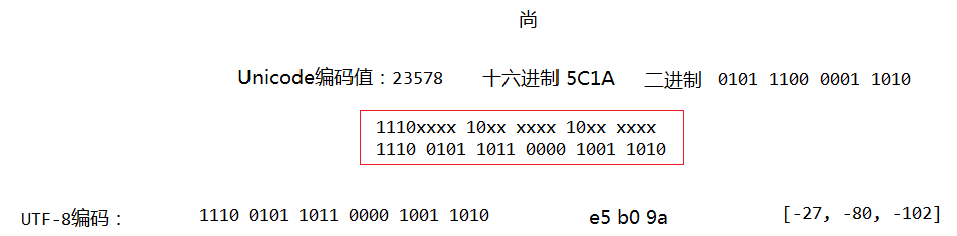
—————————————————————–

0000 0000-0000 007F | 0xxxxxxx（兼容原来的ASCII）

0000 0080-0000 07FF | 110xxxxx 10xxxxxx

0000 0800-0000 FFFF | 1110xxxx 10xxxxxx 10xxxxxx

0001 0000-0010 FFFF | 11110xxx 10xxxxxx 10xxxxxx 10xxxxxx

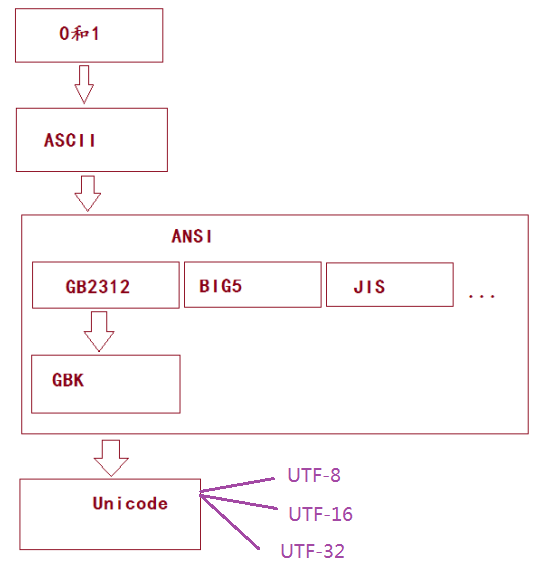


因此，Unicode只是定义了一个庞大的、全球通用的字符集，并为每个字符规定了唯一确定的编号，具体存储成什么样的字节流，取决于字符编码方案。推荐的Unicode编码是UTF-16和UTF-8。

**Unicode字符集只是定义了字符的集合和唯一编号，Unicode编码，则是对UTF-8、UCS-2/UTF-16等具体编码方案的统称而已，并不是具体的编码方案。所以当需要用到字符编码的时候，你可以写gb2312，codepage936，utf-8，utf-16，但请不要写Unicode。**

造成乱码的原因就是因为使用了错误的字符编码去解码字节流，因此当我们在思考任何跟文本显示有关的问题时，请时刻保持清醒：当前使用的字符编码是什么。只有这样，我们才能正确分析和处理乱码问题。

常见CharSet有：GBK、GB2312、US-ASCII、ISO-8859-1、UTF-8、UTF-16BE、UTF-16LE、UTF-16



## 八、练习

### 简答题

1、什么是标识符？命名规则和命名规范分别是什么？

2、基本数据类型有哪几类？包含String吗？

3、声明变量的格式？使用变量要注意哪几点？

4、char型变量中能不能存贮一个中文汉字?为什么?

5、定义float f=3.4;是否正确?

### 编程题

1、为抵抗洪水，战士连续作战89小时，编程计算共多少天零多少小时？ day=h/24 hour=h%24

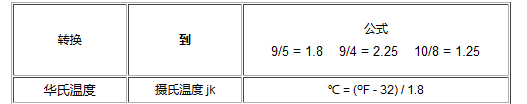
2、今天是周2，100天以后是周几？

3、编写代码实现n和m值的交换

int n = 5,m = 13;

4、如何求一个0~255范围内的整数的十六进制值，例如60十六进制表示形式3C

5、小明要到美国旅游，可是那里的温度是以华氏度为单位记录的。它需要一个程序将华氏温度（80度）转换为摄氏度，并以华氏度和摄氏度为单位分别显示该温度。



### 参考答案

#### 示例1

|  |
| --- |
| //1、为抵抗洪水，战士连续作战89小时，编程计算共多少天零多少小时？  public class TestExer1 {  public static void main(String[] args) {  int hour = 89;  int day = hour / 24;  hour = hour % 24;  System.out.println("为抵抗洪水，战士连续作战" + day + "天" + hour + "小时");  }  } |

#### 示例2

|  |
| --- |
| //1、今天是周二，100天以后是周几？  public class TestExer2 {  public static void main(String[] args) {  int week = 2;  week += 100;  week = week % 7;  System.out.println("100天后星期" + week);  }  } |

#### 示例3

|  |
| --- |
| //1、编写代码实现n和m值的交换 int n = 5,m = 13;  public class TestExer2 {  public static void main(String[] args) {  int n = 5,m = 13;  System.out.println("交换之前n=" + n + ",m=" + m);  int temp = n;  n = m;  m = temp;  System.out.println("交换之后n=" + n + ",m=" + m);  }  } |
| //1、编写代码实现n和m值的交换 int n = 5,m = 13;  public class TestExer2 {  public static void main(String[] args) {  int n = 5,m = 13;  System.out.println("交换之前n=" + n + ",m=" + m);  m = m + n;  n = m - n;  m = m - n;  System.out.println("交换之后n=" + n + ",m=" + m);  }  } |
| //1、编写代码实现n和m值的交换 int n = 5,m = 13;  public class TestExer2 {  public static void main(String[] args) {  int n = 5,m = 13;  System.out.println("交换之前n=" + n + ",m=" + m);  m = m ^ n;  n = m ^ n;  m = m ^ n;  System.out.println("交换之后n=" + n + ",m=" + m);  }  } |

#### 示例4

|  |
| --- |
| public static void main(String[] args) {  //方式一：自动实现  String str1 = Integer.toBinaryString(60);  String str2 = Integer.toHexString(60);  System.out.println("60的二进制形式" + str1);  System.out.println("60的十六进制形式" + str2);  //方式二：手动实现  int num = 60;  int temp = num&15;  String r = (temp > 9)? (char)(temp-10 + 'A')+"" : temp+"";    temp = num >>> 4;  temp = temp & 15;  String l = (temp > 9)? (char)(temp-10 + 'A')+"" : temp+"";  System.out.println("60的十六进制形式" + l + "" + r);  } |

#### 示例5

|  |
| --- |
| //小明要到美国旅游，可是那里的温度是以华氏度为单位记录的。  //它需要一个程序将华氏温度（80度）转换为摄氏度，并以华氏度和摄氏度为单位分别显示该温度。  public static void main(String[] args) {  int h = 80;  double s = (h - 32) / 1.8;  System.out.println(h + "华氏度，是摄氏度" + Math.round(s));  } |

# 第三章 流程控制语句结构

流程控制语句结构：

顺序结构：从上到下，顺序执行

分支结构：根据选择不同，执行不同的代码。if...else和switch...case

循环结构：根据条件循环重复执行某段代码。for（普通for和增强for）、while和do...while三种

## 一、顺序结构

（在同一个方法中）从上往下执行

|  |
| --- |
| /\*  流程控制语句结构：  1、顺序：（在同一个方法中）从上往下执行  从键盘输入一个值，并打印输出  2、分支  3、循环  用核心类库中的类：String,System,Math,Scanner等  如果这个类型定义在java.lang包下，那么直接使用，无需导包，直接使用简名称  但是如果这个类型定义在其他包下，那么需要写全名称或导包  全名称：java.util.Scanner  \*/  //导包语句：  //格式：import 包.类名;  //位置：必须在源文件的上面，在class的上面  import java.util.Scanner;  class TestStatement{  public static void main(String[] args){  System.out.println("欢迎使用尚硅谷登记系统：");  System.out.println("请输入个人信息：");    //System.out.println(Math.pow(2,3));    //Scanner键盘输入的工具类  //input是一个变量名，自己命名  //方式一：全名称  //java.util.Scanner input = new java.util.Scanner(System.in);    //方式二：导包，简名称  Scanner input = new Scanner(System.in);    System.out.print("请输入年龄：");  int age = input.nextInt();  System.out.println("您的年龄是：" + age);    System.out.print("请输入入学成绩：");  double score = input.nextDouble();  System.out.println("您的分数是：" + score);    System.out.print("请告知我是否帅/美：");  boolean flag = input.nextBoolean();  System.out.println("帅/美否：" + flag);    System.out.print("请输入您的姓名：");  String name = input.next();  System.out.println("您的名字是："+name);  }  } |

## 二、复合语句：

与C语言及其他语言相同，Java语言的复合语句是以整个块区为单位的语句，所以又称为块语句。**复合语句由开括号“{”开始，闭括号“}”结束。**

复合语句中的每一语句都是从上到下执行。复合语句以整个块为单位，并且在复合语句中可以嵌套复合语句。

Java代码结构：

class 类{

{

//代码块

}

方法签名{

//方法体

}

内部类{

内部类体

}

}

**复合语句为变量创建了一个作用域**。在该作用域中某个变量被创建并能够使用，如果在某个变量的作用域外使用该变量，则会发生错误。

## 三、分支结构

### 1、条件语句

#### 1、if单分支语句

语法格式：

if(条件表达式){

//当条件表达式结果为true时，需要执行的语句块

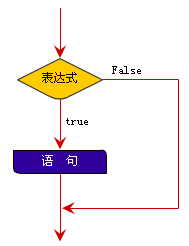
}

说明：

1. 条件表达式必须是布尔表达式（关系表达式或逻辑表达式）、布尔变量。
2. 语句块只有一个语句时，{}可以省略，但是建议保留

执行过程：

条件成立就执行，条件不成立就不执行。



#### 2、if...else双分支条件语句

语法格式：

if(条件表达式){

//当条件表达式结果为true时，需要执行的语句块1

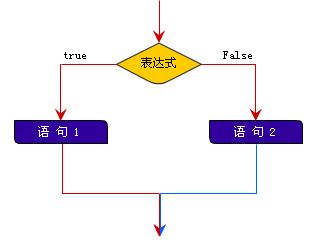
}else{

//当条件表达式结果为false时，需要执行的语句块2

}

执行过程：

当条件成立执行语句块1，不成立执行语句块2



说明：

（1）条件表达式必须是布尔表达式（关系表达式或逻辑表达式）、布尔变量。

（2）if或else的语句块只有一个语句时，{}可以省略，但是建议保留

#### 3、if...else if多分支条件语句

语法结构：

if(条件表达式1){

//当条件表达式1结果为true时，需要执行的复合语句1

}else if(条件表达式2){

//当条件表达式2结果为true时，需要执行的复合语句2

}

....

【else{

//当上述条件表达式结果都为false时，需要执行的语句

}】

执行过程：

从上至下判断条件，如果一旦某个条件成立了，就执行对应的语句块，后面的条件就不看了，如果所有条件都不满足，如果存在else，那么执行else，如果不存在else，就什么也不执行。

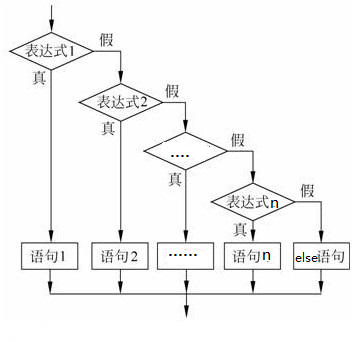
注意：

（1）每个if或者else后面的{}不是必须的，但是如果没有{}，默认只能带一条语句。即if或else后面的语句块只有一条语句时，可以省略{}，但是可读性不够好。

（2）最后的else不是必须的，可以缺省

（3）当多个条件是互斥关系时，顺序无所谓

当多个条件是包含关系时，“小上大下/子上父下”



|  |  |
| --- | --- |
| **public** **static** **void** main(String[] args) {  **int** score = 89;  **if**(score>90){  System.***out***.println("优秀");  }**else** **if**(score>80){  System.***out***.println("良好");  }**else** **if**(score>60){  System.***out***.println("及格");  }**else**{  System.***out***.println("不合格");  }  } | **public** **static** **void** main(String[] args) {  **int** score = 89;  **if**(score>90 && score<=100){  System.***out***.println("优秀");  }**else** **if**(score>80 && score<=90){  System.***out***.println("良好");  }**else** **if**(score>60 && score<=80){  System.***out***.println("及格");  }**else**{  System.***out***.println("不合格");  }  } |

#### 4、嵌套

if...else系列语句可以嵌套，在任意一个语句块中都可以再嵌套其他的条件语句。

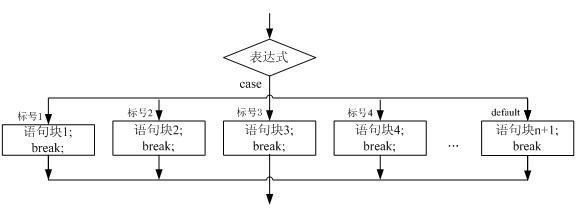
执行特点：

当外层条件满足时，内层条件才判断

### 2、选择结构

语法结构：

|  |
| --- |
| switch(表达式){  case 常量值1：  //执行语句块1  [break;]  case 常量值2：  //执行语句块2  [break;]  ......  case 常量值n：  //执行语句块n  [break;]  【default：  //执行缺省语句  [break;]  】  } |
| 如果多个case值要执行的语句相同，可以缩写为：  switch(表达式){  case 常量值1：  case 常量值2：  //执行语句块1  [break;]  ......  case 常量值n：  case 常量值n+1：  case 常量值n+2：  //执行语句块n  [break;]  【default：  //执行缺省语句  [break;]  】  } |



注意：

1. **表达式返回的结果类型只能是byte，short，char，int，枚举（JDK1.5之后），String（JDK1.7之后）（局限性-类型限制）**
2. 表达式返回的结果的类型需要与case后的值的类型保持一致
3. case后面只能跟常量值，不能是变量值或不确定的表达式值**（局限性）**
4. 同一个switch语句，case的常量值必须互不相同
5. break是可选的，当表达式返回结果与某个case后的值进行匹配成功后，执行相应分支的语句，一旦进入某个分支后，只有直到遇到break才会退出switch，否则将会继续执行下一个case的语句。
6. default是可选的。位置也不一定是最后，也可以在任意case的位置。但是不管再哪里，都是要所有case的常量值不匹配后才会进入default分支，一旦进入default分支，也是需要遇到break或switch的闭括号“}”才会停止。

总结：switch的入口，表达式值与某个case匹配，或都不匹配，从default进入。

switch的出口，遇到break或者switch的闭括号“}”结束

### 3、条件判断与选择结构的选择

当条件判断是**等值判断**，并且表达式的结果是byte,short,char,int,枚举,String类型的时候，用switch会更合适，

其他都使用条件判断

可以使用switch...case的，肯定可以使用if...else，返回来不一定。

### 4、经典案例（练习题）

1. 从键盘输入一个分数，如果60分（含）以上，就打印合格，如果60分以下，就打印不合格

提示：从键盘输入

import java.util.Scanner;

Scanner input=new Scanner(System.in);

int score=input.nextInt();//接收从键盘输入的一个整数

2、岳小鹏参加Java考试，他和父亲岳不群达成承诺：如果：

成绩为100分时，奖励一辆BMW；

成绩为(80，99]时，奖励一台iphone7plus；

当成绩为[60,80]时，奖励一个 iPad；

其它时，什么奖励也没有。

请从键盘输入岳小鹏的期末成绩，并加以判断

1. 编写程序：由键盘输入三个整数分别存入变量num1、num2、num3，对它们进行排序(使用 if-else if-else),并且从小到大输出。

4、大家都知道，男大当婚，女大当嫁。那么女方家长要嫁女儿，当然要提出一定的条件：高：180cm以上；富：财富1千万以上；帅：是。

如果这三个条件同时满足，则：“我一定要嫁给他!!!”

如果三个条件中有为真的情况，则：“嫁吧，比上不足，比下有余。”

如果三个条件都不满足，则：“不嫁！”

提示：从键盘输入

Scanner input=new Scanner(System.in);

input.nextInt();//接收从键盘输入的一个整数

input.next();输入一个字符串

input.nextDouble();//输入一个小数

input.nextBoolean();//输入一个布尔值

例如：

System.out.println(“身高:(cm)”);

input.nextInt();

System.out.println(“财富:(千万)”);

input.nextDouble();

System.out.println(“帅否:(true/false)”); (是/否)

input.nextBoolean(); 或者 input.next(); “是”.equals(str)

1. 求ax2+bx+c=0方程的根。a,b,c分别为函数的参数，

如果a≠0，那么：

1. 当b2-4ac>0，则有两个解；
2. 当b2-4ac=0，则有一个解；
3. 当b2-4ac<0，则无解；

如果a=0,b≠0，那么，

提示1：Math.sqrt(num); sqrt指平方根

例如：

求x2-4x+1=0方程的根

求x2-2x+1=0方程的根

6、假设你想开发一个玩彩票的游戏，程序随机地产生一个两位数的彩票，提示用户输入一个两位数，然后按照下面的规则判定用户是否能赢。

1)如果用户输入的数匹配彩票的实际顺序，奖金10 000美元。

2)如果用户输入的所有数字匹配彩票的所有数字，但顺序不一致，奖金 3 000美元。

3)如果用户输入的一个数字仅满足顺序情况下匹配彩票的一个数字，奖金1 000美元。

4)如果用户输入的一个数字仅满足非顺序情况下匹配彩票的一个数字，奖金500美元。

5)如果用户输入的数字没有匹配任何一个数字，则彩票作废。

提示：使用Math.random() 产生随机数

Math.random() 产生[0,1)范围的随机值

Math.random() \* 90：[0,90)

Math.random() \* 90 + 10：[10,100) 即得到 [10,99]

使用(int)(Math.random() \* 90 + 10)产生一个两位数的随机数。

7、赌数游戏：随机产生3个1-6的整数，如果三个数相等，那么称为“豹子”，如果三个数之和大于9，称为“大”，如果三个数之和小于等于9，称为“小”，用户从键盘输入押的是“豹子”、“大”、“小”，并判断是否猜对了

8、使用 switch 把阿拉伯数字转为“壹、贰、叁、肆、伍、陆、柒、捌、玖”；其它的都输出 “other”。

9、根据用于指定月份，打印该月份所属的季节。

3,4,5 春季 6,7,8 夏季 9,10,11 秋季 12, 1, 2 冬季

10、编写程序，从键盘接收整数参数.如果该数为1-7，打印对应的星期值，否则打印“非法参数”。

11、从键盘分别输入年、月、日，判断这一天是当年的第几天

注：判断一年是否是闰年的标准：

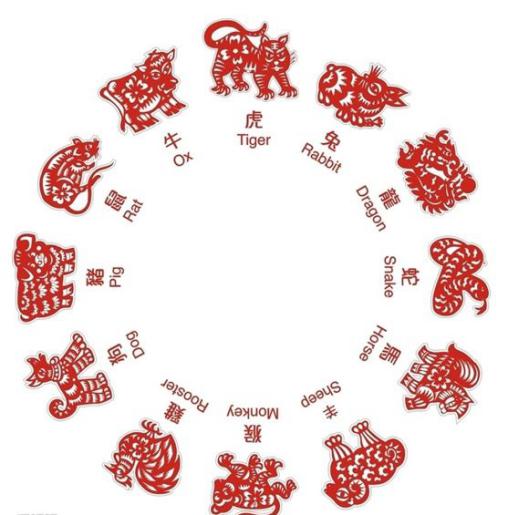
1）可以被4整除，但不可被100整除

2）可以被400整除

12、编写一个程序，为一个给定的年份找出其对应的中国生肖。中国的生肖基于12年一个周期，每年用一个动物代表：rat（鼠）、ox（牛）、tiger（虎）、rabbit（兔）、dragon（龙）、snake（蛇）、

horse（马）、sheep（羊）、monkey（候）、rooster（鸡）、dog（狗）、pig（猪）。

提示：2017年：鸡 2017 % 12 == 1



#### 示例1

|  |
| --- |
| /\*  1、从键盘输入一个分数，如果60分（含）以上，就打印合格，如果60分以下，就打印不合格  提示：从键盘输入  import java.util.Scanner;  Scanner input=new Scanner(System.in);  int score=input.nextInt();//接收从键盘输入的一个整数  \*/  import java.util.Scanner;  class TestScoreExer{  public static void main(String[] args){  Scanner scanner = new Scanner(System.in);    System.out.print("请输入成绩：");  int score = scanner.nextInt();    System.out.println(score >= 60 ? "合格" : "不合格");  }  } |
| /\*  1、从键盘输入一个分数，如果60分（含）以上，就打印合格，如果60分以下，就打印不合格  提示：从键盘输入  import java.util.Scanner;  Scanner input=new Scanner(System.in);  int score=input.nextInt();//接收从键盘输入的一个整数  \*/  import java.util.Scanner;  class TestScore{  public static void main(String[] args){  Scanner input = new Scanner(System.in);  System.out.print("请输入成绩：");  int score = input.nextInt();    if(score >= 60){  System.out.println("合格");  }else{  System.out.println("不合格");  }  }  } |

#### 示例2

|  |
| --- |
| /\*  2、岳小鹏参加Java考试，他和父亲岳不群达成承诺：如果：  成绩为100分时，奖励一辆BMW；  成绩为(80，99]时，奖励一台iphone7plus；  当成绩为[60,80]时，奖励一个 iPad；  其它时，什么奖励也没有。  请从键盘输入岳小鹏的期末成绩，并加以判断  \*/  import java.util.Scanner;  class TestJiangLiExer2{  public static void main(String[] args){  Scanner input = new Scanner(System.in);  System.out.print("请输入岳小鹏的成绩：");  int score = input.nextInt();    /\*  if(score == 100){  System.out.println("奖励一辆BMW");  }else if(score>80 && score<=99){  System.out.println("奖励一台iphone7plus");  }else if(score>=60 && score<=80){  System.out.println("奖励一个 iPad");  }else{  System.out.println("什么奖励也没有");  }  \*/    //默认成绩的范围是[0,100]，否则100以上的就会进入score>80分支中  if(score == 100){  System.out.println("奖励一辆BMW");  }else if(score>80){  System.out.println("奖励一台iphone7plus");  }else if(score>=60){  System.out.println("奖励一个 iPad");  }else{  System.out.println("什么奖励也没有");  }  }  } |

#### 示例3

|  |
| --- |
| /\*  3、编写程序：由键盘输入三个整数分别存入变量num1、num2、num3，对它们进行排序(使用 if-else if-else),并且从小到大输出。  \*/  import java.util.Scanner;  public class TestThreeNumbersSort{  public static void main(String[] args){  Scanner input = new Scanner(System.in);    System.out.print("请输入第一个数：");  int num1 = input.nextInt();    System.out.print("请输入第二个数：");  int num2 = input.nextInt();    System.out.print("请输入第三个数：");  int num3 = input.nextInt();    //方法一：罗列所有情况  if(num1 <= num2 && num2 <= num3){  System.out.println(num1 + "<=" + num2 + "<=" + num3);  }else if(num1 <= num3 && num3 <= num2){  System.out.println(num1 + "<=" + num3 + "<=" + num2);  }else if(num2 <= num1 && num1 <= num3){  System.out.println(num2 + "<=" + num1 + "<=" + num3);  }else if(num2 <= num3 && num3 <= num1){  System.out.println(num2 + "<=" + num3 + "<=" + num1);  }else if(num3 <= num1 && num1 <= num2){  System.out.println(num3 + "<=" + num1 + "<=" + num2);  }else{  System.out.println(num3 + "<=" + num2 + "<=" + num1);  }  }  } |
| /\*  3、编写程序：由键盘输入三个整数分别存入变量num1、num2、num3，对它们进行排序(使用 if-else if-else),并且从小到大输出。  \*/  import java.util.Scanner;  class TestExer3{  public static void main(String[] args){  Scanner s = new Scanner(System.in);    System.out.print("请输入第一个整数：");  int num1 = s.nextInt();    System.out.print("请输入第一个整数：");  int num2 = s.nextInt();    System.out.print("请输入第一个整数：");  int num3 = s.nextInt();    //嵌套  if(num1 <= num2){  //num1 <= num2  if(num2 <= num3){  System.out.println(num1 + "<=" + num2 + "<=" + num3);  }else{  //num1<=num2 && num3<num2  if(num1 <= num3){  //num1<=num2 && num3<num2 && num1 <= num3  System.out.println(num1 + "<=" + num3 + "<=" + num2);  }else{  //num1<=num2 && num3<num2 && num3<num1  System.out.println(num3 + "<=" + num1 + "<=" + num2);  }  }  }else{  //num1 > num2  if(num3 > num1){  //num3 > num1 > num2  System.out.println(num2 + "<=" + num1 + "<=" + num3);  }else{  //num1 > num2 && num1 > num3  if(num2 <= num3){  //num1 > num2 && num1 > num3 && num2 <= num3  System.out.println(num2 + "<=" + num3 + "<=" + num1);  }else{  //num1 > num2 && num1 > num3 && num3<num2  System.out.println(num3 + "<=" + num2 + "<=" + num1);  }  }  }  }  } |

#### 示例4

|  |
| --- |
| /\*  4、大家都知道，男大当婚，女大当嫁。那么女方家长要嫁女儿，  当然要提出一定的条件：高：180cm以上；富：财富1千万以上；帅：是。  如果这三个条件同时满足，则：“我一定要嫁给他!!!”  如果三个条件中有为真的情况，则：“嫁吧，比上不足，比下有余。”  如果三个条件都不满足，则：“不嫁！”  提示：从键盘输入  Scanner input=new Scanner(System.in);  input.nextInt();//接收从键盘输入的一个整数  input.next();输入一个字符串  input.nextDouble();//输入一个小数  input.nextBoolean();//输入一个布尔值  例如：  System.out.println(“身高:(cm)”);  scanner.nextInt();  System.out.println(“财富: (千万)”);  scanner.nextDouble();  System.out.println(“帅否: (true/false)”); (是/否)  scanner.nextBoolean(); 或者 scanner.next(); str.equals(“是”)  \*/  import java.util.Scanner;  class MarryExer4{  public static void main(String[] args){  Scanner input = new Scanner(System.in);    System.out.print("身高:(cm)");  int height = input.nextInt();    System.out.print("财富:(千万)");  double money = input.nextDouble();    System.out.print("帅否:(true/false)");  boolean h = input.nextBoolean();    //if(height>=180 && money >= 1 && h)  if(height>=180 && money >= 1 && h == true){  System.out.print("我一定要嫁给他!!!");  }else if(height>=180 || money >= 1 || h == true){  System.out.print("嫁吧，比上不足，比下有余!");  }else{  System.out.print("不嫁");  }  }  } |
| /\*  4、大家都知道，男大当婚，女大当嫁。那么女方家长要嫁女儿，  当然要提出一定的条件：高：180cm以上；富：财富1千万以上；帅：是。  如果这三个条件同时满足，则：“我一定要嫁给他!!!”  如果三个条件中有为真的情况，则：“嫁吧，比上不足，比下有余。”  如果三个条件都不满足，则：“不嫁！”  提示：从键盘输入  Scanner input=new Scanner(System.in);  input.nextInt();//接收从键盘输入的一个整数  input.next();输入一个字符串  input.nextDouble();//输入一个小数  input.nextBoolean();//输入一个布尔值  例如：  System.out.println(“身高:(cm)”);  scanner.nextInt();  System.out.println(“财富: (千万)”);  scanner.nextDouble();  System.out.println(“帅否: (true/false)”); (是/否)  scanner.nextBoolean(); 或者 scanner.next(); str.equals(“是”)  \*/  import java.util.Scanner;  class MarryExer42{  public static void main(String[] args){  Scanner input = new Scanner(System.in);    System.out.print("身高:(cm)");  int height = input.nextInt();    System.out.print("财富:(千万)");  double money = input.nextDouble();    System.out.print("帅否:(是/否)");  String h = input.next();    //h.equals("是")：比较h变量中的字符串与"是"是否相等，如果相等，返回true，否则结果为false  if(height>=180 && money >= 1 && h.equals("是")){  System.out.print("我一定要嫁给他!!!");  }else if(height>=180 || money >= 1 || h.equals("是")){  System.out.print("嫁吧，比上不足，比下有余!");  }else{  System.out.print("不嫁");  }  }  } |
| /\*  4、大家都知道，男大当婚，女大当嫁。那么女方家长要嫁女儿，  当然要提出一定的条件：高：180cm以上；富：财富1千万以上；帅：是。  如果这三个条件同时满足，则：“我一定要嫁给他!!!”  如果三个条件中有为真的情况，则：“嫁吧，比上不足，比下有余。”  如果三个条件都不满足，则：“不嫁！”  提示：从键盘输入  Scanner input=new Scanner(System.in);  input.nextInt();//接收从键盘输入的一个整数  input.next();输入一个字符串  input.nextDouble();//输入一个小数  input.nextBoolean();//输入一个布尔值  例如：  System.out.println(“身高:(cm)”);  scanner.nextInt();  System.out.println(“财富: (千万)”);  scanner.nextDouble();  System.out.println(“帅否: (true/false)”); (是/否)  scanner.nextBoolean(); 或者 scanner.next(); str.equals(“是”)  \*/  import java.util.Scanner;  class MarryExer43{  public static void main(String[] args){  Scanner input = new Scanner(System.in);    System.out.print("身高:(cm)");  int height = input.nextInt();    System.out.print("财富:(千万)");  double money = input.nextDouble();    System.out.print("帅否:(是/否)");  //char h = input.next().charAt(0);  String str = input.next();//接收一个字符串的内容  //System.out.println("输入的内容：" + str);  char h = str.charAt(0);//取字符串的第1个字符  //System.out.println("h：" + h);    if(height>=180 && money >= 1 && h == '是'){  System.out.print("我一定要嫁给他!!!");  }else if(height>=180 || money >= 1 || h == '是'){  System.out.print("嫁吧，比上不足，比下有余!");  }else{  System.out.print("不嫁");  }  }  } |

#### 示例5

|  |
| --- |
| import java.util.Scanner;  class Exer5{  public static void main(String[] args){  Scanner input = new Scanner(System.in);    System.out.println("一元二次方程：ax^2+bx+c=0");  System.out.print("请输入参数a：");  double a = input.nextDouble();    System.out.print("请输入参数b：");  double b = input.nextDouble();    System.out.print("请输入参数c：");  double c = input.nextDouble();  if(a!=0){  double temp = b\*b - 4\*a\*c;  if(temp==0){  double x = -b/(2\*a);  System.out.println("该方程是一元二次方法，有一个解：" + x);  }else if(temp>0){  double sqrt = Math.sqrt(temp);  double x1 = (-b+ sqrt)/(2\*a);  double x2 = (-b- sqrt)/(2\*a);  System.out.println("该方程是一元二次方法，两个解：" + x1 +"," + x2);  }else{  System.out.println("该方程是一元二次方法，在实数范围内无解！");  }  }else{  if(b!=0){  double x = -c/b;  System.out.println("该方程是一元一次方法，有一个解：" + x);  }else{  System.out.println("不是方程！");  }  }  }  } |

#### 示例6

|  |
| --- |
| /\*  6、假设你想开发一个玩彩票的游戏，程序随机地产生一个两位数的彩票，提示用户输入一个两位数，  然后按照下面的规则判定用户是否能赢。  1)如果用户输入的数匹配彩票的实际顺序，奖金10 000美元。  2)如果用户输入的所有数字匹配彩票的所有数字，但顺序不一致，奖金 3 000美元。  3)如果用户输入的一个数字仅满足顺序情况下匹配彩票的一个数字，奖金1 000美元。  4)如果用户输入的一个数字仅满足非顺序情况下匹配彩票的一个数字，奖金500美元。  5)如果用户输入的数字没有匹配任何一个数字，则彩票作废。  提示：使用(int)(Math.random() \* 90 + 10)产生随机数。  Math.random() : [0,1) \* 90 ?[0,90) + 10 ?[10,100) ? [10,99]  Math.sqrt(xxx)  Math.pow(x,y)  Math.random() --> [0,1) 0<=number<1  两位数[10,100) 0+10<=number+10<1+10  0\*90+10<=number\*90+10<1\*90+10  \*/  import java.util.Scanner;  class TestCaiPiao{  public static void main(String[] args){  //1、随机产生一个两位数  //System.out.println(Math.random());//产生[0,1)  int number = (int)(Math.random()\*90 + 10);//得到[10,99]，即[10,100)  //System.out.println(number);    int numberShi = number/10;  int numberGe = number%10;    //2、用户输入一个两位数  Scanner input = new Scanner(System.in);  System.out.print("请输入一个两位数：");  int guess = input.nextInt();    int guessShi = guess/10;  int guessGe = guess%10;    if(number == guess){  System.out.println("奖金10 000美元");  }else if(numberShi == guessGe && numberGe == guessShi){  System.out.println("奖金3 000美元");  }else if(numberShi==guessShi || numberGe == guessGe){  System.out.println("奖金1 000美元");  }else if(numberShi==guessGe || numberGe == guessShi){  System.out.println("奖金500美元");  }else{  System.out.println("没中奖");  }    System.out.println("中奖号码是：" + number);  }  } |

#### 示例7

|  |
| --- |
| /\*  7、赌数游戏：随机产生3个1-6的整数，  如果三个数相等，那么称为“豹子”，  如果三个数之和大于9，称为“大”，  如果三个数之和小于等于9，称为“小”，  用户从键盘输入押的是“豹子”、“大”、“小”，并判断是否猜对了  Math.random() -> [0,1) 0<=x<1  1<=x<7  0\*6+1<=x\*6+1<1\*6+1    \*/  import java.util.Scanner;  class GuessNumber{  public static void main(String[] args){  //1、产生一个[1-6]的数  int a = (int)(Math.random()\*6+1);  //System.out.println(a);  int b = (int)(Math.random()\*6+1);  //System.out.println(b);  int c = (int)(Math.random()\*6+1);  //System.out.println(c);    /\*  a = 1;  b = 1;  c = 1;  \*/    String result = "大";  if(a==b && b==c){  result = "豹子";  }else if(a+b+c <=9 ){  result = "小";  }    //2、用户输入猜的结果  Scanner input = new Scanner(System.in);  System.out.print("请押宝，买定离手：");  String guess = input.next();    //3、判断结果  if(result.equals(guess)){  System.out.println("猜对了");  }else{  System.out.println("猜错了");  }  }  } |

#### 示例8

|  |
| --- |
|  |

#### 示例9

|  |
| --- |
| /\*  9、根据用于指定月份，打印该月份所属的季节。  3,4,5 春季 6,7,8 夏季 9,10,11 秋季 12, 1, 2 冬季  \*/  import java.util.Scanner;  class TestSeason{  public static void main(String[] args){  Scanner input = new Scanner(System.in);  System.out.println("请输入当前月份：");  int month = input.nextInt();    switch(month){  case 3:  case 4:  case 5:  System.out.println("春季");  break;  case 6:  case 7:  case 8:  System.out.println("夏季");  break;  case 9:  case 10:  case 11:  System.out.println("秋季");  break;  case 12:  case 1:  case 2:  System.out.println("冬季");  break;  default:  System.out.println("非法月份");  }  }  } |

#### 示例10

|  |
| --- |
| 10、编写程序，从键盘接收整数参数.如果该数为1-7，打印对应的星期值，否则打印“非法参数”。  \*/  import java.util.Scanner;  class TestSwitch{  public static void main(String[] args){  Scanner input = new Scanner(System.in);  System.out.print("请输入星期值：");  int week = input.nextInt();    switch(week){  case 1:  System.out.println("星期一：Monday");  break;  case 2:  System.out.println("星期二：Tuesday");  break;  case 3:  System.out.println("星期三：Wednesday");  break;  case 4:  System.out.println("星期四：Thursday");  break;  case 5:  System.out.println("星期五：Friday");  break;  case 6:  System.out.println("星期六：Saturday");  break;  case 7:  System.out.println("星期天：Sunday");  break;  default:  System.out.println("非法星期值");  break;  }    }  } |

#### 示例11

|  |
| --- |
| /\*  12、从键盘分别输入年、月、日，判断这一天是当年的第几天  注：判断一年是否是闰年的标准：  1）可以被4整除，但不可被100整除  2）可以被400整除  \*/  import java.util.Scanner;  class TestDaysOfYear{  public static void main(String[] args){  Scanner input = new Scanner(System.in);    System.out.print("年：");  int year = input.nextInt();    System.out.print("月：");  int month = input.nextInt();    System.out.print("日：");  int day = input.nextInt();    int days = day;    //加前面几个月的满月天数  switch(month){  case 12:  //前面11个月的总天数  //days += 第11月的天数;  days += 30;  case 11:  //前面10个月的总天数  //days += 第10月的天数;  days += 31;  case 10:  days += 30;//九月  case 9:  days += 31;//八月  case 8:  days += 31;//七月  case 7:  days += 30;//六月  case 6:  days += 31;//五月  case 5:  days += 30;//四月  case 4:  days += 31;//三月  case 3:  days += 28;//二月  /\*if(闰年){  days++;  }  \*/  if(year % 4 ==0 && year % 100 != 0 || year%400==0){  days++;  }  case 2:  days += 31;//一月  }    System.out.println(year + "年" + month + "月" + day + "日是这一年的第" + days + "天");    /\*  int days = 0;  switch(month){  case 1:  days = day;  break;  case 2:  days = 31 + day;  break;  case 3:  //days = 31 + 二月的天数 + day;  days = 31 + 28 + day;  break;  case 4:  //days = 31 + 二月的天数 + 31 + day;  days = 31 + 28 + 31 + day;  break;  ....  }    if(闰年 && month >2){  days++;  }  \*/    }  } |

#### 示例12

|  |
| --- |
| /\*  13、编写一个程序，为一个给定的年份找出其对应的中国生肖。  中国的生肖基于12年一个周期，每年用一个动物代表：  rat（鼠）、ox（牛）、tiger（虎）、rabbit（兔）、dragon（龙）、snake（蛇）、  horse（马）、sheep（羊）、monkey（候）、rooster（鸡）、dog（狗）、pig（猪）。  提示：2017年：鸡 2017 % 12 == 1  \*/  import java.util.Scanner;  class TestShengXiao{  public static void main(String[] args){  Scanner input = new Scanner(System.in);  System.out.print("请输入年份：");  int year = input.nextInt();    switch(year%12){  case 1:  System.out.println("鸡年");  break;  case 2:  System.out.println("狗年");  break;  case 3:  System.out.println("猪年");  break;  case 4:  System.out.println("鼠年");  break;  case 5:  System.out.println("牛年");  break;  case 6:  System.out.println("虎年");  break;  case 7:  System.out.println("兔年");  break;  case 8:  System.out.println("龙年");  break;  case 9:  System.out.println("蛇年");  break;  case 10:  System.out.println("马年");  break;  case 11:  System.out.println("羊年");  break;  case 0:  System.out.println("猴年");  break;  }  }  } |

## 四、循环结构

循环语句就是在满足一定条件的情况下反复执行某一个操作。在Java中提供了3中常用的循环语句，分别是while循环语句、do...while循环语句和for循环语句。

### 1、三种循环语句的语法

#### while循环语句

语法结构：

初始化表达式1；

while(条件判断2){

//循环体3

//条件迭代4

}

执行过程：

1-2-3-4-2-3-4

#### do...while循环语句

语法结构：

初始化表达式1；

do{

//循环体3

//条件迭代4

}while(条件表达式2);

执行过程：

1-3-4-2-3-4-2。。。。。

先执行一次循环体后，再判断条件，如果条件成立，继续下一次循环体;**do...while循环至少执行一次循环体。**

#### for循环结构

语法结构：

for(初始化表达式1; 循环条件表达式2; 迭代表达式4){

//循环体语句块3

}

执行过程：

1-2-3-4-2-3-4-。。。。。。。。

注意：

* 两个分号必不可少
* 三个表达式可以省略，但要结合break，否则死循环。
* 表达式1可以有多个变量声明，但必须是同一个类型，用逗号分隔
* 表达式3可以有多个变量更新，用逗号分隔

#### foreach循环结构（后面数组和集合部分再讲）

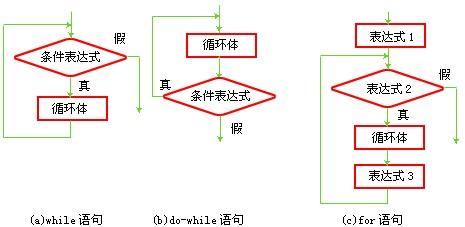
语法结构：

for(元素类型 元素临时名：数组/集合){

//循环体

}

### 2、三种循环语句的比较



都能实现循环重复执行某段代码，可以互相转换

执行顺序不同：for、while是先判断后执行，do...while是先执行后判断

执行效果不同：while和for，如果第一次条件判断不满足，那么循环体将一次都不执行，而do...while是先执行一次循环体，然后才判断条件，看是否第二次执行循环体

执行效率不同：do...while效率最高

每一种循环都有四个循环要素，而for循环体现的最明显，**循环变量初始值、循环条件、循环体、循环变量更新迭代。**

for循环适用于循环次数比较明确的，而do...while适用于至少执行一次的循环。while适用于循环条件比较明确的。

**while(true)等价于for(;;)**

### 3、嵌套循环

一个循环体中嵌套了另一个完整的循环结构。一个循环充当另一个循环的循环体。

三种循环体中都可以嵌套另外任意一种循环结构

执行特点，先执行外层循环，外循环执行一次，内循环从头至尾执行一轮。**总次数=外循环次数\*内循环次数**，详细说：总次数= 内循环第一轮的次数+内循环第二轮的次数+内循环第三轮的次数.....+内循环第n轮的次数。轮数即是外循环的次数。

例如：找出1000以内的所有完数

分析：（1）1-1000个数需要判断是否是否是完数，因此需要循环1000次

for(int i=1; i<=1000;i++){

//i是否是完数

//如果是，就打印i，如果不是就不打印i

}

（2）i是否是完数的判断，要把i的所有因子找出来，并相加

int sum = 0;

for(int j=1; j<i; j++){

if(i%j==0){

sum += j;

}

}

（3）如果是，就打印i

if(sum == i){

System.out.println(i);

}

例如：打印一个矩形4行5列的#

分析：（1）打印4行的#

for(int i=0;i<4;i++){

System.out.println(“#####”);

}

（2）每行5个#，还可以使用循环

for(int j=0;j<5;j++){

System.out.print(“#”);

}

System.out.println();

### 4、跳转

break：**用在switch和循环中，用于跳出所在的当层循环或switch**

continue：**只能用于循环中，提前结束本次循环，继续下一次循环**

return：用于结束当前方法

|  |  |
| --- | --- |
| **public** **class** TestBreakAndContinue {  **public** **static** **void** main(String[] args) {  **for** (**int** i = 1; i <=5; i++) {  **for** (**int** j = 1; j <=5; j++) {  System.***out***.print("\*");  **if**(i==j){  **break**; 跳出内循环  }  }  System.***out***.println();  }  }  } | \*  \*\*  \*\*\*  \*\*\*\*  \*\*\*\*\*  I=1 j=1 \* break  i=2 j=1 \* \* break |
| **public** **class** TestBreakAndContinue {  **public** **static** **void** main(String[] args) {  **for** (**int** i = 1; i <=5; i++) {  **for** (**int** j = 1; j <=5; j++) {  System.***out***.print("\*");  **if**(i==j){  **continue** ;  }  }  System.***out***.println();  }  }  } | \*\*\*\*\*  \*\*\*\*\*  \*\*\*\*\*  \*\*\*\*\*  \*\*\*\*\*  I=1 j=1 \* continue  I=1 j=2 \*\*  I=1 j=3 \*\*\*  I=1 j=4\*\*\*\*  I=1 j=5\*\*\*\*\*  然后 |

### 5、标签（了解）

|  |  |
| --- | --- |
| **public** **class** TestBreakAndContinue {  **public** **static** **void** main(String[] args) {  out:**for** (**int** i = 1; i <=5; i++) {  **for** (**int** j = 1; j <=5; j++) {  System.***out***.print("\*");  **if**(i==j){  System.***out***.println();  **continue** out;  }  }  }  }  } | \*  \*\*  \*\*\*  \*\*\*\*  \*\*\*\*\* |
| **public** **class** TestBreakAndContinue {  **public** **static** **void** main(String[] args) {  out:**for** (**int** i = 1; i <=5; i++) {  **for** (**int** j = 1; j <=5; j++) {  System.***out***.print("\*");  **if**(i==j){  System.***out***.println();  **break** out;  }  }  }  }  } | \* |

### 6、循环案例

1. 求1-100的和
2. 打印1-100的偶数
3. 从键盘输入整数，输入0结束，统计输入的正数、负数的个数。
4. 随机生成一个100以内的数，猜数字游戏

从键盘输入数，如果大了提示，大了，如果小了，提示小了，如果对了，就不再猜了，并统计一共猜了多少次

提示：随机数

import java.util.Random;

Random rand = new Random();

int num= rand.nextInt(100);

1. 输出所有的水仙花数，所谓水仙花数是指一个3位数，其各个位上数字立方和等于其本身。

例如： 153 = 1\*1\*1 + 5\*5\*5 + 3\*3\*3

1. 编写程序FooBizBaz.java，从1循环到150并在每行打印一个值，

另外在每个3的倍数行上打印出“foo”,在每个5的倍数行上打印“biz”,

在每个7的倍数行上打印输出“baz”。

1. 输入两个正整数m和n，求其最大公约数和最小公倍数
2. 打印1-100之间数，其中3、5、7的倍数不打印（如果用continue怎么做）

9、一个数如果恰好等于它的因子之和，这个数就称为"完数"。（因子：除去这个数本身的约数）

例如6=1＋2＋3.编程 找出1000以内的所有完数

10、九九乘法表

11、打印菱形

12、找出1-100之间所有的素数（质数）

#### 示例1

|  |
| --- |
| /\*  1、求1-100的和  \*/  public class Exer1Sum{  public static void main(String[] args){    int sum = 0;//别忘了初始值    //求1-100的累加和  for(int i=1; i<=100; i++){  sum += i;  }    //显示最终结果  System.out.println("1-100的和是：" + sum);  }  } |
| **public** **static** **void** main(String[] args) {  **int** i=1;  **int** sum = 0;  **while**(i<=100){  sum +=i;  i++;  }  System.***out***.println("1-100的和为："+sum);  } |

#### 示例2

|  |
| --- |
| /\*  2、打印1-100的偶数  Even：偶数  \*/  public class Exer2Even{  public static void main(String[] args){  System.out.println("1-100的偶数有：");  for(int i=2; i<=100; i=i+2){//步幅不一定是1，可以是其他  System.out.println(i);  }  /\*  (1)int i=2;  (2)i<=100成立，System.out.println(i);  (3)i=i+2 i=4  (4)i<=100成立，System.out.println(i);  (5)i=i+2 i=6  (6)i<=100成立，System.out.println(i);  ....  (n)i=i+2 i=102  (n+1)i<=100不成立，结束  \*/    /\*  for(int i=1; i<=100; i++){  //如果i是偶数就打印它  if(i % 2 == 0){//条件判断了100次，只有50次满足条件  System.out.println(i);  }  }  \*/  }  } |
| **public** **static** **void** main(String[] args) {  System.***out***.println("1-100之间的偶数有：");  **int** i=1;  **while**(i<=100){  **if**(i%2==0){  System.***out***.println(i);  }  }  } |

#### 示例3

|  |
| --- |
| import java.util.Scanner;  //3、从键盘输入整数，输入0结束，统计输入的正数、负数的个数。  public class TestExer3 {  public static void main(String[] args) {  Scanner input = new Scanner(System.in);    int positive = 0;  int negative = 0;  while(true){  System.out.print("请输入整数（0）结束：");  int num = input.nextInt();  if(num==0){  break;  }else if(num>0){  positive++;  }else{  negative++;  }  }  System.out.println("正数：" + positive + "，负数：" + negative);  }  } |

#### 示例4

|  |
| --- |
| import java.util.Random;  import java.util.Scanner;  public class TestDoWhileLoop{  public static void main(String[] args){  //1、随机产生一个100以内的整数  Random rand = new Random();  //int num = rand.nextInt();//产生的是任意大小的整数  int num = rand.nextInt(100);//产生[0,100)的整数  System.out.println(num);    //2、键盘输入  Scanner input = new Scanner(System.in);    //声明变量  int guess;  int count = 0;  do{  //循环体至少执行一次  System.out.print("请输入一个整数：");  guess = input.nextInt();//为变量赋值    count++;//输入一次，计数一次    if(guess>num){  System.out.println("大了");  }else if(guess < num){  System.out.println("小了");  }else{  System.out.println("猜对了");  }  }while(guess != num);    System.out.println("一共猜了：" + count + "次");  }    } |
| public static void main(String[] args) {  Random rand = new Random();  int num= rand.nextInt(100);  Scanner input = new Scanner(System.in);  int count =0 ;  do{  count++;  System.out.println("请猜：");  int temp = input.nextInt();  if(temp<num){  System.out.println("小了");  continue;  }  if(temp>num){  System.out.println("大了");  continue;  }  if(temp == num){  break;  }  }while(true);  System.out.println("总共猜了"+count+"次");  } |

#### 示例5

|  |
| --- |
| /\*  5、输出所有的水仙花数，所谓水仙花数是指一个3位数，其各个位上数字立方和等于其本身。  例如： 153 = 1\*1\*1 + 5\*5\*5 + 3\*3\*3  \*/  public class Exer5FlowerNumber{  public static void main(String[] args){  System.out.println("所有的水仙花数是：");  //判断所有的三位数，看这个三位数是否满足水仙花数的特征，如果是就打印  for(int i=100; i<=999; i++){  //i就代表每一个三位数  //判断i这个三位数是否满足水仙花数的特征  //(1)求出i的百位、十位、个位  int bai = i/100;  int shi = i/10%10;  int ge = i%10;    //(2)判断其各个位上数字立方和是否等于其本身  if(i == bai\*bai\*bai + shi\*shi\*shi + ge\*ge\*ge){  System.out.println(i);  }  }  }  } |

#### 示例6

|  |
| --- |
| /\*  6、编写程序FooBizBaz.java，从1循环到150并在每行打印一个值，  另外在每个3的倍数行上打印出“foo”,在每个5的倍数行上打印“biz”,  在每个7的倍数行上打印输出“baz”。  \*/  public class Exer6FooBizBaz{  public static void main(String[] args){  for(int i=1; i<=150; i++){  //1、打印数字  System.out.print(i);    //2、看是否打印foo,biz,baz  if(i%3==0){  System.out.print("\tfoo");  }  if(i%5==0){  System.out.print("\tbiz");  }  if(i%7==0){  System.out.print("\tbaz");  }    //3、打印换行  System.out.println();  }  }  } |

#### 示例7

|  |
| --- |
| **public** **static** **void** main(String[] args) {  Scanner input = **new** Scanner(System.***in***);  System.***out***.println("请输入m的值：");  **int** m = input.nextInt();  System.***out***.println("请输入n的值：");  **int** n = input.nextInt();    //获取m和n的较大值  **int** max = (m > n)? m : n;  //获取m和n的较小值  **int** min = (m < n)? m : n;  //求m和n的最大公约数  **for**(**int** i = min;i >= 1;i--){  **if**( m % i == 0 && n % i == 0){  System.***out***.println("m和n的最大公约数是：" + i);  **break**;  }  }  //求m和n的最小公倍数  **for**(**int** i = max;i <= m \* n;i++){  **if**( i % m == 0 && i % n == 0){  System.***out***.println("m和n的最小公倍数是：" + i);  **break**;  }  }  } |

#### 示例8

|  |
| --- |
| class TestContinue {  public static void main(String[] args) {  //打印1-100之间数，其中3、5、7的倍数不打印  for(int i=1; i<=100; i++){  if(i%3==0 | i%5==0 | i%7==0){  //跳过下面的打印语句，提前进入下一次循环，即i++语句  //break;//结束循环  continue;  }    //下面这部分循环体语句，有些情况下需要跳过  System.out.println(i);  }  System.out.println("over");  }  } |

#### 示例9

|  |
| --- |
| **public** **static** **void** main(String[] args) {  System.***out***.println("1-1000之间的完数有：");  **for**(**int** i=1;i<=1000;i++){  **int** sum = 0;  **for**(**int** j=1;j<i;j++){  **if**(i%j==0){  sum +=j;  }  }  **if**(sum == i){  System.***out***.println(i);  }  }  } |

#### 示例10

|  |
| --- |
| **public** **static** **void** main(String[] args) {  **for**(**int** i =1;i<=9;i++){  **for**(**int** j=1;j<=i;j++){  System.***out***.print(j+"\*"+i+"="+(i\*j) + "\t");  }  System.***out***.println();  }  } |

#### 示例11

|  |
| --- |
| **public** **static** **void** main(String[] args) {  //上半部分  **for**(**int** i = 0;i < 5;i++){  //输出“ ”  **for**(**int** j = 0;j < 4-i;j++){  System.***out***.print(" ");  }  //输出“\* ”  **for**(**int** k = 0;k < i+1;k++){  System.***out***.print("\* ");  }  System.***out***.println();  }  //下半部分  **for**(**int** i = 0;i < 4;i++){  **for**(**int** j = 0;j < i+1;j++){  System.***out***.print(" ");  }  **for**(**int** k = 0;k < 4-i;k++){  System.***out***.print("\* ");  }  System.***out***.println();  }  } |

#### 示例12

|  |
| --- |
| **public** **static** **void** main(String[] args) {  System.***out***.println("1-100之间的质数：");  **long** start = System.*currentTimeMillis*();  **for**(**int** i=2;i<=100;i++){  **boolean** flag = **true**;//假设i是质数  **for**(**int** j=2;j<i;j++){  **if**(i%j==0){  flag = **false**;//i不是质数  **break**;  }  }  **if**(flag){  System.***out***.println(i);  }  }  **long** end = System.*currentTimeMillis*();  System.***out***.println("运行时间："+(end - start));  } |

# 第四章 数组

## 一、数组的相关概念

### 1、数组的相关概念[了解]

**数组(Array)**，就是相同数据类型的元素按一定顺序排列的集合，就是把有限个类型相同的变量用一个名字命名，以便统一管理他们，然后用编号区分他们，这个名字称为**数组名**，编号称为**下标或索引(index)**。组成数组的各个变量称为数组的**元素(element)**。数组中元素的个数称为**数组的长度(length)**。

例如：int s1\_score = 80; int s2\_score = 90; int s3\_score = 87; .....

int[] scores = {80,90,87,68,99,100,79,96}; 元素1：scores[0]=80; 数组的长度为8

### 2、数组的特点[掌握]

**1、数组的长度一旦确定就不能修改**

**2、创建数组对象会在内存中开辟一整块连续的空间。而数组名中引用的是这块连续空间的首地址。**

3、遍历数组和获取某个元素的速度快，可以通过数组名+下标直接定位到任意一个元素。

### 3、数组类型与数组元素的类型[理解]

数组中的元素可以是任何数据类型，包括基本类型和引用类型

* **数组属引用类型，把数组看成对象(object)，数组中的每个元素相当于该对象的成员变量。**

例如：

（1）基本数据类型元素的数组

int[] scores = {80,90,87,68,99,100,79,96};

scores数组的类型是：int[]，元素的类型是int

scores是一个对象，scores[0],scores[1]等元素是scores对象的成员变量

（2）引用数据类型元素的数组

String[] strings = {“hello”,”java”,”world”,”array”};

strings数组的类型是：String[]，元素的类型是String

strings是一个对象，strings[0]，strings[1]等元素是strings对象的成员变量

### 4、数组的分类[了解]

数组按照维度分为：一维数组、二维数组、多维数组

数组按照元素的数据类型分：基本数据类型元素的数据，引用数据类型元素的数组（也称为对象数组）

## 二、一维数组

### 1、一维数组的声明[掌握]

数组的声明：

元素数据类型[] 数组名;

元素数据类型 数组名[];

**例如：int []arr; int arr[]; Student arr[];**

### 2、一维数组的初始化[掌握]

#### 1、静态初始化 直接指定元素，由元素决定数组长度

数组名 = new 元素数据类型[]{元素1的值，元素2的值，....};

说明：

1. **静态初始化时数组的长度由{}中罗列的元素的个数决定**
2. **静态初始化时[]中不可以再写长度**
3. **如果声明与初始化一起，可以简写：**

**元素数据类型[] 数组名 = {元素1的值，元素2的值，....};**

例如：int[] arr = {1,2,3,4,5,6};

String[] arr = {“hello”,”java”,”world”};

Student[] arr = {new Student(“貂蝉”), new Student(“王昭君”, new Student(“西施”),new Student(“杨玉环”) };

一维数组静态初始化案例：

升景坊单间短期出租4个月，550元/月（水电煤公摊，网费35元/月），空调、卫生间、厨房齐全。屋内均是IT行业人士，喜欢安静。所以要求来租者最好是同行或者刚毕业的年轻人，爱干净、安静。

public class ArrayTest {

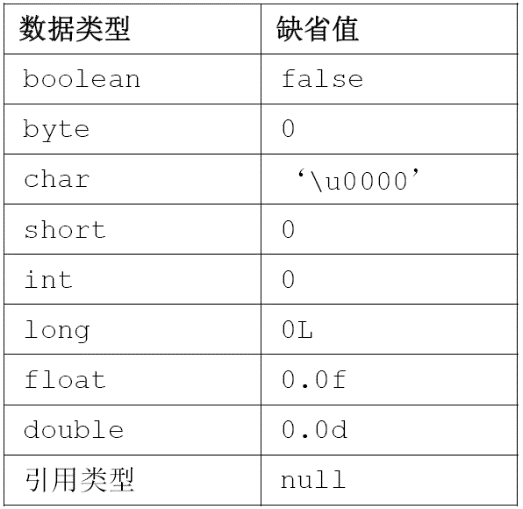
public static void main(String[] args) {

int[] arr = new int[]{8,2,1,0,3};

int[] index = new int[]{2,0,3,2,4,0,1,3,2,3,3};

String tel = "";

for(int i = 0;i < index.length;i++){

 tel += arr[index[i]];

}

System.out.println("联系方式：" + tel);

}

}

#### 2、动态初始化

数组名 = new 元素数据类型[数组的长度];

说明：

1. 动态初始化时数组的长度在[]中指定，而且必须指定
2. 动态初始化时每一个元素有默认值

**注意：**

1. 无论是静态初始化还是动态初始化必须确定数组长度。

动态初始化在创建数组对象时，显式指明长度，静态初始化，由列出的元素个数确定数组的长度

1. int[] arr = new int[5]{1,2,3,4,5};//错误
2. int[5] arr ;//错误
3. 数组的长度一旦确定，不可改变
4. 元素的数据类型是引用数据类型，在刚刚动态初始化后，元素的值是null，并没有创建元素的对象，元素的对象需要单独创建。

例如：Student[] arr = new Student[3];此时arr[i]为null，arr[i]并没有指向任何学生对象，此时arr[i].属性或arr[i].方法都会报空指针异常

String[] arr = new String[3];此时arr[i]也为null，arr[i]中并没有存储任何字符串对象

### 3、一维数组元素的访问与赋值[掌握]

数组的元素表示方式：数组名[下标]

下标的范围是：[0,数组的长度) 或[0,数组的长度-1]

数组元素的赋值：数组名[下标] = 值;

例如：

nums[0] = 15;

students[1] = new Student(“柴美女”);

### 4、一维数组的长度[掌握]

数组名.length

### 5、一维数组的遍历[掌握]

#### 普通for

for(int i=0;i<数组名.length;i++){

System.out.println(数组名[i]);

}

#### 增强for：foreach循环

for(元素的数据类型 变量名 : 被遍历的数组名){

System.out.println(变量名); //每循环一次，变量名代表一个元素

}

区别：

普通for可以修改数组元素的值，增强for不行；

普通for可以获取下标信息，增强for没有下标信息；

**仅遍历时增强for效率高**

### 6、一维数组的内存分析[理解]

数组对象是在堆中创建的，元素相当于数组对象的成员变量

创建数组对象会在堆中开辟连续的空间

#### 元素是基本数据类型的一维数组

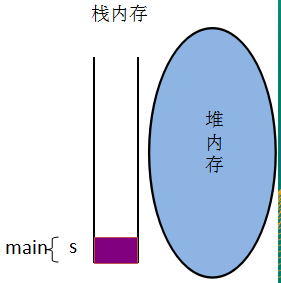
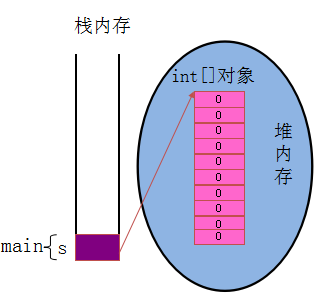
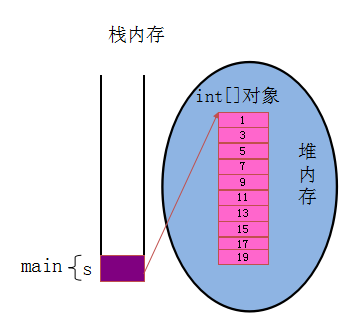
int[] s;

s = new int[10];

for ( int i=0; i<10; i++ ) {

s[i] =2\*i+1;

}

#### 引用数据类型元素的一维数组（面向对象之后讲）

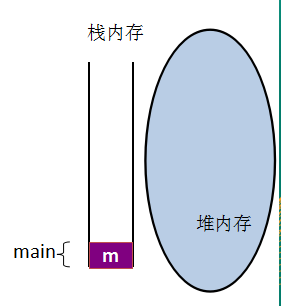
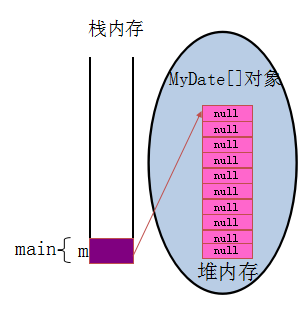
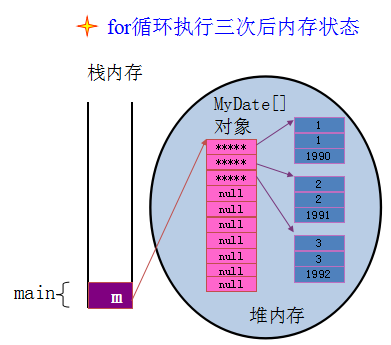
MyDate[] m; //MyDate是自定义类型，好比Student

m = new MyDate[10]; //引用类型的元素的默认值null

for ( int i=0; i<10; i++ ) {

m[i] =new MyDate(i+1, i+1,1990+i); //每一个元素又new对象

}

## 三、数组的常见异常[理解]

（1）越界 ArrayIndexOutOfBoundsException

（2）空指针 NullPointerException

## 四、数组的常见算法

### 1、找出数组中元素的最大值或最小值[掌握]

|  |
| --- |
| **public** **static** **void** main(String[] args) {  **int**[] array = { 6, 1, 4, 8, 3 };  **int** max = array[0];  **for** (**int** i = 0; i < array.length; i++) {  **if** (max < array[i]) {  max = array[i];  }  }  System.***out***.println("最大值是" + max);  } |

### 2、找出数组中元素的最大/小值及其下标[掌握]

说明：如果多个相同以第一个为准

|  |
| --- |
| **public** **static** **void** main(String[] args) {  **int**[] array = { 6, 1, 4, 8, 3 };  **int** maxIndex = 0;  int max = array[0]  **for** (**int** i = 0; i < array.length; i++) {  **if** (max < array[i]) {  max = array[i];  maxIndex = i;  }  }  System.***out***.println("最大值是" + max + "，其下标为：" + maxIndex);  } |
| **public** **static** **void** main(String[] args) {  **int**[] array = { 6, 1, 4, 8, 3 };  **int** maxIndex = 0;  **for** (**int** i = 0; i < array.length; i++) {  **if** (array[maxIndex] < array[i]) {  maxIndex = i;  }  }  System.***out***.println("最大值是" + array[maxIndex] + "，其下标为：" + maxIndex);  } |

### 3、求元素的总和/平均值[掌握]

|  |
| --- |
| **public** **static** **void** main(String[] args) {  **int**[] array = { 6, 1, 4, 8, 3 };  **int** sum = 0;  **for** (**int** i = 0; i < array.length; i++) {  sum += array[i];  }  System.***out***.println("数组元素的和：" + sum + ",平均值为：" + ((**double**) sum) / array.length);  } |

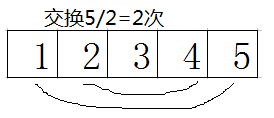
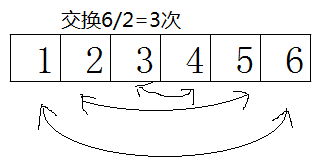
### 4、实现数组元素反转[理解]

#### 方法一：借用一个数组[理解]

|  |
| --- |
| **public** **static** **void** main(String[] args) {  **int**[] array = { 6, 1, 4, 8, 3 };  **int**[] temp = **new** **int**[array.length];  **for** (**int** i = 0; i < array.length; i++) {  temp[i] = array[array.length - 1 - i];  }  array = temp;  //遍历显示略  } |

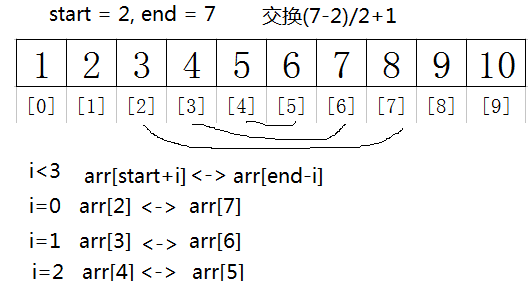
#### 方法二：交换length/2次[理解]

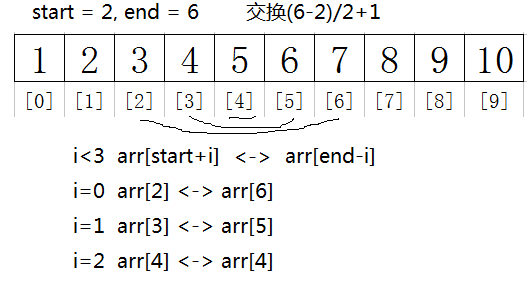
|  |
| --- |
| **public** **static** **void** main(String[] args) {  **int**[] array = { 6, 1, 4, 8, 3 };  **for** (**int** i = 0; i < array.length / 2; i++) {  **int** temp = array[i];  array[i] = array[array.length - i - 1];  array[array.length - i - 1] = temp;  }  //遍历显示略  } |



#### 反转数组的指定部分[了解]

|  |
| --- |
| **public** **static** **void** main(String[] args) {  **int**[] array = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };  *reverseArray*(array, 2, 7);  System.***out***.println(Arrays.*toString*(array));  }  **public** **static** **void** reverseArray(**int**[] array, **int** start, **int** end) {  **if** (array ==**null** || end <= start || end >= array.length || start < 0) {  **return**;  }  **for** (**int** i = 0; i < (end - start) / 2 + 1; i++) {  **int** temp = array[start + i];  array[start + i] = array[end - i];  array[end - i] = temp;  }  } |





### 5、数组的复制[掌握]

#### 复制整个数组

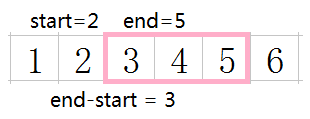
|  |
| --- |
| **public** **static** **void** main(String[] args) {  **int**[] array = {6,1,4,8,3};  // int[] copyArray = array;//不是数组的复制  **int**[] copyArray = **new** **int**[array.length];  **for**(**int** i=0;i<array.length;i++){  copyArray[i] = array[i];  }  // copyArray[2] = 100;//修改copyArray不会影响array原数组  //遍历显示略  } |

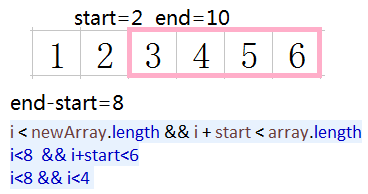
#### 复制一个指定长度的数

|  |
| --- |
| import java.util.Scanner;  public class ArrayCopy\_2{  public static void main(String[] args){  int[] arr = {1,2,3,4,5};    Scanner input = new Scanner(System.in);  int length;  while(true){  System.out.print("请输入新数组的长度(大于0)：");  length = input.nextInt();  if(length>0){  break;  }  }    //(1)创建新数组  int[] newArr = new int[length];    //(2)复制元素  //既要保证newArr[i]不越界，也要保证arr[i]不越界  for(int i=0; i<arr.length && i<newArr.length; i++){  newArr[i] = arr[i];  }    //(3)遍历显示  for(int i=0; i<newArr.length; i++){  System.out.println(newArr[i]);  }  }  } |

#### 复制数组的指定部分（了解）

|  |
| --- |
| **public** **static** **void** main(String[] args) {  **int**[] arr = { 1, 2, 3, 4, 5, 6 };  **int**[] arr2 = *copyOfRange*(arr, 2, 5);  System.***out***.println(Arrays.*toString*(arr2));  }  **public** **static** **int**[] copyOfRange(**int**[] array, **int** start, **int** end) {  **if** (array == **null** || end < start || start < 0) {  **return** array;  }  **int**[] newArray = **new** **int**[end - start];  **for** (**int** i = 0; i < newArray.length && i + start < array.length; i++) {  newArray[i] = array[i + start];  }  **return** newArray;  } |





### 6、数组元素的排序

#### 冒泡排序（Bubble Sort）[掌握]

冒泡排序是一种简单的排序算法，它不是效率最高的排序，但是所有编程语言都作为排序的入门学习算法。它重复地走访过要排序的数列，一次比较两个元素，如果他们的顺序错误就把他们交换过来。走访数列的工作是重复地进行直到没有再需要交换，也就是说该数列已经排序完成。这个算法的名字由来是因为越小的元素会经由交换慢慢“浮”到数列的顶端。

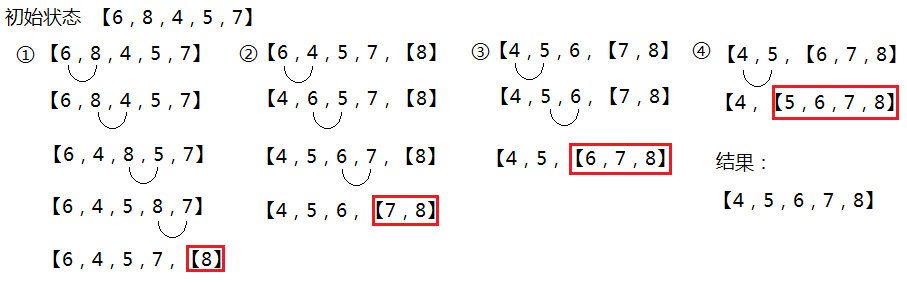
（1）小的“浮”

|  |
| --- |
| public class Sort{  public static void main(String[] args){  int[] array = { 6, 8, 4, 5, 7 };    //冒泡排序：通过两两比较，相邻元素比较，每一轮只负责把本轮“最大”或“最小”换到了正确的位置  //外循环控制的是一共比较几轮：轮数 = 元素的总个数 - 1； 轮数 = 数组的长度 - 1;  for(int i=1; i<array.length; i++){  //两种思路：  //方式一：每一轮都从[0]开始比较，从左往右  //for(int j=0; j<...; j++)  //方式二：每一轮都从[array.length-1]开始比较，从右往左  //for(int j=array.length-1; j>...; j--)    //方式二：  //第一轮：比较4次，j=4,3,2,1 j>0 i=1 j>=1 j>=i  //第二轮：比较3次，j=4,3,2 j>1 i=2 j>=2  //第三轮：比较2次，j=4,3 j>2 i=3 j>=3  //第四轮：比较1次，j=4 j>3 i=4 j>=4  for(int j=array.length-1; j>=i; j--){  //从小到大  //从右往左比  //把小的往前移动  //如果后面的元素比前面的小，就交换  //j最大的，和j最小的代入，看是否会出现下标越界  if(array[j] < array[j-1]){  int temp = array[j];  array[j] = array[j-1];  array[j-1] = temp;  }  }    }  //省略遍历显示  }  } |



（2）大的“沉”

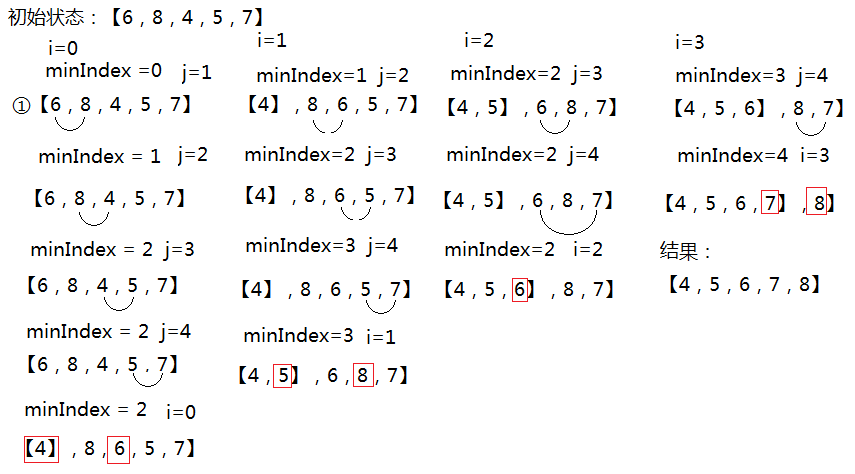
|  |
| --- |
| public class Sort{  public static void main(String[] args){  int[] array = { 6, 8, 4, 5, 7 };    //冒泡排序：通过两两比较，相邻元素比较，每一轮只负责把本轮“最大”或“最小”换到了正确的位置  //外循环控制的是一共比较几轮：轮数 = 元素的总个数 - 1； 轮数 = 数组的长度 - 1;  for(int i=1; i<array.length; i++){  //两种思路：  //方式一：每一轮都从[0]开始比较，从左往右  //for(int j=0; j<...; j++)  //方式二：每一轮都从[array.length-1]开始比较，从右往左  //for(int j=array.length-1; j>...; j--)    //选择方式一：  //本题：5个元素  //第一轮：比较4次，即要内循环循环4次， j=0,1,2,3 j<4 i=1 array.length-i 5-1=4  //第二轮：比较3次，即要内循环循环3次， j=0,1,2 j<3 i=2  //第三轮：比较2次，即要内循环循环2次， j=0,1 j<2 i=3  //第四轮：比较1次，即要内循环循环1次， j=0 j<1 i=4  for(int j=0; j<array.length-i; j++){  //从小到大  //从左到右  //把大的往右边移动  //如果前面的元素比后面的元素大，交换  if(array[j] > array[j+1]){  int temp = array[j];  array[j] = array[j+1];  array[j+1] = temp;  }  }  }  //省略遍历显示  }  } |



#### 直接选择排序(Selection sort)[了解]

选择排序(Selection sort)是一种简单直观的排序算法。它的工作原理如下。首先在未排序序列中找到最小（大）元素，存放到排序序列的起始位置，然后，再从剩余未排序元素中继续寻找最小（大）元素，然后放到已排序序列的末尾。以此类推，直到所有元素均排序完毕。

|  |
| --- |
| **public** **static** **void** main(String[] args) {  **int**[] array = { 6, 8, 4, 5, 7 };  **for** (**int** i = 0; i < array.length - 1; i++) {// 一共比较array.length-1轮  **int** minIndex = i;// 假设array[i]最小 minIndex记录本轮的真正最小值下标  **for** (**int** j = i + 1; j < array.length; j++) {// 每轮比较array.length-i次，找出本轮真正最小值的下标  **if** (array[minIndex] > array[j]) {// 每次拿假设最小值去和array[j]比较  minIndex = j;  }  }  **if** (minIndex != i) {//如果已经不是最初假设的那个，就交换“假设的最小值”与“真正的最小值”的位置  **int** temp = array[i];  array[i] = array[minIndex];  array[minIndex] = temp;  }  }  System.***out***.println(Arrays.*toString*(array));  } |



### 7、二分查找

|  |
| --- |
| /\*  数组的算法：  二分查找  效率提高的原因：  每轮排除一半  前提：数组是有序的  \*/  public class TestBinarySearch{  public static void main(String[] args){  int[] arr = {1,2,3,4,5,6,7,8,9,10};    int findValue = 1;//要找的数  int left = 0;//左边的下标  int right = arr.length - 1;//右边的下标  int mid = (right + left)/2; //中间的下标    int index = -1;//目标的下标    while(true){  if(right < left){  break;  }    if(findValue == arr[mid]){//找到了  index = mid;  break;  }else if(findValue < arr[mid]){//该值在arr[mid]的左边  right = mid - 1;  mid = (right + left)/2;  }else{//该值在arr[mid]的左边  left = mid + 1;  mid = (right + left)/2;  }  }    if(index ==-1){  System.out.println("未找到");  }else{  System.out.println(findValue + "在数组的" + index );  }  }  } |

## 五、数组的操作工具类（查看API）

java.util.Arrays类包含了用来操作数组（比如排序和搜索）的各种方法。Arrays拥有一组static方法。

* equals(数组1，数组2)：比较两个array是否相等。array拥有相同元素个数，且所有对应元素两两相等。
* fill(数组名,填充的值)：使用某个值填充array数组。
* fill(数组名,开始下标，结束下标，填充的值)：使用某个值填充array数组的某个部分。
* sort(数组名)：用来对array进行排序。
* binarySearch(数组名，要查找的数据)：在排好序的array中寻找元素。 只能用于已经排好序的数组，返回下标
* toString(数组名)：打印一个数组，格式：[元素值1，元素值2....]
* copy(原数组，新数组的长度)：复制数组，从下标为0的元素开始复制，复制n个元素。不够就默认值填充
* copyOfRange(原数组，from，to)：复制数组任意部分，从from到to（不包含）个元素
* 另：System.arraycopy(原数组，开始位置，目标数组，开始位置，长度)：array的复制。

|  |
| --- |
| /\*  练习：  1、从键盘输入本组学员的成绩，放到数组中  2、用foreach显示所有学员的成绩  3、排序：从低到高  4、查找是否有正好60分的，如果有返回位置  5、复制成绩最低三名构成新数组  6、用工具类打印成绩最低三名成绩  \*/  import java.util.Scanner;  import java.util.Arrays;  class TestArraysExer{  public static void main(String[] args){  //1、声明一个数组并创建一个数组  int[] scores = new int[5];    //2、从键盘输入成绩  Scanner input = new Scanner(System.in);  for(int i=0; i<scores.length; i++){  //成绩存在数组的元素中  //为元素赋值  System.out.print("请输入第" + (i+1) + "个学员的成绩:");  scores[i] = input.nextInt();  }    //3、显示成绩  //用foreach显示所有学员的成绩  System.out.println("本组学员的成绩如下：");  for(int s : scores){  System.out.println(s);  }  //4、排序：从低到高  Arrays.sort(scores);    System.out.println("排序后的结果：" + Arrays.toString(scores));    //5、查找60分  int index = Arrays.binarySearch(scores, 60);  if(index<0){  System.out.println("没有正好60分的");  }else{  System.out.println("60分的索引位置：" + index);  }    //6、复制成绩最低三名构成新数组  //int[] newArray = Arrays.copyOfRange(scores, 0, 3);  int[] newArray = Arrays.copyOf(scores, 3);    //7、用工具类打印成绩最低三名成绩  System.out.println("成绩最低的三名同学是：" + Arrays.toString(newArray));  }  } |

## 六、多维数组

多维数组：当数组的元素还是一个数组时，就构成了多维数组，一般使用到二维数组就算比较复杂了，再更多维的使用的不多。

### 1、二维数组的声明[掌握]

格式：元素的数据类型[][] 二维数组名;

例如：int[][] arr; Student[][] students;

通俗的理解：一维数组比喻成平房，就一层楼，有几间房

二维数组比喻成楼房，有几层楼，每层楼各有几间房

### 2、二维数组的初始化

不管静态初始化还是动态初始化，**第一维的长度必须确定**

#### 静态初始化

静态初始化：数组对象的创建和元素赋值操作同时进行

语法格式：

数组名 = new 元素的数据类型[][]{{元素1，元素2...},{元素1，元素2...},{元素1，元素2...}...};

当声明与静态初始化同时进行，可以简写：

元素的数据类型[][] 数组名 = {{元素1，元素2...},{元素1，元素2...},{元素1，元素2...}...};

例如：

int[][] arr;

arr = new int[][]{{1,2,3},{4,5,6},{7,8,9,10}};

或

int[][] arr = {{1,2,3},{4,5,6},{7,8,9,10}};

相当于：

如果把arr看成一个一维数组，那么它有三个元素，每个元素又分别是一个一维数组，分别为

arr[0]就是{1,2,3}，对于arr[0]数组的元素是arr[0][0]、arr[0][1]、arr[0][2]

arr[1]就是{4,5,6}，对于arr[1]数组的元素是arr[1][0]、arr[1][1]、arr[1][2]

arr[2]就是{7,8,9,10}，对于arr[2]数组的元素是arr[2][0]、arr[2][1]、arr[2][2]、arr[2][3]

|  |
| --- |
| //1、声明一个二维数组  int[][] scores;    //4个小组  //2、数组初始化  //（1）静态初始化  scores = new int[][]{{89,90,80},{56,99,100,88},{45,67,98},{34,100,97}};    //3、使用for循环遍历  for(int i=0; i<scores.length; i++){//遍历几行  //System.out.println(scores[i]);//scores[i]是一个一维数组  for(int j=0; j<scores[i].length; j++){//遍历每一行的几列  System.out.print(scores[i][j]+"\t");  }  //换行  System.out.println();  } |

#### 动态初始化

动态初始化：**数组对象的创建与元素的赋值分开进行**

#### （1）规则矩形

规则的二维数组：每一行的列数是一样的

（1）指定行数和列数

二维数组名 = new 元素的数据类型[行数][列数];

此时元素都是默认值

（2）为元素一一赋值

二维数组名[行下标][列下标] = 值;

例如：

arr = new int[5][6]; //声明了5行6列的数组

//或者说声明了一个5个元素一维数组arr，然后每个元素又是一个长度为6的一维数组

此时所有的数组对象都创建完毕，而且所有的元素都有默认值。

arr[0]就是{0,0,0,0,0,0}，

arr[1]就是{0,0,0,0,0,0}，

arr[2]就是{0,0,0,0,0,0}，

arr[3]就是{0,0,0,0,0,0}，

arr[4]就是{0,0,0,0,0,0}，

|  |
| --- |
| import java.util.Scanner;  public class TestTwoArrayInitialize{  public static void main(String[] args){  //1、静态初始化  //int[][] scores = {{1,2,3,4},{4,5,6,7}};    //2、动态初始化  int[][] scores = new int[3][5];//3行5列，一共是15个元素，此时元素都是默认值    /\*  scores[0][0] = 89;  scores[1][1] = 90;  scores[2][2] = 56;  \*/  Scanner input = new Scanner(System.in);  for(int i=0; i<scores.length; i++){  System.out.println("第" + (i+1) + "组：");  for(int j=0; j<scores[i].length; j++){  System.out.print("请输入第" + (i+1) + "组的" + (j+1) + "个学员的成绩：");  scores[i][j] = input.nextInt();  }  }    for(int i = 0; i<scores.length; i++){  for(int j=0; j<scores[i].length; j++){  System.out.print(scores[i][j] + "\t");  }  System.out.println();  }  }  } |

#### （2）不规则矩阵

不规则的二维数组：每一行的列数是不相同的

（1）指定行数

**二维数组名 = new 元素的数据类型[行数][];**

（2）确定每一行的列数，为每一行分配空间

二维数组名[行下标] = new 元素的数据类型[该行的列数];

（3）为元素一一赋值

二维数组名[行下标][列下标] = 值;

例如：

arr = new int[5][];

此时arr的5个子数组都未创建。

arr[0]为null，此时arr[0]，想要存储数据，必须先new int[长度]，然后再为元素赋值。

arr[1]为null，同上

arr[2]为null，同上

arr[3]为null，同上

arr[4]为null，同上

例如：

arr = new int[][]{{1,2,3,4},null,null,null};

此时有三个子数组对象未创建。但是行数已经确定，一共有4行

arr[0]就是{1,2,3,4}，对于arr[0]数组的元素是arr[0][0]、arr[0][1]、arr[0][2]、arr[0][3]

arr[1]就是null，此时arr[1]，想要存储数据，必须先new int[长度]，然后再为元素赋值。

arr[2]就是null，同上

arr[3]就是null，同上

|  |
| --- |
| import java.util.Scanner;  public class TestTwoArrayInitialize\_2{  public static void main(String[] args){  //1、静态初始化  //int[][] scores = {{1,2,3,4},{4,5,6,7}};    //2、动态初始化  int[][] scores = new int[3][];//确定有3行，此时行有几列不确定的，行的空间没有分配    scores[0] = new int[5];//指定第一行有5列  scores[1] = new int[4];//指定第二行有4列  scores[2] = new int[2];//指定第三行有2列    scores[0][0] = 89;  scores[1][1] = 79;  scores[2][1] = 89;      //遍历显示元素  for(int i = 0; i<scores.length; i++){  for(int j=0; j<scores[i].length; j++){  System.out.print(scores[i][j] + "\t");  }  System.out.println();  }  }  } |

### 3、二维数组的长度

二维数组的长度（通常指行数）：数组名.length

每一行的列数：数组名[行标].length

### 4、二维数组的遍历

#### 普通for

for(int i=0; i<行数; i++){

for(int j=0; j<每一行的列数; j++){

元素：二维数组名[i][j]

}

}

行数：二维数组名.length

每一行的列数：二维数组名[i].length

#### 增强for

for(元素的数据类型[] 行临时名称 ： 二维数组名){

for(元素的数据类型 元素的临时名称 ： 行临时名称){

}

}

### 5、二维数组的特殊写法

|  |
| --- |
| public static void main(String[] args) {  //一维数组  **int[] array1;//推荐**  int array2[];    //二维数组  int[][] array3;//推荐  int array4[][];  int[] array5[];    //面试题：array6和array7分别是什么数组？  int[] array6,array7[];  //array6是一维数组，array7是二维数组  } |

### 6、元素是引用类型的二维数组（面向对象基础后再看）

注意：区分数组对象，行对象，元素对象

需求：按小组保存全班同学的个人信息（姓名和电话号码），全班分为4个小组，每个小组的人数不等。

分别是4、5、6、6人

|  |
| --- |
| **public** **static** **void** main(String[] args) {  Student[][] all = **new** Student[4][];//4个小组  all[0]= **new** Student[4];//第一组4个人  all[1]= **new** Student[5];//第二组5个人  all[2]= **new** Student[6];//第三组6个人  all[3]= **new** Student[6];//第四组6个人  all[0][0] = **new** Student("张三","1001");//第1组第一个学生  all[0][1] = **new** Student("李四","1002");//第1组第二个学生  all[0][2] = **new** Student("王五","1003");//第1组第三个学生  all[0][3] = **new** Student("赵六","1004");//第1组第四个学生  all[1][0] = **new** Student("刘备","10086");//第2组第一个学生  //....  } |

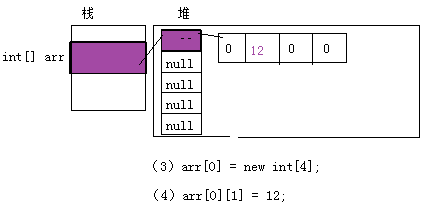
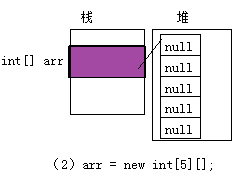
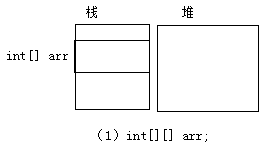
### 6、二维数组的内存分析

（1）int[][] arr;

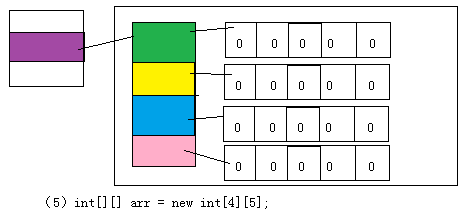
（2）arr = new int[5][];

（3）arr[0] = new int[4];

（4）arr[0][1] = 12;



（5）int[][] arr = new int[4][5];

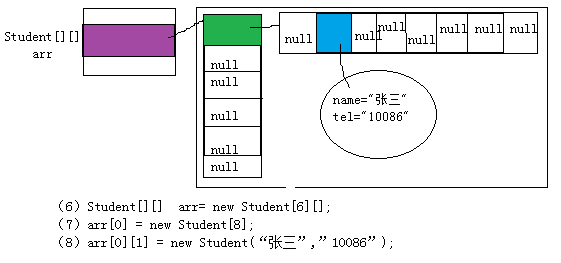


以下部分学完面向对象基础之后再看

（6）Student[][] arr= new Student[6][];

（7）arr[0] = new Student[8];

（8）arr[0][1] = new Student(“张三”,”10086”);



## 七、经典案例

1、从键盘输入本组学员人数，然后创建数组，然后在输入每一个学员的成绩，统计最高分，最低分，平均分

2、

1

2 2

3 3 3

4 4 4 4

5 5 5 5 5

3、

1 1 1 1 1

2 2 2 2 2

3 3 3 3 3

4 4 4 4 4

4、使用二维数组打印一个 10 行杨辉三角.

1

1 1

1 2 1

1 3 3 1

1 4 6 4 1

1 5 10 10 5 1

....

【提示】

1. 第一行有 1 个元素, 第 n 行有 n 个元素

2. 每一行的第一个元素和最后一个元素都是 1

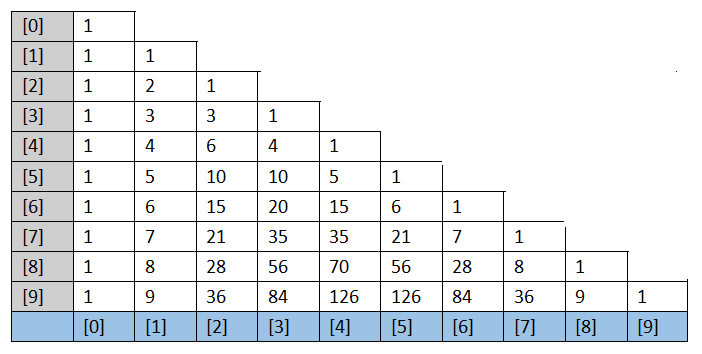
3. 从第三行开始, 对于非第一个元素和最后一个元素的元素.

yanghui[i][j] = yanghui[i-1][j-1] + yanghui[i-1][j];

yanghui[2][1] = yanghui[1][0] + yanghui[1][1] = 1+1 =2

yanghui[3][1] = yanghui[2][0] + yanghui[2][1] = 1+2 =3

yanghui[3][2] = yanghui[2][1] + yanghui[2][2] = 2+1 =3



|  |
| --- |
| public class TestYangHui {  public static void main(String[] args) {  //第一步：声明一个二维数组，并确定行数  int[][] yanghui = new int[10][];    //第二步：确定每行的列数  for(int i=0;i<yanghui.length;i++){  yanghui[i] = new int[i+1];  }    //第三步：为每一个元素赋值  /\* //第一行：  yanghui[0][0]=1;  //第二行：  yanghui[1][0]=1;  yanghui[1][1]=1;\*/    for(int i=0;i<yanghui.length;i++){  //从第三行开始, 对于非第一个元素和最后一个元素的元素.  for(int j=0;j<yanghui[i].length;j++){  if(j==0 || j==yanghui[i].length-1){  yanghui[i][j]=1;  }else{  yanghui[i][j] = yanghui[i-1][j-1] + yanghui[i-1][j];  }  }  }    //第四步：打印  for(int[] a : yanghui){  for(int num : a){  System.out.print(num+"\t");  }  System.out.println();  }  }  } |