java基础知识 第1天

**【学习目标】理解、了解、应用、记忆**

通过今天的学习，参训学员能够：（解释的时候说出二级目标的掌握程度）

1. **【应用】理解Java概述、helloworld案例**
2. 【了解】知道java语言的发展史
3. 【理解】JVM、JRE、JDK分别是什么？有什么作用？有什么关系
4. 【应用】能够操作常用的DOS命令
5. 【应用】独立下载安装JDK
6. 【应用】独立编写HelloWorld案例,并能够调试问题,使程序正常运行
7. **【理解】理解工具安装 、配置环境变量、注释、关键字**
8. 【应用】独立安装并配置Notepad软件
9. 【应用】独立配置path环境变量
10. 【理解】阐述注释、关键字的作用及分类、特点
11. **【理解】常量、变量、数据类型、标识符**
12. 【理解】阐述常量的特点及分类
13. 【理解】阐述变量的作用及定义格式
14. 【理解】阐述数据类型分类
15. 【理解】阐述标识符的组成规则及注意事项
16. 【应用】能够定义变量，使用变量
17. 【理解】阐述数据类型转换之隐式数据类型转换和强制数据类型转换的方式及注意事项
18. **【应用】完成Eclipse工具的下载、安装、配置并能使用eclipse编写代码**
19. 【应用】独立下载并安装eclipse
20. 【记忆】独立配置eclipse的工作空间
21. 【应用】应用eclipse的辅助键和快捷键编写代码
22. 【应用】能够完成导入项目操作和删除项目操作
23. **【理解】能够灵活的使用java中的各种运算符**
24. 【应用】独立编写代码使用(+,-,\*,/,%,++,--等)算数运算符完成数学运算
25. 【应用】阐述取余和除法的区别
26. 【理解】阐述字符和字符串参与加法运算操作需要注意的事项
27. 【理解】阐述++和--的作用，以及++，--在前或在后的区别
28. 【理解】阐述赋值运算符和扩展运算符有哪些
29. 【理解】阐述关系运算符有哪些，关系运算符得到的结果是什么
30. 【理解】阐述逻辑运算符的基本使用及&与&&的区别
31. 【应用】阐述三元运算符使用格式及使用场景
32. **【应用】使用Scanner完成键盘录入功能**
33. 【应用】阐述键盘录入的基本步骤
34. 【应用】独立编写代码，使用键盘录入的方式完成：求两个数的和、判断两个数是否相等、获取三个数中的最大值

# Java概述

## Java语言发展史和平台概述

A:Java语言发展史

詹姆斯·高斯林（James Gosling），1977年获得了加拿大卡尔加里大学计算机科学学士学位，1983年获得了美国卡内基梅隆大学计算机科学博士学位，毕业后到IBM工作，但不受重视。后来转至Sun公司，于1991年，和其他几位工程师一起合作参与了Green计划（绿色计划），并开发了一种称为Oak（橡树）的面向对象语言。1995年5月23日，Oak语言改名为Java，Java语言由此诞生。

SUN(Stanford University Network，斯坦福大学网络公司)

B:Java语言版本

\* Java语言诞生 1995-05-23

\* JDK 1.0 1996-01-23

\* JDK 1.1 1.1.4后都有了自己的名字 1997-02-18

\* J2SE 1.2 Playground 运动场 1998-12-04

里程碑式的版本，由此分成三大块：J2SE、J2ME、J2EE

\* J2SE 1.4.0 Merlin 灰背隼 2002-02-13

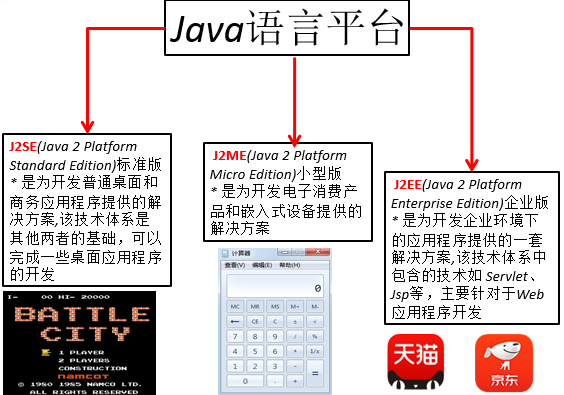
\* JAVASE 5.0 (1.5.0) Tiger 老虎 2004-09-29

\* JAVASE 6.0 (1.6.0) Mustang 野马 2006-12-11

甲骨文（Oracle）收购SUN公司 2009-04-20

\* JAVASE 7.0 (1.7.0) Dolphin 海豚 2011-07-28

\* JAVASE 8.0 (1.8.0) Spider 蜘蛛 2014-03-18



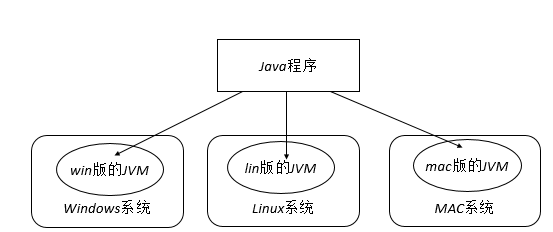
## JVM,JRE,JDK的概述

### 什么是跨平台?

平台：指的是操作系统(Windows，Linux，Mac)

跨平台：Java程序可以在任意操作系统上运行，一次编写到处运行

原理：实现跨平台需要依赖Java的虚拟机 JVM （Java Virtual Machine）



### JVM JRE JDK说明

A:什么是JVM

JVM是java虚拟机(JVM Java Virtual Machine)，java程序需要运行在虚拟机上，不同平台有自己的虚拟机，因此java语言可以跨平台。

B:什么是JRE

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。

JRE:JVM+类库。

C:什么是JDK

JDK是提供给Java开发人员使用的，其中包含了java的开发工具，也包括了JRE。所以安装了JDK，就不用在单独安装JRE了。

其中的开发工具：编译工具(javac.exe) 打包工具(jar.exe)等。

JDK:JRE+JAVA的开发工具。

D:为什么JDK中包含一个JRE

为什么JDK中包含一个JRE呢？

开发完的程序，需要运行一下看看效果。

E:JDK,JRE,JVM的作用和关系

JDK包含JRE和开发工具包。

JRE包含核心类库和JVM。

## 常用dos命令

### 打开控制台

* + win + R，然后cmd回车

### 常用命令

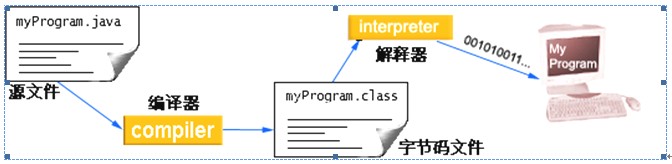
* + d: 回车 盘符切换
  + dir(directory):列出当前目录下的文件以及文件夹
  + cd (change directory)改变指定目录(进入指定目录)
    - 进入 cd 目录；cd 多级目录
    - 回退 cd.. ；cd\
  + cls : (clear screen)清屏
  + exit : 退出dos命令行

## 下载安装JDK

请参考《JDK下载安装文档.doc》安装步骤

## helloworld案例

### 执行流程



### 编写代码步骤

首先定义一个类：

public class 类名

在类定义后加上一对大括号：

{}

在大括号中间添加一个主(main)方法/函数：

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

在主方法的大括号中间添加一行输出语句：

System.out.println(“HelloWorld”);

### 案例代码一

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

运行代码步骤：

* 在命令行模式中，输入javac命令对源代码进行编译，生成字节码文件
  + javac 源文件名.java
* 编译完成后，如果没有报错信息，输入java命令对class字节码文件进行解释运行,执行时不需要添加.class扩展名，运行名是class关键字后面的类名
  + java HelloWorld

### HelloWorld案例常见问题

A:找不到文件(都演示一下，让学生看看出现的都是什么问题)

a:文件扩展名隐藏导致编译失败

b:文件名写错了

B:单词拼写问题(都演示一下，让学生看看出现的都是什么问题)

a:class写成Class

b:String写成string

c:System写成system

d:main写成mian

C:括号匹配问题(都演示一下，让学生看看出现的都是什么问题)

a:把类体的那对大括号弄掉一个

b:把方法体的那对大括号弄掉一个

c:把输出语句的那对小括号弄掉一个

D:中英文问题(都演示一下，让学生看看出现的都是什么问题)

a:提示信息：错误: 非法字符: \????的格式、错误: 编码GBK的不可映射字符

注意：java编程中需要的基本上都是英文字符

## 重点和小结

1、java语言的发展史

2、JVM、JRE、JDK分别是什么？有什么作用？有什么关系

3、常用的DOS命令

4、如何下载安装JDK

5、编写HelloWorld案例,并能够调试问题,使程序正常运行

# 环境配置

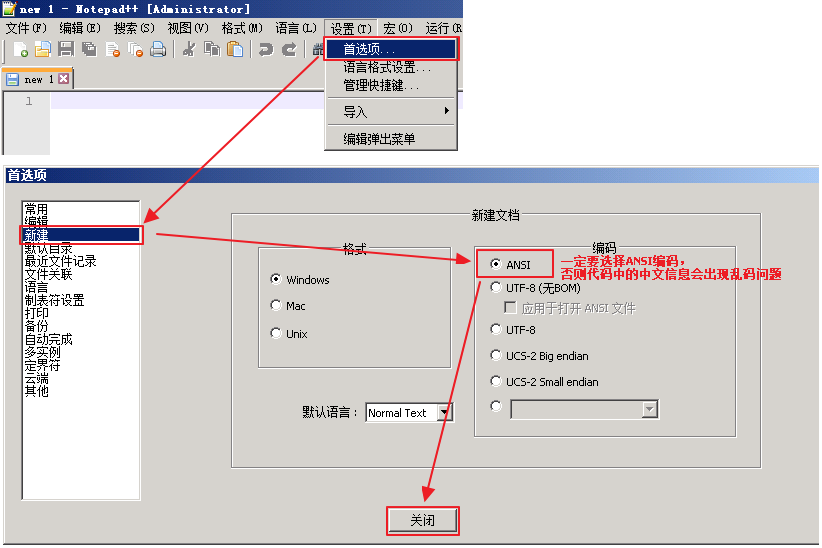
## 工具安装

### Notepad软件的安装和配置

为了让我们写的程序错误看起来更直接，我们安装一款高级记事本软件。

Notepad软件的安装和配置

设置 – 首选项 – 新建 – 默认语言和编码（ANSI）



## 环境变量配置

### 案例说明

为什么要配置

* + 程序的编译和执行需要使用到javac和java命令，所以只能在bin目录下写程序
  + 实际开发中，不可能把程序写到bin目录下，所以我们必须让javac和java命令在任意目录下能够访问

如何配置

* + 创建新的变量名称：JAVA\_HOME

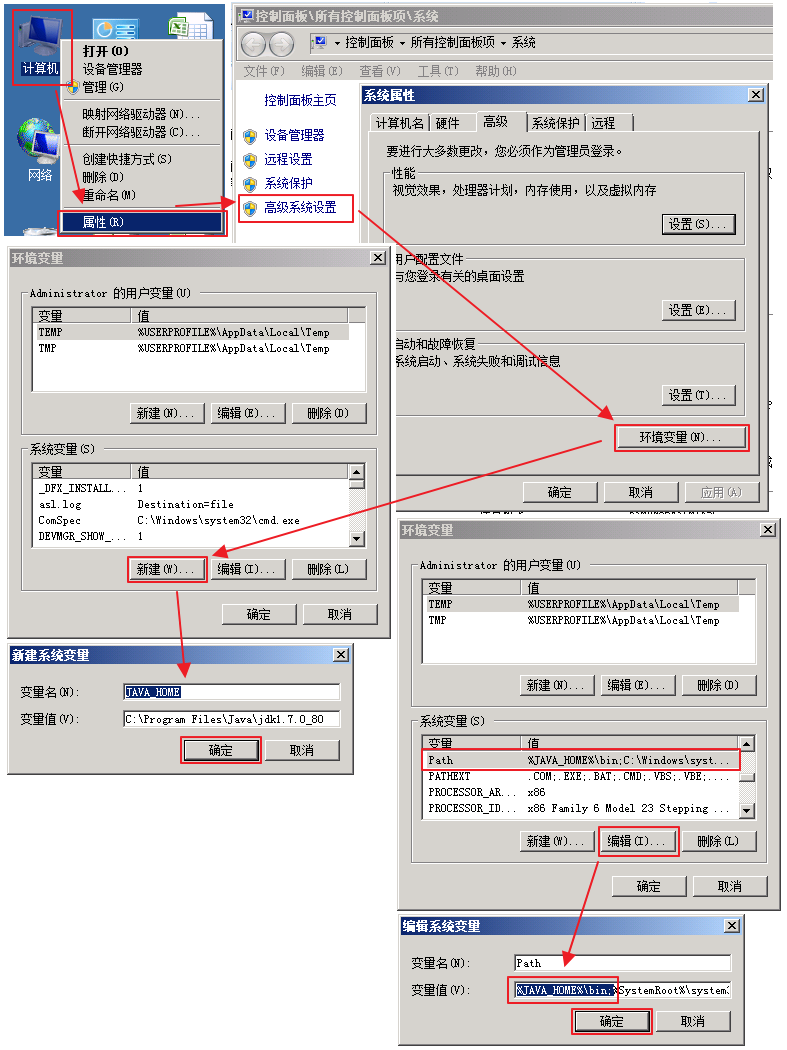
计算机-右键属性-高级系统设置-高级-环境变量-系统变量

* + 为JAVA\_HOME添加变量值：JDK安装目录

C:\Program Files\Java\jdk1.7.0\_80

* + 在path环境变量最前面添加如下内容

%JAVA\_HOME%\bin;



## 注释

### 注释概述

A: 什么是注释

* 用于解释或说明程序的文字

B: Java中注释分类

单行注释

* + 格式： //注释文字

多行注释

* + 格式： /\* 注释文字 \*/

文档注释

* + 格式：/\*\* 注释文字 \*/

C: 注释的作用

a:解释说明程序

b:提高程序的阅读性

### 案例代码二

|  |
| --- |
| /\*\*  注释：用于解释说明程序的文字    分类：  单行  多行  文档注释    作用：解释说明程序，提高程序的阅读性  \*/    //这是我的HelloWorld案例  public class HelloWorld {  /\*  这是main方法  main是程序的入口方法  所有代码的执行都是从main方法开始的  \*/  public static void main(String[] args) {  //这是输出语句  System.out.println("HelloWorld");  }  } |

## 关键字

### 关键字概述

* + 被Java语言赋予特定含义的单词

### 关键字特点

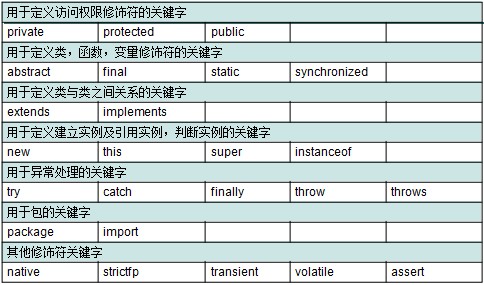
* + 组成关键字的字母全部小写
  + 常用的代码编辑器,针对关键字有特殊的颜色标记，非常直观，所以我们不需要去死记硬背，在今后的学习中重要的关键字也会不断的出来。

### 案例代码三

|  |
| --- |
| /\*  关键字：被Java语言赋予特定含义的单词    特点：  A:组成关键字的字母全部小写  B:常见的代码编辑器,针对关键字有特殊的颜色标记  \*/  public class HelloWorld {  public static void main(String[] args) {  System.out.println("HelloWorld");  }  } |

关键字举例:





## 重点和小结

1、安装并配置Notepad软件

2、配置path环境变量

3、阐述注释、关键字的作用及分类、特点

# 语法格式

## 常量

### 常量概述

* + 在程序执行的过程中，其值不可以发生改变的量

### 常量分类

* + 字符串常量 用双引号括起来的内容(“HelloWorld”)
  + 整数常量 所有整数(12,-23)
  + 小数常量 所有小数(12.34)
  + 字符常量 用单引号括起来的内容(‘a’,’A’,’0’)
  + 布尔常量 较为特有，只有true和false
  + 空常量 null(数组部分讲解)

### 案例代码四

|  |
| --- |
| /\*  常量：在程序执行的过程中，其值不可以发生改变的量    常量分类：  A:字符串常量 "HelloWorld"  B:整数常量 12,-23  C:小数常量 12.34  D:字符常量 'a','0'  E:布尔常量 true,false  F:空常量 null(后面讲解)  \*/  public class ChangLiang {  public static void main(String[] args) {  //字符串常量  System.out.println("HelloWorld");    //整数常量  System.out.println(12);  System.out.println(-23);    //小数常量  System.out.println(12.34);    //字符常量  System.out.println('a');  System.out.println('0');    //布尔常量  System.out.println(true);  System.out.println(false);  }  } |

## 变量

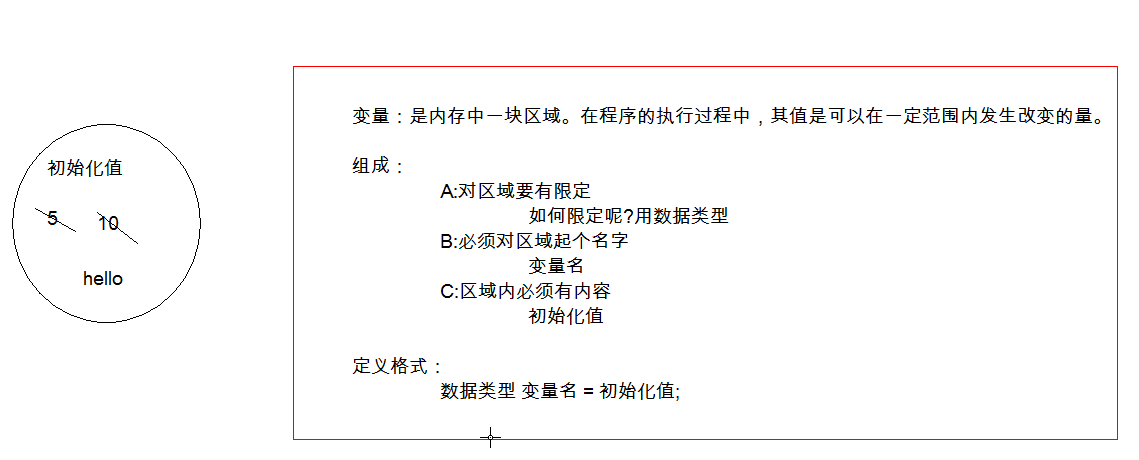
### 变量概述

* + 在程序执行的过程中，在某个范围内其值可以发生改变的量
  + 从本质上讲，变量其实是内存中的一小块区域

### 变量定义格式

* + 数据类型 变量名 = 初始化值;
  + 注意：格式是固定的，记住格式，以不变应万变

### 变量图解



## 数据类型

### 计算机存储单元

变量是内存中的小容器，用来存储数据。那么计算机内存是怎么存储数据的呢？无论是内存还是硬盘，计算机存储设备的最小信息单元叫“位（bit）”，我们又称之为“比特位”，通常用小写的字母b表示。而计算机最小的存储单元叫“字节（byte）”，通常用大写字母B表示，字节是由连续的8个位组成。

除了字节外还有一些常用的存储单位，大家可能比较熟悉，我们一起来看看：

* + 1B（字节） = 8bit
  + 1KB = 1024B
  + 1MB = 1024KB
  + 1GB = 1024MB
  + 1TB = 1024GB

### 数据类型概述和分类

A:为什么有数据类型

Java语言是强类型语言，对于每一种数据都定义了明确的具体数据类型，在内存中分配了不同大小的内存空间

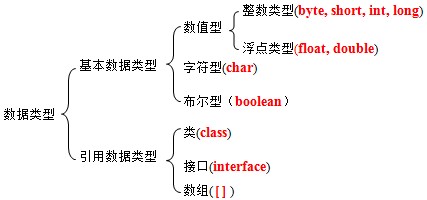
B:Java中数据类型的分类

基本数据类型

引用数据类型

面向对象部分讲解





## 标识符

### 标识符概述

A 作用

* + 给包,类,方法,变量等起名字

B 组成规则

* + 由字符(字母、数字)，下划线\_，美元符$组成
    - 这里的字符采用的是unicode字符集，所以包括英文大小写字母，中文字符，数字字符等。
  + 注意事项
  + 不能以数字开头
  + 不能是Java中的关键字

C : 命名原则:见名知意

a包

最好是域名倒过来,要求所有的字母小写

b类或者接口

如果是一个单词首字母大写

如果是多个单词每个单词首字母大写(驼峰标识)

c方法或者变量

如果是一个单词全部小写

如果是多个单词,从第二个单词首字母大写

d常量

如果是一个单词,所有字母大写

如果是多个单词,所有的单词大写,用下划线区分每个单词

### 案例代码五

|  |
| --- |
| /\*  标识符：就是给包,类,方法,变量起名字的符号。    组成规则：  A:unicode字符  数字字符,英文大小写,汉字(不建议使用汉字)  B:下划线\_  C:美元符$    注意事项  A:不能以数字开头  B:不能是java中的关键字    常见命名规则：  A:基本要求  见名知意  B:常见的命名  a:包(其实就是文件夹,用于对类进行管理)  全部小写,多级包用.隔开  举例：com，com.igeek  b:类  一个单词首字母大写  举例：Student,Car  多个单词每个单词的首字母大写  举例：HelloWorld  c:方法和变量  一个单词首字母小写  举例：age,show()  多个单词从第二个单词开始每个单词的首字母大写  举例：maxAge,getAge()  \*/  public class BiaoShiFu {  public static void main(String[] args) {  //定义变量  //数据类型 变量名 = 初始化值;  int a = 10;    //正确  int b2 = 20;  //错误  //int 2b = 30;    //不能是java中的关键字  //错误  //int public = 40;  }  } |

## 定义变量

### 基本数据类型变量的定义和使用

变量的定义格式：

数据类型 变量名 = 初始化值;

基本数据类型：

byte,short,int,long,float,double,char,boolean

注意：

整数默认是int类型，定义long类型的数据时，要在数据后面加L

浮点数默认是double类型，定义float类型的数据时，要在数据后面加F

### 案例代码六

|  |
| --- |
| public class VariableDemo {  public static void main(String[] args) {  //定义byte类型的变量  byte b = 10;  System.out.println(10);  System.out.println(b);    //定义short类型的变量  short s = 100;  System.out.println(s);    //定义int类型的变量  int i = 10000;  System.out.println(i);    //定义long类型的变量  long l = 1000000000000000L;  System.out.println(l);    //定义float类型的变量  float f = 12.34F;  System.out.println(f);    //定义double类型的变量  double d = 12.34;  System.out.println(d);    //定义char类型的变量  char c = 'a';  System.out.println(c);    //定义boolean类型的变量  boolean bb = false;  System.out.println(bb);  }  } |

### 变量定义的注意事项

* 变量未赋值,不能直接使用
  + 引出变量的第二种使用格式
* 变量只在它所属的范围内有效
  + 变量在哪对大括号内，变量就属于哪对大括号
* 一行上可以定义多个变量，但是不建议

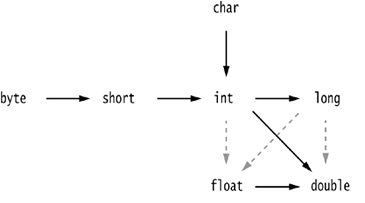
### 案例代码七

|  |
| --- |
| /\*  变量定义注意事项：  1:变量未赋值,不能直接使用  2:变量只在它所属的范围内有效  变量属于它所在的那对大括号  3:一行上可以定义多个变量,但是不建议  \*/  public class VariableDemo2 {  public static void main(String[] args) {  //定义变量  int a = 10;  System.out.println(a);    int b;  b = 20; //变量在使用前赋值都是可以的  System.out.println(b);      {  int c = 100;  System.out.println(c);  }  //System.out.println(c);    /\*  int aa,bb,cc;  aa = 10;  bb = 20;  cc = 30;  \*/    /\*  int aa = 10;  int bb = 20;  int cc = 30;  \*/    int aa=10,bb=20,cc=30;  }  } |

## 数据类型转换

### 隐式数据类型转换

取值范围小的数据类型与取值范围大的数据类型进行运算,会先将小的数据类型提升为大的,再运算



### 案例代码八

|  |
| --- |
| /\*  +:是一个运算符，做加法运算的。  我们在做运算的时候，一般要求参与运算的数据类型必须一致。    类型转换：  隐式转换  强制转换    隐式转换  byte,short,char -- int -- long -- float -- double  \*/  public class TypeCastDemo {  public static void main(String[] args) {  //直接输出了运算的结果  System.out.println(3 + 4);    //定义两个int类型的变量  int a = 3;  int b = 4;  int c = a + b;  System.out.println(c);    //定义一个byte类型,定义一个int类型  byte bb = 2;  int cc = 5;  System.out.println(bb + cc);    //我能不能不直接输出，用一个变量接受呢?  //用变量接受，这个变量应该有类型  //可能损失精度  //byte dd = bb + cc;  int dd = bb + cc;  System.out.println(dd);  }  } |

### 强制类型数据转换

强制转换的格式

\* b = (byte)(a + b);

强制转换的注意事项

\* 如果超出了被赋值的数据类型的取值范围得到的结果会与你期望的结果不同

### 案例代码九

|  |
| --- |
| /\*  强制转换：  目标类型 变量名 = (目标类型) (被转换的数据);  不建议强制转换，因为会有精度的损失。  \*/  public class TypeCastDemo2 {  public static void main(String[] args) {  int a = 3;  byte b = 4;  int c = a + b;  //byte d = a + b;  byte d = (byte) (a + b);  }  } |

## 重点和小结

1、常量的特点及分类

2、变量的作用及定义格式

3、数据类型分类

4、标识符的组成规则及注意事项

5、定义变量，使用变量

6、数据类型转换之隐式数据类型转换和强制数据类型转换的方式及注意事项

# Eclipse开发工具

## Eclipse概述和安装

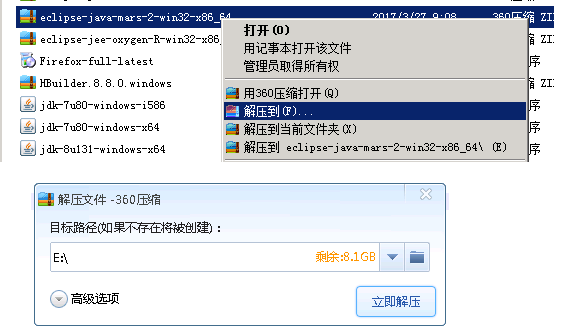
Eclipse是一个IDE(集成开发环境)IDE(Integrated Development Environment)

集成了代码编写功能，分析功能，编译功能，调试功能等一体化的开发软件。

Eclipse的特点描述：免费、纯Java语言编写、免安装、扩展性强

下载和安装、下载 http://eclipse.org/

安装：绿色版、解压就可以使用(Eclipse)

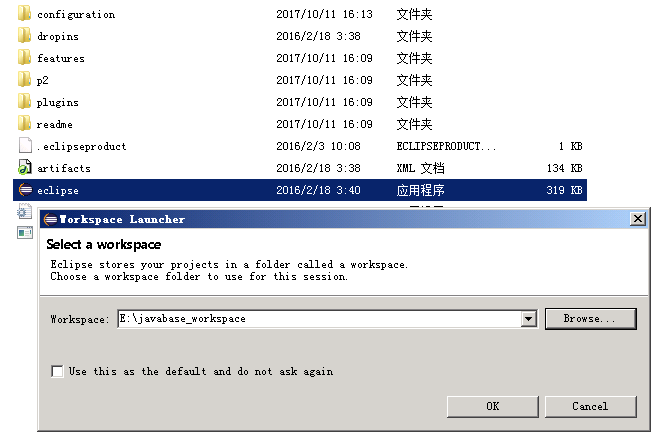


## Eclipse的基本使用

### Eclipse基本操作

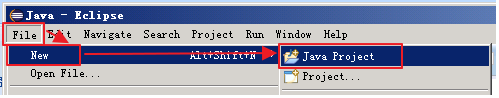
选择工作空间

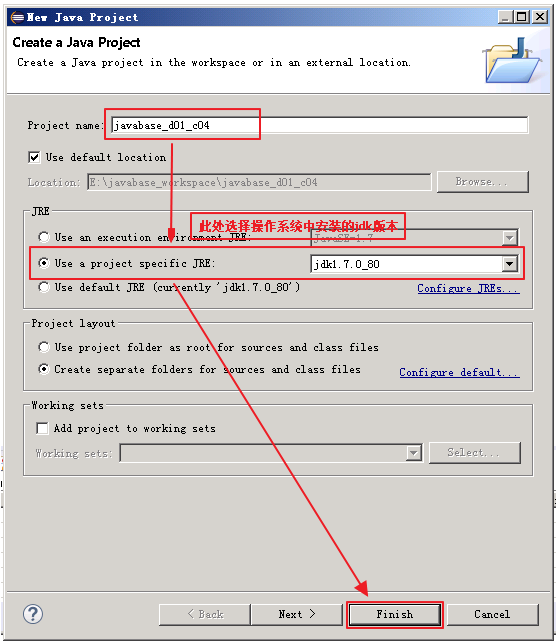
工作空间 其实就是我们写的源代码所在的目录



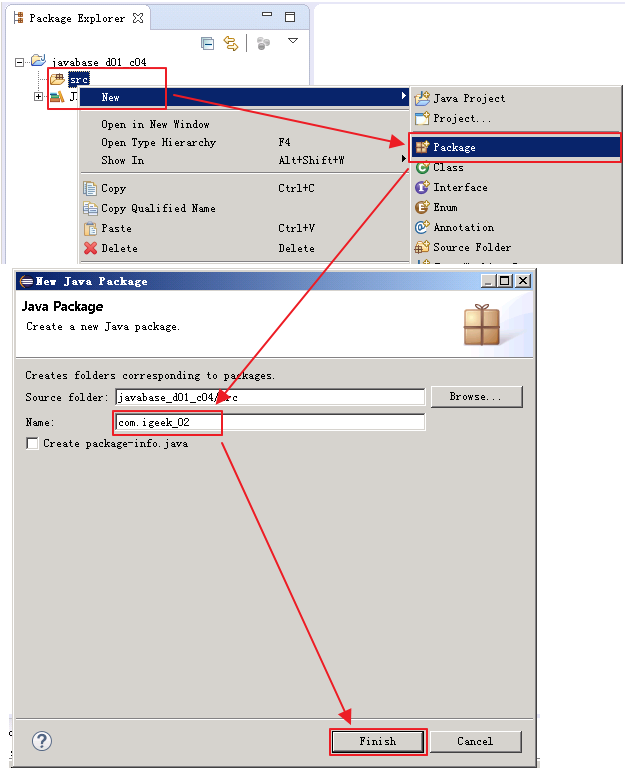
用Eclipse来完成一个HelloWorld案例

A:创建Java项目：点击File或者在最左侧空白处，选择Java项目，在界面中写一个项目名称javabase\_d01\_c04，然后Finish即可。



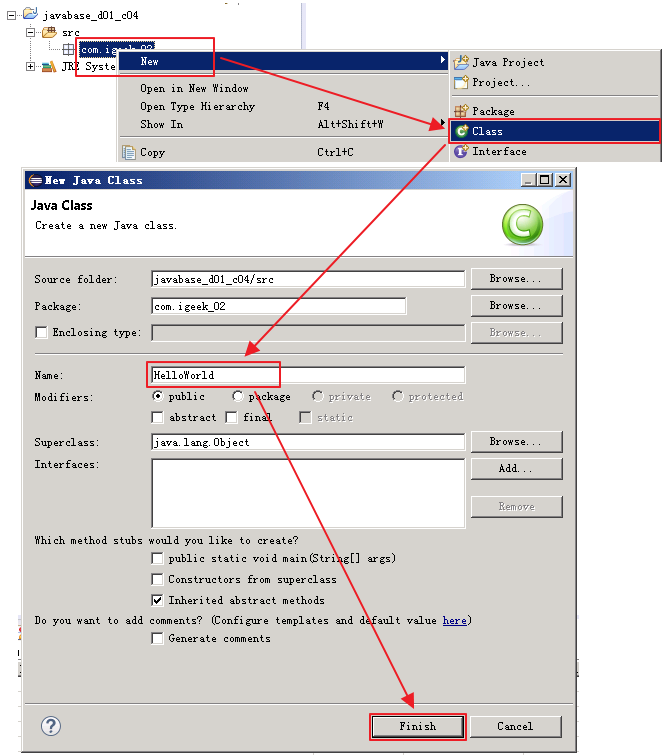


B:创建包：展开项目，在源包src下建立一个包com.igeek\_02



C:创建类：在com. igeek\_02包下建立一个类HelloWorld

在界面中写一个类名：HelloWorld，然后finish即可。



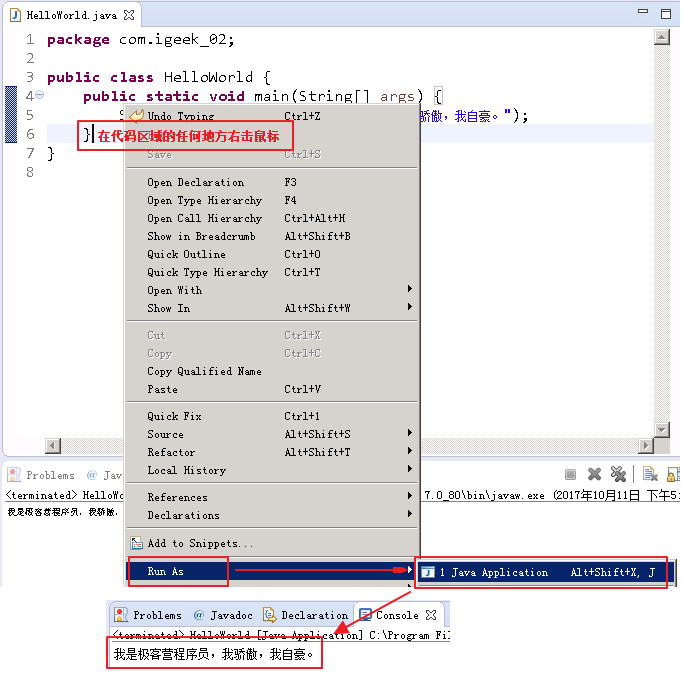
D:编写代码：在HelloWorld类写main方法，在main方法中写

一条输出语句：我是极客营程序员，我骄傲，我自豪。

E:编译：自动编译，在保存的那一刻帮你做好了

F:运行 选择要运行的文件或者在要运行的文件内容中

右键 -- Run as - Java Application即可，在控制台可以看到运行结果



### 代码案例十

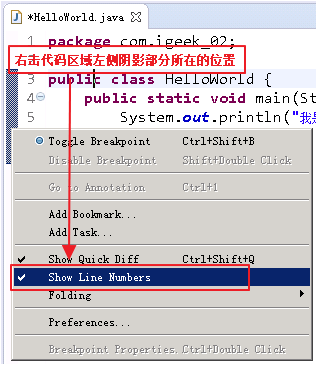
|  |
| --- |
| **package** com.igeek\_02;  /\*\*  \* **@ClassName**: HelloWorld  \* **@Description**: 在控制台打印相关信息  \* **@date** 2017年10月11日 下午6:49:38  \* Company www.igeekhome.com  \*  \*/  **public** **class** HelloWorld {  **public** **static** **void** main(String[] args) {  System.***out***.println("我是极客营程序员，我骄傲，我自豪。");  }  } |

## Eclipse工作空间的基本配置

A:行号的显示和隐藏

显示：在代码区域的最左边的空白区域，右键 -- Show Line Numbers即可。

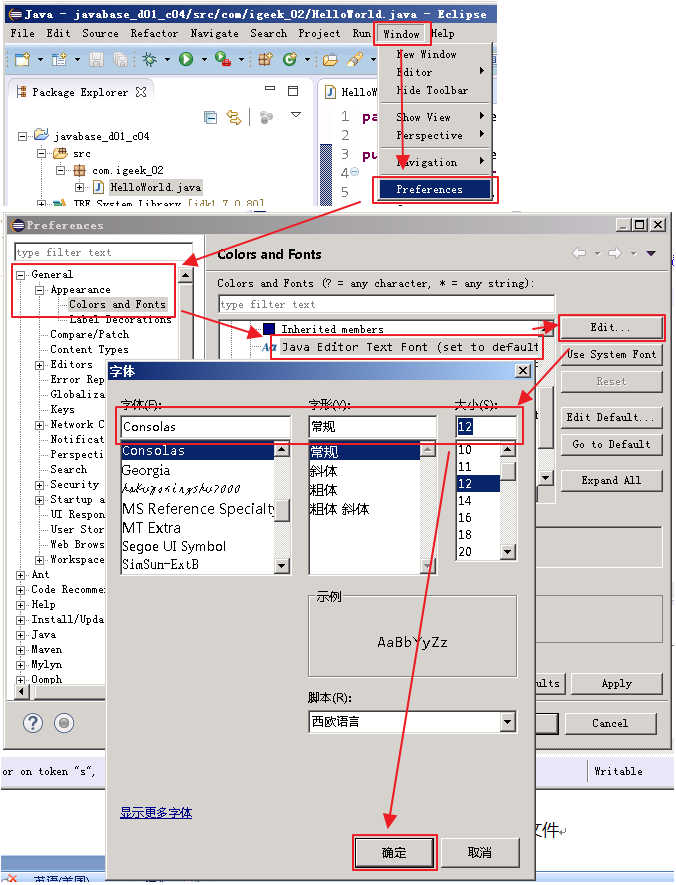
隐藏：把上面的动作再做一次。



B:字体大小及颜色

a:Java代码区域的字体大小和颜色：

window -- Preferences -- General -- Appearance -- Colors And Fonts -- Java -- Java Edit Text Font



b:控制台（类似a步骤过程）

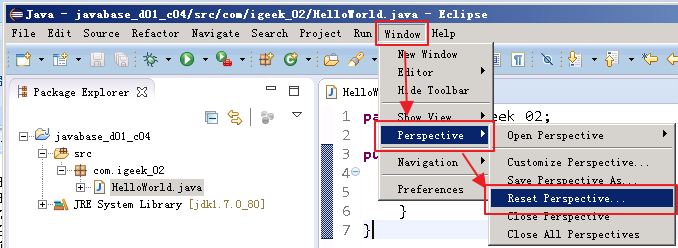
window -- Preferences -- General -- Appearance -- Colors And Fonts -- Debug -- Console font

c:其他文件（类似a步骤过程）

window -- Preferences -- General -- Appearance -- Colors And Fonts -- Basic -- Text Font

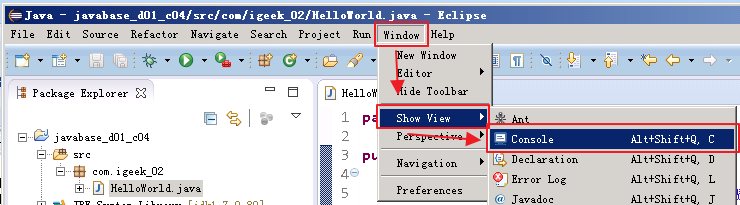
C:窗体给弄乱了，怎么办?

window -- Perspective -- Reset Perspective



D:控制台找不到了，怎么办?

Window--Show View—Console



## Eclipse中辅助键和快捷键的使用

### 常用辅助键和快捷键概述

内容辅助键 alt+/：

main 然后alt+/

sysout 然后alt+/

快捷键：

注释：

单行 选中内容，ctrl+/, 再来一次取消

多行 选中内容，ctrl+shift+/, ctrl+shift+\

格式化 ctrl+shift+f

### 代码案例十一

|  |
| --- |
| **package** com.igeek\_04;  /\*\*  \* **@ClassName**: HelloWorld  \* **@Description**: 演示快捷键的使用  \* **@date** 2017年10月11日 下午6:49:38  \* Company www.igeekhome.com  \*  \* 内容辅助键：alt+/  \* A:main方法 main,然后alt+/,回车  \* B:输出语句 sysout,然后然后alt+/,回车  \*  \* 快捷键：  \* A:注释  \* 单行 选中内容，ctrl+/,再来一次就是取消  \* 多行 选择内容，ctrl+shift+/,ctrl+shift+\  \* B:格式化  \* ctrl+shift+f  \*/  **public** **class** HelloWorld {  **public** **static** **void** main(String[] args) {  System.***out***.println("HelloWorld1");  System.***out***.println("HelloWorld2");  System.***out***.println("HelloWorld3");  System.***out***.println("HelloWorld4");  System.***out***.println("HelloWorld5");  }  } |

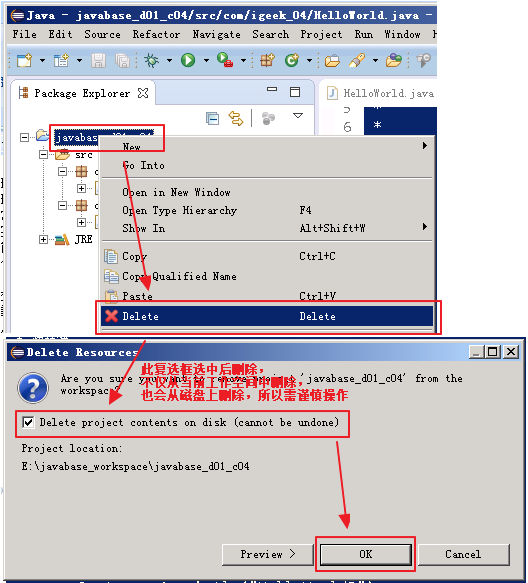
## Eclipse中项目的删除和导入

A:删除项目

选中项目 – 右键 – 删除

从项目区域中删除

从硬盘上删除



B:导入项目

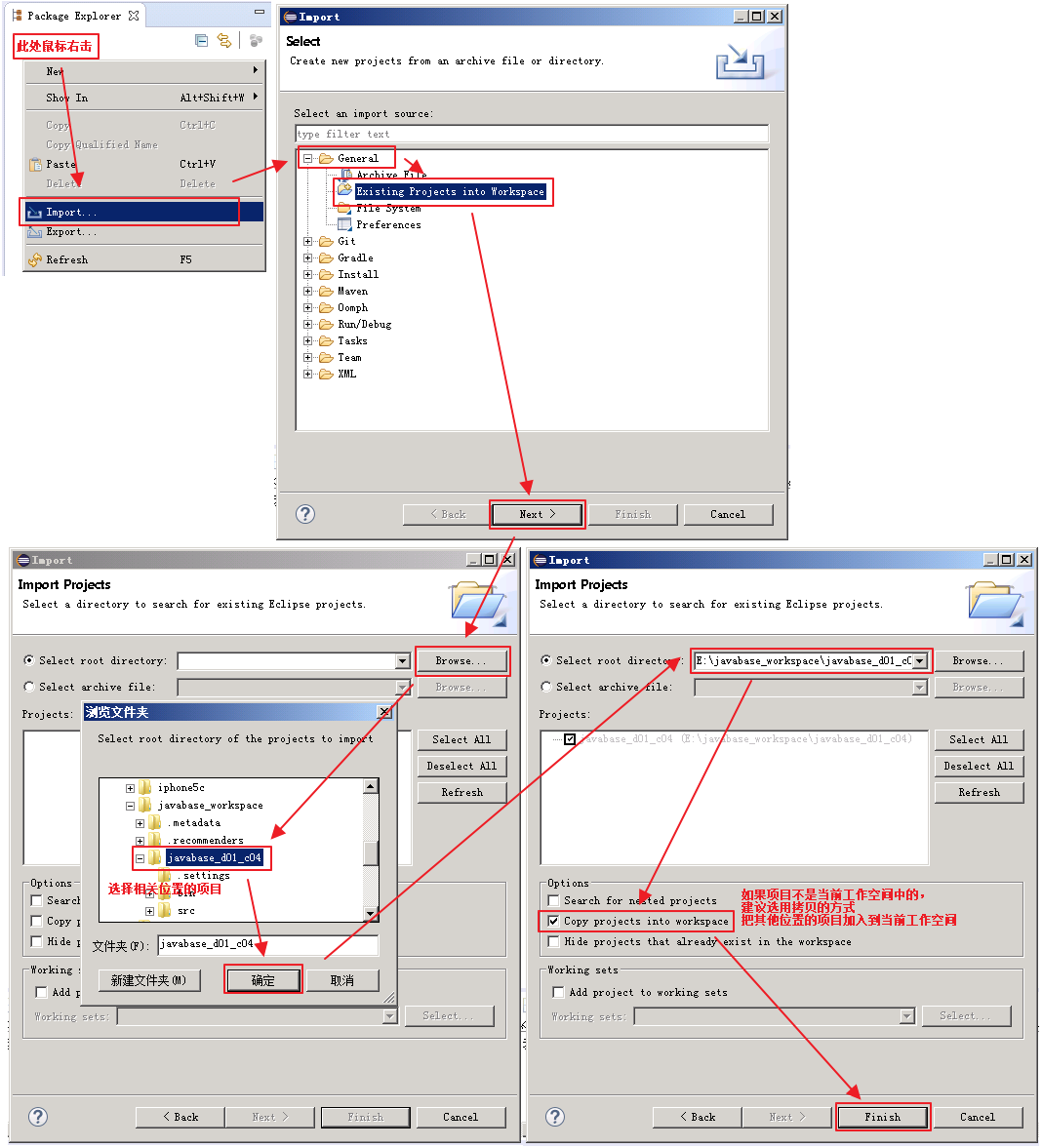
在项目区域右键找到import

找到General，展开，并找到

Existing Projects into Workspace

点击next,然后选择你要导入的项目

注意：这里选择的是项目名称



# 运算符

## 算数运算符

* 运算符

对常量和变量进行操作的符号称为运算符

* 表达式

用运算符把常量或者变量连接起来符号java语法的式子就可以称为表达式不同运算符连接的式子体现的是不同类型的表达式

定义两个int类型的变量a,b，做加法(a + b)

* 常用运算符

算术运算符

赋值运算符

关系运算符

逻辑运算符

三元运算符

### 算数运算符的基本用法

### 算数运算符概述

A:什么是运算符

就是对常量和变量进行操作的符号

B:算数运算符有哪些

\* +,-,\*,/,%,++,--

### 代码案例十二

|  |
| --- |
| **package** com.igeek\_01;  /\*\*  \* **@ClassName**: HelloWorld  \* **@Description**: 算术运算符的使用  \* **@date** 2017年10月11日 下午6:49:38  \* Company www.igeekhome.com  \*  \* 运算符：对常量和变量进行操作的符号  \* 表达式：用运算符连接起来的符合java语法的式子。不同类型的运算符连接起来的式子是不同的表达式。  \* 举例：定义两个int类型的变量a,b,  \* a + b  \*  \* 运算符分类：  \* 算术运算符，赋值运算符，关系运算符，逻辑运算符，三元运算符。  \*  \* 算术运算符：  \* +，-，\*，/的基本使用  \*/  **public** **class** OperatorDemo {  **public** **static** **void** main(String[] args) {  // 定义两个变量  **int** a = 3;  **int** b = 4;  System.***out***.println(a + b);  System.***out***.println(a - b);  System.***out***.println(a \* b);  System.***out***.println(a / b);  // 整数相除只能得到整数，要想得到小数，就必须有浮点数参与运算  System.***out***.println(3 / 4.0);  System.***out***.println(3.0 / 4);  }  } |

### 算数运算符取余和除法的区别

%：取余运算符。得到的是两个相除数据的余数。

/：除法运算符。得到是两个相除数据的商。

使用场景： %，判断两个数据是否整除。

### 代码案例十三

|  |
| --- |
| **package** com.igeek\_01;  /\*\*  \* **@ClassName**: HelloWorld  \* **@Description**: 算数运算符取余和除法的区别  \* **@date** 2017年10月11日 下午6:49:38  \* Company www.igeekhome.com  \*  \*/  **public** **class** OperatorDemo2 {  **public** **static** **void** main(String[] args) {  **int** a = 5;  **int** b = 3;  System.***out***.println(a / b);  System.***out***.println(a % b);  }  } |

### 字符和字符串参与加法操作

字符参与运算

其实是拿该字符对应的数值来操作

‘a’ 97

‘A’ 65

‘0’ 48

字符串参与运算

这里其实做的不是加法运算，而是字符串拼接。

字符串和其他类型的数据做拼接，结果是字符串类型的。

### 代码案例十四

|  |
| --- |
| **package** com.igeek\_01;  /\*\*  \* **@ClassName**: OperatorDemo3  \* **@Description**: 字符和字符串参与加法操作的区别  \* **@date** 2017年10月11日 下午7:10:27  \* Company www.igeekhome.com  \*  \* 整数的加法。  \* 字符参与加法操作。拿字符在计算机中底层存储对应的数据值来参与运算的。  \* '0' 48  \* 'a' 97  \* 'A' 65  \* 字符串参与加法操作。  \* 这里的+其实不是加法，而是字符串连接符。  \*/  **public** **class** OperatorDemo3 {  **public** **static** **void** main(String[] args) {  // 整数加法  **int** a = 10;  **int** b = 20;  System.***out***.println(a + b);  System.***out***.println("------------------");  // 字符参与加法操作  **char** c = '0';  **char** c2 = 'a';  System.***out***.println(a + c);  System.***out***.println(a + c2);  System.***out***.println("------------------");  // 字符串参与加法操作  System.***out***.println("hello" + a);  System.***out***.println("hello" + a + b); // "hello"+10,然后再和b进行拼接  System.***out***.println(a + b + "hello");  }  } |

### 算数运算符++和--的用法

++,--运算符：对变量做加1或者减1的操作。

++或者--既可以放在变量的后面，也可以放在变量的前面。

单独使用的时候，++或者--无论是放在变量的前面还是后面，结果是一样的。

参与操作的时候：

如果++或者--在变量的后面，先拿变量参与操作，后变量做++或者--

如果++或者--在变量的前面，先变量做++或者--，后拿变量参与操作

### 代码案例十五

|  |
| --- |
| **package** com.igeek\_01;  /\*\*  \* **@ClassName**: OperatorDemo4  \* **@Description**: 算数运算符++和--的用法  \* **@date** 2017年10月11日 下午7:13:51  \* Company www.igeekhome.com  \*  \*/  **public** **class** OperatorDemo4 {  **public** **static** **void** main(String[] args) {  **int** a = 10;  System.***out***.println("a:" + a);  // 单独使用  // a++;  // ++a;  // System.out.println("a:" + a);  // 参与操作使用  // int b = a++;  **int** b = ++a;  System.***out***.println("a:" + a);  System.***out***.println("b:" + b);  }  } |

## 赋值运算符

### 赋值运算符分类

基本的赋值运算符：=

扩展的赋值运算符：+=,-=,\*=,/=,%=

+=: a+=20;相当于a = (a的数据类型)(a + 20);

### 代码案例十六

|  |
| --- |
| **package** com.igeek\_02;  /\*\*  \* **@ClassName**: OperatorDemo  \* **@Description**: 赋值运算符的使用  \* **@date** 2017年10月11日 下午7:16:38  \* Company www.igeekhome.com  \*  \* 赋值运算符：  \* A:基本 =  \* B:扩展 +=,-=,\*=,...  \*  \* +=:  \* a+=20;  \* 相当于  \* a = (a的数据类型)(a + 20);  \*/  **public** **class** OperatorDemo {  **public** **static** **void** main(String[] args) {  // 把10赋值给int类型的变量a  **int** a = 10;  // += 把左边和右边的数据进行运算，最后赋值给左边。左边的只能是变量  a += 10;// 相当于a = a + 10  System.***out***.println("a:" + a);  System.***out***.println("----------------------");  **short** s = 10;  // s += 20; // 相当于 s = s + 20;  s = (**short**) (s + 20);  System.***out***.println("s:" + s);  }  } |

## 关系运算符

### 基本使用及注意事项

关系运算符包含以下内容:

==,!=,>,>=,<,<=

关系运算符的结果都是boolean型，也就是要么是true，要么是false。

注意事项:

关系运算符“==”不能误写成“=”。

### 代码案例十七

|  |
| --- |
| **package** com.igeek\_03;  /\*\*  \* **@ClassName**: OperatorDemo  \* **@Description**: 关系运算符  \* **@date** 2017年10月11日 下午7:19:36  \* Company www.igeekhome.com  \*  \* 关系运算符：  \* ==,!=,>,>=,<,<=  \* 关系运算符的结果是boolean类型。  \*  \* 注意：  \* 千万不要把==写成=  \*/  **public** **class** OperatorDemo {  **public** **static** **void** main(String[] args) {  **int** a = 10;  **int** b = 20;  **int** c = 10;  System.***out***.println(a == b);  System.***out***.println(a == c);  System.***out***.println("-----------------");  System.***out***.println(a != b);  System.***out***.println(a != c);  System.***out***.println("-----------------");  System.***out***.println(a > b);  System.***out***.println(a > c);  System.***out***.println("-----------------");  System.***out***.println(a >= b);  System.***out***.println(a >= c);  System.***out***.println("-----------------");  **int** x = 3;  **int** y = 4;  // System.out.println(x == y);  // System.out.println(x = y);// 把y赋值给x，把x的值输出  **boolean** bb = (x == y);  // 报错  // boolean cc = (x = y);  **int** cc = (x = y);  }  } |

## 逻辑运算符

### 逻辑运算符概述

A:逻辑运算符有哪些

&,|,^,!

&&,||

B:案例演示

逻辑运算符的基本用法

C:注意事项：

a:逻辑运算符一般用于连接boolean类型的表达式或者值。

b:表达式：就是用运算符把常量或者变量连接起来的符合java语法的式子。

算术表达式：a + b

比较表达式：a == b(条件表达式)

D:结论：

&逻辑与:有false则false。

|逻辑或:有true则true。

^逻辑异或:相同为false，不同为true。

!逻辑非:非false则true，非true则false。

### 代码案例十八

|  |
| --- |
| **package** com.igeek\_04;  /\*\*  \* **@ClassName**: OperatorDemo  \* **@Description**: 逻辑运算符：用于连接关系表达式  \* **@date** 2017年10月11日 下午7:23:27  \* Company www.igeekhome.com  \*  \* 逻辑运算符：用于连接关系表达式。  \* &,|,^,!  \* &&,||  \*  \* 与：& 有false则false  \* 或：| 有true则true  \* 异或：^ 相同则false,不同则true。(男女朋友)  \* 非：! true则false,false则true  \*/  **public** **class** OperatorDemo {  **public** **static** **void** main(String[] args) {  **int** a = 10;  **int** b = 20;  **int** c = 30;  System.***out***.println((a > b) & (a > c));// false & false  System.***out***.println((a < b) & (a > c)); // true & false  System.***out***.println((a > b) & (a < c)); // false & true  System.***out***.println((a < b) & (a < c)); // true & true  System.***out***.println("---------------");  System.***out***.println((a > b) | (a > c));// false | false  System.***out***.println((a < b) | (a > c)); // true | false  System.***out***.println((a > b) | (a < c)); // false | true  System.***out***.println((a < b) | (a < c)); // true | true  System.***out***.println("---------------");  System.***out***.println((a > b) ^ (a > c));// false ^ false  System.***out***.println((a < b) ^ (a > c)); // true ^ false  System.***out***.println((a > b) ^ (a < c)); // false ^ true  System.***out***.println((a < b) ^ (a < c)); // true ^ true  System.***out***.println("---------------");  System.***out***.println((a > b)); // false  System.***out***.println(!(a > b)); // !false  System.***out***.println(!!(a > b)); // !!false  }  } |

### 逻辑运算符&&与&的区别

A:&&和&的区别?

a:最终结果一样。

b:&&具有短路效果。左边是false，右边不执行。

&是无论左边是false还是true,右边都会执行

B:||和|的区别?

a:最终结果一样

b:||具有短路效果.左边是true,右边不执行

|是无论左边是false还是true,右边都会执行

### 代码案例十九

|  |
| --- |
| **package** com.igeek\_04;  /\*\*  \* **@ClassName**: OperatorDemo2  \* **@Description**: 逻辑运算符&&与&的区别  \* **@date** 2017年10月11日 下午7:25:23  \* Company www.igeekhome.com  \*  \* &&和&的结果一样  \* ||和|的结果一样  \*  \* &&和&的区别：  \* &&如果左边是false，右边不执行。  \* &无论左边是true还是false，右边都会执行。  \*/  **public** **class** OperatorDemo2 {  **public** **static** **void** main(String[] args) {  **int** a = 10;  **int** b = 20;  **int** c = 30;  System.***out***.println((a > b) && (a > c));// false && false  System.***out***.println((a < b) && (a > c)); // true && false  System.***out***.println((a > b) && (a < c)); // false && true  System.***out***.println((a < b) && (a < c)); // true && true  System.***out***.println("---------------");  System.***out***.println((a > b) || (a > c));// false || false  System.***out***.println((a < b) || (a > c)); // true || false  System.***out***.println((a > b) || (a < c)); // false || true  System.***out***.println((a < b) || (a < c)); // true || true  System.***out***.println("---------------");  **int** x = 3;  **int** y = 4;  // System.out.println((x++ > 4) & (y++ > 5)); // false & false  System.***out***.println((x++ > 4) && (y++ > 5)); // false && false  System.***out***.println("x:" + x);  System.***out***.println("y:" + y);  }  } |

## 三元运算符

### 三元运算符概述

A:格式

(关系表达式)?表达式1：表达式2；

如果条件为true，运算后的结果是表达式1；

如果条件为false，运算后的结果是表达式2；

B:示例：

获取两个数中大数。

int x=3,y=4,z;

z = (x>y)?x:y;//z变量存储的就是两个数的大数

### 代码案例二十

|  |
| --- |
| **package** com.igeek\_05;  /\*\*  \* **@ClassName**: OperatorDemo  \* **@Description**: 三元运算符  \* **@date** 2017年10月11日 下午7:28:17  \* Company www.igeekhome.com  \*  \* 三元运算符：  \*  \* 格式：  \* (关系表达式)?表达式1:表达式2;  \* 执行流程:  \* A:计算关系表达式的值，看是true还是false  \* B:如果是true,表达式1就是运算结果  \* 如果是false,表达式2就是运算结果  \*/  **public** **class** OperatorDemo {  **public** **static** **void** main(String[] args) {  **int** a = 10;  **int** b = 20;  **int** c = (a > b) ? a : b;  System.***out***.println("c:" + c);  }  } |

### 代码案例二十一(比较两个数是否相同)

|  |
| --- |
| **package** com.igeek\_05;  /\*\*  \* **@ClassName**: OperatorTest  \* **@Description**: 三元运算符的练习  \* **@date** 2017年10月11日 下午7:30:11  \* Company www.igeekhome.com  \*  \* 三元运算符的练习  \* 比较两个整数是否相同  \*/  **public** **class** OperatorTest {  **public** **static** **void** main(String[] args) {  // 定义两个int类型的变量  **int** a = 10;  **int** b = 20;  **boolean** flag = (a == b) ? **true** : **false**;  // boolean flag = (a == b);  System.***out***.println(flag);  }  } |

### 代码案例二十二(获取三个数中的最大值)

|  |
| --- |
| **package** com.igeek\_05;  /\*\*  \* **@ClassName**: OperatorTest2  \* **@Description**: 获取三个整数中的最大值  \* **@date** 2017年10月11日 下午7:31:48  \* Company www.igeekhome.com  \*  \* 获取三个整数中的最大值  \*/  **public** **class** OperatorTest2 {  **public** **static** **void** main(String[] args) {  // 定义三个int类型的变量  **int** a = 10;  **int** b = 30;  **int** c = 20;  // 先比较两个整数的大值  **int** temp = ((a > b) ? a : b);  **int** max = ((temp > c) ? temp : c);  System.***out***.println("max:" + max);  }  } |

## 重点和小结

1、使用(+,-,\*,/,%,++,--等)算数运算符

2、取余和除法的区别

3、字符和字符串参与加法运算操作需要注意的事项

4、++和--的作用，以及++，--在前或在后的区别

5、赋值运算符和扩展运算符

6、关系运算符，关系运算符的结果

7、逻辑运算符的基本使用及&与&&的区别

8、三元运算符

# 键盘录入

## 键盘录入的基本步骤

### 键盘录入数据概述

我们目前在写程序的时候，数据值都是固定的，但是实际开发中，数据值肯定是变化的，所以，把数据改进为键盘录入，提高程序的灵活性。

键盘录入数据的步骤:

A:导包(位置放到class定义的上面)

import java.util.Scanner;

B:创建对象

Scanner sc = new Scanner(System.in);

C:接收数据

int x = sc.nextInt();

### 代码案例二十三

|  |
| --- |
| **package** com.igeek\_01;  **import** java.util.Scanner;  /\*\*  \* **@ClassName**: ScannerDemo  \* **@Description**: 实现键盘录入数据  \* **@date** 2017年10月11日 下午7:34:38  \* Company www.igeekhome.com  \*  \* 为了提高程序的灵活性，我们就把数据改进为键盘录入。  \* 如何实现键盘录入呢?目前我们只能使用JDK提供的类Scanner。  \* 这个使用的步骤，目前大家记住就可以了。  \*  \* 使用步骤：  \* A:导包  \* import java.util.Scanner;  \* 类中的顺序：package > import > class  \* B:创建对象  \* Scanner sc = new Scanner(System.in);  \* C:接收数据  \* int i = sc.nextInt();  \*/  **public** **class** ScannerDemo {  **public** **static** **void** main(String[] args) {  //创建键盘录入数据的对象  Scanner sc = **new** Scanner(System.***in***);    //接收数据  System.***out***.println("请录入一个整数：");  **int** i = sc.nextInt();    //输出数据  System.***out***.println("i:"+i);  }  } |

## 键盘录入的练习

### 键盘录入两个数据并求和

键盘录入两个数据，并对这两个数据求和，输出其结果

键盘录入：

A:导包

B:创建对象

C:接收数据

### 代码案例二十四

|  |
| --- |
| **package** com.igeek\_02;  **import** java.util.Scanner;  /\*\*  \* **@ClassName**: ScannerTest  \* **@Description**: 键盘录入两个数据并求和  \* **@date** 2017年10月11日 下午7:37:48  \* Company www.igeekhome.com  \*  \*/  **public** **class** ScannerTest {  **public** **static** **void** main(String[] args) {  // 创建对象  Scanner sc = **new** Scanner(System.***in***);  // 接收数据  System.***out***.println("请输入第一个数据：");  **int** a = sc.nextInt();  System.***out***.println("请输入第二个数据：");  **int** b = sc.nextInt();  // 对数据进行求和  **int** sum = a + b;  System.***out***.println("sum:" + sum);  }  } |

### 键盘录入两个数据比较是否相等

键盘录入两个数据，比较这两个数据是否相等

### 代码案例二十五

|  |
| --- |
| **package** com.igeek\_02;  **import** java.util.Scanner;  /\*\*  \* **@ClassName**: ScannerTest2  \* **@Description**: 键盘录入两个数据，比较这两个数据是否相等  \* **@date** 2017年10月11日 下午7:39:40  \* Company www.igeekhome.com  \*  \*/  **public** **class** ScannerTest2 {  **public** **static** **void** main(String[] args) {  // 创建对象  Scanner sc = **new** Scanner(System.***in***);  // 接收数据  System.***out***.println("请输入第一个数据：");  **int** a = sc.nextInt();  System.***out***.println("请输入第二个数据：");  **int** b = sc.nextInt();  // 比较两个数据是否相等  // boolean flag = ((a == b) ? true : false);  **boolean** flag = (a == b);  System.***out***.println("flag:" + flag);  }  } |

### 键盘录入三个数据获取最大值

键盘录入三个数据，获取这三个数据中的最大值

### 代码案例二十六

|  |
| --- |
| **package** com.igeek\_02;  **import** java.util.Scanner;  /\*\*  \* **@ClassName**: ScannerTest3  \* **@Description**: 键盘录入三个数据，获取这三个数据中的最大值  \* **@date** 2017年10月11日 下午7:41:20  \* Company www.igeekhome.com  \*  \*/  **public** **class** ScannerTest3 {  **public** **static** **void** main(String[] args) {  // 创建对象  Scanner sc = **new** Scanner(System.***in***);  // 接收数据  System.***out***.println("请输入第一个数据：");  **int** a = sc.nextInt();  System.***out***.println("请输入第二个数据：");  **int** b = sc.nextInt();  System.***out***.println("请输入第三个数据：");  **int** c = sc.nextInt();  // 如何获取三个数据的最大值  **int** temp = (a > b ? a : b);  **int** max = (temp > c ? temp : c);  System.***out***.println("max:" + max);  }  } |

重点和总结

1、Java概述、helloworld案例

2、工具安装 、配置环境变量、注释、关键字

3、常量、变量、数据类型、标识符

4、Eclipse的安装、配置及使用

5、运算符：+,-,\*,/,%,++,--,&&,&,||,|

6、键盘录入