# 尚马教育 JAVA 基础课程

# Java语言基础

文档编号：A02

创建日期： 2017-04-12

最后修改日期：2020-04-29

版 本 号：V4.0

电子版文件名：尚马教育-第一阶段-2.Java语言基础专题课程.docx

**文档修改记录：**

|  |  |  |  |
| --- | --- | --- | --- |
| 更新日期 | 更新作者 | 更新说明 | 版本号 |
| 2017-07-30 | 张元林 | 初始版本 | V1.0 |
| 2018-08-01 | 王绍成 | Java基础版本更新 | V2.0 |
| 2019-08-09 | 徐丽莎 | Java基础版本更新 | V3.0 |
| 2020-04-20 | 刘蕾 | Java基础版本更新 | V4.0 |

**主讲人：**

**武汉-刘蕾**

目录

[尚马教育 JAVA 基础课程 1](#_Toc15928)

[Java语言基础 1](#_Toc14088)

[1. 变量 3](#_Toc20397)

[1.1. 内存存储数据 3](#_Toc31605)

[1.2. 创建变量语法 3](#_Toc23080)

[1.3. 变量命名规则 4](#_Toc3678)

[2. 数据类型 4](#_Toc23933)

[2.1. 划分数据类型的意义 4](#_Toc22218)

[2.2. Java数据类型 5](#_Toc30994)

[2.2.1. 基本数据类型 5](#_Toc24859)

[2.2.2. 引用数据类型 6](#_Toc4109)

[2.2.3. 案例代码 6](#_Toc4867)

[2.2.4. 练习 7](#_Toc19441)

[3. 数据类型转换 7](#_Toc18470)

[3.1. 基本数据类型转换 7](#_Toc22842)

[3.1.1. 自动数据类型转换（隐式转换） 8](#_Toc25812)

[3.1.2. 强制数据类型转换(显示转换) 8](#_Toc21941)

[3.1.3. 课堂练习 9](#_Toc6011)

[3.2. 引用数据类型转换 9](#_Toc26270)

[4. 引用类型 VS 基本数据类型 9](#_Toc1731)

[4.1. 数据类型内存存储 10](#_Toc23770)

[5. 运算符 10](#_Toc22898)

[5.1. 赋值运算符 10](#_Toc16319)

[5.1.1. 语法 11](#_Toc11915)

[5.1.2. 案例 11](#_Toc6975)

[5.2. 算术运算符 11](#_Toc3908)

[5.2.1. 语法 12](#_Toc5253)

[5.2.2. 案例 12](#_Toc29123)

[5.2.3. 扩展 13](#_Toc8246)

[5.3. 比较运算符 13](#_Toc23243)

[5.3.1. 案例 14](#_Toc20466)

[5.3.2. 扩展 14](#_Toc23995)

[5.4. 逻辑运算符 15](#_Toc6359)

[5.4.1. 语法 15](#_Toc30089)

[5.4.2. 案例 15](#_Toc17165)

[5.4.3. 扩展 16](#_Toc18087)

[5.5. 三元(三目)运算符 16](#_Toc29339)

[5.5.1. 语法 16](#_Toc6380)

[5.5.2. 案例 16](#_Toc31775)

[6. 作业 17](#_Toc4607)

## 标识符

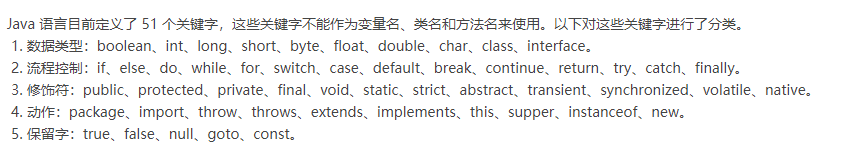
* **什么是标识符？**
  + 简单的理解为就是一个名称，可以是类名、变量、常量、数组、集合、方法等的名称。
  + Java中标识符可以由数字、字母【大小写都可以】、下划线【\_】和美元符号组成，但是不能以数字开头。



* **命名规则：**
  + 驼峰式命名规则： --> 跟骆驼的背一样，高低起伏的。
  + 小驼峰命名：首字母小写，第二个单词的首字母大写。 --> 一般用于变量、常量、数组、集合、方法等的命名。 例如：helloWorld.
  + 大驼峰命名：首字母大写，第二个单词的首字母大写。 --> 一般用于类名的命名。 例如：HelloWorld.
* **不合法的标识符案例：**

|  |
| --- |
|  |

* **不能使用关键字来命名标识符。 🡪  高亮显示的。**

****

## 变量与常量

### 常量

* **什么是常量？**
  + 常量：恒定不变的量，说白了就是一个固定的值。 在Java中，利用关键字final指示常量。

|  |
| --- |
| **public class** Constants {  **public static void** main(String[] args) {  *//π：3.14 是一个固定不变的量，称之为常量* **final float** PAI = 3.14f;  } } |

* + - 关键字final表示这个变量只能被赋值一次。一旦被赋值之后，就不能够再更改了。而且，习惯上，常量名使用全大写。
    - 在Java中，经常希望某个常量可以在一个类中多个地方都能够访问，通常将这些常量称为类常量。可以使用static final修饰。并且定义在类中。

|  |
| --- |
| **public class** Constants {  *//π：3.14 该常量可以在该类的任何地方访问* **static final float *PAI*** = 3.14f;   */\*\*  \* 程序的入口  \** ***@param args*** *\*/* **public static void** main(String[] args) {  System.***out***.println(***PAI***);  } } |

* + - 注意：如果希望在项目的任何地方访问，还可以使用public关键字来进行修饰。那么其他类中的方法也可以使用这个常量。
* **常量分为：**
  + 字面常量【比如：1、2、“张三”这些等等具体的值都是字面常量】。
  + 字符常量【通过final关键字修饰的常量】。
  + 注意：常量一旦被初始化将不能修改，只要能修改的就不是常量则为变量了。
* **字面常量又分为：** 
  + 整数型常量：1、2、3
  + 浮点型常量/实型常量：1.23、3.14
  + 布尔型常量/逻辑常量：true或者false   🡪 需注意：逻辑常量只有true或者false
  + 字符串常量：“你好”
  + 字符型常量：‘A’

### 变量

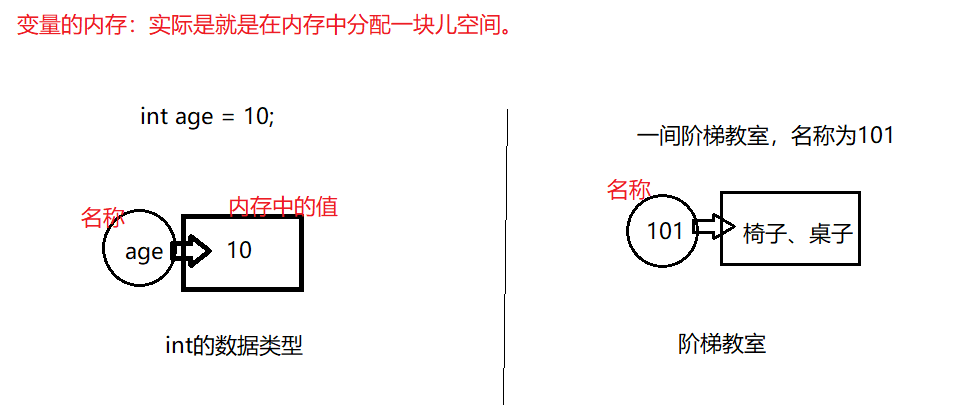
* **什么是变量？**
  + 变量：可以改变的量。
  + 在Java中变量就是数据存储空间的表示，不同数据存储在不同的内存地址空间里，是相互独立存在的。
  + 简单的认为，变量就是存储数据的一个基本单元。
* **变量分类：** 
  + 局部变量：声明在方法中。 🡪 需要自己手动赋值。
  + 全局变量：声明在类中。 🡪 是有默认值的。

### 变量的声明与使用

* **变量的声明/创建：**
  + 数据类型 变量名 [,变量名2]; 🡪 注： [ ]里面的内容代表可以省略
  + 数据类型 变量名;
  + 数据类型 变量名 , 变量名2;
* **变量的赋值：**
  + 变量名 = 具体的值;
* **声明和赋值同时进行：**
  + 数据类型 变量名1 = 值[,变量名2 = 值];
  + 数据类型 变量名1 = 值；
  + 数据类型 变量名1 = 值，变量名2 = 值；🡪 这里的变量1和变量2是同一个数据类型。
* **变量的使用：**
  + 直接使用变量的名称即可，但是需要注意的是局部变量在使用之前必须赋值。

### 变量在内存中的存储

* **变量的内存：**
  + 变量的内存实际上就是在内存中分配一块儿空间。
  + 思考:通过java程序存储一个学生的基本信息。
  + 电脑使用内存来记忆计算时所使用的数据
  + 数据各式各样，要先根据数据的需求（即类型）为它申请一块合适的空间



## 数据类型

* 目标：使用变量并结合数据类型定义学生的基本信息
  + 学生的姓名 张三
  + 学生的年龄 20
  + 学生的成绩 80.5
  + 学生的性别 男
  + 是否是一个好学生 是

### 划分数据类型的意义

* 数据有各式各样的，使用不同的变量来代表不同的数据，就需要不同的数据类型。
* 如果不把数据划分为不同的类型，那么就没有办法区分出数据之间的差别；
* 计算机可以根据不同的数据类型，把数据“合理”地存放到内存中；
* 计算机从内存中读取数据时，也可以根据其数据类型就能确认取到的数据的特征，从而正确地去处理；

### Java数据类型

|  |
| --- |
|  |

#### 基本数据类型

一般称为四类八种:

|  |
| --- |
|  |

* 整数默认类型：int
* 小数默认类型：double
* 计算机存储设备的最小信息单元叫“位（bit）”，又称之为“比特位”；
* 连续的8个位成为1个“字节（byte）”；
* 操作系统分配内存最少1个字节，即8 个位，而不是1个位；
* 长度越长，所表示的范围就越大；
* 每种具体类型都有不同的默认值；

|  |
| --- |
|  |

#### 引用数据类型

* String，类，接口，数组等都属于引用数据类型

#### 案例代码

因此，学生信息可以结合相对应的数据类型

|  |
| --- |
| //定义变量  String stuName = "吕子尚";//字符串不是基本数据类型  **int** stuAge = 20;//学生年龄  **double** stuScore = 80.5;//学生成绩  **char** stuGender = '男';//学生性别  **boolean** isGoodStudent = **true**;//是否是好学生  //使用变量  System.***out***.println(stuAge);  System.***out***.println(stuName);  System.***out***.println(stuScore);  System.***out***.println(isGoodStudent);  System.***out***.println(stuGender); |

使用其它数据类型定义变量：

|  |
| --- |
| **byte** by = 10;//-128-127  **short** sh = 200;  **int** num = 200;  **long** lon = 200L;  **float** f = 1.0F;  **double** dou = 2.0375;  **char** ch = 'a';  **char** ch1 = 97;//ASCII  **char** ch3 = '\u0021';  **boolean** flag = **false**;  **boolean** flag1 = **true**; |

#### 练习

|  |
| --- |
| 使用变量存储以下手机信息，并打印输出   * + **品牌（brand）：小米X**   + **重量（weight）：12.4**   + **电池容量（capacity）：4050**   + **价格（price）：6999**   + **厚度（thickness）：5.3** |

## 数据类型转换

### 基本数据类型转换

* 不同的基本数据类型之间可以进行转换
* 基本数据类型范围排序：

|  |
| --- |
|  |

#### 自动数据类型转换（隐式转换）

* 从表示范围小的类型转换为表示范围大的类型，可以直接转换，称为隐式转换

|  |
| --- |
| **byte** by = 10;  **int** num = by;//自动类型转换 表示范围小的可以直接转换为表示范围大的类型  **long** lon = num;  **double** dou = lon;  System.***out***.println(dou); |

#### 强制数据类型转换(显示转换)

* 从表示范围大的类型转换为表示范围小的类型，需要强制转换，称为显式转换
* (将高级别类型赋值给低级别，必须进行强制转换.)

|  |
| --- |
| **int** a = 200;  **byte** b = (**byte**)a;  System.***out***.println(b);  **double** c = 12.1234D;  **int** d = (**int**)c;  System.***out***.println(d);  **char** ch1 = 100;//d  **int** f = ch1;  System.***out***.println(f);  **char** ch2 = (**char**)f;  System.***out***.println(ch2);  **byte** b1 = 10;  **byte** b2 = 10;  **byte** b3 = (**byte**)(b1+b2);  System.***out***.println(b3); |

* 虽然类型之间可以进行强制的隐式转换，但是也需要有一定的前提；
* 数值类型和boolean类型之间就不能转换；强制也不可以；

|  |
| --- |
| **byte** by1 = 1;  **boolean** boo1 = **false**;  // "风马牛不相及"的boolean和数值类型，强制也不能转换；  by1 = (**boolean**)boo1; //编译无法成功(会报错) |

#### 课堂练习

* + 去年Apple笔记本所占的市场份额是20，今年增长的市场份额是9.8，求今年所占的份额
  + 某班第一次Java考试平均分81.29，第二次比第一次多2分，计算第二次考试平均分？

### 引用数据类型转换

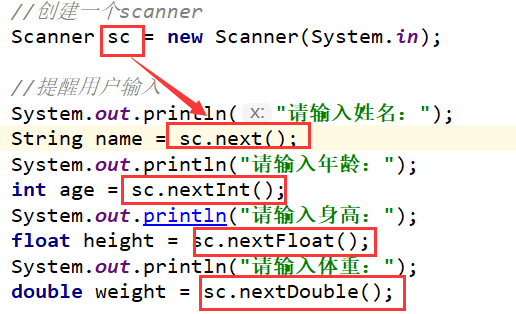
* 在后面的学习中会接触相关的知识。

## Scanner

* Scanner是Java提供的一个类，用来获取用户的输入。

### 语法

* 创建Scanner的基本语法：
  + Scanner s = new Scanner(System.in);
  + 导入java.util.Scanner包。
  + Scanner：简单的理解是一个扫描仪。
  + System.in:简单的理解为就是从控制台输入。标准输入！
* 通过Scanner类中的方法获取输入的值：



* + nextInt(): 获取一个int类型的值。
  + nextFloat()：获取一个float类型的值。
  + nextDouble():获取一个double类型的值。 …
  + next()和nextLine()：都是用来获取字符串的值。
* next()和nextLine()的区别？

|  |  |
| --- | --- |
| next() | 1、一定要读取到有效字符后才可以结束输入。  2、对输入有效字符之前遇到的空白，next() 方法会自动将其去掉。  3、只有输入有效字符后才将其后面输入的空白作为分隔符或者结束符。  4、next() 不能得到带有空格的字符串。 |
| nextLine() | 1、以Enter为结束符,也就是说 nextLine()方法返回的是输入回车之前的所有字符。  2、可以获得空白。 |

### 案例

* 案例分析：

|  |
| --- |
| **public** **static** **void** main(String[] args) {  Scanner input = **new** Scanner(System.***in***);//通过input变量获得控制台的数据  System.***out***.println("请录入学生的姓名");  //动态获得用户录入的字符串数据  String name = input.next();  System.***out***.println("请录入"+name+"的成绩");  **int** score = input.nextInt();//动态获得用户录入的数据  **byte** by = input.nextByte();  **long** lon = input.nextLong();  //Scanner无法获得char类型的数据 都当成字符串进行处理  String gender = input.nextLine();  System.***out***.println(name+"的成绩是:"+score);  } |

## 运算符

* 我们已经学习了数据类型，数据与数据之间是需要进行运算的，接下来我们来学习运算符的使用。
* 从功能角度分，Java中的运算符可以分为算术、关系、位、逻辑运算符四类；
* 运算符既可以对变量进行运算，也可以对常量进行运算，被运算的数据称作操作数

### 赋值运算符

* 将数据赋值给某个变量。
* = , +=, -=, \*=, /=, %=

#### 语法

|  |
| --- |
|  |
|  |

#### 案例

|  |
| --- |
| // 赋值运算符 = += -= \*= /= %=  **int** num = 100;//将100的数据赋值给num变量 =是一个赋值符号  num+=10;//等同于 num = num+10; 将num+10之后的结果再赋值给num变量  System.***out***.println(num);//110  num-=10;  System.***out***.println(num);//100  num\*=10;  System.***out***.println(num);//1000  num/=10;  System.***out***.println(num);//100  num%=10;  System.***out***.println(num);//0 |

### 算术运算符

* 变量与变量之间(数据与数据之间)进行加减乘除等相关的运算。

|  |
| --- |
|  |

#### 一元运算符

* 一元运算符也称之为**自增（++）自减（--）运算符**。这是一种特殊的运算符，在算术运算符中需要两个操作数来进行运算。但是自增自减是一个操作数。
* 观察以下代码，注意观察++a和a++的区别？

|  |
| --- |
| //++ 自增加1 -- 自减减1  **int** c = b++;  //++ 在变量名之后 先赋值 先b的值赋值给c b再自增加1  System.***out***.println(b);//4  System.***out***.println(c);//3  c = ++b;  //++ 在变量名之后 b++ b=5  System.***out***.println(b);//5  System.***out***.println(c);//5 |
| **int** x = 10;  x = x++;  //变量缓存机制  // int temp = x;//10  // x++;  // x = temp;  System.***out***.println(x);//10 |

* a++和++a的区别？
  + 前缀自增自减法（++a/--a）：先计算，再使用！
  + 后缀自增自减法（a++/a--）：先使用，再计算！

#### 二元运算符

* 二元运算符的作用和数学中的作用是一样的。

##### 语法

|  |
| --- |
| **int** a = 20;  **int** b = 3;  System.***out***.println(a+b);//a与b进行算数运算  System.***out***.println(a-b);  System.***out***.println(a\*b);  System.***out***.println(a/b);  System.***out***.println(a%b); |

##### 案例

* 某学员3门课成绩如右表所示，编程实现：

1、Java课和SQL课的分数之差

2、3门课的平均分



##### 扩展

* + 拼接符号

|  |
| --- |
| //拼接 +  System.***out***.println(10+100+"hello");  System.***out***.println("hello"+(10+100));  System.***out***.println("hello"+",world"); |

### 比较运算符

* 用来运算两个操作数(变量)的大小关系，返回值是true或false；

|  |
| --- |
|  |

#### 案例

|  |
| --- |
| //比较运算符 > < >= <= != ==  // 结果: boolean true false  **int** num1 = 10;  **int** num2 = 5;  System.***out***.println(num1>num2);  System.***out***.println(num1<num2);  System.***out***.println(num1>=num2);  System.***out***.println(num1<=num2);  System.***out***.println(num1!=num2);  System.***out***.println(num1==num2); |

#### 扩展

使用==进行比较的时候，比的是什么？

|  |
| --- |
| // == 相等于  //如果是基本数据类型而言 比较的是数据  //如果是引用数据类型 比较的是对象在内存里面地址值  //判断两个数据是否相等  **int** num1 = 100;  **int** num2 = 100;  System.***out***.println(num1==num2);  //如果== 比较基本类型数据 比较的是数据  String str1 = "hello";  String str2 = "hello";  String str3 = **new** String("hello");  System.***out***.println(str2);  System.***out***.println(str3);  System.***out***.println(str1==str2);  System.***out***.println(str2==str3);//false  //比较字符串内容是否一致  System.***out***.println(str2.equals(str3)); |
|  |

### 逻辑运算符

#### 语法

* 逻辑运算符用于连接布尔型表达式。在Java中不可以写成3<x<6，应该写成x>3 & x<6 。
* 表达式：用运算符连接的式子。布尔型表达式就是运算结果为boolean类型的表达式。

|  |
| --- |
|  |

#### 案例

|  |
| --- |
| System.***out***.println((5>4)&&(5<3));  System.***out***.println((5>4)&(5<3));  System.***out***.println((5<3)&&(5>4));//短路与  System.***out***.println((5<3)&(5>4)); |

#### 扩展

* “&”和“&&”的区别：
  + 单&时，左边无论真假，右边都进行运算；
  + 双&时，如果左边为真，右边参与运算，如果左边为假，那么右边不参与运算。
  + “|”和“||”的区别同理，双或时，左边为真，右边不参与运算
  + &&,||和&以及|的区别：结果相同，但是有短路效果，一般建议使用&&,||

### 三元(三目)/条件运算符

#### 语法

* 返回最终结果 = (关系表达式) ? 结果1 ：结果2;
  + 如果关系表达式成立(true) ，则运算后为结果1。
  + 如果关系表达式不成立(false) ，则运算后为结果2。
* 要求：
  + 关系表达式结果为boolean类型。
  + 一定会返回最终结果。
  + 结果1与结果2、最终结果的数据类型必须一致。
  + ?和:都是英文的。

#### 案例

|  |
| --- |
| // (表达式)?第一个表达式:第二个 表达式结果;  String result = (5>4)?"hello":"world";  System.***out***.println(result);  **int** num = (**int**)((1>0)?1:2.0);  System.***out***.println(num); |
| // 获取两个数之间的最大数  **int** a=3,b=4,z;  z=(a>b)?a:b; //z变量存储a与b之间的最大值 |

### instanceof运算符

* 该运算符用于操作对象实例，检查该对象是否是一个特定类型（类的类型或者接口的类型）。
* Instanceof运算符的使用语法：

|  |
| --- |
| 对象 Instanceof 类的类型或者接口的类型   * 如果运算符的左侧变量的类型与Instanceof右侧的类型一致，则会直接返回true，否则为false。 |

* 案例：

|  |
| --- |
| **public static void** main(String[] args) {  *//姓名* String name = **"tom"**;  *//判断name的类型是否是String？* **boolean** b = name **instanceof** String;  *//由于name是String，所以返回true* System.***out***.println(b); } |

### 位运算符（了解）

* 处理整数类型时，可以直接对组成整数型值的各个位完成操作。
* Java定义了位运算符应用于整数（int）、长整型（long）、短整型（short）、字符型（char）和字节型（byte）。
* 位运算符作用在所有的位上，并且按照位来运算。
* 常用的位运算符有：

|  |  |
| --- | --- |
| 操作符 | 描述 |
| 按位与（&） | 参与运算的两个值，如果两个相应位都是1，则该位的结果为1，否则为0。 |
| 按位或（|） | 参与运算的两个值，如果两个相应位都是0，则该位的结果为0，否则为1。 |
| 按位非（~） | 按位取反，对数据的每个二进制数据进行取反操作。将1变成0，将0变成1。 |
| 按位异或（^） | 参与运算的两个值，如果两个相应位值相同，则结果为0,否则为1. |
| 按位左移动运算符（<<） | 运算符的各个二进制位全部左移若干位，由”<<”右边的数指定移动的位数，高位丢弃，低位补0.【数字缩小】 |
| 按位右移动运算符（>>） | 把”>>”左边的运算符的各个二进制全部右移若干位，由”>>”右边的数指定移动的位数。【数字扩大】 |

#### 案例

|  |
| --- |
| */\*\*  \** ***@Author：liulei*** *\** ***@Version：1.0*** *\** ***@Date：2021/5/6-16:40*** *\** ***@Since:jdk1.8*** *\** ***@Description:todo*** *\*/* **public class** Demo1 {  **public static void** main(String[] args) {  **int** a = 5; *//0000 0101* **int** b = 6; *//0000 0110   //获取对应的二进制数字* System.***out***.println(**"a的二进制："**+Integer.*toBinaryString*(a));  System.***out***.println(**"b的二进制："**+Integer.*toBinaryString*(b));   *//按为取反 1111 1010 : 先减-1 再取反：1111 1001 --> 0000 0110 --> -6* System.***out***.println(~a);  */\*  0101  0110  ----  0100 --> 4  \*/* System.***out***.println(a&b);  */\*  0101  0110  ----  0111 --> 7  \*/* System.***out***.println(a|b);  */\*  0101  0110  ----  0011 --> 3  \*/* System.***out***.println(a^b);  */\*  0101  0010[1] --> 2  \*/* System.***out***.println(5 >> 1);  */\*  0101  0101[0] --> 8 + 2 = 10  \*/* System.***out***.println(5 << 1);  } } |

## 运算符优先级：

