1.1. Java基础语法

1.1.1. 上节回顾

>

1.1.2. 教学目标

> 1.掌握什么是标识符 2.标识符的命名规范 3.掌握关键字和保留字 4.掌握变量声明、初始化、赋值 5.掌握八种基本数据类型 6.掌握运算符

1.1.3. 标识符、关键字、保留字

1.1.3.1. 什么是标识符

Java对包、类、方法、参数和变量等要素命名时使用的字符序列称为标识符。也就是名字。

1.1.3.2. 标识符的命名规则

1. 由Unicode字符（含英、中、日、俄等）、数字、下划线（_）和美元符号（$ 音：dollar）组成。
2. 不能以数字开头。
3. 区分大小写。
4. 长度无限制。（一般不要超过15个字符）
5. 不能是单独的保留字和关键字。

1.1.3.3. 标识符命名习惯

1. 见名知意，使用英文单词，如：name、age等，不要用a、b、c，尽量不要用拼音。
2. 驼峰式命名法：变量名，方法名，参数名首字母小写，如果多个单词组成，第一个单词首字母小写，剩下的单词首字母大写，如：firstName
3. 帕斯卡命名法：类名单词首字母大写，多个单词组成每个单词首字母大写，如：Person

1.1.3.4. 关键字、保留字

关键字：Java中有一些赋予特定的含义，有专门用途的字符串称为关键字（keyword），如：public class等。全部为小写，在对应章节学习。

保留字：没有定义用途，但保留备用。**goto、const**

1.1.4. 常量、变量

> 官网教程：https://docs.oracle.com/javase/tutorial/java/nutsandbolts/variables.html

1.1.4.1. 什么常量、变量

程序是用来处理数据的，当我们需要处理某些数据时，需要在内存中申请空间，以便存放数据。如以下这行代码处理的数据为1，代码运行到此处时会**在内存中分配一个区域，这个区域中放的数据是1**：

```
System.out.println(1);
```

存放1的这个内存空间在申请后，**只能使用一次，不可修改**，而且其中存放的数据就是字面上看到1，这个内存空间我们称为`字面常量`,即使随后执行以下代码，也是重新申请了一个内存空间，刚才存放1的内存空间已不能重复使用：

```
System.out.println(2);
```

如果我们想要重复使用这个内存空间，可以给这个内存空间起个名称(标识符)，随后通过名称可多次使用这个内存区域，如：

```
int age;
age=1;
System.out.println(age);
age=2;
System.out.println(age);
```

> `int age;`：程序运行到这个语句的时候，申请了一个内存空间，而且命名为age；
>
> `age=1;`：在age对应的这个内存空间中存入1。
>
> `println(age)`：打印age对应的这个内存空间中的数据。
>
> `age=2;`：在age对应的这个内存空间中存入2。
>
> `println(age)`：打印age对应的这个内存空间中的数据。

age这个内存空间**可以多次读取，而且其中可再次存放的他数据**，这个的内存空间我们称为`变量`，即age变量，如果只是想多次使用某个内存空间中存放的数据，而不希望修改它里面的值，则可以在变量声明语句前加final：

```
final int sex=1;//sex只能赋值一次
System.out.println(sex);
System.out.println(sex);
```

sex对应的这个内存空间中的值只能是1，**可多次读取，不可修改**，我们称sex对应的内存空间为`常量`，即sex常量。

总结：

1. 字面常量：申请的空间只可读一次，放入的数据就是字面上的数据
2. 变量：申请的空间可以多次读写
3. 常量：申请的空间可以多次读，但是只能写一次

1.1.4.2. 变量的声明、初始化和赋值

1. **变量声明**

申请一个内存空间，给这个内存空间起个名称，语法如下：

```
数据类型  变量名;
```

如：

```
int age;
```

2. **赋值**

给变量中放入数据的过程称为`赋值`,语法如下：

```
变量名=数据;
```

如：

```
age=1;
```

3. **初始化**

第一次赋值称为`初始化`。可以将变量声明和初始化放在一个语句中，,语法如下：

```
数据类型  变量名=数据;
```

如：

```
int age=1;
```

随堂练习：

练习1：银行账户存储1000元，年利率是0.05，计算一年后账户余额?

要求：使用变量保存数据

练习2：使用变量存储数据，实现个人简历信息的输出(String str="");

姓名，年龄，工作年限，技术方向、兴趣爱好

要求：使用变量保存数据

1.1.4.3. 变量的分类

1. 按变量的数据类型划分： **基本类型变量**：可放入**基本类型**的值。(原始数据类型) 引用类型变量：持有引用值。(某个对象的地址，不是该对象本身) String 字符串，OOP中细说
2. 按被声明的位置划分： **局部变量**：方法或语句块内部定义的变量

   实例变量：又叫成员变量或者字段，方法外部、类的内部定义的变量，不加static，OOP中细说

   类变量：又叫静态成员变量或者静态字段，方法外部、类的内部定义的变量，加static，OOP中细说

   方法参数：在方法声明中定义的变量，方法中细说

```
public class Variable {

    static  int sex;
    /**
     * 变量作用域
     *
     * @param args
     */
    public static void main(String[] args) {
        {
            int age;
            System.out.println(age);
            System.out.println(sex);
        }
        {
            System.out.println(age);
        }
    }
}
```

> 局部变量和成员变量的区别：
>
> 1. 作用域（有效范围）不同： 局部变量的**作用域就是它所在的方法或语句块**，如：age是个局部变量，作用域只能是第一个化括号内 成员变量的作用域是整个类体。如：sex是个成员变量，作用域是类中的所有代码
> 2. 是否拥有默认值：
>
>   成员变量有默认值，如：sex有默认值，不需要初始化就可用 局部变量没有默认值，**所以使用前必须初始化**，如：age没有默认值，必须初始化才能用，否则编译不了

1.1.5. 数据类型

> 官网教程：https://docs.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.html

声明变量时，需要告诉虚拟机我们使用这个空间放什么类型的数据，虚拟机好确定空间大小及随后判断对该空间可做的操作。

Java数据类型可分为两大类:**基本数据类型**和**引用数据类型**

- 基本数据类型：byte、short、int、long、float（单精度）、double（双精度）、char、boolean

- 引用数据类型：类、接口、数组、枚举（后面讲解）、注解

1.1.5.1. 基本数据类型

四类八种，由于char类型的变量中存放的字符是通过在字符编码表中的位置表示的，所以char也可以归结为整数型：

> 整数型---- byte（字节）, short（短整型）, int（整型）, long（长整型）
>
> 浮点型---- float（单精度浮点型）, double（双精度浮点型）
>
> 字符型---- char
>
> 布尔型---- boolean

1.1.5.2. 基本数据类型总览

Java中所有的基本数据类型都有固定的存储范围和所占内存空间的大小，而不受具体操作系统的影响，以保证Java程序的可移植性

类型	长度	范围	默认值	大致上限
byte	1个字节	-128 到 127	0	百
short	2个字节	-32768到 32767	0	万
int	4个字节	-2147483648到2147483647	0	十亿
long	8个字节	-9223372036854775808 to 9223372036854775807	0L	N多
float	4个字节		0.0F	
double	8个字节		0.0D	
char	2个字节	0-65535	'\u0000'	
boolean	不确定	true、false	false	

> '\u0000'：表示的是空字符(NUL)，不是空格(\u0020，十进制32)，打印出来是一个小方块，没什么意义。ASCII 里的所有控制字符(如：上箭头\u0018，十进制24)打印出来都是一个小方块。

1.1.5.3. 整数型

Java语言的字面整型数值默认为int型，如：int i = 3;要声明long型可以加'l'或' L'，如：long l = 3L;

```
public static void main(String[] args) {
    //1个字节 ，范围：有符号整型， -128 到 127，
    byte maxByte = +127;
    byte minByte = -128;

    // 2个字节 ，范围： 有符号整型，-32768到 32767，默认值：0
    short maxShort = +32767;
    short minShort = -32768;


    //4个字节 ，范围： 有符号整型，-2147483648 to 2147483647，默认值：0
    int maxInt = +2147483647;
    int minInt = -2147483648;

    //8个字节，范围： 有符号整型，-9223372036854775808 to 9223372036854775807，默认值
0L
    long maxLong = +9223372036854775807L;
    long minLong = -9223372036854775808L;


    //所有整型值可以通过以下方式表示：
    //数字26，十进制
    int decVal = 26;
    //数字26，十六进制
    int hexVal = 0x1a;
    //数字26，二进制,Java7以上
    int binVal = 0b11010;

    //数值默认是int型，可以加l或者L表示long型，建议用L，不容器混淆
    int intVal = 26;
    long longVal1 = 26L;
    long longVal2 = 0b11010L;

    //在数字文字中使用下划线字符
    //在Java7及更高版本中，下划线字符（_）可以出现在数字之间的任何位置
    long creditCardNumber = 1234_5678_9012_3456L;
    long socialSecurityNumber = 999_99_9999L;
    float pi = 3.14_15F;
    long hexBytes = 0xFF_EC_DE_5E;
    long hexWords = 0xCAFE_BABE;
    byte nybbles = 0b0010_0101;
    long bytes = 0b11010010_01101001_10010100_10010010;

  }
```

> 不能在以下地方放置下划线：
>
> - 在数字的开头或结尾
> - 小数点前后
> - F或L后缀之前
> - 在预期有一串数字的位置

1.1.5.4. 浮点型

浮点数也就是我们常说的带小数点的数，Java支持两种浮点数：

- float: 单精度浮点数（精度7,8位） 4个字节
- double:双精度浮点数 (精度15,16位) 8个字节

  Java浮点型字面常量**默认为double型**,也可以在数字后面加d或D明确声明，如要声明一个字面常量为float型，则需在数字后面加f或F，

```
public static void main(String[] args) {
    //float: 单精度浮点数（精度7,8位） 4个字节
    float floatValue=1.1;

    //双精度浮点数 (精度15,16位)  8个字节
    double doubleValue=1.1;

    //科学计算法表示
    double doubleEValue=3.14e2;

    //二进制不能准确的表示1/10等分数，所以部分浮点数只能无限趋近
    System.out.println(floatValue==doubleValue);//false
}
```

> 1. 浮点数存在精度损失，原因在于Java的数据存储采用的都是二进制形式，二进制不能准确的表示1/10等分数，只能无限趋近。如果需要进行精确数字计算，需要使用BigDecimal类。
> 2. 避免比较中使用浮点数

1.1.5.5. 字符型

`char`和`String`可以包含任何Unicode（UTF-16，\u0000 (0)---\uffff(65535)）字符，使用char可以使用单引号(')括起来，String使用双引号("")起来。char存储的实际上是字符在对应字符编码中的编码，所以可以使用整形值直接赋值,或者使用Unicode转义序列表示。

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

    char charValue1 = '中';

    char charInt1=20013;//使用的是'中'这个字在unicode字符集中的序号，不能超过65535

    char  charU1 = '\u4e2d';;//Unicode转义，unicode字符的序号使用16进制，相当于'中'

    char charValue = 'a';

    char charInt=97;//相当于'a'

    char  charU = '\u0061';

    System.out.print("\n");
  }
```

> - 对于部分特殊字符，需要通过转义字符输出:`\t`（制表符），`\n`（换行），`\r`（回车），`\"`（双引号），`\'`（单引号）和`\\`（反斜杠）

> **字符集与字符编码：**
>
> ASCII (美国信息交换标准代码）：到目前为止共定义了128个字符
>
> GB2312(国标2312)：基本集共收入汉字6763个和非汉字图形字符682个
>
> GBK(汉字编码扩展规范)：共收录了21003个汉字，兼容GB2312
>
> Unicode(统一码): 2020年推出的Unicode 13.0.0已经包含 137,374 个字符。Unicode只是收集了一组字符，并且给这些字符分配了**序号(Unicode编码)**，至于字符在计算机中通过什么样的二进制串表示，是由**字符编码**来确定的。UTF-16、UTF-32的二进制序列用的就是Unicode的序号，但是UTF-8的编码不同于**字符的序号**。如：中的序号是20013，UTF-16中的编码是0x4E0x2D，UTF-32中的编码是0x00 0x00 0x4E 0x2D，转为10进制都是20013，但是UTF-8中的编码是0xE40xB80xAD
>
> UTF-8：变长码，长度可能是1、2、3、4，绝大多数的汉字是用三个字节的，但是部分特殊汉字使用的是4个字节(主要是一些异体字)。
>
> **异体字：**
>
> 异体字是一个字的正体之外的写法，字音和字义相同而字形不同的一组字，现代汉语中基本已经不用。如：克(异体)=充，學(异体)=學(繁体)=学，搜狗输入法是输入不了异体字的。

1.1.5.6. 布尔类型

boolean类型适于逻辑运算，一般用于程序流程控制。boolean类型数据只允许取值true或false，不可以用0或非0的整数替代true和false。

boolean类型的长度在语言规范中没有给出精确的定义，《Java虚拟机规范》单变量4个字节，和boolean数组1个字节的定义，具体还要看虚拟机实现是否按照规范来，所以1个字节、4个字节都是有可能的。

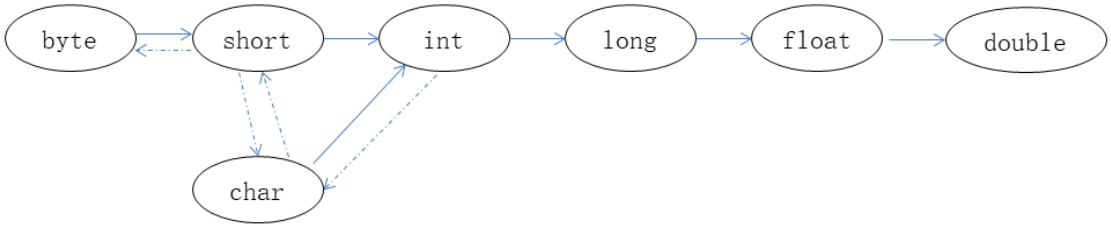
```
public static void main(String[] args) {
    boolean booleanFalse = false;
    boolean booleanTrue = true;
}
```

1.1.5.7. 基本数据类型的转换

boolean 类型不能转换成任何其它基本数据类型。

1. 自动类型转换

   数据类型取值范围小的可以直接赋值给取值范围大的，JVM自动完成数据类型转换，也就是byte可以赋值给short，short可以直接赋值给int等



> - int与byte、short、char**可以直接相互赋值**，但是不能超过彼此的表示范围，如：'耀'的整形是32768，则存放'耀'的变量是不能赋值给short的
> - 如果数值型混合运算，则**向其中的最高范围转换。**float、int都是4个字节，但是由于float的整数部分采用**指数**表示，所以可表示的整数范围是超过int的，甚至long型的值可以直接赋值给float

```
public static void main(String[] args) {
    short c1=32767;

    int i1=c1; //自动转换

    byte byteValue=1;

    short shortValue=1;

    char charValue='耀';

    int intValue=1;

    float floatValue=1.1F;

    //浮点型
    System.out.println(byteValue+shortValue+charValue+intValue+floatValue);

    System.out.println(shortValue+shortValue);//int型

    char c=2+'2';  //自动转

    short s1=1+1; //自动转，编译器可以确定赋值符右侧的值其实是个常量

    int a=1;

    //不可自动转，编译器无法预测变量a的值(只通过当前行预测)，可能超出short的范围
    short s2=a+1;
}
```

2. 强制类型转换

   数据类型取值范围大的转为取值范围小的. 需要进行强制转换，也就是（要转换的数据类型）。

```
long l = 100L;
int i = (int)l;
```

> 但有可能造成精度降低或数据溢出，使用时要小心。

随堂练习：

1. 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。
2. 去年Java所占市场份额是20，今年增长的市场份额是9.8，求今年所占份额?

```
int beforeyear=20;
int year=(int)(beforeyear+9.8);
```

1.1.6. 运算符和表达式

> 官网教程：https://docs.oracle.com/javase/tutorial/java/nutsandbolts/operators.html

1.1.6.1. 运算符的分类

1. 算术运算符：`+ 、 - 、 * 、 / 、 % 、 ++ 、 --`
2. 赋值运算符：`= 、 += 、 -= 、 *= 、 /= 、 %=`
3. 关系运算符：`> 、 < 、 >= 、 <= 、 == 、 !=`
4. 逻辑运算符：`! 、 && 、 ||`
5. 位运算符：`& 、 | 、 ^ 、 ~ 、 >> 、 << 、 >>>(了解)`
6. 字符串连接运算符：`+`
7. 三目运算符 `?:`

1.1.6.2. 算数运算符

描述	运算符	运算符	描述
加	+	**%**	求余、取模
减	-	**++**	自增
乘	*	**--**	自减
除	**/**	+、-	取正没什么意义、取负

随堂练习：

1. 编写以下代码以熟悉算术运算符

```
public static void main(String[] args) {
    System.out.println("***************** + *****************");
    // 结果是：3
    System.out.println(" 1 + 2 = " + (1 + 2));


    System.out.println("***************** - *****************");

    // 结果是：2
    System.out.println(" 3 - 1 = " + (3 - 1));


    System.out.println("***************** * *****************");

    // 结果是：4
    System.out.println(" 2 * 2 = " + 2 * 2);
```

```
        System.out.println("***************** / *****************");

        // 结果是：2
        System.out.println(" 4 / 2 = " + 4 / 2);

        // 结果是：1
        System.out.println(" 3 / 2 = " + 3 / 2);

        // 结果是：1.5
        System.out.println(" 3 / 2 = " + 3 / 2.0);


        System.out.println("***************** % *****************");
        // 结果是：3
        System.out.println(" 10 % 7 = " + 10 % 7);

        System.out.println("***************** - *****************");
        int original = -1;
        int result = -original;
        // 结果是：1
        System.out.println(" - " + original + "=" + result);

        System.out.println("***************** -- *****************");
        original=3;
        // 先用后减，结果是：3，result中是2
        System.out.println(" original(3) -- =" + original--);

        System.out.println(" original = " + original);

        // 先用后减，结果是：1，original中是21
        System.out.println(" -- original(2)=" + --original);

        System.out.println("***************** ++ *****************");
        original=3;
        // 先用后减，结果是：3，result中是4
        System.out.println(" original(3) ++ =" + original++);

        System.out.println(" original = " + original);

        System.out.println(" ++ original(4)  = " + ++original);

        byte byteValue = 127;

        //自增会在取值范围内打圈
        System.out.println(" ++byteValue = "+(++byteValue));
    }
```

1.1.6.3. 赋值运算符

运算符	描述	运算符	描述
=	赋值 比如:int i=10;	/=	除等于
+=	加等于	%=	模等于
-=	减等于		
*=	乘等于		

随堂练习:

1. 编写以下代码以熟悉赋值运算符

```
public static void main(String[] args) {
    int i = 34;
    //i = i + 45;
    i += 45;
    //i = i -45;
    i -= 45;
    i *= 45;
    i /= 45;
    i %= 45;
    System.out.println("i= " + i);
}
```

1.1.6.4. 关系运算符

运算符	描述	运算符	描述
==	相等于	<=	小于等于
!=	不等于	>=	大于等于
<	小于		
>	大于		

随堂练习:

1. 编写以下代码以熟悉关系运算符

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

    int value1 = 1;
    int value2 = 2;

    System.out.println("value1 == value2: " + (value1 == value2));
    System.out.println("value1 != value2: " + (value1 != value2));
    System.out.println("value1 > value2: " + (value1 > value2));
    System.out.println("value1 < value2: " + (value1 < value2));
    System.out.println("value1 <= value2: " + (value1 <= value2));
    System.out.println("value1 >= value2: " + (value1 >= value2));
}
```

1.1.6.5. 逻辑运算符

逻辑运算符用于对boolean型数据进行运算，运算结果总是boolean型

运算符	描述	运算符	描述
&&	短路与	!	逻辑非
||	短路或		

> `&&`、`&&`表现出“短路”行为，这意味着仅在需要时才计算第二个操作数

随堂练习：

1. 编写以下代码以熟悉逻辑运算符

```
public static void main(String[] args) {
    int value1 = 1;
    int value2 = 2;

    System.out.println("(value1 == 1) && (value2 == 2): " +
                       ((value1 == 1) && (value2 == 2)));
    System.out.println("(value1 == 1) || (value2 == 1): " +
                        ((value1 == 1) || (value2 == 1)));
    System.out.println("!(value2 == 1): " + (!(value2 == 1)));
}
```

1.1.6.6. 位运算符(了解)

异或、符号位

运算符	描述	运算符	描述
&	按位与	>>	右移
|	按位或	<<	左移
^	按位异或	>>>	无符号右移
~	按位取反		

> - &是按位与，和&&没有关系
> - ^两个值不同结果为1，相同结果为0

随堂练习：

1. 编写以下代码以了解位运算符

```
public static void main(String[] args) {
    byte bitmask = 0b1111101;
    System.out.println(bitmask);
    System.out.println(~bitmask);//按位取反
}
```

1.1.6.7. 字符串连接符

字符串连接符用于连接多个字符串，也可以连接字符串与其它变量及常量，这时会将其它变量及常量的值转为字符串。

```
public static void main(String[] args) {
    System.out.println("Hello"+"World!");
    System.out.println("Hello = "+ 5);// 字符串和基本数据类型变量 常量连接以后 都变成了字符串
    System.out.println("Hello = "+ (5+8);
    System.out.println(5+"");
}
```

1.1.6.8. 三目运算符

唯一一个需要三个操作数据的运算符，相当于简单的if-else语句，比if-else语句更加紧凑

```
expr ? expr1 : expr2
```

> expr为boolean类型表达式，先计算expr的值，若为true，整个三目运算的结果为表达式expr1的值，否则整个运算结果为表达式expr2的值。expr1、expr2必须保持数据类型属于一个系列，否则没法赋值

随堂练习：

1. 编写以下代码熟悉三目运算符

```
public static void main(String[] args) {
    int score = 99;
    //boolean falg = score>80;
    String str = score>80? "非常优秀" : "优秀";
    char c = '男';
    int i = c=='男'? 5 : (int)(4.0);

    System.out.println(i);
    // 需求：大于90输出"非常优秀“，大于等于60”输出及格“，否则小于60输出”下个班见“
    String str2 = score>90?"非常优秀":score>=60?"及格":"下个班见";
    System.out.println(str2);

    //数据类型最好保持一致，否则无法将结果赋值给变量
    System.out.println(1>2?"1>2":1);
}
```

1.1.6.9. 运算符的优先级

当运算符混合使用时，计算的先后顺序由运算符的优先级确定。除了赋值运算符之外的所有二元运算符都是从左到右计算的; 赋值运算符从右到左进行计算。

运算符	结合性
后缀	`expr++` `expr--`
一元	`++expr` `--expr` `+expr` `-expr` `~` `!`
乘除取余	`*` `/` `%`
加减	`+` `-`
位移	`<<` `>>` `>>>`
关系	`<` `>` `<=` `>=` `instanceof`
等于	`==` `!=`
按位与	`&`
按位异或	`|`
按位或	`^`
逻辑AND	`&&`
逻辑或	`||`
三元	`?` `:`
赋值	`=` `+=` `-=` `*=` `/=` `%=`

> instanceof：类型判断运算符，判断引用型变量指向的对象的类型，OOP部分学习，如：
>
> ```
> System.out.println( "1" instanceof  String);
> ```

1.1.6.10. 表达式、语句、语句块

一： 表达式

根据语法规则将操作数(变量、常量)，运算符和方法调用组合在一起，形成表达式，表达式返回一个结果值。如**以下代码的粗体部分**：

---

int **age= 0** ; //赋值表达式

System.out.println("---------- ") ;//方法调用表达式

int **result = 1 + 2** ; //算术表达式

String name= **new String()** ;//对象创建达单式

> - 表达式返回的值的数据类型取决于表达式中使用的元素 ，如：表达式`age= 0`返回一个`int`，因为赋值运算符返回与其左侧操作数相同的数据类型的值;
> - 在编写复合表达式时，可以通过括号优先计算子表达式，如：`(x +y) / 100`，而且平时写代码时也应该在复合表达式中使用括号进行子表达式的分组，以使代码更易于阅读和维护，如：x +(y / 100)

二： 语句

语句是一个完成的执行单元，通过以分号结束表达式，形成表达式语句，如：

---

int age= 0 ; //赋值语句

aValue ++; //自增语句

System.out.println("---------- ") ;//方法调用语句

String name= new String() ;//对象创建语句

> **除了表达式语句之外，还有另外两种语句：声明语句和流程控制语句。**声明语句如：
>
> ```
> int age;
> ```

三：语句块

可以通过花括号将0条或者多条语句括起来形成一个语句块，语句块可以用在任何单条语句可以出现的地方，如：

```
{
    int age;
    System.out.println(age);
    System.out.println(sex);
}
{
    System.out.println(age);
}
```

1.1.7. Scanner的简单使用

目前为止，数据都是我们在代码中直接写入的，现代程序基本上很少是这种模式的，数据一般来源于键盘、硬盘、摄像机等设备，为了更好的演示知识点，此处介绍命令行模式下从键盘输入数据到Java程序的方法：

```
//Scanner的使用
//1 先导入Scanner  import java.util.Scanner;
//2 创建对象    Scanner input=new Scanner(System.in);
//3 接收键盘录入的数据

import java.util.Scanner;
public class Demo11{
    public static void main(String[] args){
        Scanner input=new Scanner(System.in);
        //3.1接收字符串类型数据 next(),遇到空格截断 nextLine(); 遇到换行结束
        //String s=input.next();scanner.useDelimiter("\n");
        String s=input.nextLine();
        System.out.println(s);

        //3.2接收整型数据 nextInt();
        int num=input.nextInt();
        System.out.println(num);
        //3.3接收浮点数
        double d=input.nextDouble();
        System.out.println(d);

        //Math.random():返回带正号的 double 值，该值大于等于 0.0 且小于 1.0。
        //生成0到4的随机整数
        System.out.println((int)(Math.random()*5));
    }

}
```

> - **nextLine与nextInt不要混合使用**，如：先nextInt，然后再nextLine，由于所有多次输入的数据都在缓冲区中，next*只是通过正则获取数据，int的正则中只用了数字，没有用\n结尾，会导致buffer的position比实际小1，从而下一个nextLine返回空行。如果要混合使用，nextInt后调用一个空的nextLine
> - 特殊代码记忆就好，如：new Scanner(System.in);这行代码要解释清楚需要太多的基本知识点，后继学习OOP和IO后即可理解。

随堂练习：

1. 从控制台输入学员3门课程成绩(html，java，sql)，编写程序实现

（1）Java课和SQL课的分数之差

（2）3门课的平均分

```
import java.util.Scanner;  //导入Scanner类
public class ScoreStat {
     public static void main(String[] args) {
         //创建对象
        Scanner input = new Scanner(System.in);
        System.out.print("htmls的成绩是：");
        int html = input.nextInt();      //html分数
        int java = input.nextInt();
         int sql = input.nextInt();
        int diffen;        //分数差
        double avg;        //平均分
         //省略输出成绩单代码.....
        diffen = java - sql; //计算Java课和SQL课的成绩差
        System.out.println("Java和SQL的成绩差：" + diffen);
        avg = (html + java + sql) / 3;  //计算平均分
        System.out.println("3门课的平均分是：" + avg);
     }
}
```

1.1.8. 关键知识点默写

1. 变量的分类——按照数据类型来分：
    1. 基本数据类型(8种):
        - 数值型:
            - 整型:byte(-128~127) short int long
            - 浮点型: float double
        - 字符型:char占的是两个字节
        - 布尔型:boolean 注意,布尔类型的变量只有两种值:true false
    2. 引用数据类型：类、接口、数组、枚举

       注意：除了基本数据类型以外的数据类型都是引用数据类型.
2. 数据类型转换： 发生的场景：不同类型的变量之间进行赋值、运算
    1. 自动类型转换:byte short char ==>int==>long==>float==>double 注意：a.当byte short char 之间进行运算时结果属于int型； b.boolean类型不可以转换为其它的数据类型。
    2. 强制类型转换：将容量大的数据类型强制转换成容量小的数据类型，必须使用"(目标数据类型)"符号； 格式是：小的数据类型 = (小的数据类型)大的数据类型。 注意：强转后会造成精度的损失(丢失),丢的是高位。
3. 算术运算符：+ - * / % ++ --
    1. 取模：% 取余数，结果的符号取决于被模数的符号。
    2. ++：让变量的值自增1

    - 前++：++在变量的前面，先自增1，后运算
    - 后++：++在变量的后面，先运算，再自增1
  3. --：让变量的值自减1
    - 前--：--在变量的前面，先自减1，再运算
    - 后--：--在变量的后面，先运算，再自减1
4. 赋值运算符： = += -= *= /= %=
5. 比较运算符：== < > <= >= !=
6. 逻辑运算符： && || ！（运算符两端是条件表达式）

1.1.9. 常见面试题

1. 如何不借助第三个变量，交换两个数据?
2. char型变量中能不能存储一个中文汉字?为什么?
3. String属于基本数据类型吗?

1.1.10. 补充

1、UTF-8编码如何识别单字节和多字节字符?

字节0x00到0x7F仅用于ASCII，没有其他用途。0x7F以上的字节仅用于多字节序列：

- 1个字节：0xxxxx
- 2个字节：110xxxxx 10xxxxxx
- 3个字节：1110xxxx 10xxxxxx 10xxxxxx
- 4个字节：11110xxx 10xxxxxx 10xxxxxx 10xxxxxx