Java 语言基础

Author：zhangzhang

Version：1.0.0

一、引言

1.1 生活中变量

> 变量来源于数学，是计算机语言中能储存计算结果或能表示值抽象概念。变量可以通过变量名访问。

二、变量【 重点 】

2.1 变量的概念

> 概念：计算机内存中的一块存储空间，是存储数据的基本单元。

> - 整个内存就好像是酒店，当中包含了多个房间。

> - 房间的容量（大小）不同（单人间、两人间...）
> - 每个房间都有一个唯一的门牌号。
> - 每个房间的住客（类型）也不同。

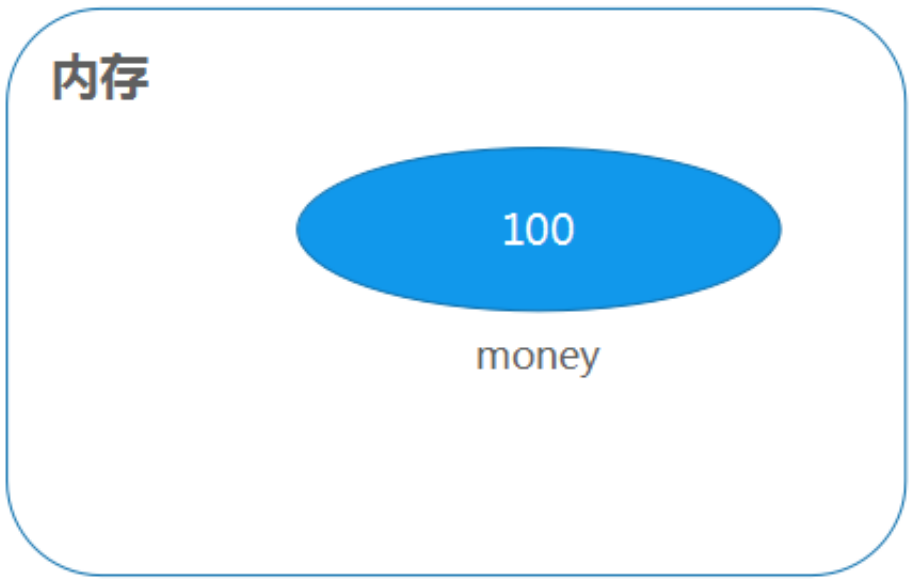
> - 酒店的房间 — 变量
>   - 房间的类型 — 数据类型
>   - 房间门牌号 — 变量名
>   - 房间的住客 — 值

2.2 变量的定义流程

> 声明：数据类型 变量名;int money; （开辟整数变量空间）。

> 赋值：变量名 = 值;money = 100; （将整数值赋给变量）。

> 应用：System.out.print(money);



注意：Java是强类型语言，变量的类型必须与数据的类型一致。

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

    //声明变量，语法：数据类型 变量名;
    int money; //在内存中开辟了一块整数空间

    //赋值，语法：变量名 = 值;
    money = 100; //将100赋值给money变量空间

    System.out.println(money); //打印变量中的值

    System.out.println("money"); //打印文本

  }
}
```

2.3 变量的定义方式

> 声明变量的3种方式：

> - 先声明，再赋值：【常用】
>   - 数据类型 变量名;
>   - 变量名 = 值;

> - 声明并赋值：【常用】
>   - 数据类型 变量名 = 值;

> - 多个同类型变量的声明与赋值：【了解】
>   - 数据类型 变量1 , 变量2 , 变量3 = 值3 , 变量4 , 变量5 = 值5;

```
public class TestVarDefined{

  public static void main(String[] args){

    //声明并赋值
    int age = 10; //将声明的语法与赋值的语法合二为一

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

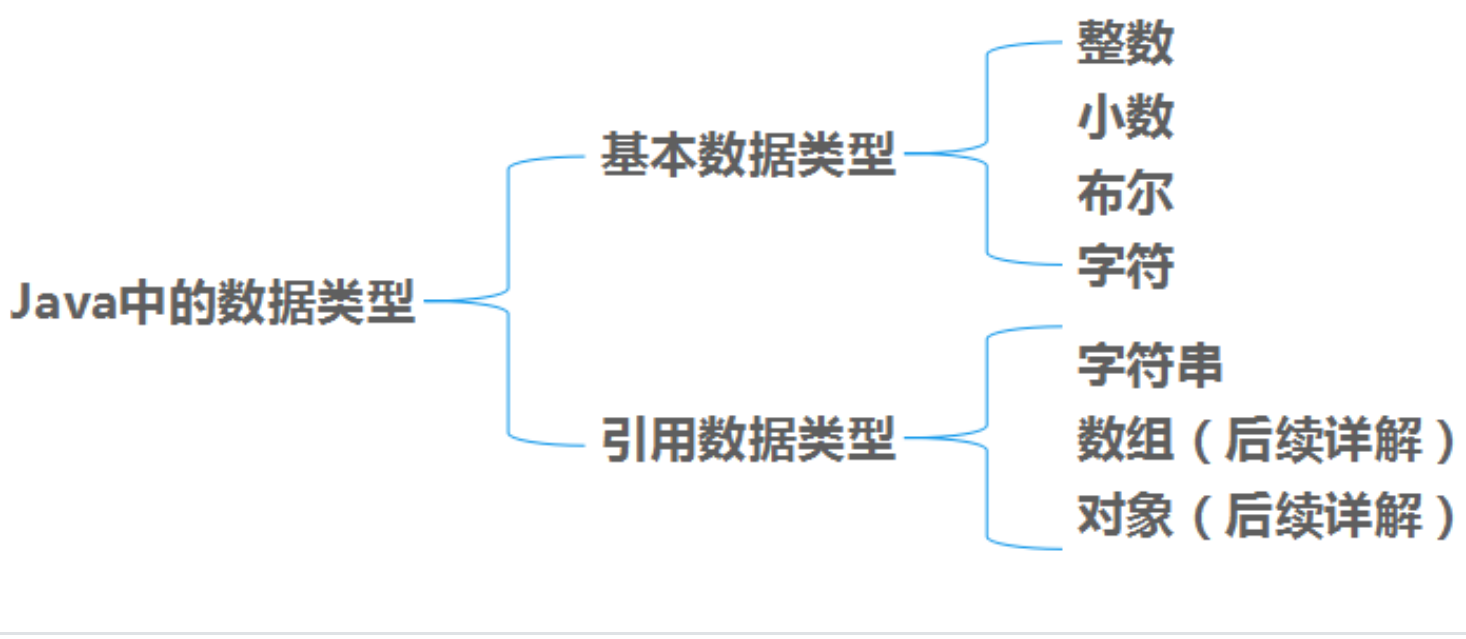
```
    //同时声明多个同类型变量
    int a , b , c , d = 44 , e = 55;

    a = 11;
    b = 22;
    c = 33;

    System.out.println(a);
    System.out.println(b);
    System.out.println(c);
    System.out.println(d);
    System.out.println(e);
  }
}
```

三、数据类型【 重点 】

3.1 数据类型

> Java中的变量具有严格的数据类型区分。（强类型语言）

> 在Java语言中，任何一个值，都有其对应类型的变量。



3.1.1 基本数据类型（整数）

类型	字节	取值范围（二进制）	取值范围（十进制）
byte	1字节	-2^7 ~ 2^7-1	-128 ~ 127
short	2字节	-2^15 ~ 2^15-1	-32768 ~ 32767
int	4字节	-2^31 ~ 2^31-1	-2147483648 ~2147483647
long	8字节	-2^63 ~ 2^63-1	-9223372036854775808 ~9223372036854775807

注意：int为整数的默认类型，如需为long类型赋值较大整数时，需在值的后面追加“L”。

```
/**
  基本数据类型（整数）
*/
public class TestType{
  public static void main(String[] args){

    //数据类型 变量名 = 值;

    byte b = 127;// -128 ~ 127 （共计256个整数）

    System.out.println(b);


    short s = 32767;//-32768 ~ 32767 （共计65536个整数）
```

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


    int i = 2147483647;//-2147483648 ~ 2147483647  (共计42亿多个整数)

    System.out.println(i);



    //Java中所有的“整数字面值”的默认类型是int，当整数字面值超过int的取值范围时，则提醒“过大的整数”

    long l1 = 2147483648L;//显示告知JVM，此值为long类型
    long l2 = 9223372036854775807L;//-9223372036854775808L ~ 9223372036854775807L

    System.out.println(l1);
    System.out.println(l2);
  }
}
```

3.1.2 基本数据类型（小数/浮点数）

类型	字节	负数取值范围	正数取值范围
float	4字节	-3.4E+38 ~ -1.4E-45	1.4E-45 ~ 3.4E+38
double	8字节	-1.7E+308 ~ -4.9E-324	4.9E-324 ~ 1.7E+308

- 浮点型数值采用科学计数法表示：
  - 2E3 等价于 2 * 10 ^ 3（结果：2000.0）
  - 3E5 等价于 3 * 10 ^ 5（结果：300000.0）

注意：double为浮点数的默认类型，如需为float类型赋值时，需要在值的后面追加“F”。

```
/**
  基本数据类型（小数/浮点数）
*/
public class TestType2{
  public static void main(String[] args){

    //float 单精度浮点型、double 双精度浮点型


    double d = 1.2;

    System.out.println(d);


     double d2 = 1;

     System.out.println(d2);


    float f = 1.5F; //任何“小数字面值”默认类型都是double，如果要存储到float变量中，需显示追加“F”

    System.out.println(f);

    //科学计数法
    double d3 = 2E3; // 2 * 10 ^ 3
    System.out.println(d3);

    double d4 = 2E7; // 2 * 10 ^ 7   20000000.0
    System.out.println(d4);

    float f2 = 5E4F; // 5 * 10 ^ 4 追加F代表float
    System.out.println(f2);


    //取值范围的差异

    //float正数取值范围：0.00000000000000000000000000000000000000000014F ~ 340000000000000000000000000000000000.0F

    float floatMin = 0.00000000000000000000000000000000000000000014F; //float的最小正数

    float floatMax = 340000000000000000000000000000000000.0F; //float的最大正数
```

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

    System.out.println(floatMax);


    //float负数取值范围：-340000000000000000000000000000000000000.0F ~
-0.00000000000000000000000000000000000000000000014F

    float floatMin2 = -340000000000000000000000000000000000000.0F;//个十百千万亿兆京垓。。。。。那由他

    float floatMax2 = -0.00000000000000000000000000000000000000000000014F;

    System.out.println(floatMin2);

    System.out.println(floatMax2);


    //double正数取值范围：
    double doubleMin = 4.9E-
324;//0.0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000049;
    double doubleMax =
1.7E308;//17000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000.
0;

    System.out.println(doubleMin);
    System.out.println(doubleMax);

  }
}
```

3.1.3 基本数据类型（布尔）

类型	字节	取值范围	描述
boolean	1字节	true / false	仅可描述“真”或者“假”

- 可直接赋值true / false
- 也可赋值一个结果为true / false的表达式

注意：Java中的boolean不能参与算数运算。

```
public class TestType3{

  public static void main(String[] args){

    boolean b1 = false; // true / false

    System.out.println(b1);


    boolean b2 = 5 > 4;

    System.out.println(b2);
  }
}
```

3.1.4 基本数据类型（字符-1）

- 前置知识：
  - ASCII(American Standard Code for Information Interchange)美国信息交换标准码。
  - 基于拉丁字母的一套电脑编码系统，主要用于显示现代英语和其他西欧语言。
  - ASCII是最通用的信息交换标准，为英文字符设定了统一并且唯一的二进制编码。

ASCII值	控制字	ASCII值	控制字	ASCII值	控制字	ASCII值	控制字	ASCII值	控制字	ASCII值	控制字	ASCII值	控制字	ASCII值	控制字
0	NUT	16	DLE	32	(space)	48	0	64	@	80	P	96	、	112	p
1	SOH	17	DCI	33	!	49	1	65	A	81	Q	97	a	113	q
2	STX	18	DC2	34	"	50	2	66	B	82	R	98	b	114	r
3	ETX	19	DC3	35	#	51	3	67	C	83	S	99	c	115	s
4	EOT	20	DC4	36	$	52	4	68	D	84	T	100	d	116	t
5	ENQ	21	NAK	37	%	53	5	69	E	85	U	101	e	117	u
6	ACK	22	SYN	38	&	54	6	70	F	86	V	102	f	118	v
7	BEL	23	TB	39	,	55	7	71	G	87	W	103	g	119	w
8	BS	24	CAN	40	(	56	8	72	H	88	X	104	h	120	x
9	HT	25	EM	41	)	57	9	73	I	89	Y	105	i	121	y
10	LF	26	SUB	42	*	58	:	74	J	90	Z	106	j	122	z
11	VT	27	ESC	43	+	59	;	75	K	91	[	107	k	123	{
12	FF	28	FS	44	,	60	<	76	L	92	/	108	l	124	|
13	CR	29	GS	45	-	61	=	77	M	93	]	109	m	125	}
14	SO	30	RS	46	.	62	>	78	N	94	^	110	n	126	`
15	SI	31	US	47	/	63	?	79	O	95	_	111	o	127	DEL

3.1.5 基本数据类型（字符-2）

- Unicode编码
- Unicode(万国码)是计算机科学领域里的一项业界标准，包括字符集、编码方案等。
- 它为每种语言中的每个字符设定了统一并且唯一的二进制编码。
- 以满足跨语言、跨平台进行文本转换、处理的要求，（其中包含了ASCII编码）。

U+	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
4f00	伀	企	伂	伃	伄	伅	伆	伇	伈	伉	伊	伋	伌	伍	伎	伏
4f10	伐	休	伒	伓	伔	伕	伖	众	优	伙	会	伜	伝	伞	伟	
4f20	传	伡	伢	伣	伤	伥	伦	伧	伨	伩	伪	伫	伬	伭	伯	
4f30	估	伯	伱	伲	伳	伴	伵	伶	伷	伸	伹	伺	伻	似	伽	伾
4f40	佀	佁	佂	佃	佄	佅	但	佇	佈	佉	佊	佋	佌	位	低	住
4f50	佐	佑	佒	体	佔	何	佖	佗	佘	余	佚	佛	作	佝	佞	佟
4f60	你	佡	佢	佣	佤	佥	佦	佧	佨	佩	佪	佫	佬	佭	佮	佯
4f70	佰	佱	佲	佳	佴	併	佶	佷	佸	佹	佺	佻	佼	佽	佾	使
4f80	侀	侁	侂	侃	侄	侅	來	侇	侈	侉	例	侌	侍	侏	侐	
4f90	侀	侑	侒	侓	侔	侕	侖	侗	侘	侙	侚	供	侜	依	侞	侟
4fa0	侠	価	侢	侣	侤	侥	侦	侧	侨	侩	侪	侫	侬	侭	侮	侯
4fb0	侰	侱	侲	侳	侴	侵	侶	侷	侸	侹	侺	侻	侼	侽	侾	便
4fc0	俀	俁	係	促	俄	俅	俆	俇	俈	俉	俊	俋	俌	俍	俎	俏
4fd0	俐	俑	俒	俓	俔	俕	俖	俗	俘	俙	俚	俛	俜	保	俞	俟
4fe0	俠	信	修	俭	俯	俰	俱	俲	俳	俴	俵	俶	俷	俸	俹	俺
4ff0	俼	俽	俾	倀	倁	倂	倃	倄	倅	倆	倇	倈	倉	個	倊	個
.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....

3.1.6 基本数据类型（字符-3）

类型	字节	取值范围	字符编码
char	2字节	0 ~ 65535	Unicode字符集（万国码）

- Unicode中每个字符都对应一个二进制整数，可以使用多种方式赋值。
- 字符赋值：char c1 = 'A';（通过''描述为字符赋值）
- 整数赋值：char c2 = 65;（通过十进制数65在字符集中对应的字符赋值）
- 进制赋值：char c3 = '\u0041';（通过十六进制数41在字符集中所对应的字符赋值）

```
public class TestType4{

  public static void main(String[] args){

    char c1 = 'A'; //字符赋值，原生、基本的赋值方式（常用）

    System.out.println(c1);


    char c2 = 65;//整数赋值（十进制）

    System.out.println(c2);


    char c3 = '\u0041';//进制赋值（十六进制）

    System.out.println(c3);
  }
}
```

3.1.7 转义字符（1）

如果需要在程序中输出一个单引号字符，该如何完成?

```
package demo;

public class TestChar {
    public static void main(String[] args) {
        char c = ''';
    }
}
```

编译错误：
未结束的字符文字

为了解决这一问题，Java采用了转义字符来表示单引号和一些特殊符号。

3.1.8 转义字符（2）

转义字符	描述
\n	换行符
\t	缩进（制表位）
\\	反斜线
\'	单引号
\''	双引号

```
public class TestSign{

  public static void main(String[] args){

    char c1 = '\'';

    System.out.println(c1);


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

    System.out.println("Hello\tWorld");

    System.out.println("Hello\nWorld");

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

    System.out.println("u0041");

    System.out.println("\u0041");
  }
}
```

3.2 引用数据类型（字符串）

类型	取值范围	字符编码
String	任何” ”之间的字面值	Unicode字符序列

- String类型的字面取值：
- String str1 = ''Hello'';
- String str2 = ''您好'';

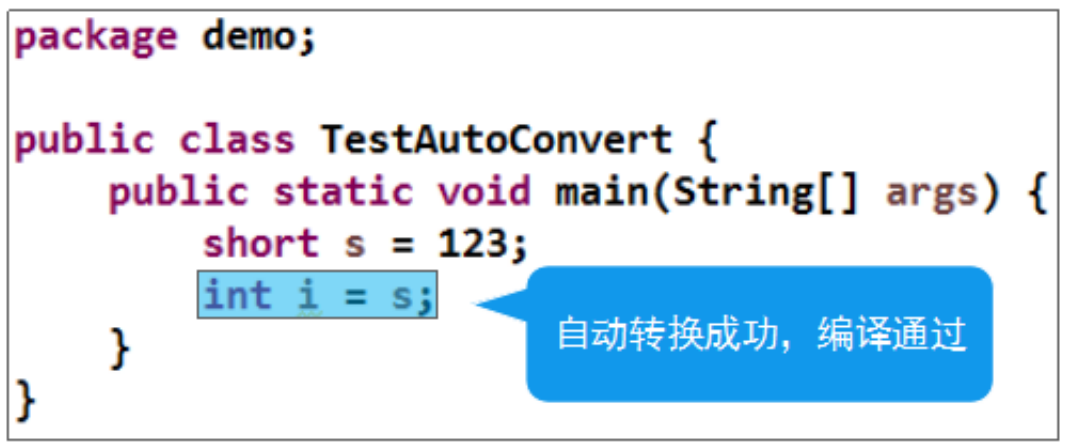
- String str3 = ''Java Engineer'';
- String str4 = ''微服务架构师'';

```
public class TestString{

  public static void main(String[] args){

    String str1 = "HelloWorld";

    System.out.println(str1);

    System.out.println("HelloWorld");

    String str2 = "Hello  Everyone";

    System.out.println(str2);

  }
}
```

四、类型转换

4.1 类型转换（1）

- 自动类型转换：
  - 两种类型相互兼容。
  - 目标类型大于源类型。




```
public class TestAutoConvert{

  public static void main(String[] args){

    //整数 - 整数

    short s = 123;

    int i = s;//将源类型值存入到目标类型变量中（自动类型转换）

    System.out.println(i);


    byte b = 100;

    short s2 = b;//自动类型转换

    System.out.println(s2);


    //小数 - 小数

    float f = 100.0F;

    double d = f;//自动类型转换

    System.out.println(d);


    //小数 - 整数

    int i2 = 100;

    double d2 = i2;//自动类型转换

    System.out.println(d2);
```

```
    //字符 - 整数

    char c = 'A';

    int i3 = c;//自动类型转换

    System.out.println(i3);


    //字符 - 小数

    char c2 = 'a';

    double d3 = c2;

    System.out.println(d3);


    //boolean无法与其他类型进行转换

    boolean bool = true;//true | flase

    int i4 = bool;//不兼容的类型

  }
}
```

4.2 类型转换（2）

> - 强制类型转换：
>   - 两种类型相互兼容。
>   - 目标类型小于源类型。


```
public class TestForceConvert{

  public static void main(String[] args){

    //长度足够，数据完整
    short s = 123;

    byte b = (byte)s;//强制类型转换（数据完整）

    System.out.println(b);

    //长度不够，数据截断

    short s2 = 257;

    byte b2 = (byte)s2;//强制类型转换（数据截断）

    System.out.println(b2);


    short s3 = 130;

    byte b3 = (byte)s3;

    System.out.println(b3);


    //小数 强转 整数

    double d = 2.999;

    int i = (int)d;

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

```
    //字符 强转 整数

    char c = 'A';

    int i2 = c;//自动类型转换

    System.out.println(i2);


    char c2 = (char)i2;//强制类型转换

    System.out.println(c2);


    //字符与整数转换的注意事项

    short s4 = -1;// -32768 ~ 32767

    char c3 = (char)s4;//强制类型转换

    System.out.println(c3);
  }
}
```

五、运算符【重点】

5.1 算数运算符

> 算数运算符：两个操作数进行计算。

操作符	描述
+	加、求和
-	减、求差
*	乘、求积
/	除、求商
%	模、求余

> 算数运算符：一元运算符（只有一个操作数）。

操作符	描述
++	递增，变量值+1
- -	递减，变量值-1

```
public class TestOperation1{

  public static void main(String[] args){

    int a = 10;

    int b = 3;

    System.out.println( a / b );//求商 = 3

    System.out.println( a % b );//求余 = 1


    double d = 10.0;

    int c = 3;

    System.out.println(d / c);//求商 3.33.......


    int num1 = 10;

    num1++;//自增1
```

```
        System.out.println(num1);



        int num2 = 10;

        num2--;//自减1

        System.out.println(num2);



        int num3 = 5;

        //前++ ： 先++，再打印自增后的值

        //后++ ：  先打印当前值，再++

        System.out.println( ++num3 );

        System.out.println( num3 );


        int num4 = 100;

        //前++ ： 先++，再赋值

        //后++ ：  先赋值，再++

        int num5 = num4++;

        System.out.println(num5);

        System.out.println(num4);


    }
}
```

5.2 赋值运算符

> 赋值运算符：等号右边赋值给等号左边。

操作符	描述
=	直接赋值
+=	求和后赋值
-=	求差后赋值
*=	求积后赋值
/=	求商后赋值
%=	求余后赋值

```
public class TestOperation2{

    public static void main(String[] args){

        int a = 10; //赋值运算符

        a += 5; //在a基础上+5      a = a + 5;


        System.out.println(a);


        int b = 20;

        b -= 3;// b = b - 3;

        System.out.println(b);


        int c = 30;

        c %= 4;// c = c % 4;

        System.out.println(c);
    }
```

```
}
```

5.3 关系运算符

> 关系运算符：两个操作数进行比较。

操作符	描述
>	大于
<	小于
>=	大于等于
<=	小于等于
==	等于
!=	不等于

```
public class TestOperation3{

  public static void main(String[] args){

    int a = 10;
    int b = 6;

    System.out.println( a > b );
    System.out.println( a < b );


    System.out.println( a >= b );
    System.out.println( a <= b );


    System.out.println( a == b );
    System.out.println( a != b );


  }
}
```

5.4 逻辑运算符

> 逻辑运算符：两个boolean类型的操作数或表达式进行逻辑比较。

操作符	语义	描述
&&	与（并且）	两个操作数，同时为真，结果为真
||	或（或者）	两个操作数，有一个为真，结果为真
!	非（取反）	意为“不是”，真即是假，假即是真

```
public class TestOperation4{

  public static void main(String[] args){

    int javaScore = 100;

    int webScore = 99;

    //比较两者是否相等
    System.out.println( javaScore == webScore);


    //别分判断二者是否为满分
    System.out.println( javaScore == 100);
    System.out.println( webScore == 100);


    //一次性判断二者是否均为满分

    //                  false
    //             true       &&    false     两个表达式同时为真
    System.out.println( javaScore == 100  && webScore == 100 );
```

```
    //一次性判断二者是是否有一个满分

    //                            true
    //             true     ||     false
    System.out.println( javaScore == 100  ||  webScore == 100 );



    boolean result = javaScore == 100;

    //Java的成绩是满分吗?
    System.out.println(result);//true

    //Java的成绩不是满分吗?
    System.out.println( !result );//false

  }
}
```

5.5 三元运算符

> 三元运算符：将判断后的结果赋值给变量。

操作符	语义	描述
？：	布尔表达式?结果1:结果2	当表达式结果为真，获得结果1 当表达式结果为假，获得结果2

```
public class TestOperation5{

  public static void main(String[] args){

    //1.判断
    //2.赋值
    //布尔表达式 ? 值1 : 值2

    int javaScore = 100;

    String result = javaScore == 100 ? "恭喜" : "加油" ;

    System.out.println(result);


    int webScore = 99;

    int result2 = webScore == 100 ? 666 : 111;

    System.out.println(result2);

  }
}
```

5.6 表达式

> 使用运算符连接的变量或字面值，并可以得到一个最终结果。

- **例如：**
  - 1 + 2;
  - int a = 3;　　a - 2;
  - int b = 10;　　int c = 20;　　b * c;　　c / b;
  - short d = 100;　　int e = 200;　　d > e;　　d <= e;
  - ……

六、类型提升

6.1 自动类型提升

> - 进行算数运算时：
>   - 两个操作数有一个为double，计算结果提升为double。
>   - 如果操作数中没有double，有一个为float，计算结果提升为float。
>   - 如果操作数中没有float，有一个为long，计算结果提升为long。
>   - 如果操作数中没有long，有一个为int，计算结果提升为int。
>   - 如果操作数中没有int，均为short或byte，计算结果仍旧提升为int。

注意：任何类型与String相加（+）时，实为拼接，其结果自动提升为String。

```
public class TestTypeRaise{

  public static void main(String[] args){

    double d1 = 10.0;

    int i1 = 5;

    double d2 = d1 + i1;

    System.out.println(d2);

    //----------------------------------------

    float f1 = 5.0F;

    short s1 = 20;

    float f2 = f1 + s1;

    System.out.println(f2);

    //----------------------------------------

    long l1 = 100;

    byte b1 = 50;

    long l2 = l1 + b1;

    System.out.println(l2);

    //----------------------------------------

    int i3 = 123;

    short s3 = 456;

    int i4 = i3 + s3;

    System.out.println(i4);

    //----------------------------------------

    short s4 = 321;

    byte b3 = 111;

    int s5 = s4 + b3;

    System.out.println(s5);


    //----------------------------------------

    //特殊：String的字符串拼接

    String str = "Hello";

    int i5 = 123;

    String str2 = str + i5;

    System.out.println(str2);


    int javaScore = 100;

    String str3 = "Java的成绩是：" + javaScore;

    System.out.println(str3);

    System.out.println( "Java的成绩是：" + javaScore );
```

```
  }
}
```

七、控制台录入

> 程序运行中，可在控制台（终端）手动录入数据，再让程序继续运行。

> 导包语法：import 包名.类名;//将外部class文件的功能引入到自身文件中。

- 使用顺序：
  - 导入 java.util.Scanner。
  - 声明 Scanner 类型的变量。
  - 使用Scanner类中对应的方法（区分类型）：
    - .nextInt(); //获得整数
    - .nextDouble(); //获得小数
    - .next(); //获得字符串
    - .next().charAt(0);//获得单个字符

注：如果输入了不匹配的数据，则会产生 java.util.InputMismatchException。

```
//package 必须在源文件的首行

import java.util.Scanner;//1.引入外部文件

public class TestScanner{

  public static void main(String[] args){

    //2.声明Scanner类型的变量
    Scanner input = new Scanner(System.in);

    System.out.println("请输入一个整数: ");

    //3.使用
    int i = input.nextInt(); //控制台获取一个整数

    System.out.println("您输入的值为: " + i);
  }
}
```

```
import java.util.Scanner;

public class TestScanner2{

  public static void main(String[] args){

    Scanner input = new Scanner(System.in);

    System.out.println("请输入值: ");

    int i = input.nextInt();//接收整数

    double d = input.nextDouble();//接收小数

    String s = input.next();//接收字符串

    char c = input.next().charAt(0);//接收字符（接收一个完整的字符串，获取其中的第一个字符）


    System.out.println("整数: " + i);
    System.out.println("小数: " + d);
    System.out.println("字符串: " + s);
    System.out.println("字符: " + c);


  }
}
```