Java Day 03

局部变量和实例变量

局部变量：

1）定义在方法的里面

2）必须先赋值后使用

3）作用域：声明点到方法的右半个花括号

实例变量：

1）定义在类的里面方法的外面

2）可以不用初始化直接使用（有默认值）

byte	short	int	long	float	double	char	boolean	All reference types
0	0	0	0L	0.0f	0.0d	'\u0000'	false	null

'\u0000'代表的应该是 nul,一个控制字符，表示没有值，不可见，输出控制台是空，但不是真正的空格，因为真正的空格的 unicode 编码是'\u0020'

3）作用域：类的内部

作用域：指标识符可以使用的有效范围

可见性：指标识符可以被访问、引用的范围

作用域内不一定可见，但可见一定在作用域内

补充知识

任意进制到十进制的转换原理

把系数*基数的权次幂相加即可

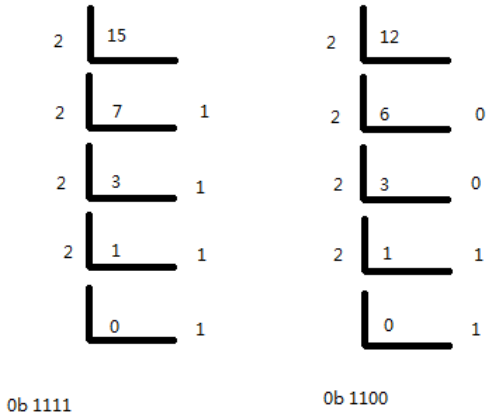
系数：就是每一位上的数据

基数：X 进制，基数就是 X

权：在右边，从 0 开始编号，对应位上的编号即为该位的权

十进制到任意进制的转换原理

除积倒取余



原码反码补码

1）为什么要学习原码反码补码？

后面要学习位运算、强制类型转换，如果不知道有原反补会看不懂结果

2）有符号数据表示法的几种方式

原码

就是二进制定点表示法，即最高位为符号位，“0”表示正，“1”表示负，其余位表示数值的大小。

通过一个字节,也就是 8 个二进制位表示+7 和-7

0(符号位) 0000111

1(符号位) 0000111

反码

正数的反码与其原码相同；负数的反码是对其原码逐位取反，但符号位除外。

11111000(-7)

补码

正数的补码与其原码相同；负数的补码是在其反码的末位加 1。

11111001(-7)

注意：计算机中，数都以补码的形式存放，

正数的补码是其本身，负数的补码是其绝对值取反加 1；

如果补码的符号位为“0”,表示是一个正数,所以补码就是该数的原码

如果补码的符号位为“1”,表示是一个负数,求原码的操作可以是:符

号位为 1,其余各位取反,然后再整个数加 1

操作符

赋值操作符

```
int a = 10;
```

1）数据类型要匹配

2）左值只能是变量

3）复合赋值运算符会将右值的类型隐式地转换为左值的类型，可能会造成数据损失

instanceof

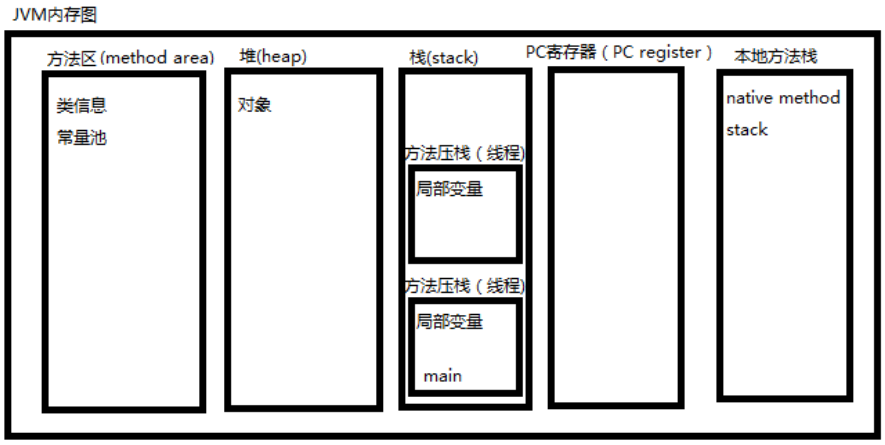
判断对象是否是某一类类型；不能操作基本数据类型

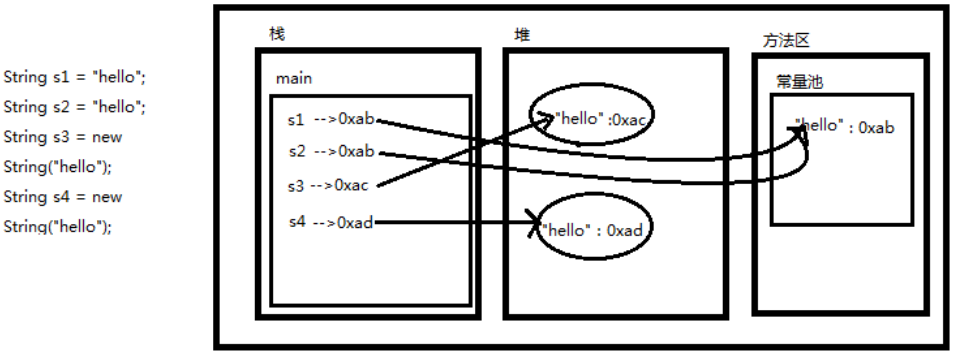
`java.lang.Object` 所有类的父类

==

对于基本数据类型，判断值是否相等；对于类类型，判断地址是否相同

内存图讲解字符串比较





移位运算

```
int a = 12; a << 2;
```

左移运算符用“<<”表示，是将运算符左边的对象，向左移动运算符右边指定的位数，并且在低位补零；其实向左移 n 位，就相当于乘上 2 的 n 次方。

右移运算符用符号“>>>”表示，逻辑右移，是将运算符左边的对象向右移动运算符右边指定的位数，并且在高位补 0；其实右移 n 位，就相当于除上 2 的 n 次方。

带符号的右移运算符用符号“>>”表示，算数右移，是将运算符左边的运算对象，向右移动运算符右边指定的位数。如果是正数，在高位

补零，如果是负数，则在高位补 1

位运算符

&：与 0 相&，将某位清零

|：与 1 相|，将某位置一

^：与 1 相^，将某位反转

逻辑操作符

&&：逻辑运算符，具有逻辑短路功能（连接的两个条件，如果第一个条件为假，后面的条件不进行运算）

&：位运算符，但可以连接多个条件，但没有短路功能

？ ： 三目运算符

++、--

优先级	运算符	名称
1.	()	括号
2.	[ ] ， .	后缀运算符
3.	！ ， ~ ， ++ ， -- ， - （一元运算符，取负数）	一元运算符
4.	* ， / ， %	乘，除，取模
5.	+ ， -	加，减
6.	>> ， << ， >>>	移位运算符
7.	> ， < ， >= ， <= ， instanceof	关系运算符
8.	= = ，！=	等于，不等于
9.	&	按位与
10.	^	按位异或
11.	|	按位或
12.	&&	逻辑与
13.	||	逻辑或
14.	?:	条件运算符
15.	= ， += ， - = ， *= ， /= ， %=	（算术）赋值运算符

类型转换

1）隐式类型转换

a）基本数据类型：位宽度窄的向位宽度宽的转

b）类类型：子类向父类转

2）显示类型转换

a）基本数据类型：位宽度宽的向位宽度窄的转

b）类类型：父类向子类转

小题目

---

1)复合赋

```
short x = 0;
int i = 123456;
```

```
x += i;//编译通过，x = -7616
```

```
x = x + i;//编译错误
```

注意：复合赋值表达式自动地将所执行计算的结果转型为其左侧变量的类型

2）判断奇数

```
public boolean isOdd(int i){
   return i % 2 == 1;
   //return i % 2 != 0;
}
```

注意：当取余操作返回一个非零的结果时，它与左操作数具有相同的

正负符号

3）逻辑操作符

&&和&区别

注意：&是位运算符，可以进行逻辑运算，但不会逻辑短路

4）条件运算符

```
int a = 20;
int b = 10;
char c = 'x';
int i = 0;
System.out.println(a > b ? c : 0);
System.out.println(a < b ? i : c);
```

注意：除非彻底理解表达式行为的复杂规范，通常最好在条件表达式

中使用类型相同的第二和第三操作数

5）自增运算

```
int i = 10;
int j = 20;
System.out.println(i+++j);
System.out.println(i+++++j);
```

词法解析器，利用贪心算法，将尽量多的字符解析成一个整体进行运

算

6）类型转换

```
System.out.println((int)(char)(byte) - 1);
```

类型转换时, 如果最初的数值类型是有符号的, 那么就执行符号扩展；

如果它是 char，那么不管将要被转换成什么类型，都执行零扩展

练习

不引入第三个变量，交换两个数

```
int x =5,y=10; //定义两个变量
x = x^y;
y = x^y;  //y=x^y^y=x
x = x^y;  //x=x^y^x=y
System.out.println("x="+x+"y="+y);
```

^的特点：一个数据对另一个数据位异或两次，该数本身不变。