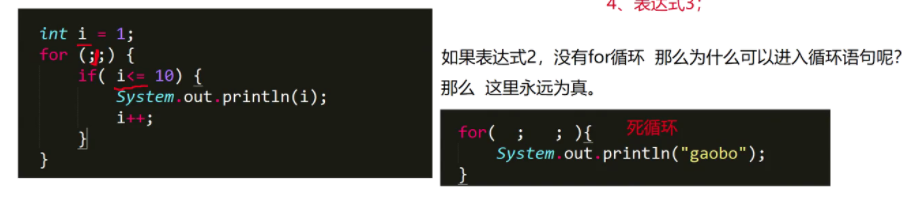
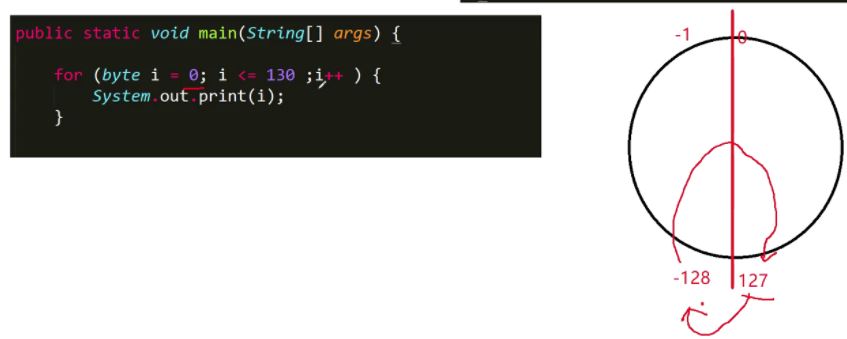
if(布尔表达式){
}
switch(整形|枚举|字符|字符串){
case :
break;
default :
break;
}
for()

**long float double boolean 不可以作为switch的判断依据**

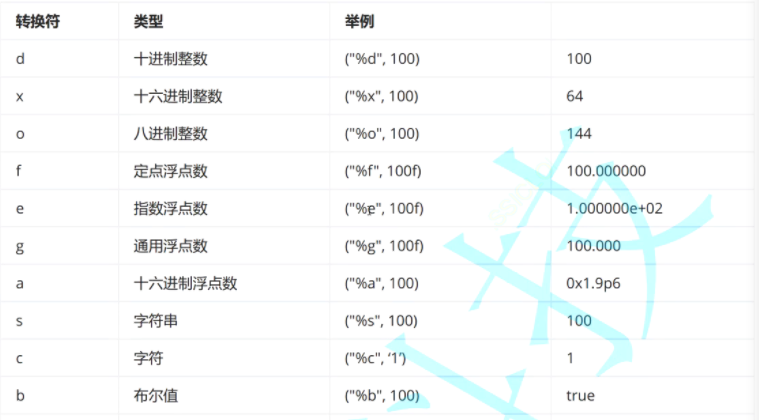
**for循环之死循环：（表达式2为空时，默认为true）**

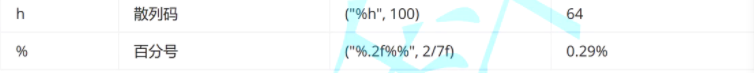




continue结束本次循环，继续下一次循环

break退出整个循环结束（break只能在switch或者循环中出现）





**生成随机数：**

Random ran=new Random();
int random=ran.nextInt(100);[0,100)

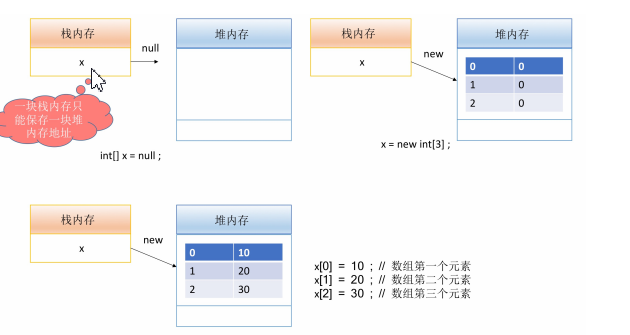
方法（public static):返回值可以是void或者引用类型及基本类型

方法重载：方法名称相同，参数列表不同（参数的类型、顺序或个数不同）

方法的签名：指的是方法名与参数，返回类型不是签名的⼀部分

**数组（存放相同数据类型的集合）**

数组是一个引用数据类型，存在内存分配问题，在使⽤之前⼀定要开辟空间（实例化），否则就会产⽣*NullPoninterException*



**动态初始化:**先开辟内存空间，再利用索引进⾏内容的设置

**静态初始化:**在定义的同时可以设置内容

数据类型[] 数组名称 = {值，值，....}

数据类型[] 数组名称 = new 数据类型[] {值，值，....}

**匿名数组：**

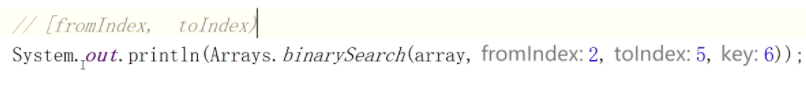
System.out.println(new int[]{1,2,5,55,555,223,45545,666465,6443}.length) ;

**数组排序：** java.util.Arrays.sort(arrayName) ;内部为快速排序

**数组拷贝：System.arraycopy > Arrays.copyOf > for > clone() 均为浅拷贝**

System.arraycopy(源数组名称，源数组开始点，⽬标数组名称，⽬标数组开始点，拷⻉⻓度);
java.util.Arrays.copyOf(源数组名称,新数组⻓度)

**二分查找法：**

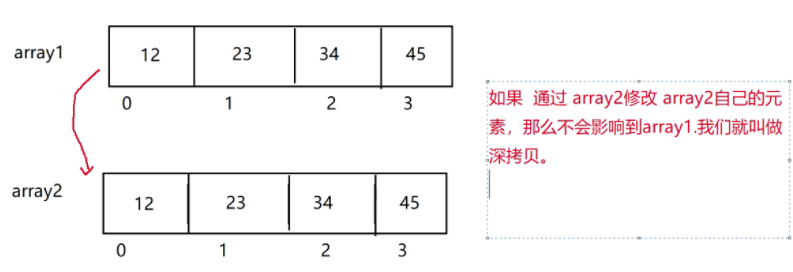


**判断两个数组内容是否相同（Arrays.equals)**

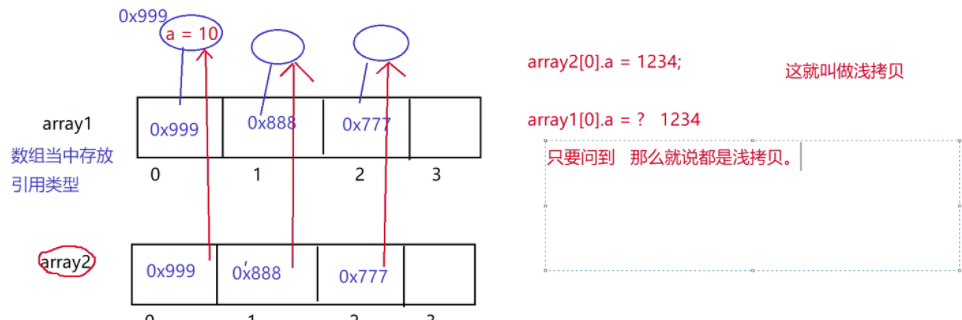
**判断两个引用是否相同（equals)**

缺点：容易发生越界问题

**深拷贝：**



**浅拷贝：**



**内存划分：**

