[**equals和==的区别小结**](http://www.cnblogs.com/Eason-S/p/5524837.html)

**==：**

**== 比较的是变量(栈)内存中存放的对象的(堆)内存地址，用来判断两个对象的地址是否相同，即是否是指相同一个对象。比较的是真正意义上的指针操作。**

1、比较的是操作符两端的操作数是否是同一个对象。  
2、两边的操作数必须是同一类型的（可以是父子类之间）才能编译通过。  
3、比较的是地址，如果是具体的阿拉伯数字的比较，值相等则为true，如：  
int a=10 与 long b=10L 与 double c=10.0都是相同的（为true），因为他们都指向地址为10的堆。

**equals：**

**equals用来比较的是两个对象的内容是否相等，由于所有的类都是继承自java.lang.Object类的，所以适用于所有对象，如果没有对该方法进行覆盖的话，调用的仍然是Object类中的方法，而Object中的equals方法返回的却是==的判断。**

　　String s="abce"是一种非常特殊的形式,和new 有本质的区别。它是java中唯一不需要new 就可以产生对象的途径。以String s="abce";形式赋值在java中叫直接量,它是在常量池中而不是象new一样放在压缩堆中。这种形式的字符串，在JVM内部发生字符串拘留，即当声明这样的一个字符串后，JVM会在常量池中先查找有有没有一个值为"abcd"的对象,如果有,就会把它赋给当前引用.即原来那个引用和现在这个引用指点向了同一对象,如果没有,则在常量池中新创建一个"abcd",下一次如果有String s1 = "abcd";又会将s1指向"abcd"这个对象,即以这形式声明的字符串,只要值相等,任何多个引用都指向同一对象.  
　　而String s = new String("abcd");和其它任何对象一样.每调用一次就产生一个对象，只要它们调用。

　　也可以这么理解: String str = "hello"; 先在内存中找是不是有"hello"这个对象,如果有，就让str指向那个"hello".如果内存里没有"hello"，就创建一个新的对象保存"hello". String str=new String ("hello") 就是不管内存里是不是已经有"hello"这个对象，都新建一个对象保存"hello"。

具体可以看下面的代码：

[复制代码](javascript:void(0);)

1 public class test1 {

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

3 String a = new String("ab"); // a 为一个引用

4 String b = new String("ab"); // b为另一个引用,对象的内容一样

5 String aa = "ab"; // 放在常量池中

6 String bb = "ab"; // 从常量池中查找

7 if (aa == bb) // true

8 System.out.println("aa==bb");

9 if (a == b) // false，非同一对象

10 System.out.println("a==b");

11 if (a.equals(b)) // true

12 System.out.println("aEQb");

13 if (42 == 42.0) { // true

14 System.out.println("true");

15 }

16 }

17 }

[复制代码](javascript:void(0);)

**equals和==的区别**

equals方法最初是在所有类的基类Object中进行定义的，源码是

|  |  |
| --- | --- |
| 1  2  3 | public boolean equals(Object obj) {      return (this == obj);      } |

　　由equals的源码可以看出这里定义的equals与==是等效的（Object类中的equals没什么区别），不同的原因就在于有些类（像String、Integer等类）对equals进行了重写，但是没有对equals进行重写的类（比如我们自己写的类）就只能从Object类中继承equals方法，其equals方法与==就也是等效的，除非我们在此类中重写equals。

　　对equals重新需要注意五点：

　　1   自反性：对任意引用值X，x.equals(x)的返回值一定为true；  
　　2   对称性：对于任何引用值x,y,当且仅当y.equals(x)返回值为true时，x.equals(y)的返回值一定为true；  
　　3   传递性：如果x.equals(y)=true, y.equals(z)=true,则x.equals(z)=true ；  
　　4   一致性：如果参与比较的对象没任何改变，则对象比较的结果也不应该有任何改变；  
　　5   非空性：任何非空的引用值X，x.equals(null)的返回值一定为false 。

String类对equals的重写如下：

[复制代码](javascript:void(0);)

1 public boolean equals(Object anObject) {

2 if (this == anObject) {

3 return true;

4 }

5 if (anObject instanceof String) {

6 String anotherString = (String)anObject;

7 int n = count;

8 if (n == anotherString.count) {

9 char v1[] = value;

10 char v2[] = anotherString.value;

11 int i = offset;

12 int j = anotherString.offset;

13 while (n-- != 0) {

14 if (v1[i++] != v2[j++])

15 return false;

16 }

17 return true;

18 }

19 }

20 return false;

21 }

[复制代码](javascript:void(0);)

　　另外，"=="比"equals"运行速度快,因为"=="只是比较引用。