Java核心API (Object类)

在Java体系中，java.lang.Object类位于顶端（是所有对象的直接或间接父类）。如果一个类没有写extends关键字声明其父类，则该类默认继承java.lang.Object类。Object类定义了"对象"的基本行为，被子类默认继承。

## toString方法

toString()方法：返回一个可以表示该对象属性内容的字符串。

|  |
| --- |
| MyObject obj=new MyObject(); // MyObject类自定义  String info=obj.toString();  System.out.println(info); |

A. 上例MyObject 类没有toString方法，对象也能调用toString()？

因为所有的类都继承自Object，而toString方法是Ojbect定义的，我们直接继承了这个方法。Object的toString方法帮我们返回一个字符串，这个字符串的格式是固定的：类型@hashcode，这个hashcode是一串数字，在java中叫句柄，或叫地址（但不是真实的物理地址，是java自己的一套虚拟地址，防止直接操作内存的）。

//只能用public，重写的方法访问权限要大于等于父类中方法的权限

public String toString(){

return "这个是我们自己定义的toString方法的返回值MyObject!";

}

B．为什么要重写toString方法？

toString定义的原意是返回能够描述当前这个类的实例的一串文字，我们看一串hashcode没意义，所以是要重写的。

## equals方法：

equals()：用于对象的"相等"逻辑。

A．在Object中的定义：

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

由此可见，this==obj与直接的==（双等于）效果一样，仅仅是根据对象的地址（句柄，那个hashcode值）来判断对象是否相等。因此想比较对象与给定对象内容是否一致，则必须重写equals方法。

B．"=="与equals的区别：

用"=="比较对象时，描述的是两个对象是否为同一个对象！根据地址值判断。而equals方法力图去描述两个对象的内容是否相等，内容相等取决于业务逻辑需要，可以自行定义比较规则。

C．equals方法的重写：如，判断两点是否相等。

|  |
| --- |
| public boolean equals(Object obj){  /\* 注意参数，若给定的对象obj的地址和当前对象地址一致，那么他们是同一个对象，  \* equals方法中有大量的内容比较逻辑时，加上这个判断会节省性能的开销！  \*/  if(this == obj){  return true;  }  /\* equals比较前要进行安全验证，确保给定的对象不是null！若obj是null，  \* 说明该引用变量没有指向任何对象，那么就不能引用ojb所指象的对象（因为对象不存在）  \* 的属性和方法！若这么做就会引发NullPointException，空指针异常！  \*/  if(obj == null){  return false;  }  /\* 直接将Object转为子类是存在风险的！我们不能保证Object和我们要比较的对象是  \* 同一个类型的这会引发ClassCastException！我们称为：类造型异常。  \* 重写equals时第一件要做的事情就是判断给定的对象是否和当前对象为同一个类型，  \* 不是同类型直接返回false，因为不具备可比性！  \*/  if( !(obj instanceof Point) ){  return false;  }  // 是同一类型，转换  Point p=(Point)obj;  // 最后返回内容比较( 根据实际情况决定 )  return this.x==p.x && this.y==p.y;  } |

## hashCode()方法

hashcode是用来查找的，例如内存中有这样的位置： 0 1 2 3 4 5 6 7，而我有个类，这个类有个字段叫ID，我要把这个类存放在以上8个位置之一，如果不用hashcode而任意存放，那么当查找时就需要到这八个位置里挨个去找，或者用二分法一类的算法。但如果用hashcode那就会使效率提高很多。我们这个类中有个字段叫ID，那么我们就定义我们的hashcode为ID％8，然后把我们的类存放在取得得余数那个位置。比如我们的ID为9，9除8的余数为1，那么我们就把该类存在1这个位置，如果ID是13，求得的余数是5，那么我们就把该类放在5这个位置。这样，以后在查找该类时就可以通过ID除 8求余数直接找到存放的位置。

但是如果两个类有相同的hashcode怎么办那（我们假设上面的类的ID不是唯一的），例如9除以8和17除以8的余数都是1，那么这是不是合法的，回答是：可以这样。那么如何判断呢？在这个时候就需要定义 equals了。也就是说，我们先通过 hashcode来判断两个类是否存放某个桶里，但这个桶里可能有很多类，那么我们就需要再通过 equals 来在这个桶里找到我们要的类。 那么。重写了equals()，为什么还要重写hashCode()呢？ 想想，你要在一个桶里找东西，你必须先要找到这个桶啊，你不通过重写hashcode()来找到桶，光重写equals()有什么用啊。

注意：编码时候，可以采用eclipse快捷工具生成对应的equals()和hashCode()方法。