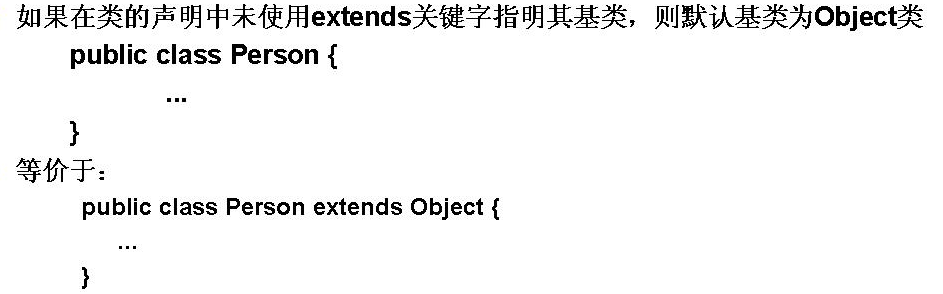
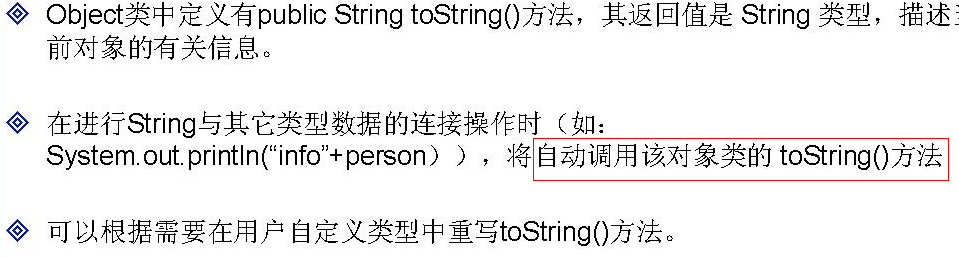
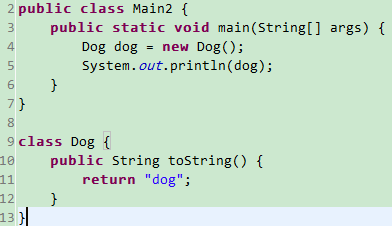
1. Object类是所有Java类的根基类，如果在类的声明中未使用extends关键字指明其基类。则默认基类为Object类。



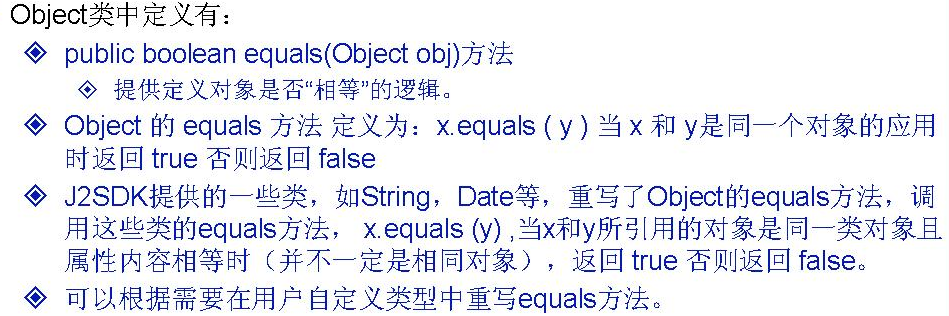
1. toString()方法



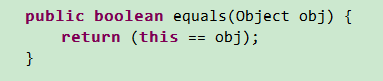
如下例，会输出“dog”。



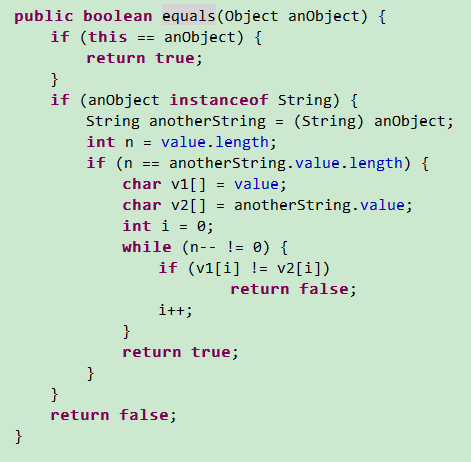
1. equals()方法: public boolean equals(Object obj)



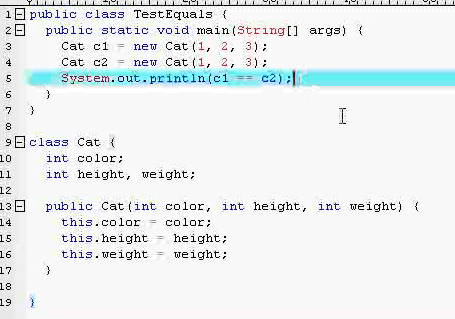
* 1. Object类中的equals实现



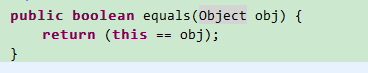
* 1. String类中equals的实现



例，输出false。因为比较的是c1和c2这两个引用的值。显然不相等。



若打印c1.equals(c2)，依然输出false。调用的是Object类中的equals方法，如下：



所以Object类中的equals方法并没有真正实现比较对象中的内容是否相等，即等价于c1 == c2。所以需要重写该方法。



**说明：instanceof：判断是不是该类的类型(该类或该类的子类)**

String、Date等类中重写了Object类中的equals方法，其比较的是真正的内容是否相等。

输出false

输出true

1. **hashCode**()方法：public int hashCode()

HashSet、HashMap等使用hashCode来达成存取速度较快的存储方法。

1. 两个对象怎样才算相等呢？

hashCode值相等，并且equals

1. hashCode()与equals()的相关规定：API文件有对对象的状态指定出必须遵守的规则
   1. 如果两个对象相等，则hashcode必须也是相等的
   2. 如果两个对象相等，对其中一个对象调用equals()必须返回true。也就是说，若a.equals(b)则b.equals(a)
   3. 如果两个对象有相同的hashcode(碰撞产生)，他们也不一定是相等的。但是若两个对象相等，则hashcode值一定是相等的
   4. 因此若equals()被覆盖过，则hashCode()也必须被覆盖
   5. hashCode的默认行为是对在heap上的对象产生的独特的值。如果没有override过hashCode()，则该class的两个对象怎样都不会被认为是相同的
   6. equals()的默认行为是执行==的比较(Object中实现的)，也就是说会去测试两个引用是否堆上heap上同一个对象。如果equals()没有被override过，两个对象永远不会被视为相同的，因为不同的对象有不同的字节组合。a.equals(b)必须与a.hashCode()==b.hashCode()等值。但a.hashCode()==b.hashCode()不一定要与a.equals(b)等值
2. HashSet如何检查重复：hashCode()与equals()

当把对象加入到HashSet时，它会使用对象的hashcode值来判断对象加入的位置。同时也会与其他已经加入的对象的hashcode值作比较，如果没有相符的hashcode，HashSet会假设新对象没有重复出现。也就是说，如果hashcode是相异的，则HashSet会假设对象不可能是相同的。如果hashCode()找到相同的hashcode的对象，它会调用其中一个的equals()来检验hashcode相等的对象是否真的相同。