**1. java中equals方法和“==”的区别：**

equals 方法是 java.lang.Object 类的方法。

有两种用法说明：

（1）对于字符串变量来说，使用“==”和“equals()”方法比较字符串时，其比较方法不同。

“==”比较两个变量本身的值，即两个对象在内存中的首地址。

“equals()”比较字符串中所包含的内容是否相同。

比如：

String s1,s2,s3 = "abc", s4 ="abc" ;

s1 = new String("abc");

s2 = new String("abc");

那么：

s1==s2 是 false //两个变量的内存地址不一样，也就是说它们指向的对象不 一样，

故不相等。

s1.equals(s2) 是 true //两个变量的所包含的内容是abc，故相等。

注意（1）：

如果： StringBuffer s1 = new StringBuffer("a");

StringBuffer s2 = new StringBuffer("a");

结果： s1.equals(s2) //是false

解释：StringBuffer类中没有重新定义equals这个方法，因此这个方法就来自Object类，

而Object类中的equals方法是用来比较“地址”的，所以等于false.

注意（2）：

对于s3和s4来说，有一点不一样要引起注意，由于s3和s4是两个字符

串常量所生成的变量，其中所存放的内存地址是相等的，

所以s3==s4是true（即使没有s3=s4这样一个赋值语句）

（2）对于非字符串变量来说，"=="和"equals"方法的作用是相同的都是用来比较其

对象在堆内存的首地址，即用来比较两个引用变量是否指向同一个对象。

比如：

class A

{

A obj1 = new A();

A obj2 = new A();

}

那么：obj1==obj2是false

obj1.equals(obj2)是false

但是如加上这样一句：obj1=obj2;

那么 obj1==obj2 是true

obj1.equals(obj2) 是true

总之：equals方法对于字符串来说是比较内容的，而对于非字符串来说是比较

其指向的对象是否相同的。

== 比较符也是比较指向的对象是否相同的也就是对象在对内存中的的首地址。

String类中重新定义了equals这个方法，而且比较的是值，而不是地址。所以是true。

关于equals与==的区别从以下几个方面来说：

（1）如果是基本类型比较，那么只能用==来比较，不能用equals

（2）对于基本类型的包装类型，比如Boolean、Character、Byte、Shot、Integer、Long、Float、Double等的引用变量，==是比较地址的，而equals是比较内容的。

**2.浅拷贝，深拷贝和完全拷贝的区别**

答：如果一个类拥有资源，当这个类的对象发生复制过程的时候，资源重新分配，这个过程就是深拷贝，反之，没有重新分配资源，就是浅拷贝。

浅拷贝:拷贝所考虑对象，不拷贝所引用对象；

深拷贝：所引用的都拷贝；

完全拷贝：所有的都拷贝。

**3.单例模式与序列化的比较**

答：单例模式的目的是使得类的一个对象成为系统中的唯一实例。要实现这一点，可以从客户端对其进行实例化开始。因此需要用一种只允许生成对象类的唯一实例的机制，“阻止”所有想要生成对象的访问。使用工厂方法来限制实例化过程。这个方法应该是静态方法；

序列化将对象的状态信息转换为可以存储或传输的形式的过程。在序列化期间，对象将其当前状态写入到临时或持久性存储区。以后，可以通过从存储区中读取或反序列化对象的状态，重新创建该对象。

序列化使其他代码可以查看或修改那些不序列化便无法访问的对象实例数据。通常，对象实例的所有字段都会被序列化，这意味着数据会被表示为实例的序列化数据。这样，能够解释该格式的代码有可能能够确定这些数据的值，而不依赖于该成员的可访问性。类似地，反序列化从序列化的表示形式中提取数据，并直接设置对象状态，这也与可访问性规则无关。

对于任何可能包含重要的安全性数据的对象，如果可能，应该使该对象不可序列化。如果它必须为可序列化的，请尝试生成特定字段来保存不可序列化的重要数据。如果无法实现这一点，则应注意该数据会被公开给任何拥有序列化权限的代码，并确保不让任何恶意代码获得该权限。