==操作比较的是两个变量的值是否相等，对于引用型变量表示的是两个变量在堆中存储的地址是否相同，即栈中的内容是否相同。equals操作表示的两个变量是否是对同一个对象的引用，即堆中的内容是否相同。==比较的是2个对象的地址，而equals比较的是2个对象的内容。

full clone指克隆出的系统和原始系统一模一样，可以独立运行。Full Clone是完全独立的虚拟机，它的执行效率与被克隆虚拟机相同，但与Linked Clone相比，它占用空间多，克隆慢等缺点。

Serializable接口是启用其序列化功能的接口。public interface Serializable类通过实现 java.io.Serializable 接口以启用其序列化功能。未实现此接口的类将无法使其任何状态序列化或反序列化。可序列化类的所有子类型本身都是可序列化的。序列化接口没有方法或字段，仅用于标识可序列化的语义。

深拷贝和浅拷贝是两个概念，不一定谁比谁好，要按照实际的工作来确定如何去拷贝一个对象。如果在数据库操作方面，为了取出一张表时不涉及其他的表，肯定需要使用浅拷贝，而在框架的Serializable中，虽然耗时，但是深拷贝是非常有必要的。

clone顾名思义就是复制， 在Java语言中， clone方法被对象调用，所以会复制对象。所谓的复制对象，首先要分配一个和源对象同样大小的空间，在这个空间中创建一个新的对象。

有浅克隆（shallow clone）和深克隆（deep clone）。所谓的浅克隆，顾名思义就是很表面的很表层的克隆，如果我们要克隆Administrator对象，只克隆他自身以及他包含的所有对象的**引用地址**。而深克隆，就是非浅克隆。克隆除自身以外所有的对象，包括自身所包含的所有**对象实例**。至于深克隆的层次，由具体的需求决定，也有“N层克隆”一说。

原型模式是一种创建型设计模式,它通过复制一个已经存在的实例来返回新的实例,而不是新建实例.被复制的实例就是我们所称的原型,这个原型是可定制的。原型模式多用于创建复杂的或者耗时的实例, 因为这种情况下,复制一个已经存在的实例可以使程序运行更高效,或者创建值相等,只是命名不一样的同类数据。原型模式中的拷贝分为"浅拷贝"和"深拷贝"。浅拷贝: 对值类型的成员变量进行值的复制,对引用类型的成员变量只复制引用,不复制引用的对象。深拷贝: 对值类型的成员变量进行值的复制,对引用类型的成员变量也进行引用对象的复制。

原型（Prototype）模式主要采用的是java的clone克隆，克隆有克隆的前提（即满足的条件），也有深、浅之分。首先，我们来看看克隆满足的条件：1、对任何的对象x，都有 x.clone() != x。即克隆对象与原对象不是同一个对象。2、对任何的对象x，都有 x.clone().getClass() == x.getClass()，即克隆对象与原对象的类型是一样。3、如果对象x的equals()方法定义恰当的话，那么 x.clone().equals(x)应当是成立的。 在java语言的API中，凡是提供了clone()方法的类，都满足上面的这些条件。（一般来说，clone前两个条件必须，后一个可选）。

其次，我们来看看深浅克隆（即拷贝）的问题。浅克隆是指仅仅克隆所考虑的对象，而不克隆它所引用的对象，而深克隆是指把要克隆的对象及其所引用的对象都克隆一遍，这种对引用到的对象的克隆称之为间接克隆。

在创建型模式中，单例（Singleton）模式和原型（Prototype）模式相对来说其用意更为简单明了。单例（Singleton）模式确保某类只有一个实例，且自行实例化并向整个系统提供这个实例；原型（Prototype）模式通过给出一个原型对象来指明所要创建的对象类型，并通过Clone的方式创建出所需的同类型的对象。接下来，我们针对这两种模式的一些常见问题给出简单阐述。

#单例（Singleton）模式

#特点：1、只能有一个实例；2、必须自行创建这个实例；3、必须自行向整个系统提供这个实例；4、因构造方法似有，不可被继承。其中，1是通过构造方法来保障的，2&3是通过工厂方法来实现的。

#使用单例的必要条件（使用场景）：一个系统要求一个类只有一个实例时。

#实现方式及其存在特点：单例（Singleton）模式常被拿来作为笔试考察的，侧重于单例模式的实现（所谓的饿汉、懒汉、注册、双重检查等实现方式优劣以及多类加载器、多JVM情况下Singleton的表现）及其使用场景。