1. 24设计模式-原型模式

```
原型模式概念介绍
原型模式uml图介绍
原型模式-浅克隆
原型模式-深克隆
super.clone()
原型模式-浅克隆代码
原型模式-深克隆代码
```

2. 为什么需要接口Cloneable

```
cloneable其实就是一个标记接口，只有实现这个接口后，然后在类中重写Object中的clone方法，然后通过类调用clone方法才能克隆成功，如果不实现这个接口，则会抛出CloneNotSupportedException(克隆不被支持)异常。
java对象如果想被克隆，它对应的类需要implements标志接口Cloneable。如果不重写clone()方法，则在调用clone()方法实现的是浅复制（所有的引用对象保持不变，意思是如果原型里这些对象发生改变会直接影响到复制对象）。重写clone()方法，一般会先调用super.clone()进行浅复制，然后再复制那些易变对象，从而达到深复制的效果。
```

3. java什么时候会抛出CloneNotSupportedException异常?

```
在另一个包中调用clone且没有实现Cloneable接口的时候
```

4. 流的概念

```
当程序需要读取数据的时候，就会开启一个通向数据源的流。这个数据源可以是文件，内存，或是网络连接。类似的，当程序需要写入数据的时候，就会开启一个通向目的地的流。这时候你就可以想象数据好像在其中“流”动一样。
```

5. 什么是序列化和反序列化

```
序列化的过程，就是一个“freeze”的过程，它将一个对象freeze（冷冻）住，然后进行存储，等到再次需要的时候，再将这个对象de-freeze就可以立即使用。

*序列化是将对象状态转换为可保存或传输的格式的过程。与序列化相对的是反序列化，它将流转换为对象。这两个过程结合起来，可以轻松地存储和传输数据。

通过对象输出流将 对象 首先转换为了一组字节，这个过程称为：对象序列化

对象的输出流将指定的对象写入到文件的过程，就是将对象序列化的过程，对象的输入流将指定序列化好的文件读出来的过程，就是对象反序列化的过程。
```

6. Serializable接口

```
Serializable接口是一个里面什么都没有的接口Serializable接口是启用其序列化功能的接口。
实现java.io.Serializable 接口的类是可序列化的。没有实现此接口的类将不能使它们的任意状态被序列化或逆序列化。
```

7.ByteArrayOutputStream

```
对byte类型数据进行写入的类 相当于一个中间缓冲层，将类写入到文件等其他outputStream。
```

8. ObjectOutputStream

```
对应序列化
该流可以将一个对象写出，或者读取一个对象到程序中
```

9. ByteArrayInputStream

```
是字节数组输入流,在内存中创建了一个字节数组,将输入流中读取的数据保存到字节数组的缓存区中.也就是说字节数组输入流将读取数据放到字节数组缓冲区中.
```

10.ObjectInputStream

```
对应反序列化
```

11.原型模式优缺点

优:

```
简化对象的创建过程，通过复制一个已有实例可以提高新实例的创建效率
扩展性较好
简化创建结构，原型模式中产品的复制是通过封装在原型类中的克隆方法实现的，无须专门的工厂类来创建产品
可以使用深克隆的方式保存对象的状态，以便在需要的时候使用，可辅助实现撤销操作
```

缺:

```
需要为每一个类配备一个克隆方法，而且该克隆方法位于一个类的内部，当对已有的类进行改造时，需要修改源代码，违背了开闭原则
在实现深克隆时需要编写较为复杂的代码，而且当对象之间存在多重的嵌套引用时，为了实现深克隆，每一层对象对应的类都必须支持深克隆，实现起来可能会比较麻烦
```