```
package com.sxt.singleton;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.InputStream;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.io.OutputStream;
import java.lang.reflect.Constructor;

/**
 * 测试反射和反序列化破解单列模式
 * @author 江
 *
 */

public class Client02 {
   public static void main(String[] args) throws Exception {
        SingletonDemo06 s1=SingletonDemo06.getInstance();
        SingletonDemo06 s2=SingletonDemo06.getInstance();

        System.err.println(s1);
        System.err.println(s2);


        //通过反射的方式直接调用私有构造器
        System.err.println("通过反射的方式直接调用私有构造器");
        Class<SingletonDemo06> clazz=
(Class<SingletonDemo06>)Class.forName("com.sxt.singleton.SingletonDemo06");
        Constructor<SingletonDemo06> c=clazz.getDeclaredConstructor();
        c.setAccessible(true);

        SingletonDemo06 s3=c.newInstance();
        SingletonDemo06 s4=c.newInstance();
```

```
        System.err.println(s3);
        System.err.println(s4);


        //通过反序列化的方式构造多个对象
        System.err.println("通过反序列化的方式构造多个对象");
        OutputStream fos=new FileOutputStream(new
File("D:\\workspace\\Pattern\\a.txt"));
        ObjectOutputStream oos=new ObjectOutputStream(fos);
        oos.writeObject(s1);
        oos.close();
        fos.close();

        InputStream fis=new FileInputStream(new
File("D:\\workspace\\Pattern\\a.txt"));
        ObjectInputStream ois=new ObjectInputStream(fis);

        SingletonDemo06 s5=(SingletonDemo06)ois.readObject();
        System.err.println(s5);



}
}
```