```
package com.sxt.others;
/**
 * 单例模式(DCL)：懒汉式套路基础上加入并发控制，保证在多线程环境下，对外存在一个对象
 * 1.构造器私有化-->避免外部new对象
 * 2.提供私有的静态属性-->存储对象的地址
 * 3.提供公共的静态方法-->获取属性
 * @author 江
 *
 */

public class DoubleCheckedLocking {
	//2.提供私有的静态属性
	//没有volatile其他线程可能访问一个没有初始化的对象
	private static volatile DoubleCheckedLocking instance;
	//1.构造器私有化
	private DoubleCheckedLocking() {

	}
	//3.提供公共的静态方法-->获取属性
	public static  DoubleCheckedLocking getInstance() {
	  //再次检测
		if(null!=instance) {  //避免不必要的同步,已经存在对象
			return instance;
		}
	  synchronized(DoubleCheckedLocking.class) {
		if(null==instance) {
			instance=new DoubleCheckedLocking();
			//1.开辟空间
			//2.初始化对象信息
			//3.返回对象的地址给引用
		}
	  }
		return instance;
	}
```

```
    public static void main(String[] args) {
                Thread t=new Thread(()->{
                        System.err.println(DoubleCheckedLocking.getInstance());
                }) ;
                t.start();
                System.err.println(DoubleCheckedLocking.getInstance());
        }
}
```