K:首先解释一下为什么fina可以保证对象正确的初始化，首先要知道原因。因为：

A a = new A() 这个语句需要两步才能执行完毕。就是这两步是有可能发生重排序。而final在可以达到不重排序的目的。更何况远远不止两步。（内存分配，初始化对象，设置实例指向内存空间，初次访问对象）

Q：回想一下，static是如何实现正确初始化的？或者说类初始化方案是如何成立的

A：解决思路就是jvm级别保证的类初始化，同时也叫作基于类初始化的静态方案

基于类的初始化方案：

第一种非延迟初始化方案，直接使用静态域并new，触发类的初始化，jvm会保证初始化正确，但是这种问题就是如果有些域不用就会浪费，一般我们会选择延迟初始化。如果仅仅是在静态方法里new，

|  |
| --- |
| public class SingleTest {  private static CaptureImgBean *captureImgBean*;  private CaptureImgBean getInstance(){  *captureImgBean* = new CaptureImgBean();  return *captureImgBean*;  } }  延迟初始化可以做到，但是每次调用都会new一个对象，如果想要单例可以这样改进：  public class SingleTest {  private static CaptureImgBean *captureImgBean*;  private static CaptureImgBean getInstance(){  if(*captureImgBean* == null) {  *captureImgBean* = new CaptureImgBean();  }  return *captureImgBean*;  } }  这种有什么问题呢？当你判断为空后，这个时候可能另一个线程正好初始化完，不管加多少if(*captureImgBean* == null)都有可能这种情况  继续改进：  public class SingleTest {  private static CaptureImgBean *captureImgBean*;  private static CaptureImgBean getInstance(){  if(*captureImgBean* == null) {  synchronized (SingleTest.class) {  *captureImgBean* = new CaptureImgBean();  }  }  return *captureImgBean*;  } }  这个在进入锁后还是有可能已经不为空  public class SingleTest {  private static CaptureImgBean *captureImgBean*;  private static CaptureImgBean getInstance(){  if(*captureImgBean* == null) {  synchronized (SingleTest.class) {  if(*captureImgBean* == null) {  *captureImgBean* = new CaptureImgBean();  }  }  }  return *captureImgBean*;  } }  双重检查，但是还有问题，由于重排序，可能虽然不为空，但是并没有正常的初始化  这一点我不理解，因为既然静态变量，那么必然是结果类初始化的，为什么会出现未初始化完全的对象呢？除非*captureImgBean不是一个静态变量，这样我可以理解，但是这里不是，我认为作者说的不对。而且这样我可以推翻基于类类初始化创建对象的方案。如果他是对的，有一种解释，延迟加载场景下，并没有类初始化一说，而且你也看到了，基于类初始化的方案也是用到了静态类。*  *目前的解释：静态属性在方法中进行new，并不会受到jvm锁的保证。只有这样的解释才是行得通。*  *为了保证其能够正确初始化：*  public class SingleTest {  private static volatile CaptureImgBean *captureImgBean*;  public static CaptureImgBean getInstance(){  System.*err*.println("getInstance");  if(*captureImgBean* == null) {  synchronized (SingleTest.class) {  if(*captureImgBean* == null) {  *captureImgBean* = new CaptureImgBean();  }  }  }  return *captureImgBean*;  } }  有人会问问什么不用final，因为final在这里编译不会通过，final会要求声明时就进行初始化。  然后再说下基于类初始化的方案：  public class SingleTest {  static class InstanceHolder{  static CaptureImgBean *captureImgBean* = new CaptureImgBean();  }  public static CaptureImgBean getInstance(){  return InstanceHolder.*captureImgBean*;  } } |

Q:到底应该锁什么？还是锁什么都可以？