JVM第五天-指令乱序性

**乱序性**

实际上，如果两个语句操作之间不存在依赖关系，是有可能会被乱序执行的，这是因为指令重排序的原因。

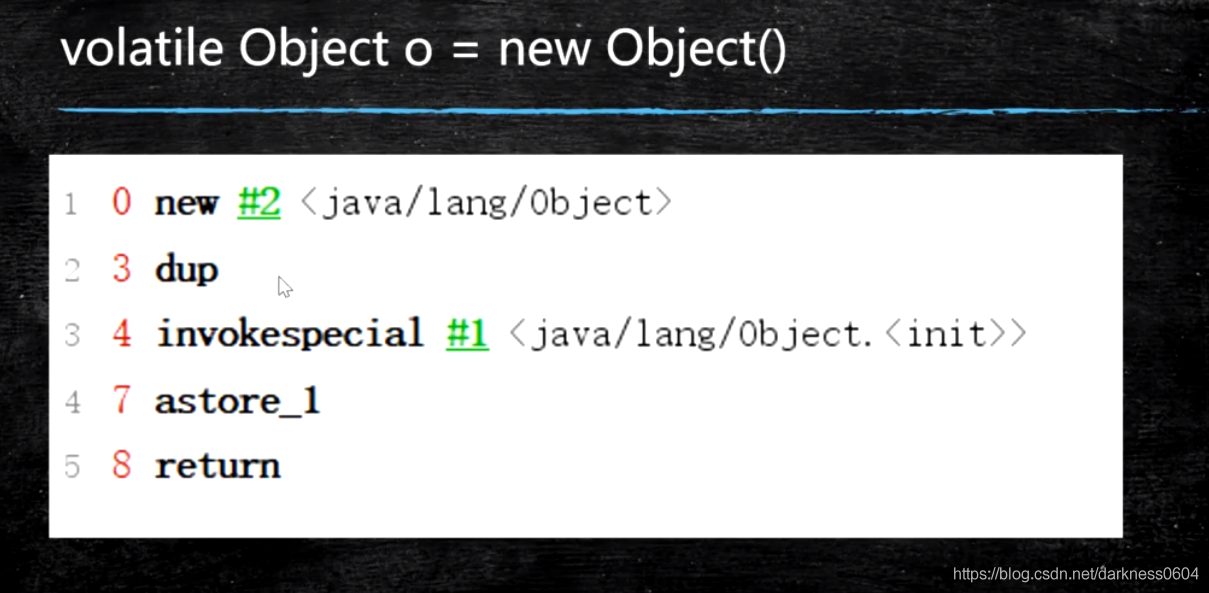
这也意味着，一个对象的创建在多线程情况下是可能有问题的。这里我们拿设计模式中的单例模式举例：

|  |
| --- |
| public class Mgr06 {  private static Mgr06 INSTANCE;  private Mgr06() {  }  public static Mgr06 getInstance() {  if (INSTANCE == null) {  //双重检查  synchronized (Mgr06.class) {  if(INSTANCE == null) {  try {  Thread.sleep(1);  } catch (InterruptedException e) {  e.printStackTrace();  }  INSTANCE = new Mgr06();  }  }  }  return INSTANCE;  }  public void m() {  System.out.println("m");  }  public static void main(String[] args) {  for(int i=0; i<100; i++) {  new Thread(()->{  System.out.println(Mgr06.getInstance().hashCode());  }).start();  }  }  } |

这是单例模式中double check的典型实现，这个代码有没有问题？

**分析原因**

其实会有，原因要从一个对象的new的细节来说



看上面图，一个对象new的语句 Object o=new Object()，它被解释成指令的时候，实际上会解释成上面图的多条指令。

我们关注重点指令：

指令0是向内存申请这个对象需要的空间，然后给对象赋予默认值。

指令4是执行这个对象的属性值的初始化操作

指令7是把对象的引用赋予给局部变量o

这样看是没问题的，但如果发生了指令重排序，指令7先执行，指令4后执行的呢？

那是不是意味着某个时刻是有可能得到一个半初始化状态的对象呢（只有默认值）？

回到之前的单例代码中，看INSTANCE = new Mgr06();这句话，如果指令被重排序，那么在多线程情况下，另外一个线程很可能会得到一个半初始化状态的对象！

**解决方案**

有没有解决方法？ 自然是有的，只需要给属性值赋予volatile的修饰，就可以避免指令重排序了。

|  |
| --- |
| private static volatile Mgr06 INSTANCE; |