多线程编程实战篇秘籍(二) PDF转换可能丢失图片或格式,建议阅读原文 https://www.100test.com/kao_ti2020/145/2021_2022__E5_A4_9A_E7_BA_BF_E7_A8_8B_E7_c104_145442.htm 多线程编程的设计模式 不变模式(二) 不变模式(Immutable Pattern)顾名思义,它的状态在它的生命周期内是永恒的(晕,永恒的日月星晨,对象如人,太渺小,谈不上永恒!),不会改变的.对于其中的不变类(Immutable Class),它的实例可以在运行期间保持状态永远不会被改变,所以不需要采取共享互斥机制来保护,如果运用得当可以节省大量的时间成本. 请注意上面这段话,不变模式其中的不变类,说明不变类只是不变模式中一个组成部分,不变类和与之相辅的可变类,以及它们之间的关系才共同构成不变模式!所以在涉及不变模式的时候一定要研究一个类是不变的还是可变的(Mutable).在jdk中的String类和StringBuffer类就组成了一个不变模式.还是先看具体的例子:final class Dog{ private final String name. private final int age. public Dog(String name,int age){ this.name = name. this.age = age. } public String getName(){return this.name.} public int getAge(){return this.age.} public String toString(){ return "Dogs name = " this.name ",age = " this.age. }}1.Dog类本身被声明为final,可以保证它本身的状态不会被子类扩展方法所改变.2.Dog类的所有成员变量都是final的,保证它在构造后不会被重新赋值.而且Dog类所有属性是private的,只提供getter访问.3.Dog类的能传入的参数本身是Immutable的.这一点非常重要将在下面具体说明.以上条件都是必要条件,而不是充要条件.class DisplayDog extends Thread{ private Dog dog. public DisplayDog(Dog dog){ this.dog = dog. } public void run(){

while(true){ System.out.println(this.getName() " display: " dog). } }}DisplayDog类是把一个Dog对象传入后,不断显示这个dog的属性.我们会同时用多个线程来显示同一dog对象,看看它们在共享同一对象时对象的状态:public class Test { public static void main(String[] args) throws Exception { Dog dog = new Dog("Sager",100). new DisplayDog(dog).start(). new DisplayDog(dog).start(). new DisplayDog(dog).start(). }}运行这个例子你可以等上一个月,虽然运行一年都正常并不能说明第366天不出现异常,但我们可以把这样的结果认为是一种说明.多个线程共享一个不变类的实例时,这个实例的状态不会发生改变.事实上它没有地方让你去改变.在临界区模式中有些操作必须只允许有一个线程操作,而这个类本身以及对它的访问类中并不需要进行临界区保护,这就让多个线程不必等待从而提高了性能.既然有这么好的优势,那我们在需要临界区保护的对象为什么不都设计成不变类呢?1.不变类设计起来有一定难度.对于上面这个用来示例的Dog,由于其本身的属性,方法都很简单,我们还可以充分地考虑到可以改变它状态的各种情况.但对于复杂的类,要保证它的不变性,是一个非常吃力的工作. 不变类中,任何一个必要都件都不是充要条件,虽然连老骨灰都没有这么说过,但我还是要真诚地目光深邃语气凝重地告诉你.没有任何条件是充要条件的意思就是如果任何一个必要条件你没考虑到,那它就会无法保证类的不可变性.没有规范,没有模板,完全看一人设计人员的经验和水平.也许你自以为考虑很全面的一个"不变类"在其他高手面前轻而易举地就"可变"了.2.不变类的种种必要条件限制了类设计的全面性,灵活性.这点不用多说,简单说因为是不变类,所以你不能A,因为是不变类,你不能B.当

然,如果你是一人很有经验的设计者,你能成功地设计一个不变类,但因为它的限制而失去一些功能,你就要以使用与之相辅的可变类.并且它们之间可以相互转换,在需要不变性操作的时候以不变类提供给用户,在需要可变性操作的时候以可变类提供给用户.在jdk中String被设计为不可变类,一旦生成一个String对象,它的所有属性就不会被变,任何方法要么返回这个对象本身的原始状态,要么抛弃原来的字符串返回一个新字符串,而绝对不会返回被修改了的字符串对象.但是很多时候返回新字符串抛弃原来的字符串对象这样的操作太浪费资源了.特别是在循环地操作的时候: String s = "Axman". for(int i=0.i那么这种时候需要将原始的不变的s包装成可变的StringBuffer来操作,性能的改变可能是成千上万倍: StringBuffer sb = new StringBuffer(s). //将不变的String包装成可变的String. for(int i=0.i sb.append("x"). s = new String(sb). //将可变类封装成不变类.虽然可以调用toString(),但那不是可变到不变的转换. 