多线程编程的设计模式不变模式(二) PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/144/2021_2022__E5_A4_9A_E7_BA_BF_E7_A8_8B_E7_c104_144591.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. 