JAVA认证:在Java中应用设计模式SingletonJava认证考试 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/583/2021_2022_JAVA_E8_AE_A4_E8_AF_81_c104_583594.htm 基本概念 Singleton 是一种创建性模型,它用来确保只产生一个实例,并提供一个访问它的全局访问点.对一些类来说,保证只有一个实例是很重要的,比如有的时候,数据库连接或 Socket 连接要受到一定的限制,必须保持同一时间只能有一个连接的存在.再举个例子,集合中的 set 中不能包含重复的元素,添加到set里的对象必须是唯一的,如果重复的值添加到 set,它只接受一个实例.JDK中正式运用了Singleton模式来实现 set 的这一特性,大家可以查看java.util.Collections里的内部静态类SingletonSet的原代码.其实Singleton是最简单但也是应用最广泛的模式之一,在 JDK 中随处可见. 简单分析 为了实现 Singleton 模式,我们需要的是一个静态的变量,能够在不创建对象的情况下记忆是否已经产生过实例了.静态变量或静态方法都可以在不产生具体实例的情况下直接调用,这样的变量或方法不会因为类的实例化而有所改变.在图1的结构中可以看到,uniqueInstance 就是这个独立的静态变量,它可以记忆对象是否已经实例化了,在静态方法 Instance 中对这个变量进行判断,若没有实例化过就产生一个新的对象,如果已经实例化了则不再产生新的对象,仍然返回以前产生的实例. 图1: Singleton 模式结构 具体实施 实现 Singleton 模式的办法通常有三种. 一. 用静态方法实现 Singleton 这种方法是使用静态方法来监视实例的创建.为了防止创建一个以上的实例,我们最好把构造器声明为 private. 这样可以防止客户程序员通过除由我们提供的方法之外的任意方式来创建一个实

例,如果不把构造器声明为private,编译器就会自作聪明的自动同步一个默认的friendly构造器.这种实现方法是最常见的,也就是图1中结构的标准实现. 1 public class Singleton { 2 3 private static Singleton s. 4 5 private Singleton(){}. 6 7 /** 8 9 * Class method to access the singleton instance of the class. 10 11 */ 12 13 public static Singleton getInstance() { 14 15 if (s == null) 16 17 s = new Singleton(). 18 19 return s. 20 21 } 22 23 } 24 25 // 测试类 26 27 class singletonTest { 28 29 public static void main(String[] args) { 30 31 Singleton s1 = Singleton.getInstance(). 32 33 Singleton s2 = Singleton.getInstance(). 34 35 if (s1==s2) 36 37 System.out.println("s1 is the same instance with s2"). 38 39 else 40 41 System.out.println("s1 is not the same instance with s2"). 42 43 } 44 45 } 46 47 singletonTest运行结果是: s1 is the same instance with s2 这证明我们只创建了一个实例. 二. 以静态变量为标志实现 Singleton 在类中嵌入一个静态变量做为标志,每次都在进入构造器的时候进行检查.