1. 双向加锁功能的孤子模式

Code：

DoubleCheckedLockingSingleton.java

public class DoubleCheckedLockingSingleton {  
 // 使用 volatile 关键字确保可见性和禁止指令重排序  
 private static volatile DoubleCheckedLockingSingleton *instance*;  
  
 // 私有构造函数，防止外部实例化  
 private DoubleCheckedLockingSingleton() {}  
  
 // 公共静态方法，用于获取单例实例  
 public static DoubleCheckedLockingSingleton getInstance() {  
 // 第一次检查，避免不必要的同步  
 if (*instance* == null) {  
 // 同步块，确保线程安全  
 synchronized (DoubleCheckedLockingSingleton.class) {  
 // 第二次检查，防止多个线程同时通过第一次检查  
 if (*instance* == null) {  
 *instance* = new DoubleCheckedLockingSingleton();  
 }  
 }  
 }  
 return *instance*;  
 }  
}

1. 具有可变用例数目的孤子模式

Code：

VariableInstancesSingleton.java

import java.util.ArrayList;  
import java.util.List;  
import java.util.concurrent.ThreadLocalRandom;  
  
public class VariableInstancesSingleton {  
 // 最大实例数  
 private static final int *MAX\_INSTANCES* = 3;  
 // 存储单例实例的列表  
 private static final List<VariableInstancesSingleton> *instances* = new ArrayList<>();  
  
 // 静态代码块，初始化实例列表  
 static {  
 for (int i = 0; i < *MAX\_INSTANCES*; i++) {  
 *instances*.add(new VariableInstancesSingleton());  
 }  
 }  
  
 // 私有构造函数，防止外部实例化  
 private VariableInstancesSingleton() {}  
  
 // 公共静态方法，用于获取随机的单例实例  
 public static VariableInstancesSingleton getInstance() {  
 int randomIndex = ThreadLocalRandom.*current*().nextInt(*MAX\_INSTANCES*);  
 return *instances*.get(randomIndex);  
 }  
}

测试代码：

Main.java

public class Main {  
 public static void main(String[] args) {  
 // 测试双向加锁的孤子模式  
 DoubleCheckedLockingSingleton singleton = DoubleCheckedLockingSingleton.*getInstance*();  
 System.*out*.println(singleton);  
  
 // 测试可变用例数目的孤子模式  
 VariableInstancesSingleton instance = VariableInstancesSingleton.*getInstance*();  
 System.*out*.println(instance);  
 }  
}

运行结果：







