线程安全的单例模式

单例模式是一种常见的软件设计模式，核心思想是一个类只有一个实例对象。

JVM：栈内存、堆内存

单线程模式下的单例模式

```
package com.southwind.test;

public class SingletonDemo {

  private static SingletonDemo singletonDemo;

  private SingletonDemo() {
    System.out.println("创建了SingletonDemo...");
  }

  public static SingletonDemo getInstance() {
    if(singletonDemo == null) {
      singletonDemo = new SingletonDemo();
    }
    return singletonDemo;
  }

}
```

多线程模式下的单例模式

```
package com.southwind.test;

public class SingletonDemo {

  private static SingletonDemo singletonDemo;

  private SingletonDemo() {
    System.out.println("创建了SingletonDemo...");
  }

  public synchronized static SingletonDemo getInstance() {
    if(singletonDemo == null) {
      singletonDemo = new SingletonDemo();
    }
    return singletonDemo;
  }

}
```

双重检测，synchronized 修饰代码块。

1、线程同步是为了实现线程安全，如果只创建一个对象，那么线程就是安全的。

2、如果 synchronized 锁定的是多个线程共享的数据（同一个对象），那么线程就是安全的。

3、

```
package com.southwind.test;

public class SingletonDemo {

  private volatile static SingletonDemo singletonDemo;

  private SingletonDemo() {
    System.out.println("创建了SingletonDemo...");
  }

  public static SingletonDemo getInstance() {
    if(singletonDemo == null) {
      synchronized (SingletonDemo.class) {
        if(singletonDemo == null) {
          singletonDemo = new SingletonDemo();
        }
      }
    }
    return singletonDemo;
  }

}
```

volatile 的作用时候可以使内存中的数据对象线程可见。

主内存对线程是不可见的，添加 volatile 关键字之后，主内存对线程可见。