单例设计模式就是采取一定的方法保证在整个的软件系统中，对某个类**只能存在一个对象实例**，并且该类**只提供一个取得其对象实例的方法**。如果我们要让类在一个虚拟机中只能产生一个对象，我们首先必须**将类的构造方法的访问权限设置为private**，这样，就不能用new操作符在类的外部产生类的对象了，但在类内部仍可以产生该类的对象。因为在类的外部开始还无法得到类的对象，只能**调用该类的某个静态方法**以返回类内部创建的对象，静态方法只能访问类中的静态成员变量，所以，指向类内部产生的**该类对象的变量也必须定义成静态的**。

* **单例模式——饿汉式**

/\*\*  
 \* 设计模式  
 \* <p>  
 \* 单例设计模式  
 \* 1、解决的问题：使得一个类只能创建一个对象  
 \* 2、如何实现  
 \* 饿汉式  
 \*/  
public class TestSingleton {  
  
 public static void main(String[] args) {  
 Singleton s1 = Singleton.getInstance();  
 Singleton s2 = Singleton.getInstance();  
 System.***out***.println(s1 == s2);  
 }  
}  
  
/\*\*  
 \* 只能创建Singleton的单个实例。  
 \*/  
class Singleton {  
  
 /\*\*  
 \* （1）私有化构造器,使得在类的外部不能调用此构造器。  
 \*/  
 private Singleton() {  
  
 }  
  
 // （2）在类内部创建一个类的实例。  
 private static Singleton instance = new Singleton();  
  
 /\*\*  
 \* （3）私有化此对象，通过公共的方法来调用。  
 \* （4）此公共方法,只能通过类来调用，因此设置为static的，同时类的实例也必

\* 须为static。  
 \*/  
 public static Singleton getInstance() {  
 return instance;  
 }  
}

* **单例模式——懒汉式 可能存在线程安全问题（讲到多线程时可修复）**

/\*\*  
 \* 懒汉式  
 \*/  
public class TestSingleton1 {  
  
 public static void main(String[] args) {  
 Singleton1 s1 = Singleton1.getInstance();  
 Singleton1 s2 = Singleton1.getInstance();  
 System.***out***.println(s1 == s2);  
 }  
}  
  
/\*\*  
 \*  
 \*/  
class Singleton1 {  
  
 /\*\*  
 \* （1）私有化构造器  
 \*/  
 private Singleton1() {  
  
 }  
  
 /\*\*  
 \* （2）声明类的引用  
 \* （4）需要配合static的方法，用static修饰此类的引用。  
 \*/  
 private static Singleton1 instance = null;  
  
 /\*\*  
 \* （3）设置公共的方法来访问类的实例  
 \*/  
 public static Singleton1 getInstance() {  
 // 如果类的实例未创建，那就先创建，然后返回给调用者。  
 if (instance == null) {  
 instance = new Singleton1();  
 }  
 // 若有了类的实例，直接返回给调用者。  
 return instance;  
 }  
}