<https://github.com/>

设计模式（Design pattern）：通常被有经验的面向对象的软件开发人员使用

通俗说就是解决固定的问题的一般套路，模型化的解题步骤，是经过总结的，非常好使用的。

Java有23种设计模式，当然，设计模式有它自己的原则。六大原则。

单例模式（Singleton Pattern）：一个类只有一个实例化的对象，在该类的内部实例化一个对象，并将其设为静态私有，并将构造方法私有化，以确保其他函数不能实列化此类的新的对象，然后提供一个公共的接口以便于其他方法调用该类的唯一对象。在设置变量和方法时都是静态的。

优点：剩内存。

单例模式的几种实现方式：

1 懒汉式（线程不安全，不支持多线程）

**class** Single{

**private** **static** Single *instance*;

**private** Single(){}

**public** **static** Single getInstance(){

**if**(*instance*==**null**){

Single instance=**new** Single();

}

**return** *instance*;

}

}

2 懒汉式（线程安全，支持多线程）

**class** Single{

**private** **static** Single *instance*;

**private** Single(){}

**public** **static** **synchronized** Single getInstance(){

**if**(*instance*==**null**){

*instance*=**new** Single();

}

**return** *instance*;

}

}

优点：第一次调用时才初始化，避免了内存浪费。

3 饿汉式（推荐使用）

缺点：加载时就初始化浪费内存。

**class** Single{

**private** **static** **int** *age*=10;

**private** **static** String *name*="abc";

**private** **static** **final** Single ***instance***=**new** Single();

**public** **static** Single getInstance(){

**return** ***instance***;

}

}

不new对象却想调用类中的相应方法，该方法肯定是静态的，静态方法只能调用静态成员，所以成员也是静态的。在类中创建的唯一的一个实例化的对象，也需要时public static‘的否则通过类名.变量名就可以在其他方法中调用，造成不可控，其他方法只能通过其提供的public static的方法来调用实例化的唯一的对象，想让该类实例化的对象共享定义的变量，即只能实例化一一公共的变量，无非是变量名不一样，调用的东西还是一样的，static，不想让其他方法能直接访问，private。

饿汉式：在定义的同时实例化并用final修饰

懒汉式：在第一次调用方法时初始化。

步骤总结：设计模式--单例模式

体会：博客里查最好找3个左右的文章，避免低概率的错误。多看。

发现的问题：

解决办法：