1，设计模式：

是在大量的实践中总结和理论化之后优选的代码结构、编程风格以及解决问题的思考方式

单例设计模式：

2，采取一定的方法保证在整个的软件系统中，对某个只能存在一个对象的实例，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，就不能使用new关键字在类的外部产生类的对象但是在类的内部仍然可以创建一个新的对象，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内部类创建的对象，静态方法只能访问类中静态成员变量，所以，指向类内部产生的该类对象的变量也必须是定义成为一个静态变量

如何解决使得一个类只能够创建一个对象：

①私有化构造器，使得在类的外部不能调用此构造器

②在类的内部创建一个类的实例

③私有化对象，通过公共的方法来调用

④此公共的方法，只能通过类来调用，因为设置为static的，同时的实例也必须为static声明的

1，

```
//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够调用此构造器
    private Singleton(){

    }
    //2.在类的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.此公共的方法，只能通过类来调用，因为设置为static的,同时类的实例也必须为static声明的
    public static Singleton getInstance(){
        return instance;
    }
```

2，可能存在安全问题的时候

```java
//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);

    }
}
```

```java
class Singleton1{
    //1.
    private Singleton1(){

    }
    //2.
    private static Singleton1 instance = null;
    //3.
    public static Singleton1 getInstance(){

        if(instance == null){

            instance = new Singleton1();
        }
```