**饿汉式**是立即加载的方式，无论是否会用到这个对象，都会加载。  
如果在构造方法里写了性能消耗较大，占时较久的代码，比如建立与数据库的连接，那么就会在启动的时候感觉稍微有些卡顿。  
  
**懒汉式**，是延迟加载的方式，只有使用的时候才会加载。 并且有[线程安全](https://how2j.cn/k/thread/thread-synchronized/355.html" \l "step793)的考量，使用懒汉式，在启动的时候，会感觉到比饿汉式略快，因为并没有做对象的实例化。 但是在第一次调用的时候，会进行实例化操作，感觉上就略慢。  
  
看业务需求，如果业务上允许有比较充分的启动和初始化时间，就使用饿汉式，否则就使用懒汉式

何为单例模式

1. 构造方法私有化
2. 静态属性指向实例

3. public static的 getInstance方法，返回第二步的静态属性

单例模式又叫做 Singleton模式，指的是一个类，在一个JVM里，只有一个实例存在。

package charactor;

public class GiantDragon {

//私有化构造方法使得该类无法在外部通过new 进行实例化

private GiantDragon(){

}

//准备一个类属性，指向一个实例化对象。 因为是类属性，所以只有一个

private static GiantDragon instance = new GiantDragon();

//public static 方法，提供给调用者获取12行定义的对象

public static GiantDragon getInstance(){

return instance;

}

}

package charactor;

public class GiantDragon {

//私有化构造方法使得该类无法在外部通过new 进行实例化

private GiantDragon(){

}

//准备一个类属性，用于指向一个实例化对象，但是暂时指向null

private static GiantDragon instance;

//public static 方法，返回实例对象

public static GiantDragon getInstance(){

//第一次访问的时候，发现instance没有指向任何对象，这时实例化一个对象

if(null==instance){

instance = new GiantDragon();

}

//返回 instance指向的对象

return instance;

}

}