单例模式的四种实现方式

> 原本单纯的我一直认为这世界上的单例模式，只有饿汉和懒汉呢，今天发现了，原来单例模式有四种实现方式。

饿汉模式

```
public class Singleton {
    /**
     * 饿汉式
     */
    private Singleton() {

    }

    private static final Singleton SINGLETON = new Singleton();

    public static Singleton getInstance(){
        return SINGLETON;
    }

    public void system(){
        System.out.println("---lin---> singleton");
    }

}
```

懒汉模式

```
public class Singleton2 {
    /**
     * 懒汉式
     */
    private Singleton2() {

    }

    private static Singleton2 singleton2 = null;

    public static Singleton2 getInstance() {
        if (singleton2 == null) {
            synchronized (Singleton.class) {
                if (singleton2 == null) {
                    singleton2 = new Singleton2();
                }
            }
        }
        return singleton2;
    }

    public void system(){
        System.out.println("---lin---> singleton2");
    }
}
```

枚举模式

```
public enum Singleton3 {
    INSTANCE;
    private Singleton3(){

    }

    public void system(){
        System.out.println("---lin---> singleton3");
    }
}
```

Holder模式

```
public class Singleton4 {
    /**
     * 带有Holder的方式
     * 类级内部类，也就是静态的成员内部类，该内部类的实例与外部类的实例没有绑定关系
     * 只有被调用的时候才会装在，从而实现了延迟加载
     */
    private Singleton4() {

    }

    private static class SingletonHolder {
        /**
         * 静态初始化器，由JVM来保证线程安全
         */
        public static final Singleton4 INSTANCE = new Singleton4();
    }

    public static Singleton4 getInstance() {
        return SingletonHolder.INSTANCE;
    }

    public void system() {
        System.out.println("---lin---> singleton4");
    }
}
```