单例(Singleton)

波波老师~研发总监/资深架构师

波波微课

spring2go.com

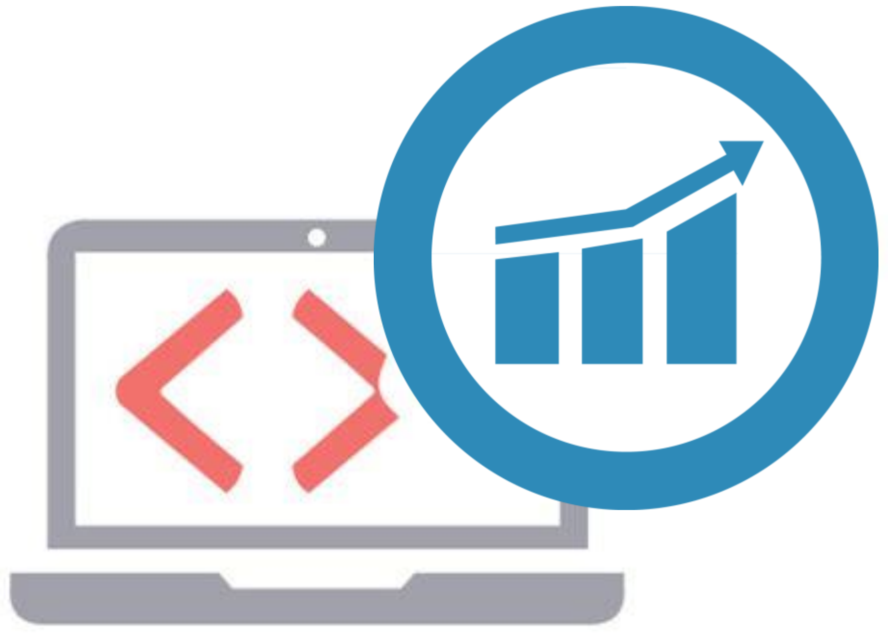
单例定义

- 单例模式限制类的实例化，保证在一个JVM中类的实例只有一个。
- 单例类提供一个可以全局访问单例的入口。
- 常用于
  - 日志logging
  - 配置设置类
  - 驱动程序对象
  - 缓存对象
  - 线程池对象

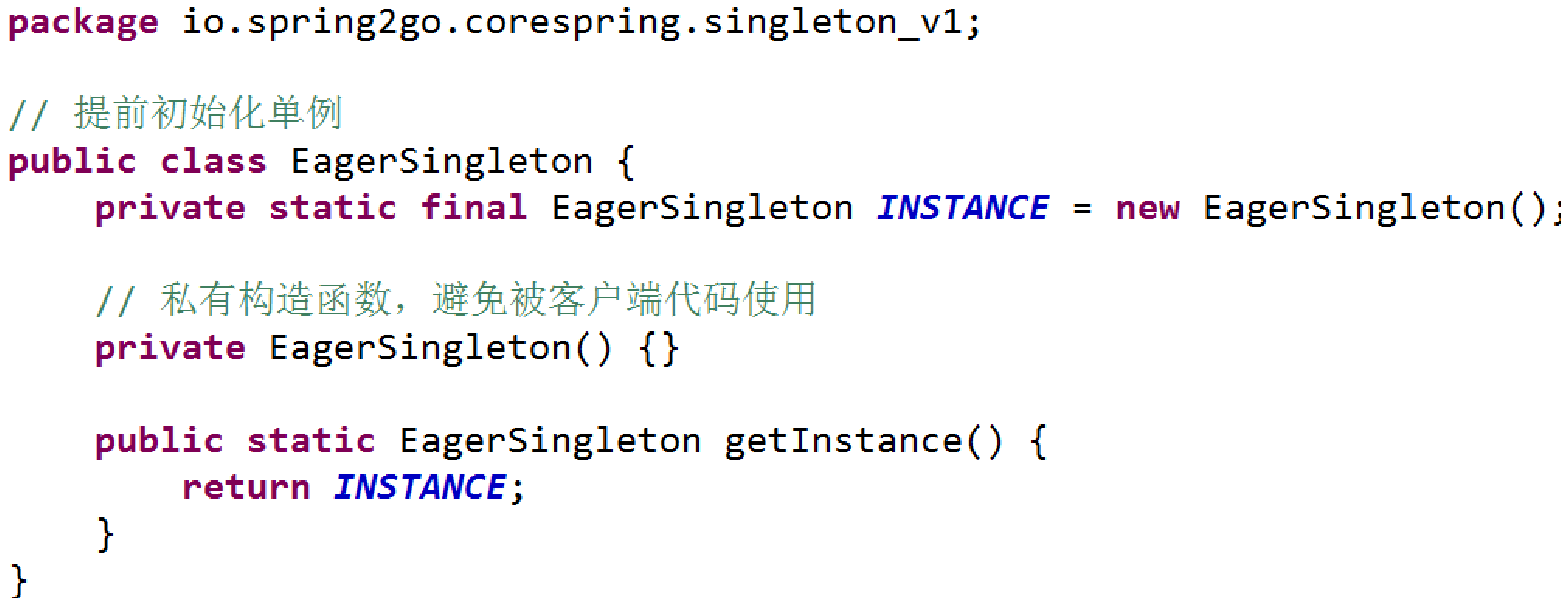
Top 1 Java面试题

- 提前(eager)和延迟(lazy)初始化
- 静态初始化块
- 多线程安全
- 双重检查锁定(double checked locking)
- JVM/JIT指令优化重排
- 反射(reflection)
- 序列化(serialization)
- 克隆(clone)

代码优化之旅

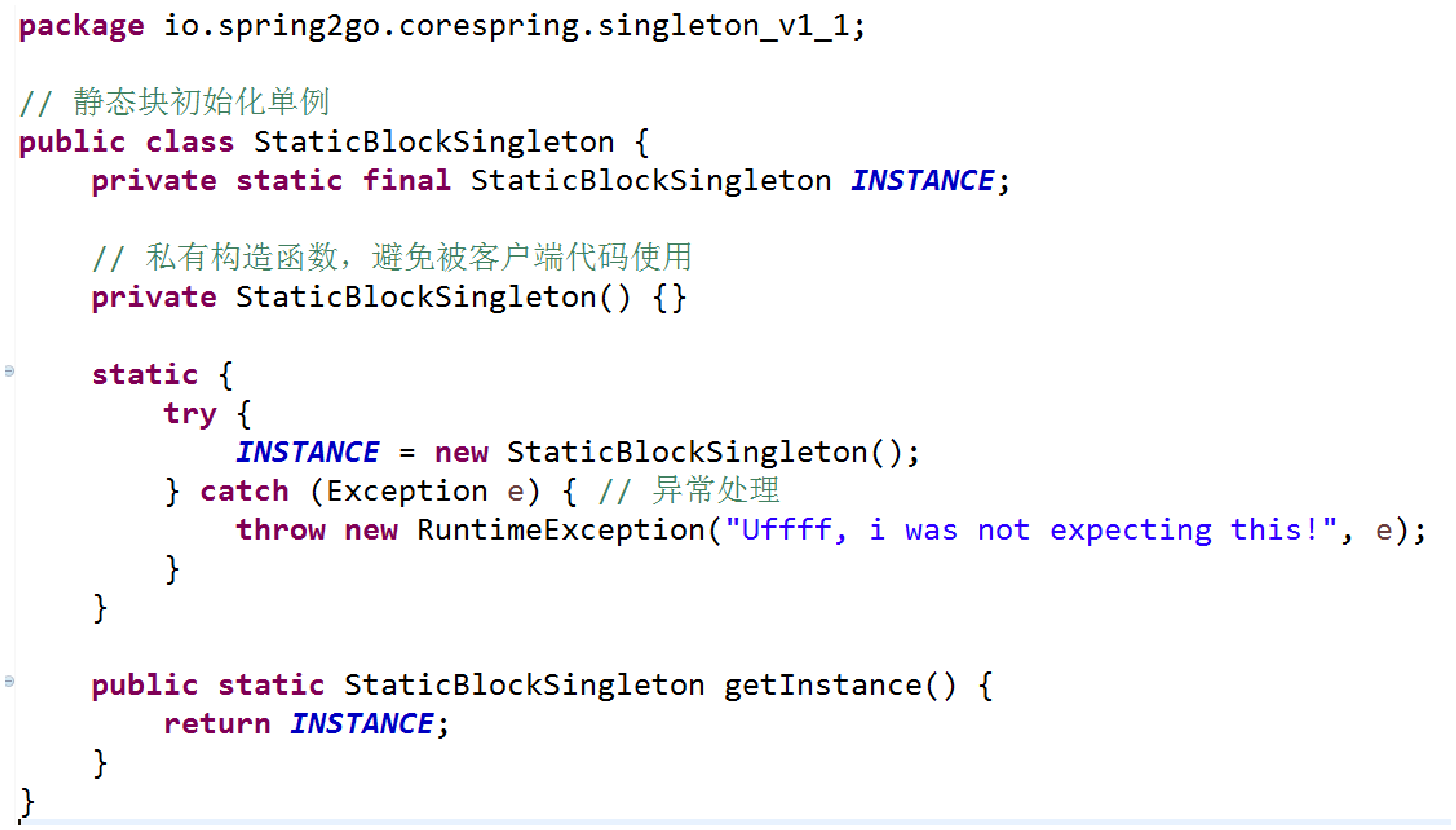


单例1.0~提前(Eager)初始化单例



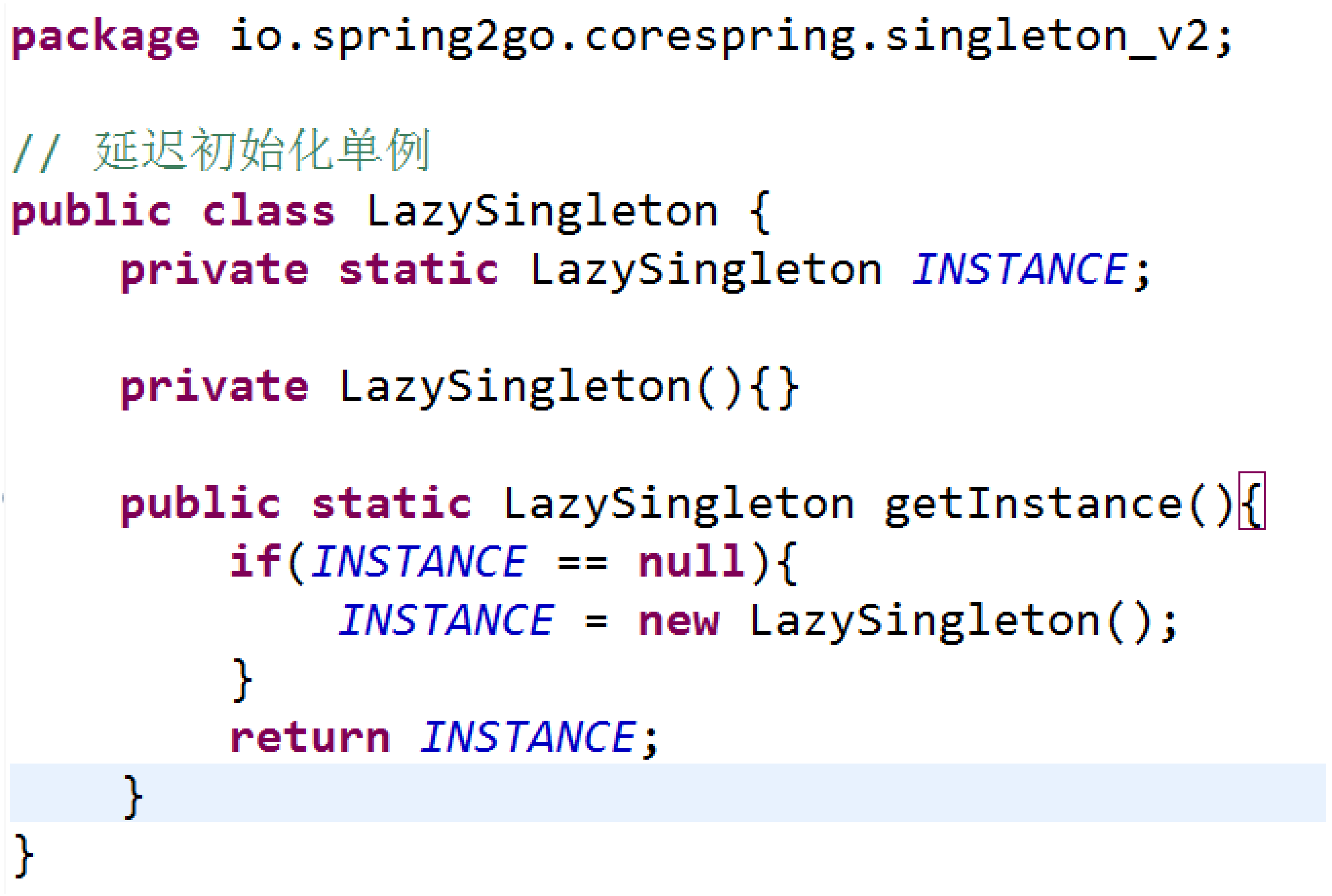
```java
package io.spring2go.corespring.singleton_v1;

// 提前初始化单例
public class EagerSingleton {
    private static final EagerSingleton INSTANCE = new EagerSingleton();

    // 私有构造函数，避免被客户端代码使用
    private EagerSingleton() {}

    public static EagerSingleton getInstance() {
        return INSTANCE;
    }
}
```

单例1.1~静态初始化块单例



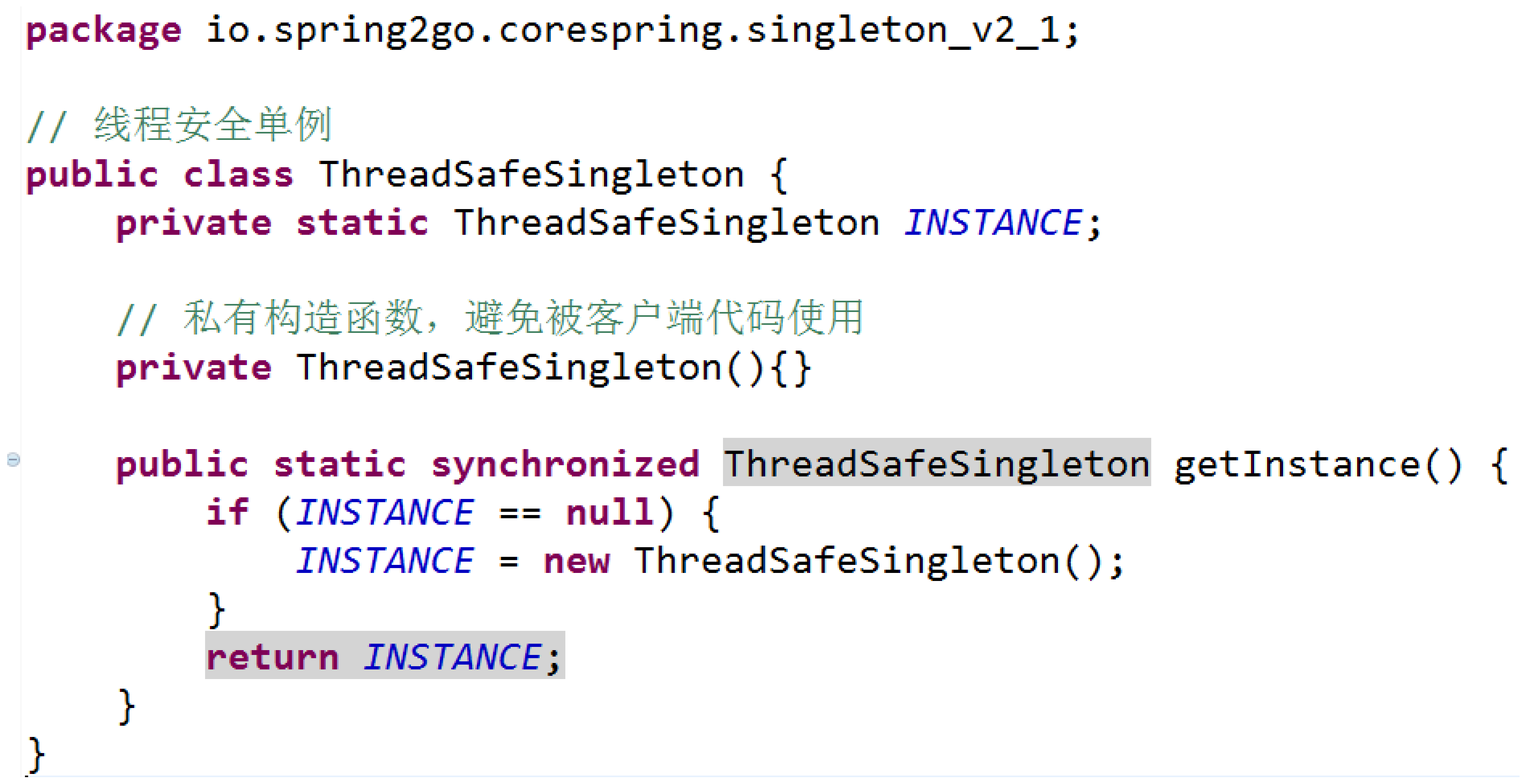
```java
package io.spring2go.corespring.singleton_v1_1;

// 静态块初始化单例
public class StaticBlockSingleton {
    private static final StaticBlockSingleton INSTANCE;

    // 私有构造函数，避免被客户端代码使用
    private StaticBlockSingleton() {}

    static {
        try {
            INSTANCE = new StaticBlockSingleton();
        } catch (Exception e) { // 异常处理
            throw new RuntimeException("Uffff, i was not expecting this!", e);
        }
    }

    public static StaticBlockSingleton getInstance() {
        return INSTANCE;
    }
}
```

单例2.0~延迟(Lazy)初始化单例



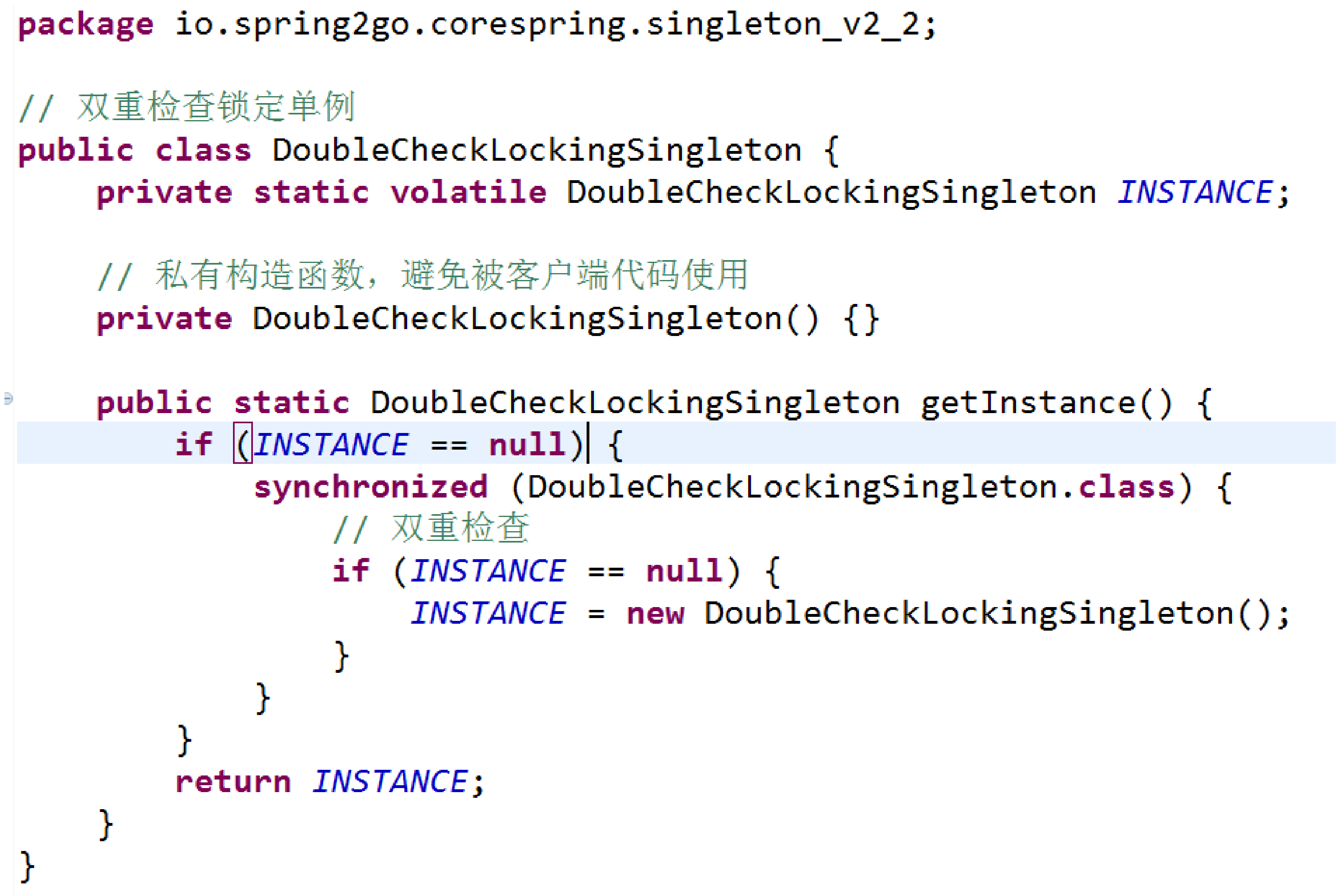
```java
package io.spring2go.corespring.singleton_v2;

// 延迟初始化单例
public class LazySingleton {
    private static LazySingleton INSTANCE;

    private LazySingleton(){}

    public static LazySingleton getInstance(){
        if(INSTANCE == null){
            INSTANCE = new LazySingleton();
        }
        return INSTANCE;
    }
}
```

单例2.1~多线程安全单例



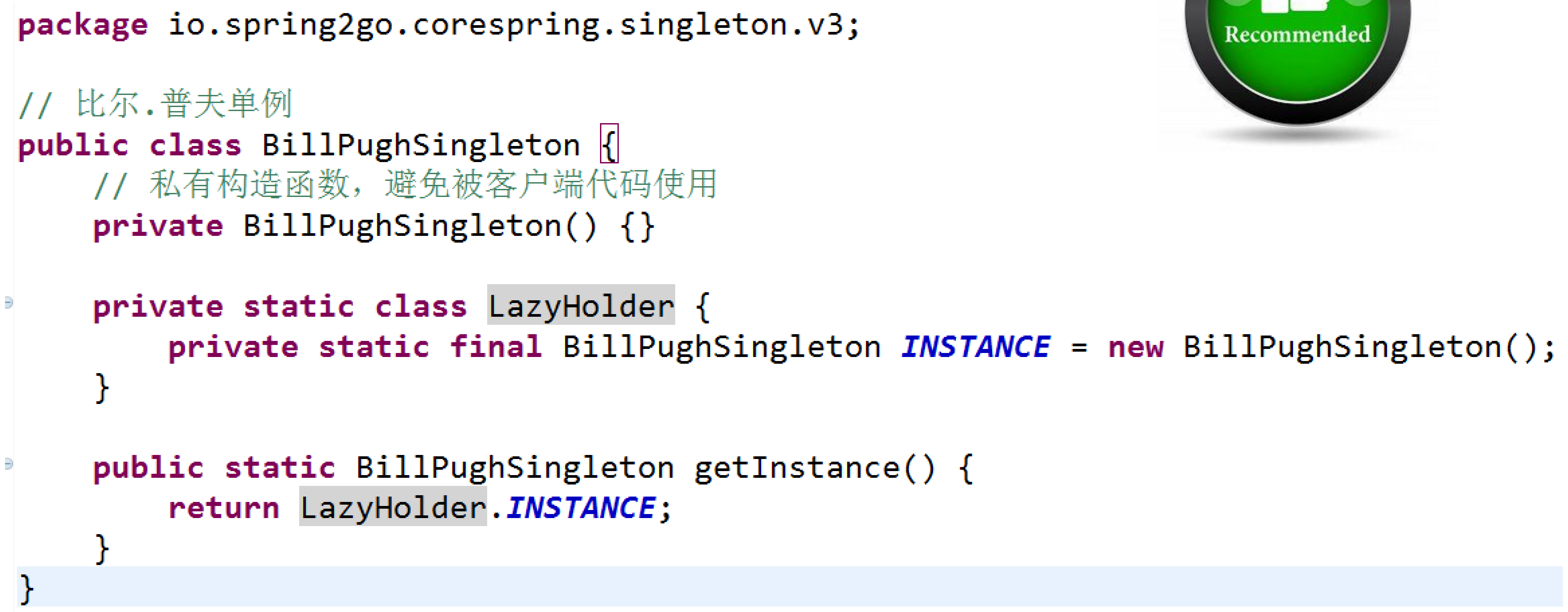
```java
package io.spring2go.corespring.singleton_v2_1;

// 线程安全单例
public class ThreadSafeSingleton {
    private static ThreadSafeSingleton INSTANCE;

    // 私有构造函数，避免被客户端代码使用
    private ThreadSafeSingleton(){}

    public static synchronized ThreadSafeSingleton getInstance() {
        if (INSTANCE == null) {
            INSTANCE = new ThreadSafeSingleton();
        }
        return INSTANCE;
    }
}
```

单例2.2~双重检查锁定单例



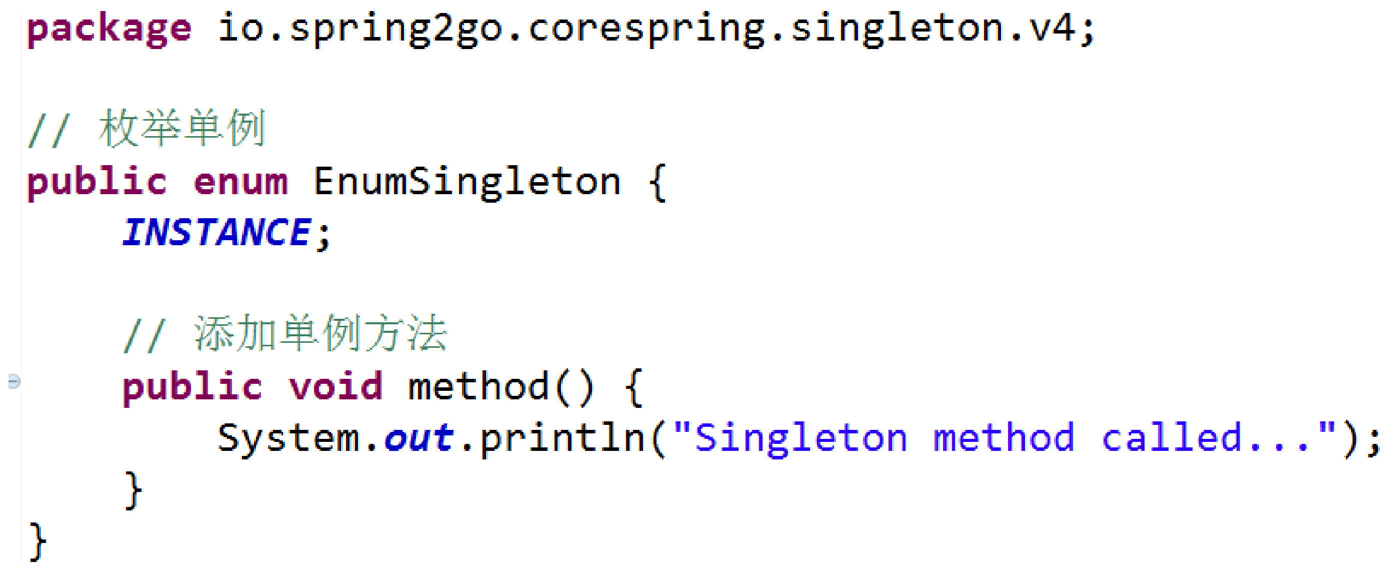
```java
package io.spring2go.corespring.singleton_v2_2;

// 双重检查锁定单例
public class DoubleCheckLockingSingleton {
    private static volatile DoubleCheckLockingSingleton INSTANCE;

    // 私有构造函数，避免被客户端代码使用
    private DoubleCheckLockingSingleton() {}

    public static DoubleCheckLockingSingleton getInstance() {
        if (INSTANCE == null) {
            synchronized (DoubleCheckLockingSingleton.class) {
                // 双重检查
                if (INSTANCE == null) {
                    INSTANCE = new DoubleCheckLockingSingleton();
                }
            }
        }
        return INSTANCE;
    }
}
```

单例3.0~比尔.普夫单例

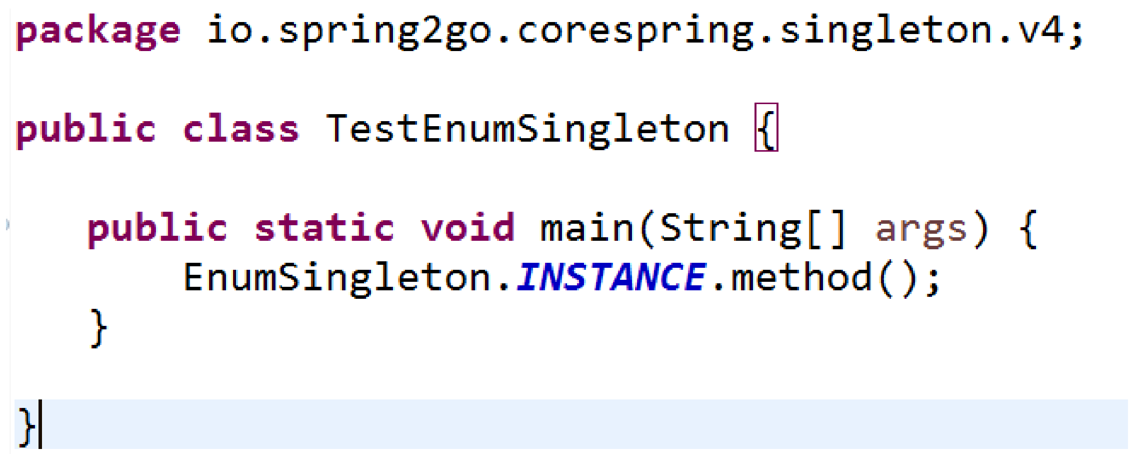


```java
package io.spring2go.corespring.singleton.v3;

// 比尔.普夫单例
public class BillPughSingleton {
    // 私有构造函数，避免被客户端代码使用
    private BillPughSingleton() {}

    private static class LazyHolder {
        private static final BillPughSingleton INSTANCE = new BillPughSingleton();
    }

    public static BillPughSingleton getInstance() {
        return LazyHolder.INSTANCE;
    }
}
```

单例4.0~枚举单例



```java
package io.spring2go.corespring.singleton.v4;

// 枚举单例
public enum EnumSingleton {
    INSTANCE;

    // 添加单例方法
    public void method() {
        System.out.println("Singleton method called...");
    }
}
```

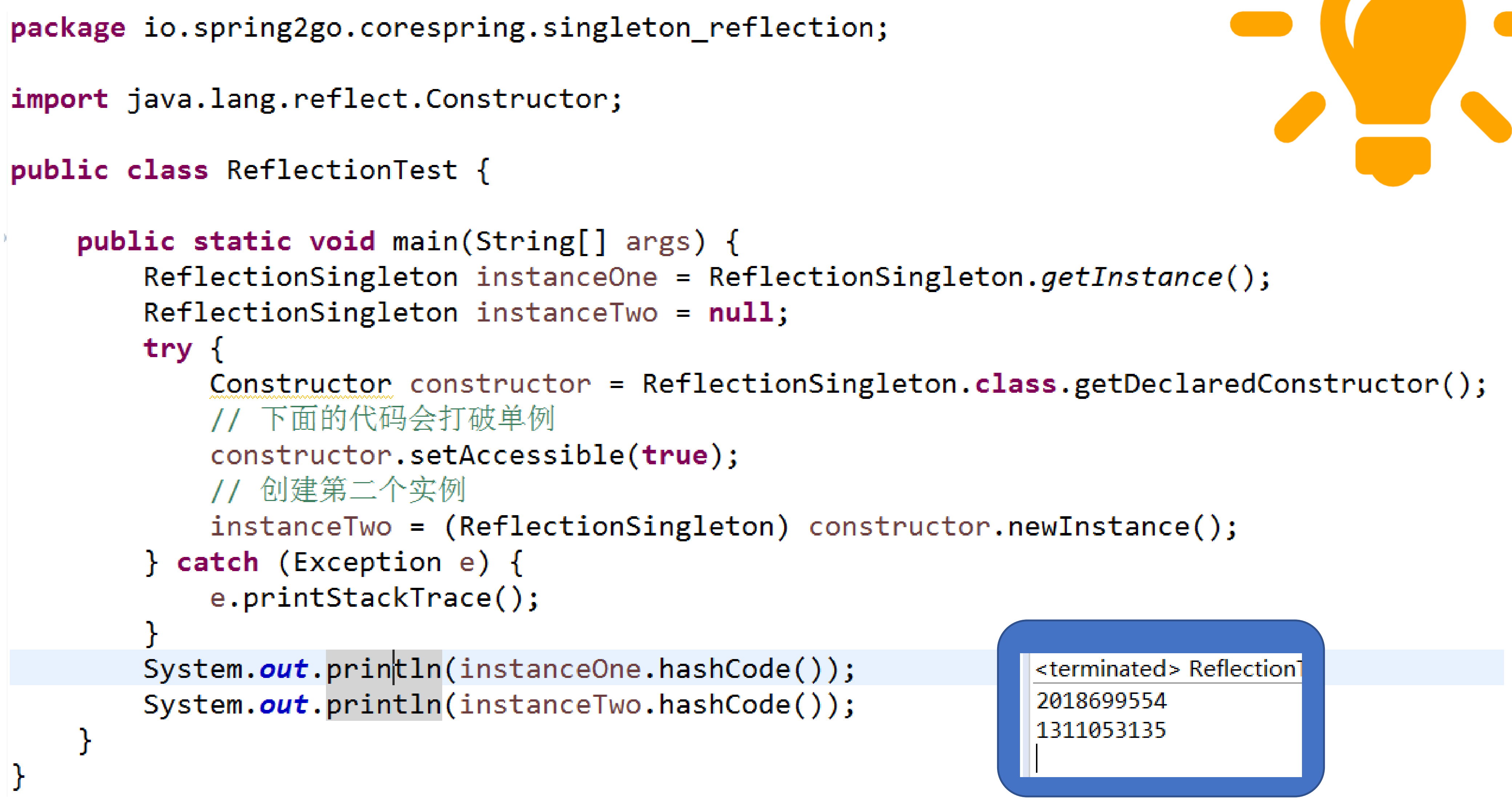


```java
package io.spring2go.corespring.singleton.v4;

public class TestEnumSingleton {

    public static void main(String[] args) {
        EnumSingleton.INSTANCE.method();
    }

}
```

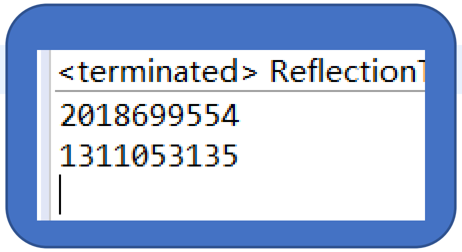
问题

- 如何破坏单例？

办法1~反射破坏单例

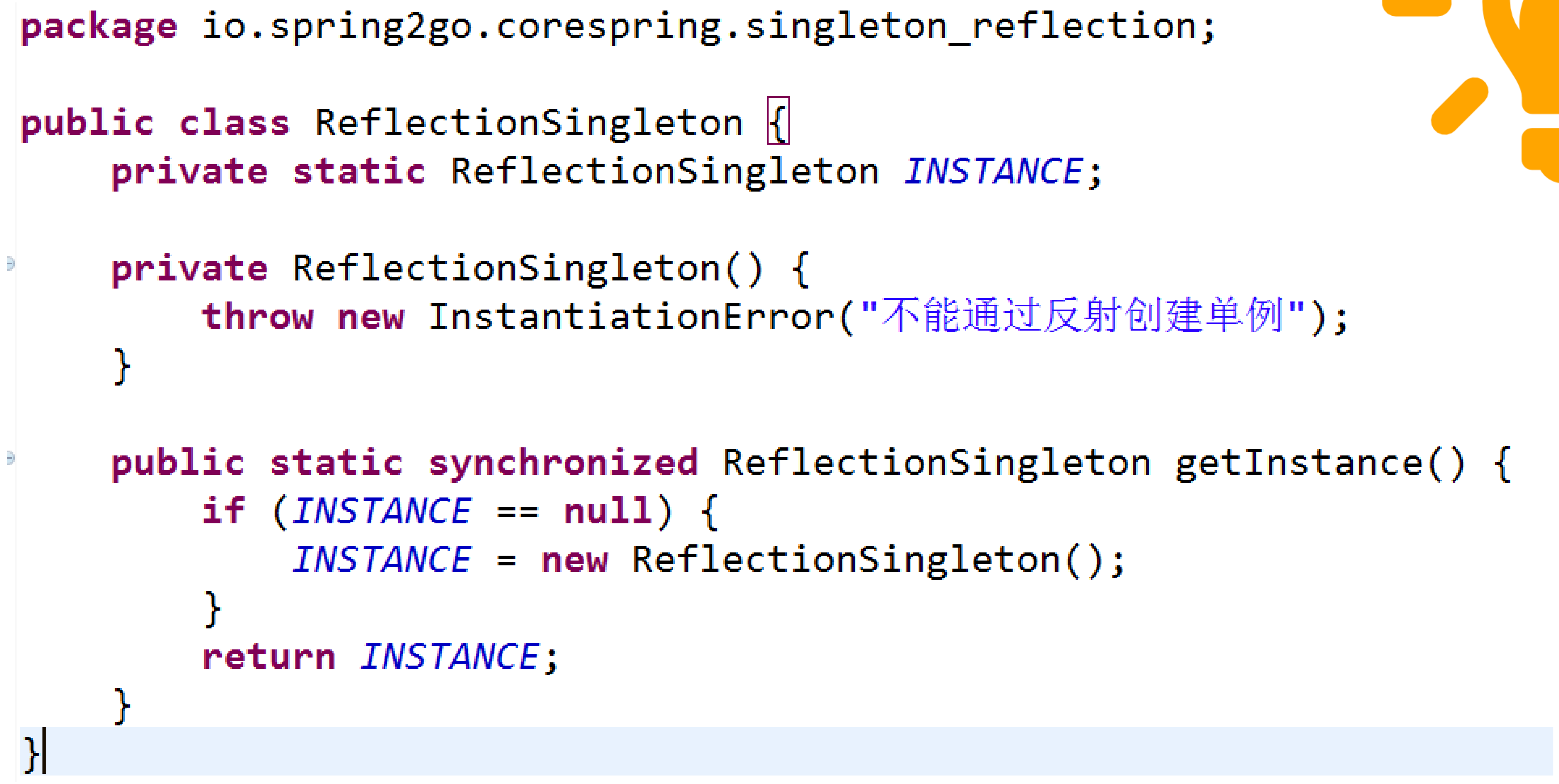


```java
package io.spring2go.corespring.singleton_reflection;

import java.lang.reflect.Constructor;

public class ReflectionTest {

    public static void main(String[] args) {
        ReflectionSingleton instanceOne = ReflectionSingleton.getInstance();
        ReflectionSingleton instanceTwo = null;
        try {
            Constructor constructor = ReflectionSingleton.class.getDeclaredConstructor();
            // 下面的代码会打破单例
            constructor.setAccessible(true);
            // 创建第二个实例
            instanceTwo = (ReflectionSingleton) constructor.newInstance();
        } catch (Exception e) {
            e.printStackTrace();
        }
        System.out.println(instanceOne.hashCode());
        System.out.println(instanceTwo.hashCode());
    }
}
```



```
<terminated> Reflection
2018699554
1311053135
```

解决办法



```java
package io.spring2go.corespring.singleton_reflection;

public class ReflectionSingleton {
    private static ReflectionSingleton INSTANCE;

    private ReflectionSingleton() {
        throw new InstantiationError("不能通过反射创建单例");
    }

    public static synchronized ReflectionSingleton getInstance() {
        if (INSTANCE == null) {
            INSTANCE = new ReflectionSingleton();
        }
        return INSTANCE;
    }
}
```

单例应用案例

- Core Java
  - java.lang.Runtime
  - java.awt.Desktop
- Spring容器
  - Singleton Scope(per container)

参考

- Singleton Design Pattern in Java
  - https://www.journaldev.com/1377/java-singleton-design-pattern-best-practices-examples
- Singleton Design Pattern Interview Questions
  - http://www.topjavatutorial.com/java-interview-questions/singleton-design-pattern-interview-questions/

问题

- 除了反射之外，还有哪些方法可以破坏单例？

序列化和发序列化

代码

- https://github.com/spring2go/core-spring-patterns

波波微课
spring2go.com

波波微课
spring2go.com