APKBUS

Java 线程安全的本质

朱凯

多线程：一听就头疼的话题

多线程：一听就头疼的话题

- 多线程怎么用
- 线程安全
- 线程间交互

多线程：一听就头疼的话题

- 多线程怎么用
- 线程安全
- 线程间交互

「线程相关面试题 100 道」的痛

「线程相关面试题 100 道」的痛

- synchronized 关键字怎么用？在 Java 虚拟机里它做了什么事？
- volatile 有什么作用？和 synchronized 有什么区别？
- AtomicXxx 和 volatile 有什么区别？
- wait() 和 notify() 方法应该怎么用？有什么注意事项？
- yield() 方法做了什么事？
- ……

「线程相关面试题 100 道」的痛

- 面试官问了你没背的题怎么办？

本质

线程安全是什么?

线程安全是什么？

从

synchronized

说起

```java
public class Config {

    private static Config instance;

    private Config() {
    }

    public static synchronized Config getInstance() {
        if (instance == null) {
            instance = new Config();
        }
        return instance;
    }

    public static Config getInstanceUsingDoubleLocking() {
        if (instance == null) {
            synchronized (Config.class) {
                if (instance == null) {
                    instance = new Config();
                }
            }
        }
        return instance;
    }
}
```

```java
public class Config {

    private static Config instance;

    private Config() {
    }

    public static synchronized Config getInstance() {
        if (instance == null) {
            instance = new Config();
        }
        return instance;
    }

    public static Config getInstanceUsingDoubleLocking() {
        if (instance == null) {
            synchronized (Config.class) {
                if (instance == null) {
                    instance = new Config();
                }
            }
        }
        return instance;
    }
}
```

```java
public class UserManager {
    private User user;

    public UserManager(User user) {
        this.user = user;
    }

    public void updateBy(User newUser) {
        user.setName(newUser.getName());
        user.setGender(newUser.getGender());
    }
}
```

```java
public class UserManager {
    private User user;

    public UserManager(User user) {
        this.user = user;
    }

    public void updateBy(User newUser) {
        user.setName(newUser.getName());
        user.setGender(newUser.getGender());
    }
}
```

```java
UserManager manager = new UserManager(user);

......

// 线程1
manager.updateBy(user1); // name: 扔物线; gender: 男

......

// 线程2
manager.updateBy(user2); // name: 丢物线; gender: 女
```

```java
public class UserManager {
    private User user;

    public UserManager(User user) {
        this.user = user;
    }

    public void updateBy(User newUser) {
        user.setName(newUser.getName());
        user.setGender(newUser.getGender());
    }
}
```

```java
UserManager manager = new UserManager(user);

......

// 线程1
manager.updateBy(user1); // name: 扔物线; gender: 男

......

// 线程2
manager.updateBy(user2); // name: 丢物线; gender: 女
```

```java
public class UserManager {
    private User user;

    public UserManager(User user) {
        this.user = user;
    }

    public void updateBy(User newUser) {
        user.setName(newUser.getName());
        user.setGender(newUser.getGender());
    }
}
```

线程一：name = "扔物线";

线程二：name = "丢物线";

线程二：gender = "女";

线程一：gender = "男";

结果：name = "丢物线", gender = "男"

```java
UserManager manager = new UserManager(user);

......

// 线程1
manager.updateBy(user1); // name: 扔物线; gender: 男

......

// 线程2
manager.updateBy(user2); // name: 丢物线; gender: 女
```

```java
public class UserManager {
    private User user;

    public UserManager(User user) {
        this.user = user;
    }

    public void updateBy(User newUser) {
        user.setName(newUser.getName());
        user.setGender(newUser.getGender());
    }
}
```

```java
UserManager manager = new UserManager(user);

......

// 线程1
manager.updateBy(user1); // name: 扔物线; gender: 男

......

// 线程2
manager.updateBy(user2); // name: 丢物线; gender: 女
```

```java
public class UserManager {
    private User user;

    public UserManager(User user) {
        this.user = user;
    }

    public synchronized void updateBy(User newUser) {
        user.setName(newUser.getName());
        user.setGender(newUser.getGender());
    }
}
```

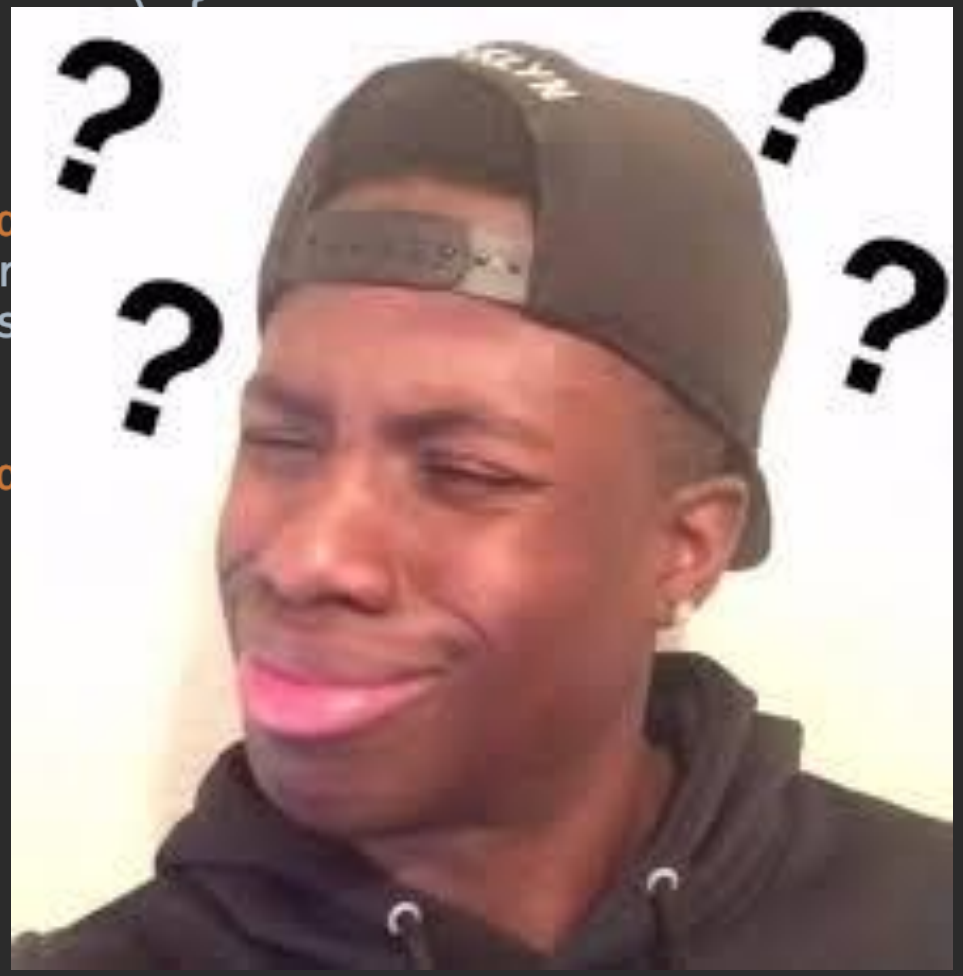
```java
UserManager manager = new UserManager(user);

......

// 线程1
manager.updateBy(user1); // name: 扔物线; gender: 男

......

// 线程2
manager.updateBy(user2); // name: 丢物线; gender: 女
```

“啊啊，这是个锁！”

```java
public class UserManager {
    private User user;

    public UserManager(User user) {
        this.user = user;
    }

    public synchronized void updateBy(User newUser) {
        user.setName(newUser.getName());
        user.setGender(newUser.getGender());
    }
}
```

```java
public class UserManager {
    private User user;

    public UserManager(User user) {
        this.user = user;
    }

    public synchronized void updateBy(User newUser) {
        user.setName(newUser.getName());
        user.setGender(newUser.getGender());
    }

    public synchronized void method2() {

    }
}
```

```
public class UserManager {
    private User user;

    public UserManager(User
        this.user = user;
    }

    public synchronized voic
        user.setName(newUser
        user.setGender(newUs
    }

    public synchronized voic

    }
}
```



```java
public class UserManager {
    private User user;

    public UserManager(User user) {
        this.user = user;
    }

    public synchronized void updateBy(User newUser) {
        user.setName(newUser.getName());
        user.setGender(newUser.getGender());
    }

    public synchronized void method2() {

    }
}
```

```java
public class UserManager {
    private User user;

    public UserManager(User user) {
        this.user = user;
    }

    public synchronized void updateBy(User newUser) {
        user.setName(newUser.getName());
        user.setGender(newUser.getGender());
    }

    public void printUser() {
        Log.d( tag: "UserManager",  msg: "name: " + user.getName()
                + ", gender: " + user.getGender());
    }
}
```

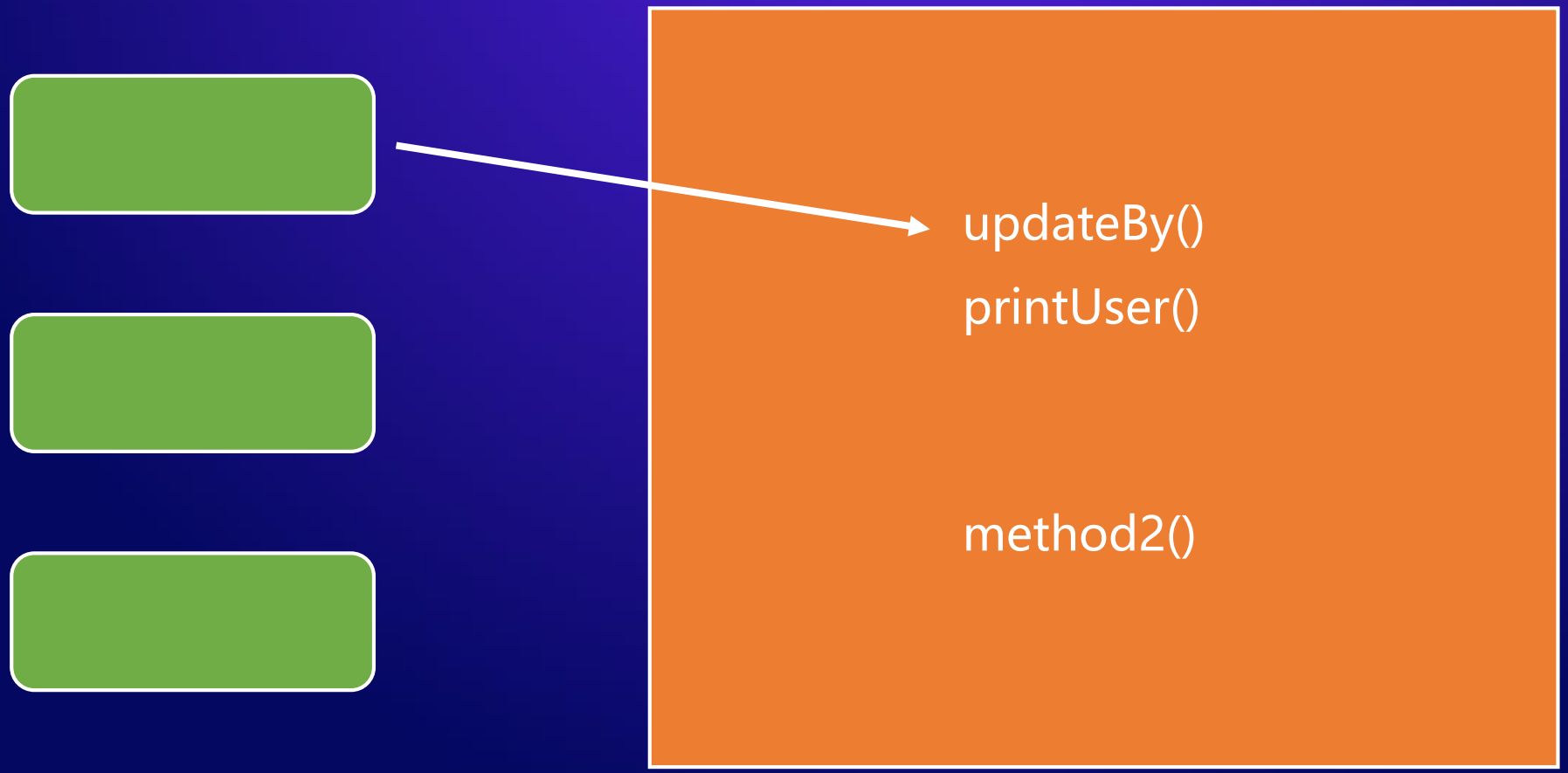
```java
public class UserManager {
    private User user;

    public UserManager(User user) {
        this.user = user;
    }

    public synchronized void updateBy(User newUser) {
        user.setName(newUser.getName());
        user.setGender(newUser.getGender());
    }

    public synchronized void printUser() {
        Log.d( tag: "UserManager",  msg: "name: " + user.getName()
                + ", gender: " + user.getGender());
    }
}
```

```java
public class UserManager {
    private User user;

    private final Object monitor2 = new Object();

    public UserManager(User user) {
        this.user = user;
    }

    public synchronized void updateBy(User newUser) {
        user.setName(newUser.getName());
        user.setGender(newUser.getGender());
    }

    public synchronized void printUser() {
        Log.d( tag: "UserManager",  msg: "name: " + user.getName()
                + ", gender: " + user.getGender());
    }

    public void method2() {
        synchronized (monitor2) {
            // 做事……
        }
    }
}
```

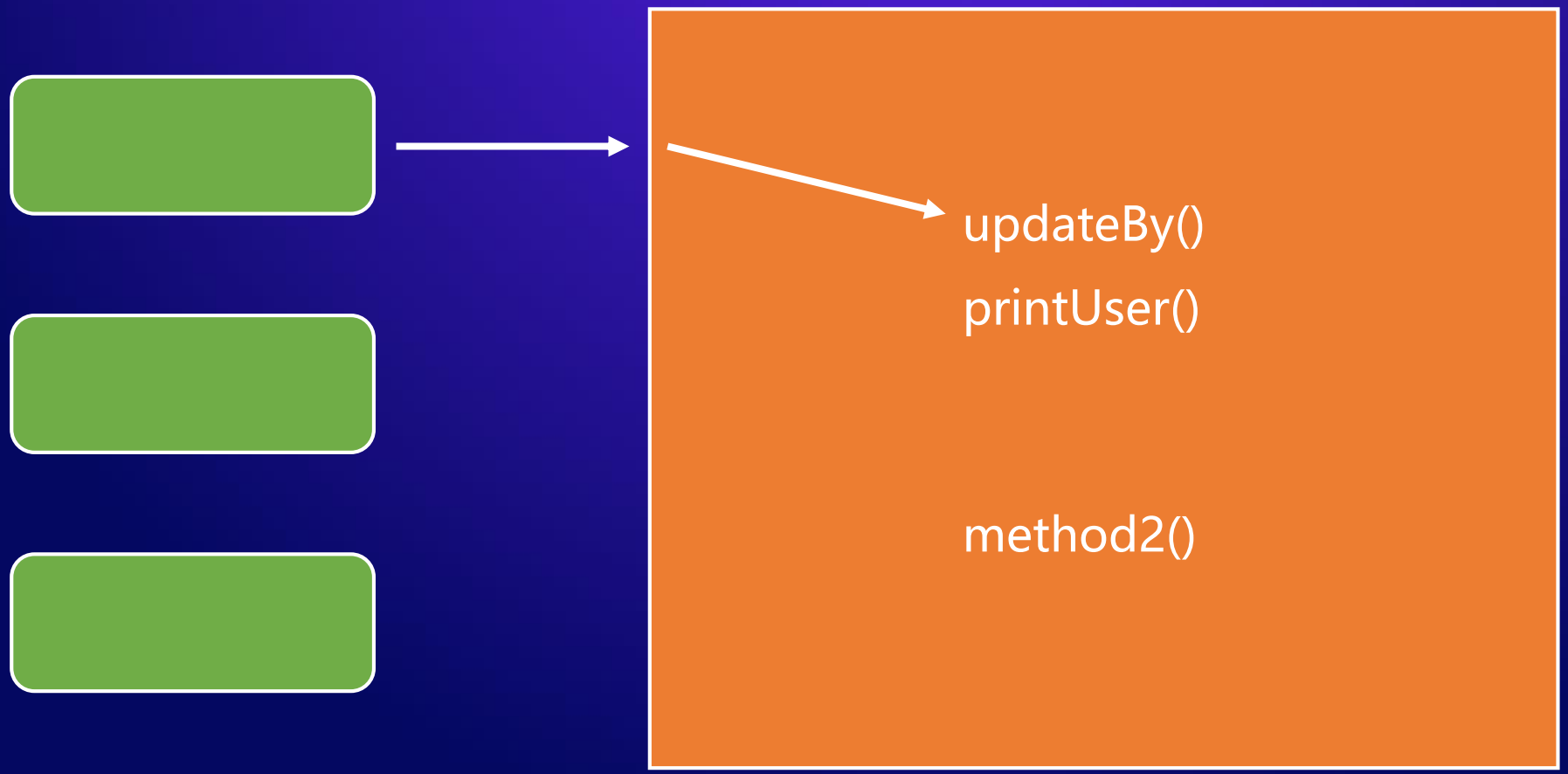
```java
public class UserManager {
    private User user;

    private final Object monitor2 = new Object();

    public UserManager(User user) {
        this.user = user;
    }

    public synchronized void updateBy(User newUser) {
        user.setName(newUser.getName());
        user.setGender(newUser.getGender());
    }

    public synchronized void printUser() {
        Log.d( tag: "UserManager",  msg: "name: " + user.getName()
                + ", gender: " + user.getGender());
    }

    public void method2() {
        synchronized (monitor2) {
            // 做事……
        }
    }
}
```

synchronized 到底做了什么

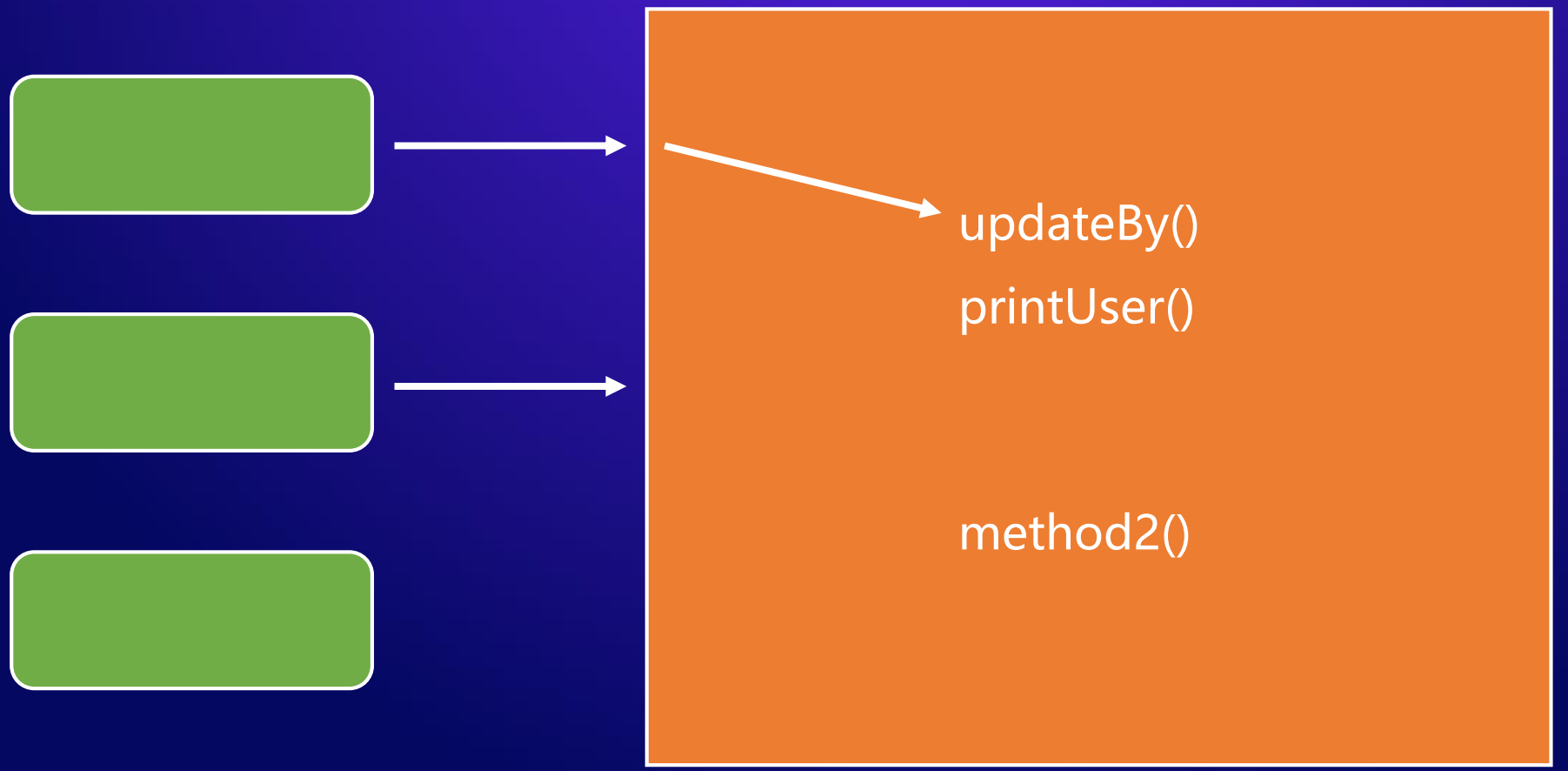
synchronized 到底做了什么



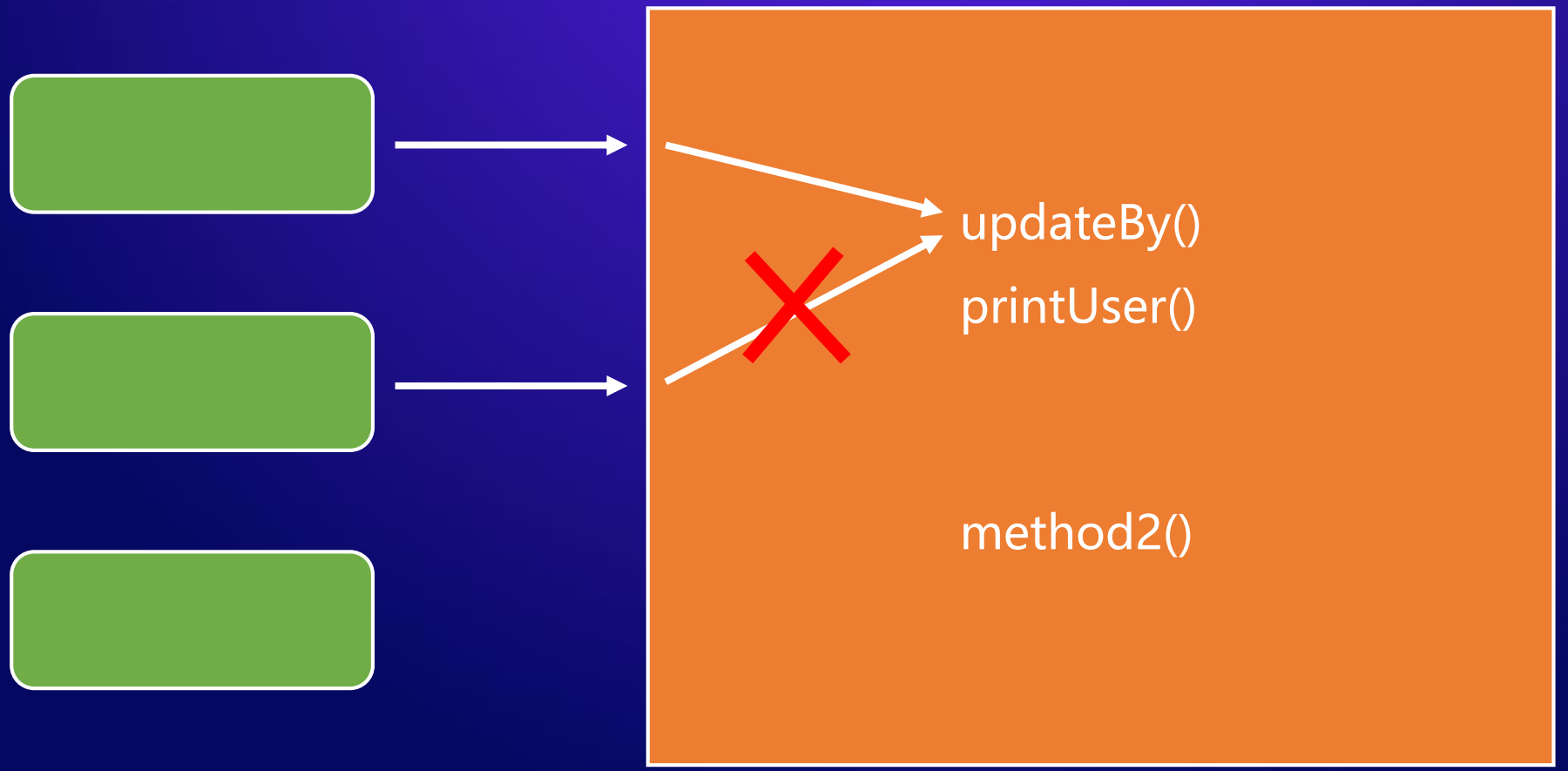
synchronized 到底做了什么



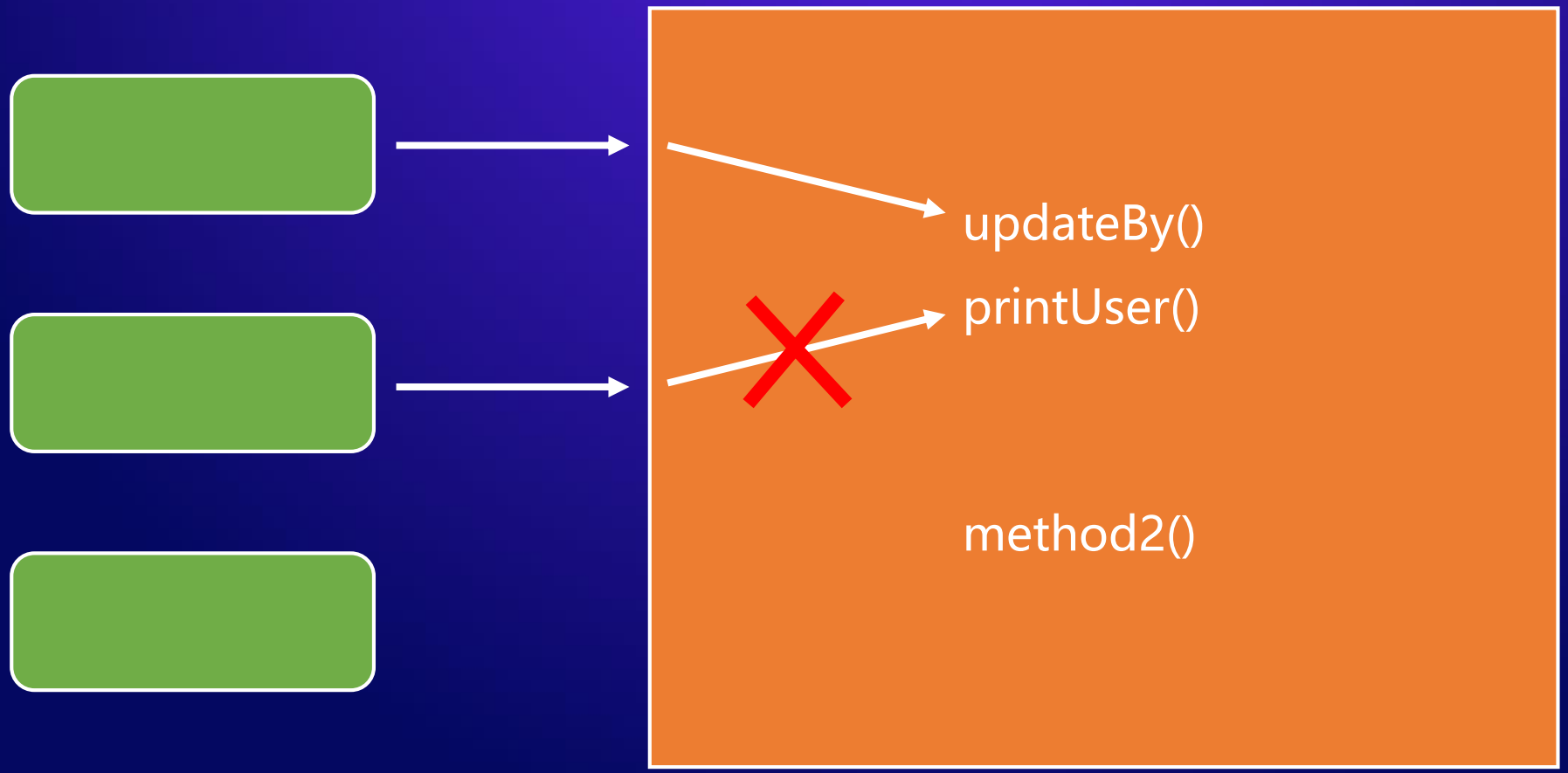
synchronized 到底做了什么



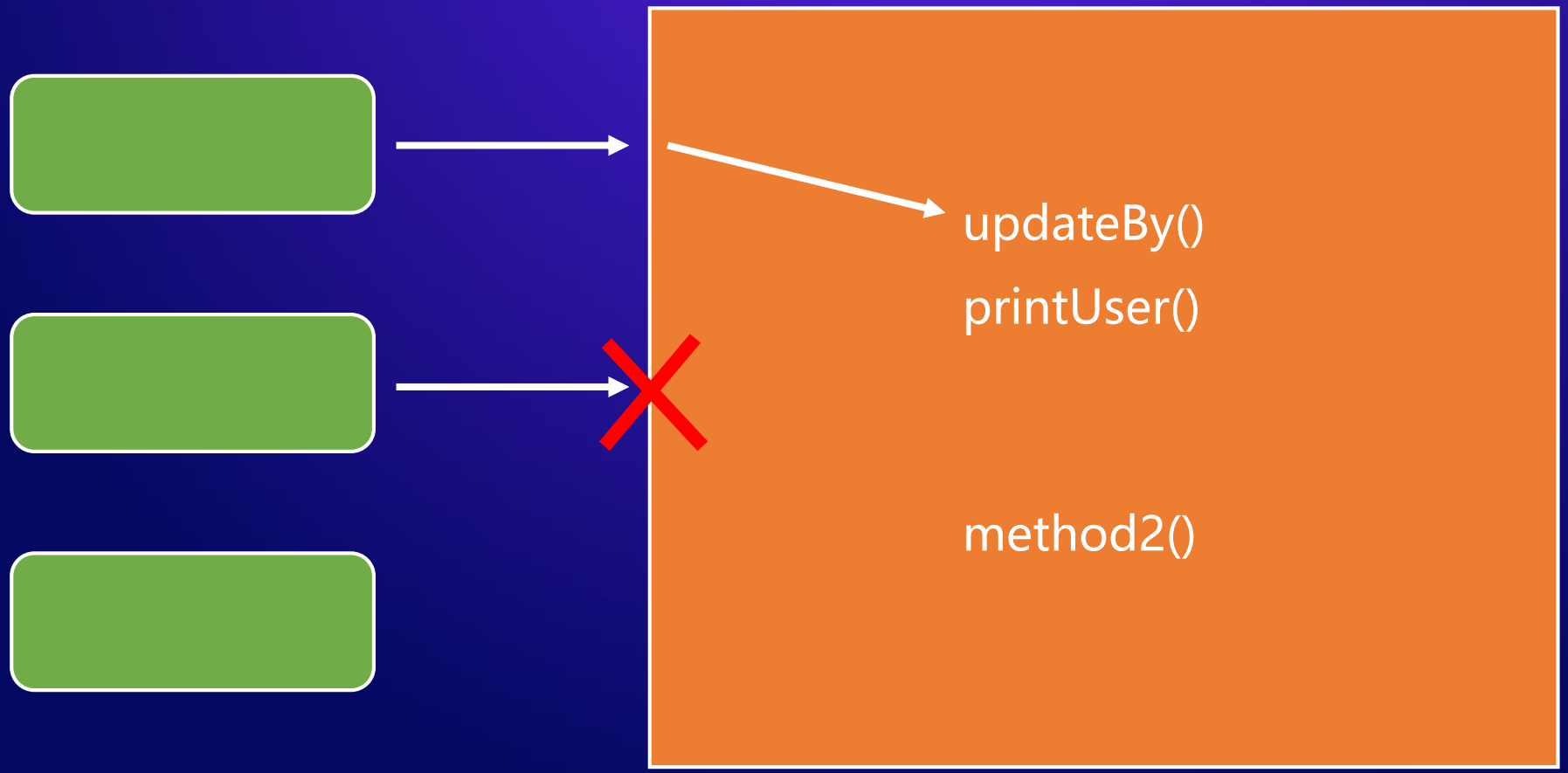
synchronized 到底做了什么



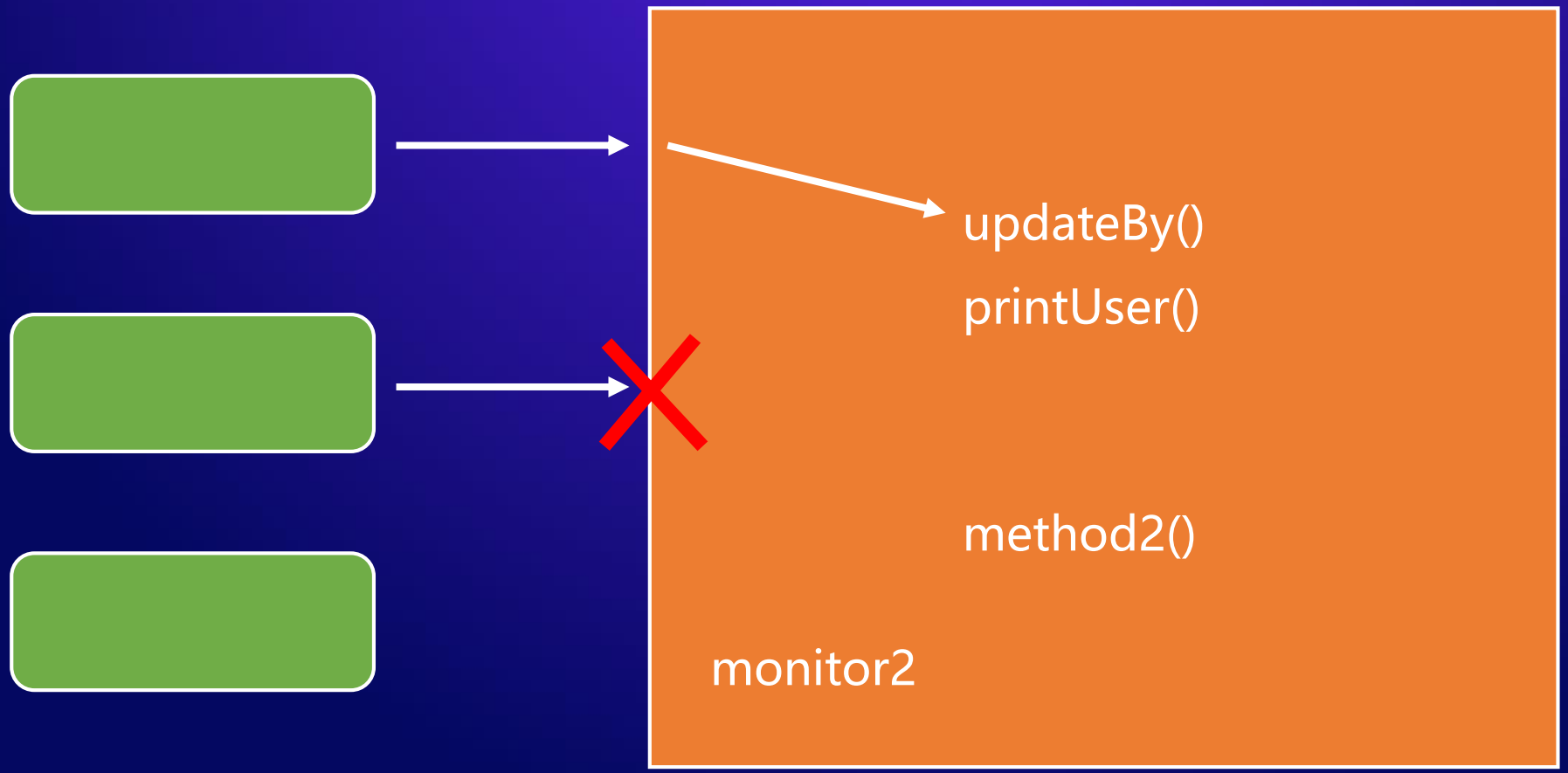
synchronized 到底做了什么



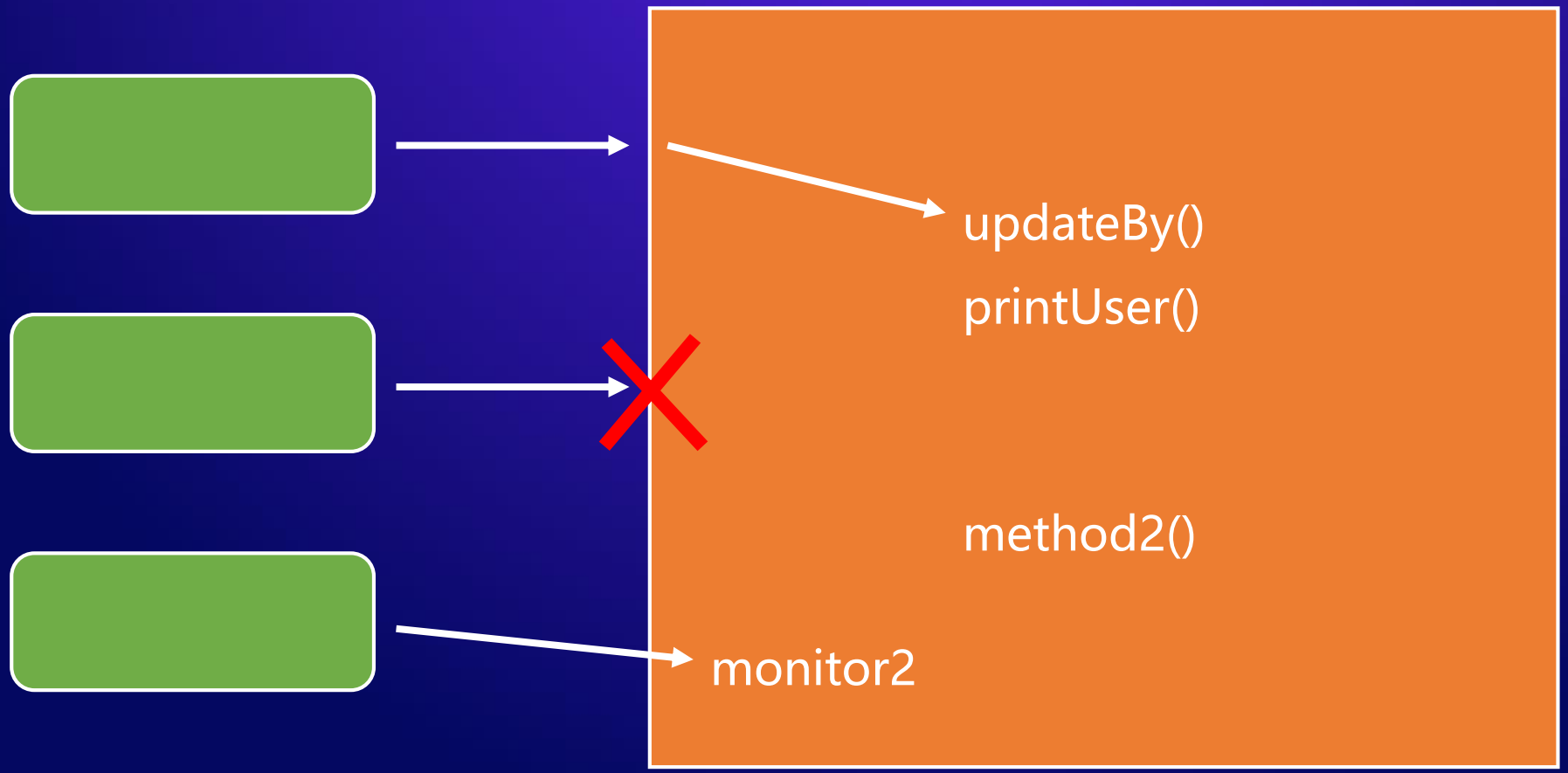
synchronized 到底做了什么



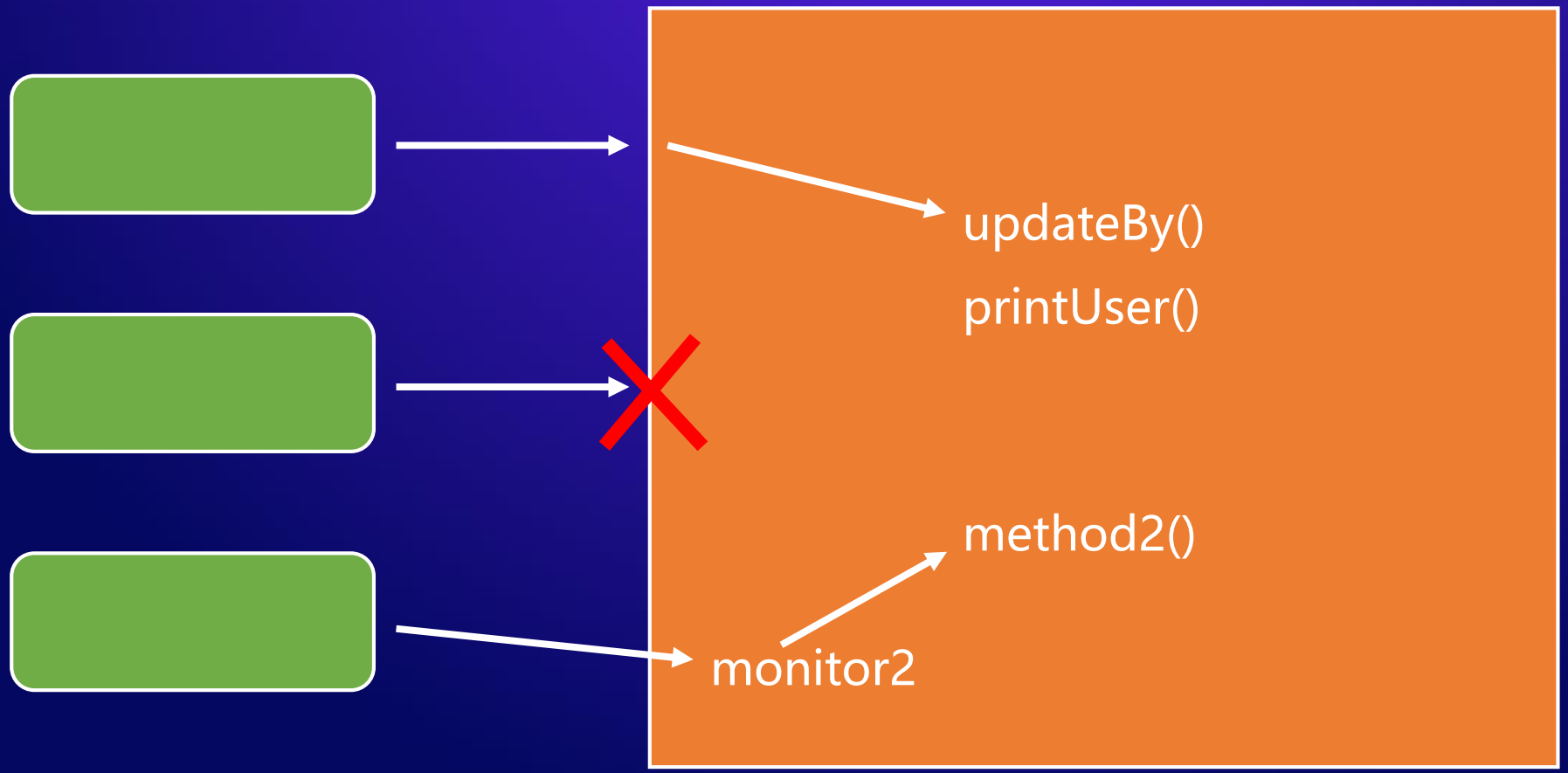
synchronized 到底做了什么



synchronized 到底做了什么

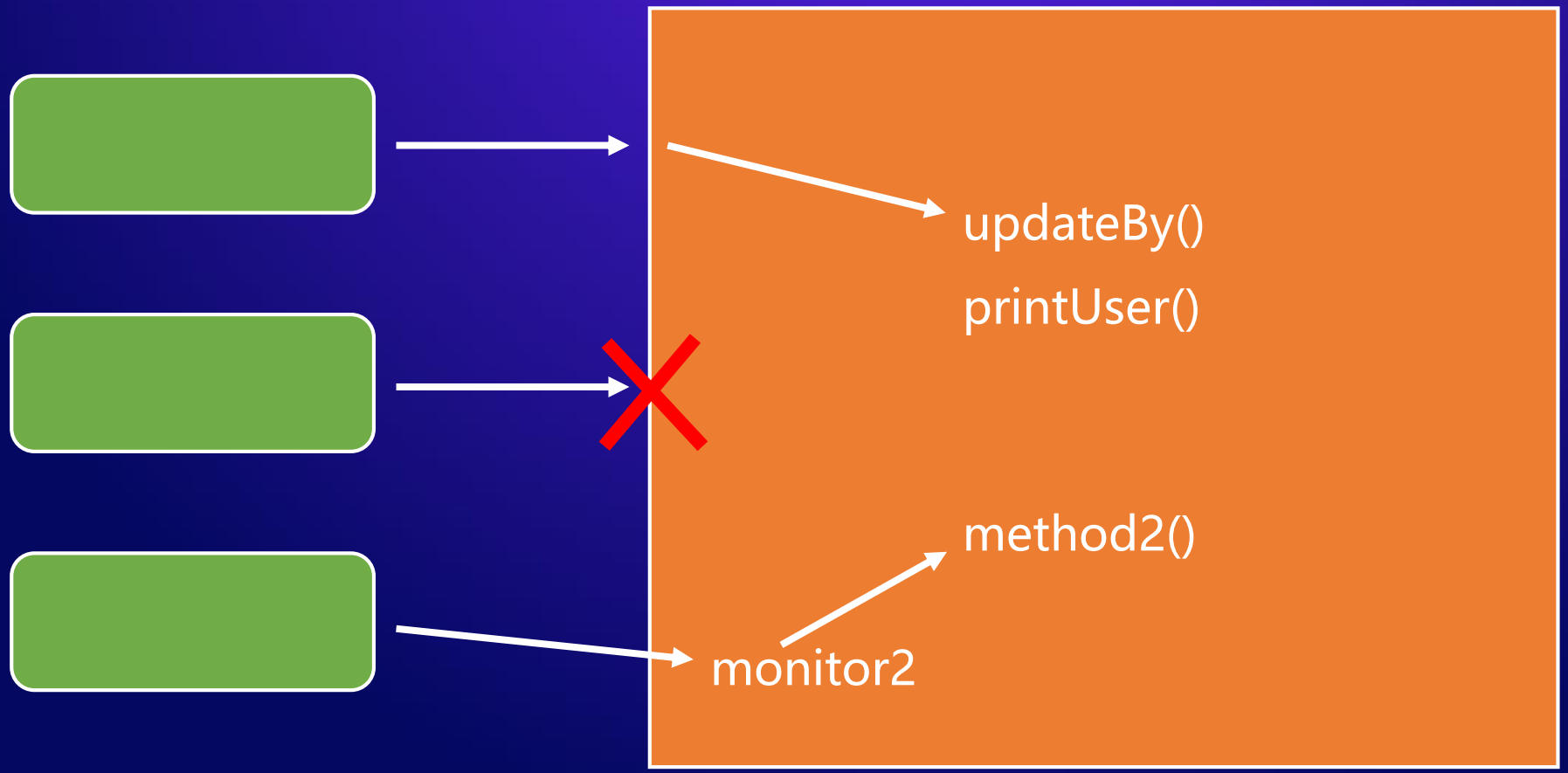


synchronized 到底做了什么



保护资源

synchronized 到底做了什么



线程「同步」？

线程安全怎么就安全了?

线程安全怎么就安全了?

- 哪些资源需要保护?
- 怎么保护?
- synchronized、volatile、Lock ……应该怎么选择?

线程安全怎么就安全了?

- 哪些资源需要保护?
- 怎么保护?
- synchronized、volatile、Lock ……应该怎么选择?
- 线程「不安全」的根本原因

线程安全怎么就安全了?

- 哪些资源需要保护?
- 怎么保护?
- synchronized、volatile、Lock ……应该怎么选择?
- 线程「不安全」的根本原因：写操作

```java
private final ReentrantReadWriteLock lock = new ReentrantReadWriteLock();
private final ReentrantReadWriteLock.ReadLock readLock = lock.readLock();
private final ReentrantReadWriteLock.WriteLock writeLock = lock.writeLock();

public synchronized void updateBy(User newUser) {
    writeLock.lock();
    try {
        user.setName(newUser.getName());
        user.setGender(newUser.getGender());
    } finally {
        writeLock.unlock();
    }
}

public synchronized void printUser() {
    readLock.lock();
    try {
        Log.d( tag: "UserManager",  msg: "name: " + user.getName()
                + ", gender: " + user.getGender());
    } finally {
        readLock.unlock();
    }
}
```

```java
private final ReentrantReadWriteLock lock = new ReentrantReadWriteLock();
private final ReentrantReadWriteLock.ReadLock readLock = lock.readLock();
private final ReentrantReadWriteLock.WriteLock writeLock = lock.writeLock();

public synchronized void updateBy(User newUser) {
    writeLock.lock();
    try {
        user.setName(newUser.getName());
        user.setGender(newUser.getGender());
    } finally {
        writeLock.unlock();
    }
}

public synchronized void printUser() {
    readLock.lock();
    try {
        Log.d( tag: "UserManager",  msg: "name: " + user.getName()
                + ", gender: " + user.getGender());
    } finally {
        readLock.unlock();
    }
}
```

总结

总结

- 线程安全问题的本质

总结

- 线程安全问题的本质
- 锁机制的本质

总结

- 线程安全问题的本质
- 锁机制的本质
- 核心：共享资源

建议：「问题」有无数多，永远要学习本质

建议：「问题」有无数多，永远要学习本质

- 例如：为什么 Thread.sleep() 一定要抛异常？

```
try {
    Thread.sleep( millis: 5000);
} catch (InterruptedException e) {
    e.printStackTrace();
}
```

欢迎关注安卓巴士公众号

www.apkbus.com