Java
Locking (LCK)

Mario Alviano

University of Calabria, Italy

A.Y. 2016/2017

Use private final lock objects to synchronize classes that may interact with untrusted code

- ✗ If the object or class are visible, also their intrinsic locks are visible
- ✗ The intrinsic lock mechanism is also messy
- ✓ Prefer block synchronization to method synchronization
- ✓ Use the *private lock object* idiom

```
https://www.securecoding.cert.org/confluence/
display/java/LCK00-J.+Use+private+final+lock+
objects+to+synchronize+classes+that+may+interact+
with+untrusted+code
```

Do not synchronize on objects that may be reused

- ✗ References to some boxed primitives and strings may be reused
- ✗ It's messy, don't synchronize on them
- ✓ Just synchronize on private final `Object` instances

```
https://www.securecoding.cert.org/confluence/
display/java/LCK01-J.+Do+not+synchronize+on+
objects+that+may+be+reused
```

Do not synchronize on the class object returned by getClass()

- ✗ If the method is overridden, `getClass()` will return a different object
- ✓ If you really want to synchronize on the class object, refer the class explicitly

```
https://www.securecoding.cert.org/confluence/
pages/viewpage.action?pageId=43647087
```

Do not synchronize on a collection view if the backing collection is accessible

- ✗ If the backing collenction is alterated, you may experience nondeterministic behavior
- ✓ Synchronize on the backing collection
- ✓ Use the *private lock object* idiom

```
https:
//www.securecoding.cert.org/confluence/display/
java/LCK04-J.+Do+not+synchronize+on+a+collection+
view+if+the+backing+collection+is+accessible
```

Do not use an instance lock to protect shared static data

- ✗ Instance lock will only synchronize that instance
- ✓ The lock object must be also static

```
https://www.securecoding.cert.org/confluence/
display/java/LCK06-J.+Do+not+use+an+instance+lock+
to+protect+shared+static+data
```

Avoid deadlock by requesting and releasing locks in the same order

- ✗ If you need multiple locks, don't create cyclic dependencies
- ✓ Enforce a partial order
- ✓ Use `ReentrantLock` to break cycles

```
https://www.securecoding.cert.org/confluence/
display/java/LCK07-J.+Avoid+deadlock+by+
requesting+and+releasing+locks+in+the+same+order
```

Ensure actively held locks are released on exceptional conditions

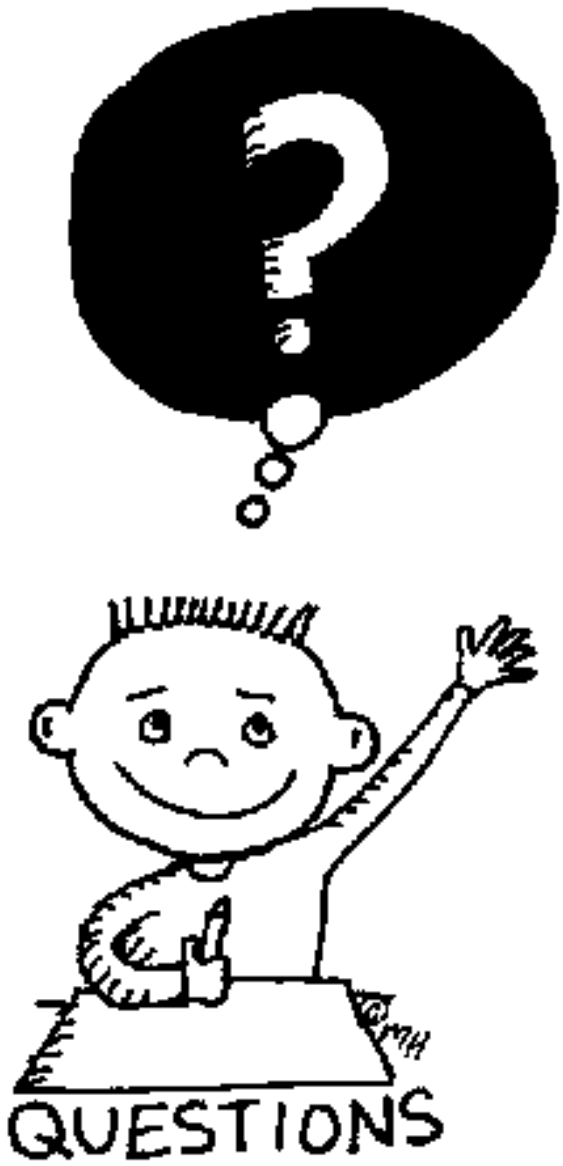
- ✗ If an exception is thrown after a lock has been acquired, it must be unlocked anyhow
- ✓ Unlock in `final` blocks
- ✓ Check for checked and unchecked exceptions

```
https://www.securecoding.cert.org/confluence/
display/java/LCK08-J.+Ensure+actively+held+locks+
are+released+on+exceptional+conditions
```

Do not perform operations that can block while holding a lock

- ✗ Avoid starvation
- ✓ Wait, don't sleep
- ✓ Only synchronize what has to be synchronized

```
https://www.securecoding.cert.org/confluence/
display/java/LCK09-J.+Do+not+perform+operations+
that+can+block+while+holding+a+lock
```



END OF THE LECTURE