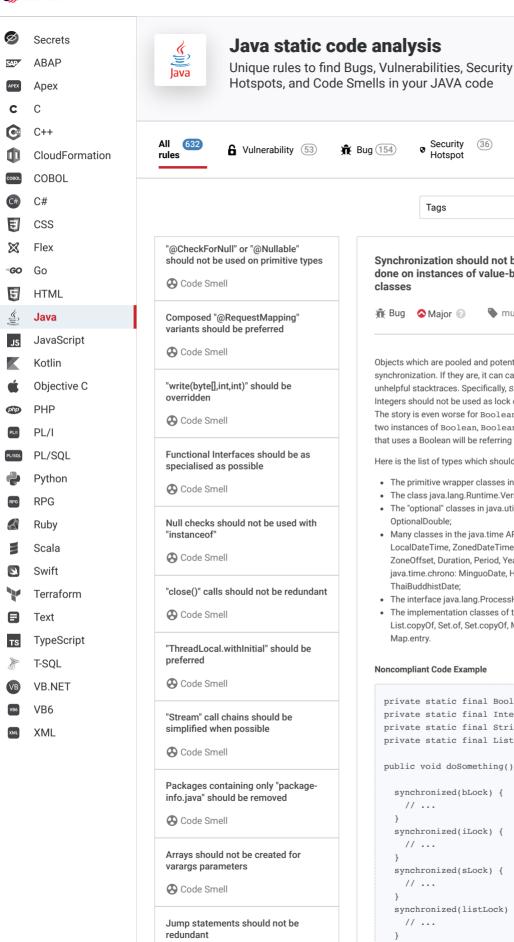


Products >



Code Smell

naming convention

Test classes should comply with a

Synchronization should not be done on instances of value-based classes

Security

Hotspot

Tags

(36)

Analyze your code

Quick 42

Q

Objects which are pooled and potentially reused should not be used for synchronization. If they are, it can cause unrelated threads to deadlock with unhelpful stacktraces. Specifically, String literals, and boxed primitives such as Integers should not be used as lock objects because they are pooled and reused. The story is even worse for Boolean objects, because there could possibly be only two instances of Boolean, Boolean. TRUE and Boolean. FALSE and every class that uses a Boolean will be referring to one of the two.

multi-threading cert

⊗ Code

Smell

(389)

Search by name.

Here is the list of types which shouldn't be used for synchronization:

- · The primitive wrapper classes in java.lang;
- · The class java.lang.Runtime.Version;
- The "optional" classes in java.util: Optional, OptionalInt, OptionalLong, and OptionalDouble:
- Many classes in the java.time API: Instant, LocalDate, LocalTime, LocalDateTime, ZonedDateTime, Zoneld, OffsetTime, OffsetDateTime, ZoneOffset, Duration, Period, Year, YearMonth, and MonthDay, and, in java.time.chrono: MinguoDate, HijrahDate, JapaneseDate, and ThaiBuddhistDate;
- The interface java lang Process Handle and its implementation classes:
- The implementation classes of the collection factories in java.util: List.of, List.copyOf, Set.of, Set.copyOf, Map.of, Map.copyOf, Map.ofEntries, and Map.entry.

## **Noncompliant Code Example**

```
private static final Boolean bLock = Boolean.FALSE;
private static final Integer iLock = Integer.valueOf(0);
private static final String sLock = "LOCK";
private static final List<String> listLock = List.of("a", "b
public void doSomething() {
  synchronized(bLock) { // Noncompliant
  synchronized(iLock) { // Noncompliant
  synchronized(sLock) { // Noncompliant
  synchronized(listLock) { // Noncompliant
```

**Compliant Solution** 

A Code Smell Loggers should be named for their enclosing classes Code Smell Methods should not return constants Code Smell "private" methods called only by inner classes should be moved to those Code Smell "enum" fields should not be publicly

mutable

```
private static final Object lock1 = new Object();
private static final Object lock2 = new Object();
private static final Object lock3 = new Object();
private static final Object lock4 = new Object();
public void doSomething() {
  synchronized(lock1) {
  synchronized(lock2) {
  synchronized(lock3) {
    // ...
  synchronized(lock4) {
```

## See

- CERT, LCK01-J. Do not synchronize on objects that may be reused
- JEP-390. JEP 390: Warnings for Value-Based Classes

Available In:

sonarlint ⊕ | sonarcloud ♦ | sonarqube



© 2008-2022 SonarSource S.A., Switzerland. All content is copyright protected. SONAR, SONARSOURCE, SONARLINT, SONARQUBE and SONARCLOUD are trademarks of SonarSource S.A. All other trademarks and copyrights are the property of their respective owners. All rights are expressly reserved. Privacy Policy