Java static code analysis

Unique rules to find Bugs, Vulnerabilities, Security Hotspots, and Code Smells in your JAVA code

Values should not be uselessly incremented
Bug

Silly String operations should not be made
Bug

Non-serializable classes should not be written
Bug

"hashCode" and "toString" should not be called on array instances
Bug

Collections should not be passed as arguments to their own methods
Bug

"BigDecimal(double)" should not be used
Bug

Invalid "Date" values should not be used
Bug

Reflection should not be used to check non-runtime annotations
Bug

Custom serialization method signatures should meet requirements
Bug

"Externalizable" classes should have no-arguments constructors
Bug

Classes should not be compared by name
Bug

"Object.wait(...)" and "Condition.await(...)" should be called inside a "while" loop

Analyze your code

Code Smell Critical multi-threading cert

According to the documentation of the Java `Condition` interface:

> When waiting upon a `Condition`, a "spurious wakeup" is permitted to occur, in general, as a concession to the underlying platform semantics. This has little practical impact on most application programs as a Condition should always be waited upon in a loop, testing the state predicate that is being waited for. An implementation is free to remove the possibility of spurious wakeups but it is recommended that applications programmers always assume that they can occur and so always wait in a loop.

The same advice is also found for the `Object.wait(...)` method:

> waits should always occur in loops, like this one:
>
> ```
> synchronized (obj) {
>   while (<condition does not hold>){
>     obj.wait(timeout);
>   }
>    ... // Perform action appropriate to condition
> }
> ```

Noncompliant Code Example

```
synchronized (obj) {
  if (!suitableCondition()){
    obj.wait(timeout);   //the thread can wake up even if th
  }
   ... // Perform action appropriate to condition
}
```

Compliant Solution

```
synchronized (obj) {
  while (!suitableCondition()){
    obj.wait(timeout);
  }
   ... // Perform action appropriate to condition
}
```

See

- CERT THI03-J. - Always invoke wait() and await() methods inside a loop

Available In:

sonarlint sonarcloud sonarqube

Related "if/else if" statements should not have the same condition

Bug

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

Bug

"Iterator.hasNext()" should not call "Iterator.next()"

Bug

Identical expressions should not be used on both sides of a binary operator

Bug