




 Secrets


 ABAP


 Apex


 C


 C++


 CloudFormation


 COBOL


 C#


 CSS


 Flex


 Go


 HTML


 **Java**


 JavaScript


 Kotlin


 Objective C


 PHP


 PL/I


 PL/SQL


 Python


 RPG


 Ruby


 Scala


 Swift


 Terraform


 Text


 TypeScript

 T-SQL

 VB.NET

 VB6

 XML



Java static code analysis

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

All rules632

Vulnerability53

Bug154

Security Hotspot36

Code Smell389

Quick Fix42

Tags ▾

Search by name... 🔍

Interalis should be explicit

Code Smell

Null should not be returned from a "Boolean" method

Code Smell

Classes should not access their own subclasses during initialization

Code Smell

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

Code Smell

IllegalMonitorStateException should not be caught

Code Smell

JUnit assertions should not be used in "run" methods

Code Smell

Class names should not shadow interfaces or superclasses

Code Smell

"Cloneables" should implement "clone"

Code Smell

Try-with-resources should be used

Code Smell

"readResolve" methods should be inheritable

Code Smell

"for" loop increment clauses should modify the loops' counters

Code Smell

Fields in a "Serializable" class should either be transient or serializable

Code Smell

Members ignored during record serialization should not be used

Analyze your code

BugCritical?java16

In Records, serialization is not done the same way as for ordinary serializable or externalizable classes. The serialized representation of a record object will be a sequence of values (record components). During the deserialization of records, the stream of components is read and components are constructed. Then the record object is recreated by invoking the record's canonical constructor with the component values serving as arguments (or default values for absent arguments).

This process cannot be customized, so any class-specific writeObject, readObject, readObjectNoData, writeExternal, and readExternal methods or serialPersistentFields fields in record classes are ignored during serialization and deserialization.

However, there is a way to substitute serialized/deserialized objects in writeReplace and readResolve.

This rule raises an issue when any of writeObject, readObject, readObjectNoData, writeExternal, readExternal or serialPersistentFields are present as members in a Record class.

Noncompliant Code Example

```
record Record() implements Serializable {
    @Serial
    private static final ObjectOutputStreamField[] serialPersistentF
    @Serial
    private void writeObject(ObjectOutputStream out) throws IO
        ...
    }
}
record Record() implements Externalizable {
    @Override
    public void writeExternal(ObjectOutput out) throws IOExcep
        ...
    }
    @Override
    public void readExternal(ObjectInput in) throws IOExceptio
        ...
    }
}
```

Compliant Solution


```
record Record() implements Serializable {}

record Record() implements Serializable {
    private Object writeReplace() throws ObjectStreamException
        ...
    }
    private Object readResolve() throws ObjectStreamException
        ...
    }
```


https://rules.sonarsource.com/java/RSPEC-6209

1/2


Package declaration should match source file directory

 Code Smell


Generic wildcard types should not be used in return types

 Code Smell

"switch" statements should have "default" clauses

 Code Smell

Execution of the Garbage Collector should be triggered only by the JVM

 Code Smell

```
}  
}
```

See

- [Records specification](#)
- [serialization of records](#)

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](#)

https://rules.sonarsource.com/java/RSPEC-6209

2/2