sonar RULES Products

Java static code analysis

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

All rules 632 | Vulnerability 53 | Bug 154 | Security Hotspot 36 | Code Smell 389 | Quick Fix 42

Tags | Search by name...

Members ignored during record serialization should not be used

Analyze your code

Bug Critical 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 ObjectStreamField[] 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
    ...
```

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