Java static code analysis

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

Using hardcoded IP addresses is security-sensitive
Security Hotspot

'serialVersionUID' field should not be set to '0L' in records
Code Smell

Permitted types of a sealed class should be omitted if they are declared in the same file
Code Smell

Switch arrow labels should not use redundant keywords
Code Smell

Text blocks should not be used in complex expressions
Code Smell

Pattern Matching for "instanceof" operator should be used instead of simple "instanceof" + cast
Code Smell

Call to Mockito method "verify", "when" or "given" should be simplified
Code Smell

Character classes should be preferred over reluctant quantifiers in regular expressions
Code Smell

Consecutive AssertJ "assertThat" statements should be chained
Code Smell

Chained AssertJ assertions should be simplified to the corresponding dedicated assertion
Code Smell

Exception testing via JUnit @Test annotation should be avoided

Non-serializable classes should not be written

Analyze your code

Bug Major serialization

Nothing in a non-serializable class will be written out to file, and attempting to serialize such a class will result in an exception being thrown. Only a class that `implements Serializable` or one that extends such a class can successfully be serialized (or de-serialized).

Noncompliant Code Example

```
public class Vegetable {  // neither implements Serializable
  //...
}

public class Menu {
  public void meal() throws IOException {
    Vegetable veg;
    //...
    FileOutputStream fout = new FileOutputStream(veg.getName
    ObjectOutputStream oos = new ObjectOutputStream(fout);
    oos.writeObject(veg);  // Noncompliant. Nothing will be
  }
}
```

Compliant Solution

```
public class Vegetable implements Serializable {  // can now
  //...
}

public class Menu {
  public void meal() throws IOException {
    Vegetable veg;
    //...
    FileOutputStream fout = new FileOutputStream(veg.getName
    ObjectOutputStream oos = new ObjectOutputStream(fout);
    oos.writeObject(veg);
  }
}
```

Available In:

sonarlint sonarcloud sonarqube

Code Smell

Escape sequences should not be used in text blocks

Code Smell

Simple string literal should be used for single line strings

Code Smell

Boxed "Boolean" should be avoided in boolean expressions

Code Smell

Type parameters should not shadow other type parameters