Java static code analysis

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

"==" and "!=" should not be used when "equals" is overridden

Analyze your code

Code Smell Minor cwe cert suspicious

It is equivalent to use the equality `==` operator and the `equals` method to compare two objects if the `equals` method inherited from `Object` has not been overridden. In this case both checks compare the object references.

But as soon as `equals` is overridden, two objects not having the same reference but having the same value can be equal. This rule spots suspicious uses of `==` and `!=` operators on objects whose `equals` methods are overridden.

Noncompliant Code Example

```
String firstName = getFirstName(); // String overrides equal
String lastName = getLastName();

if (firstName == lastName) { ... }; // Non-compliant; false
```

Compliant Solution

```
String firstName = getFirstName();
String lastName = getLastName();

if (firstName != null && firstName.equals(lastName)) { ... }
```

Exceptions

Comparing two instances of the `Class` object will not raise an issue:

```
Class c;
if(c == Integer.class) { // No issue raised
}
```

Comparing `Enum` will not raise an issue:

```
public enum Fruit {
   APPLE, BANANA, GRAPE
}
public boolean isFruitGrape(Fruit candidateFruit) {
  return candidateFruit == Fruit.GRAPE; // it's recommended
}
```

Comparing with `final` reference will not raise an issue:

```
private static final Type DEFAULT = new Type();

void foo(Type other) {
  if (other == DEFAULT) { // Compliant
  //...
```

Local-Variable Type Inference should be used

Code Smell

Migrate your tests from JUnit4 to the new JUnit5 annotations

Code Smell

Track uses of disallowed classes

Code Smell

Track uses of "@SuppressWarnings" annotations

Code Smell

```
    }
  }
```

Comparing with `this` will not raise an issue:

```
  public boolean equals(Object other) {
    if (this == other) {  // Compliant
      return false;
    }
  }
```

Comparing with `java.lang.String` and boxed types `java.lang.Integer`, ... will not raise an issue.

See

- {rule:java:S4973} - Strings and Boxed types should be compared using "equals()"
- MITRE, CWE-595 - Comparison of Object References Instead of Object Contents
- MITRE, CWE-597 - Use of Wrong Operator in String Comparison
- CERT, EXP03-J. - Do not use the equality operators when comparing values of boxed primitives
- CERT, EXP50-J. - Do not confuse abstract object equality with reference equality

Available In:

sonarlint | sonarcloud | sonarqube