Java static code analysis

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

Classes that don't define "hashCode()" should not be used in hashes
Bug

Floating point numbers should not be tested for equality
Bug

Increment (++) and decrement (--) operators should not be used in a method call or mixed with other operators in an expression
Code Smell

Limited dependence should be placed on operator precedence
Code Smell

Custom getter method should not be used to override record's getter behavior
Code Smell

Tests should use fixed data instead of randomized data
Code Smell

Spring's ModelAndViewAssert assertions should be used instead of other assertions
Code Smell

Lambdas should not have too many lines
Code Smell

"@EnableAutoConfiguration" should be fine-tuned
Code Smell

Enum values should be compared with "=="
Code Smell

Spring components should use constructor injection

"Map.get" and value test should be replaced with single method call

Analyze your code

Code Smell Major java8

It's a common pattern to test the result of a `java.util.Map.get()` against `null` or calling `java.util.Map.containsKey()` before proceeding with adding or changing the value in the map. However the `java.util.Map` API offers a significantly better alternative in the form of the `computeIfPresent()` and `computeIfAbsent()` methods. Using these instead leads to cleaner and more readable code.

Note that this rule is automatically disabled when the project's `sonar.java.source` is not 8.

Noncompliant Code Example

```
V value = map.get(key);
if (value == null) {  // Noncompliant
  value = V.createFor(key);
  if (value != null) {
    map.put(key, value);
  }
}
if (!map.containsKey(key)) {  // Noncompliant
  value = V.createFor(key);
  if (value != null) {
    map.put(key, value);
  }
}
return value;
```

Compliant Solution

```
return map.computeIfAbsent(key, k -> V.createFor(k));
```

Exceptions

This rule will not raise an issue when trying to add `null` to a map, because `computeIfAbsent` will not add the entry if the value returned by the function is `null`.

See Also

- {rule:java:S6104} - Map "computeIfAbsent()" and "computeIfPresent()" should not be used to add "null" values.

Available In:

sonarlint sonarcloud sonarqube

Code Smell

Regex patterns should not be created needlessly

Code Smell

Track uses of disallowed constructors

Code Smell

Java 8's "Files.exists" should not be used

Code Smell

"Optional" should not be used for parameters

Code Smell