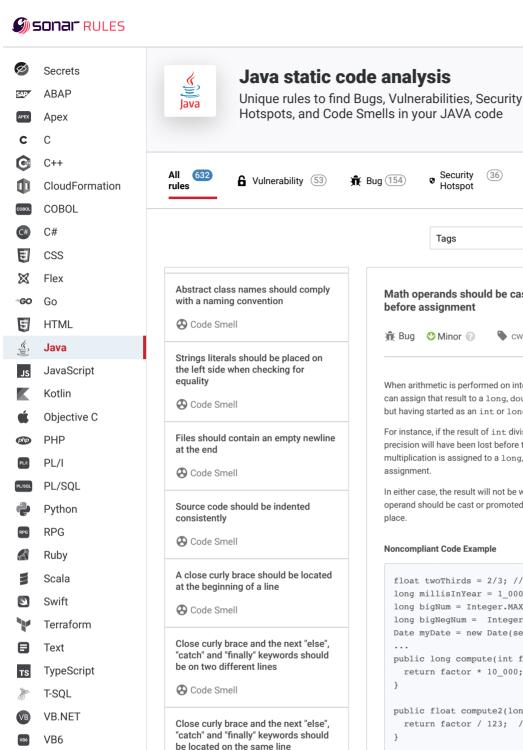
Products ✓

O Quick 42 Fix



Code Smell

Code Smell

at the end of a line

Code Smell

Code Smell

Code Smell

used

at the beginning of a line

An open curly brace should be located

An open curly brace should be located

Tabulation characters should not be

Functions should not be defined with a variable number of arguments

```
Search by name.
Math operands should be cast
                                              Analyze your code
before assignment
Rug Minor 🕝
                         cwe overflow sans-top25 cert
When arithmetic is performed on integers, the result will always be an integer. You
can assign that result to a long, double, or float, with automatic type conversion.
but having started as an int or long, the result will likely not be what you expect.
For instance, if the result of int division is assigned to a floating-point variable.
precision will have been lost before the assignment. Likewise, if the result of
multiplication is assigned to a long, it may have already overflowed before the
In either case, the result will not be what was expected. Instead, at least one
operand should be cast or promoted to the final type before the operation takes
Noncompliant Code Example
  float twoThirds = 2/3; // Noncompliant; int division. Yields
  long millisInYear = 1_000*3_600*24*365; // Noncompliant; int
  long bigNum = Integer.MAX_VALUE + 2; // Noncompliant. Yields
 long bigNegNum = Integer.MIN_VALUE-1; //Noncompliant, gives
  Date myDate = new Date(seconds * 1_000); //Noncompliant, won
 public long compute(int factor){
    return factor * 10_000; //Noncompliant, won't produce the
 public float compute2(long factor){
    return factor / 123; //Noncompliant, will be rounded to c
Compliant Solution
  float twoThirds = 2f/3; // 2 promoted to float. Yields 0.666
  long millisInYear = 1_000L*3_600*24*365; // 1000 promoted to
  long bigNum = Integer.MAX_VALUE + 2L; // 2 promoted to long.
  long bigNegNum = Integer.MIN_VALUE-1L; // Yields -2_147_483
```

Date myDate = new Date(seconds * 1 000L);

public long compute(int factor){ return factor * 10_000L;

public float compute2(long factor){

float twoThirds = (float)2/3; // 2 cast to float

long millisInYear = (long)1_000*3_600*24*365; // 1_000 cast

return factor / 123f;

⊗ Code

Smell

(389)

(36)

XML

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

```
long bigNum = (long)Integer.MAX_VALUE + 2;
long bigNegNum = (long)Integer.MIN_VALUE-1;
Date myDate = new Date((long)seconds * 1_000);
...
public long compute(long factor){
   return factor * 10_000;
}

public float compute2(float factor){
   return factor / 123;
}
```

See

- MITRE, CWE-190 Integer Overflow or Wraparound
- CERT, NUM50-J. Convert integers to floating point for floating-point operations
- <u>CERT, INT18-C.</u> Evaluate integer expressions in a larger size before comparing or assigning to that size
- SANS Top 25 Risky Resource Management

Available In:

sonarlint ⊖ | sonarcloud 👌 | sonarqube

vitzerland. All content is copyright protected. SONAR,

© 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