

P5 - Source Code Review

Pavit Srivatsasn

August 2021

1 Introduction

The source code review for function F2: (tanx) is performed using the tool SonarLint. SonarLint is an IDE extension that helps you detect and fix quality issues as you write code. Like a spell checker, SonarLint squiggles flaws so that they can be fixed before committing code.

2 SonarLint Features

SonarLint highlights code issues with markers on open files. It also provides an issues summary table for a selected component in the IDE, including the creation time of the issue.

2.1 Code Reliability

Catch tricky bugs to prevent undefined behaviour from impacting end-users.

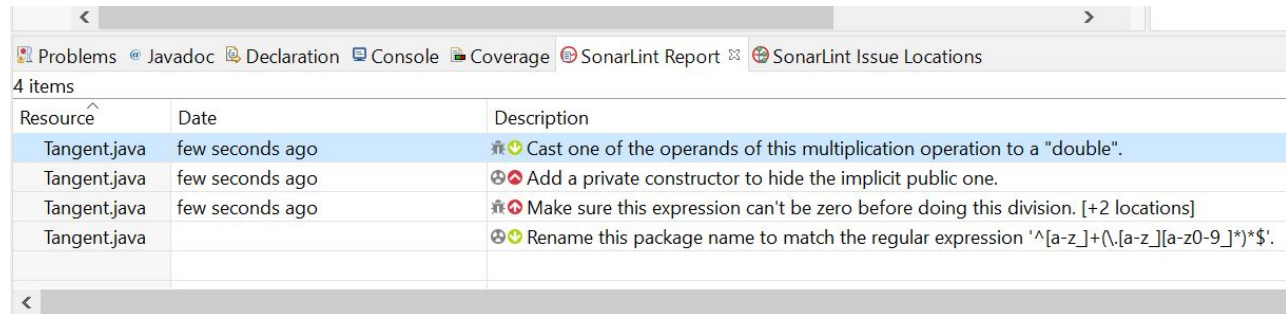
2.2 Application Security

Fix vulnerabilities that compromise the app, and learn AppSec along the way with Security Hotspots.

2.3 Technical Debt

Make sure your codebase is clean and maintainable, to increase developer velocity!

3 SonarLint Result



Resource	Date	Description
Tangent.java	few seconds ago	Cast one of the operands of this multiplication operation to a "double".
Tangent.java	few seconds ago	Add a private constructor to hide the implicit public one.
Tangent.java	few seconds ago	Make sure this expression can't be zero before doing this division. [+2 locations]
Tangent.java		Rename this package name to match the regular expression '^[a-z_]+(\.[a-z_][a-z0-9_]*)*\$'.

Figure 1: Sonar Lint Result for Tangent Function F2

Google Checkstyle is used for implementation.

3.1 Review Result 1

Cast one of the operands of this multiplication operation to a "double".

Code:

```
number = wholePart * 10000000; [line : 46]
```

Description:

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 assignment. 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 place.

3.2 Review Result 2

Add a private constructor to hide the implicit public one.

Description:

Utility classes, which are collections of static members, are not meant to be instantiated. Even abstract utility classes, which can be extended, should not have public constructors. Java adds an implicit public constructor to every class which does not define at least one explicitly. Hence, at least one non-public constructor should be defined.

3.3 Review Result 3

Make sure this expression can't be zero before doing this division. [+2 locations]

.

Code:

```
double result = calculateSin(angle) / calculateCos(angle); [line : 159]
```

Description:

If the denominator to a division or modulo operation is zero it would result in a fatal error. When working with double or float, no fatal error will be raised, but it will lead to unusual result and should be avoided anyway. This rule supports primitive int, long, double, float as well as BigDecimal and BigInteger.

3.4 Review Result 4 (enviornment - ignored)

Rename this package name to match the regular expression. .