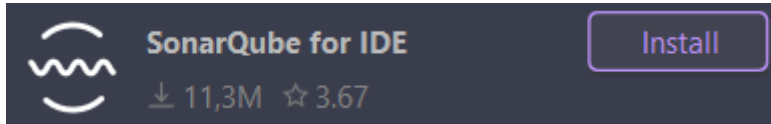


Dokumentation – Static Code Analysis

Verwendetes Tool

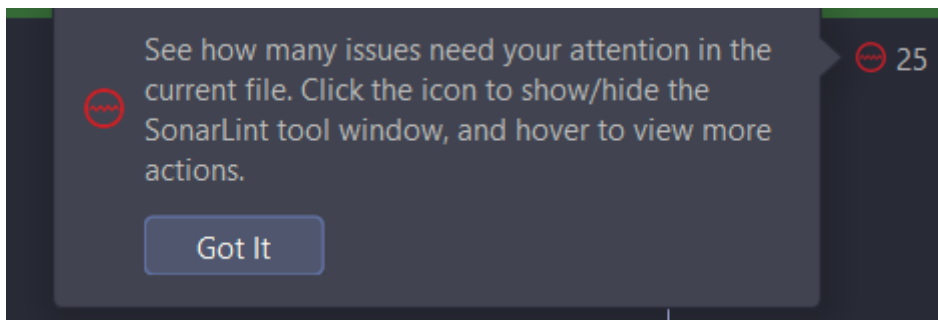
Für die statische Codeanalyse habe ich das Plugin **SonarQube for DIE** verwendet. Dieses Plugin ist über den Marketplace der DIE installierbar.



Einrichtung

Die Installation war sehr einfach:

- Nach dem Herunterladen aus dem Marketplace musste die DIE neu gestartet werden.
- Danach war SonarQube for IDE automatisch aktiviert.
- Die Analyse startet entweder „on the fly“ während des Codens oder kann manuell ausgelöst werden (z.B. über das Kontextmenü, Tastenkombinationen oder die SonarQube-Konsole in der IDE).



Funktionsweise

Das Tool zeigt automatisch alle gefundenen Probleme im Code an. Diese sind in drei Kategorien eingeteilt:

- Low Impact on Code
- Medium Impact on Code
- High Impact on Code


Vorgehensweise

Ich habe mit Hilfe von SonarQube 10 Fehler im Projekt identifiziert. Diese habe ich anschließend korrigiert und nach jeder Änderung im Git-Repository einzeln committed. Dadurch lässt sich der Fortschritt klar nachvollziehen.

Alle Fehler sind:

"switch" statements should have "default" clauses

Intentionality issue | Not clear

Maintainability 


java:S131

[Learn more](#)Why is this an issue?[More Info](#)

The requirement for a final `default` clause is defensive programming. The clause should either take appropriate action, or contain a suitable comment as to why no action is taken.

String literals should not be duplicated

Adaptability issue | Not distinct

Maintainability 


java:S1192

[Learn more](#)Why is this an issue?

Duplicated string literals make the process of refactoring error-prone, since you must be sure to update all occurrences.

Unused "private" fields should be removed

Intentionality issue | Not clear

Maintainability 


java:S1068

[Learn more](#)Why is this an issue?

If a `private` field is declared but not used in the program, it can be considered dead code and should therefore be removed. This will improve maintainability because developers will not wonder what the variable is used for. Note that this rule does not take reflection into account, which means that issues will be raised on `private` fields that are only accessed using the reflection API.

The default unnamed package should not be used


Adaptability issue | Not modular

Maintainability 

java:S1220

Sections of code should not be commented out

Intentionality issue | Not clear

Maintainability 


java:S125

[Learn more](#)Why is this an issue?

Programmers should not comment out code as it bloats programs and reduces readability. Unused code should be deleted and can be retrieved from source control history if required.

Class variable fields should not have public accessibility


Adaptability issue | Not modular

Maintainability 

java:S1104

Unused "private" fields should be removed

Intentionality issue | Not clear

Maintainability 


java:S1068

[Learn more](#)**Why is this an issue?**

If a `private` field is declared but not used in the program, it can be considered dead code and should therefore be removed. This will improve maintainability because developers will not wonder what the variable is used for. Note that this rule does not take reflection into account, which means that issues will be raised on `private` fields that are only accessed using the reflection API.

"InterruptedException" and "ThreadDeath" should not be ignored

Intentionality issue | Not complete

Reliability 


java:S2142

[Learn more](#)**Why is this an issue?****More Info**

If an `InterruptedException` or a `ThreadDeath` error is not handled properly, the information that the thread was interrupted will be lost. Handling this exception means either to re-throw it or manually re-interrupt the current thread by calling `Thread.interrupt()`. Simply logging the exception is not sufficient and counts as ignoring it. Between the moment the exception is caught and handled, is the right time to perform cleanup operations on the method's state, if needed.

Unused "private" fields should be removed


Intentionality issue | Not clear

Maintainability 

java:S1068

Sections of code should not be commented out

Intentionality issue | Not clear

Maintainability 

java:S125

[Learn more](#)**Why is this an issue?**

Programmers should not comment out code as it bloats programs and reduces readability. Unused code should be deleted and can be retrieved from source control history if required.