

Static Code Analysis Findings

Before refactoring

SonarQube analysis was done before and after refactoring the codebase. Screenshots were taken of the overview, issues and measures.

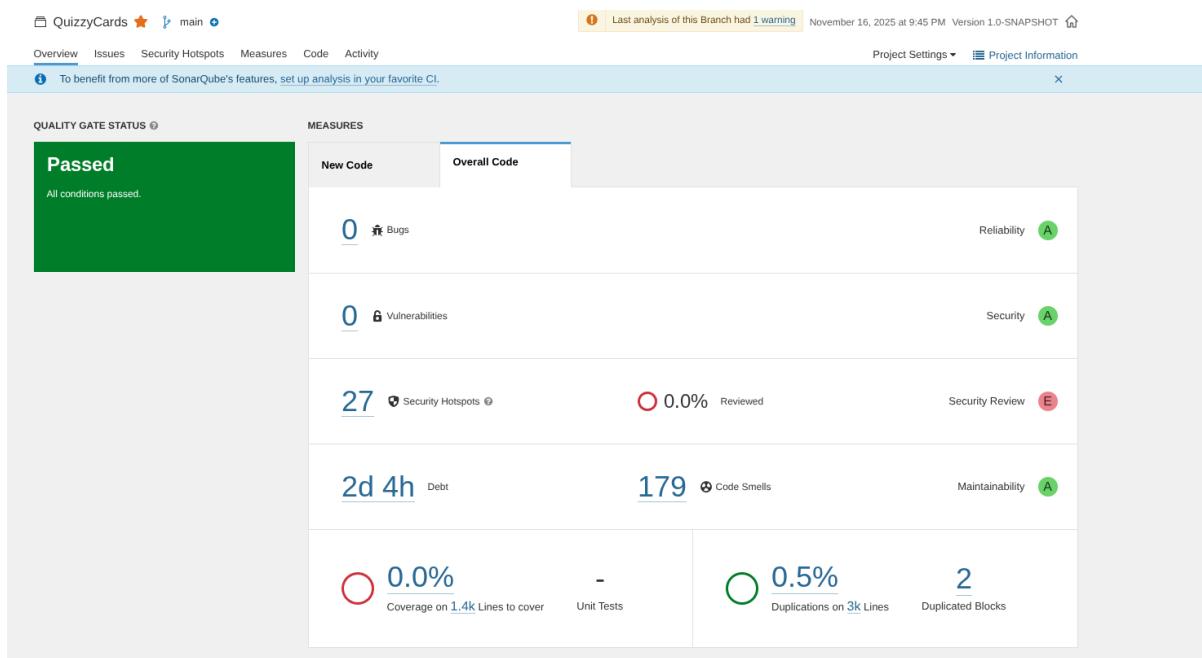
Measures before refactoring:

- 179 issues
- 410 cyclomatic complexity
- 11 lines of code per method
- 19 duplicated lines

11 lines of code per method is a very good number and cyclomatic complexity is good too. If total cyclomatic complexity is divided by total number of functions (269) to get the average cyclomatic complexity the result is 1.53, which is excellent.

So the biggest improvements that can be made is reducing the number of issues as much as possible without restructuring the codebase too much.

Screenshots before refactoring:



The screenshot shows two main sections of the SonarQube interface:

- Issues Section:** This section is titled "My Issues" and displays a list of 179 issues across various files. The issues are categorized by type (Bug, Vulnerability, Code Smell), severity (Blocker, Critical, Major, Minor, Info), and status (Open, Not assigned). A summary at the top right indicates 1 / 179 issues and 2d 4h effort.
- Measures Section:** This section is titled "Measures" and displays various code metrics. It includes a "Risk" bubble chart showing the relationship between Coverage, Technical Debt, and Risk. Other metrics listed include Density, Duplicated Lines, Duplicated Blocks, Duplicated Files, Lines of Code, Lines, Statements, Functions, Classes, Files, Comment Lines, Comments (%), Cyclomatic Complexity, and Cognitive Complexity. The "Issues" tab is also visible in the navigation bar.

First refactoring

Measures after refactoring:

- 45 issues
- 411 cyclomatic complexity
- 11 lines of code per method
- 42 duplicated lines

By refactoring the codebase, we removed 134 issues. Some issues remain, but they are not critical issues. Still, we will reduce these to zero.

Duplicated lines increased after refactoring. That needs fixing.

After refactoring (failed due to sonarqube incorrectly thinking coverage is 0, JaCoCo works):

Screenshot 1: Quality Gates Status

The Quality Gate Status is **Failed**. Coverage on New Code is less than 0.0% (80.0%).

Screenshot 2: Issues Page

Issues page showing a list of code smells:

- src/main/java/controllers/MainWindowController.java: Make the enclosing method "static" or remove this set. (Code Smell, Critical, Open, Not assigned, 20min effort)
- src/main/java/controllers/MainWindowController.java: Define and throw a dedicated exception instead of using a generic one. (Code Smell, Major, Open, Not assigned, 20min effort)
- src/main/java/controllers/AnalyticsCardController.java: Define and throw a dedicated exception instead of using a generic one. (Code Smell, Major, Open, Not assigned, 20min effort)
- src/main/java/controllers/CreateTeacherDialogController.java: Define and throw a dedicated exception instead of using a generic one. (Code Smell, Major, Open, Not assigned, 20min effort)
- src/main/java/controllers/FlashcardAnalyticsCardController.java: Remove this unused method parameter "performanceRate". (Code Smell, Major, Open, Not assigned, 5min effort)
- src/main/java/controllers/FlashcardAnalyticsCardController.java: Define and throw a dedicated exception instead of using a generic one. (Code Smell, Major, Open, Not assigned, 20min effort)
- src/main/java/controllers/FlashcardAnalyticsCardController.java: Remove this unused method parameter "setStats". (Code Smell, Major, Open, Not assigned, 5min effort)
- src/main/java/controllers/FlashcardAnalyticsCardController.java: Define and throw a dedicated exception instead of using a generic one. (Code Smell, Major, Open, Not assigned, 20min effort)

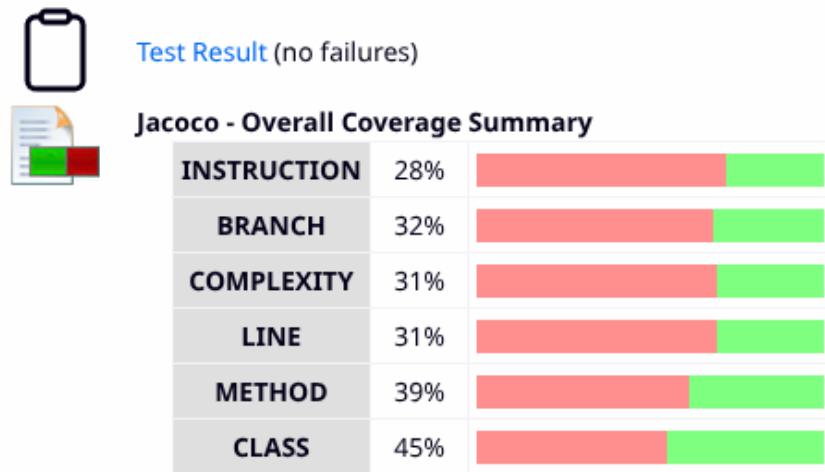
Screenshot 3: Measures Page

Measures page showing a bubble chart for QuizzyCards:

- Risk:** Color: Worse of Reliability Rating and Security Rating; Size: Lines of Code.
- Technical Debt:** 0min, 20min, 30min, 40min, 50min, 1h.
- Coverage:** 0.0%, 20.0%, 40.0%, 60.0%, 80.0%, 100%.

The chart displays several large green bubbles representing significant portions of the codebase with high risk and low coverage.

Code coverage that is not shown on sonarqube:



Could be a lot better, but the whole backend has over 90% coverage, frontend has no coverage. Due to this, the results are skewed.

Final refactoring

Measures after final refactoring:

- 0 issues
- 429 cyclomatic complexity
- 11 lines of code per method
- 0 duplicated lines

SonarQube Project Overview (Quality Gates)

Quality Gate Status: Failed (1 condition failed)

Measures:

- New Code: Since November 18, ... Started 14 days ago
- Overall Code:**
 - Bugs: 0
 - Vulnerabilities: 0
 - Security Hotspots: 0 (100% Reviewed, Security Review A)
 - Debt: 0
 - Code Smells: 0
 - Maintainability: 0
- Coverage: 0.0% (Coverage on 1.5K Lines to cover, Unit Tests: -)
- Duplications: 0.0% (Duplications on 3.2k Lines, Duplicated Blocks: 0)

SonarQube Issues Overview

Filters:

- Issues in new code
- Type: Bug (0), Vulnerability (0), Code Smell (0)
- Severity: Blocker (0), Critical (0), Info (0), Major (0)
- Scope: Resolution, Status, Security Category, Creation Date, Language, Rule, Tag, Directory, File, Assignee, Author

No Issues. Hooray!

SonarQube Measures Overview (QuizzyCards)

Measure Details: QuizzyCards, main, December 2, 2025 at 3:15 PM, Version 1.0-SNAPSHOT

Overview: Coverage (0.0%), Duplications (0.0%), Size (362), Complexity (Cyclomatic Complexity: 429, Cognitive Complexity: 223).

Measures Chart: A bubble chart showing Coverage (Y-axis, 0.0% to 100%) vs. Technical Debt (X-axis, 0 to 1h 20min). The chart shows a single green bubble centered around 0.0% coverage and 0 minutes technical debt.

All issues in the code were fixed, security testing was done and reviewed. Code has no vulnerabilities. All duplications were fixed. SpotBugs tool was also run on the application.

SpotBugs Report

Project Information

Project: QuizzyCards

SpotBugs version: 4.8.3

Code analyzed:

- /home/otto/dev/IdeaProjects/OTP1/target/classes

Metrics

2086 lines of code analyzed, in 50 classes, in 15 packages.

Metric	Total	Density*
High Priority Warnings		0.00
Medium Priority Warnings	26	12.46
Low Priority Warnings	28	13.42
Total Warnings	54	25.89

(* Defects per Thousand lines of non-commenting source statements)

Contents

- [Bad practice Warnings](#)
- [Malicious code vulnerability Warnings](#)
- [Multithreaded correctness Warnings](#)
- [Performance Warnings](#)
- [Dodgy code Warnings](#)
- [Details](#)

SpotBugs report displays 54 warnings, none of which are high priority. The warnings SpotBugs showcased are not critical and a lot of the warnings were false positives.