Análise e Teste de Software

Detection and removal of Code Smells

Universidade do Minho

2020/2021

1 Code Smells

Code Smells are patterns or occurrences in source code that lessen the quality of software, and they may be indicative of deeper problems in the source code. Examples such as duplicated code or incredibly large methods or classes easily illustrate why this is a problem, and not fixing this problem has bad long-term consequences.

There are several tools which can be used to lessen the burden of finding and correcting code smells from the developer. Some examples are SonarQube, which allows us to find code smells, track their impact and associated technical debt, and IntelliJ, which has refactoring tools that allow easy removal of some code smells.

2 Exercises

The following exercises focus on working on one project developed in the Object-Oriented Programming course lectioned last year. It was graded highly, and as such it could be expected to be a well-developed project, with low count of code smells.

Exercise 1. Set up the project to compile with Maven, as it will be needed for SonarQube analysis.

1. Prepare a *pom.xml* file and rearrange the project folders appropriately.
2. Compile and package the executable through the use of the commands `mvn compile` and `mvn package` respectively. Note that, if Manen doesn't find the source code, it will simply mention that no source was found, without displaying an error message.
3. Run the executable generated in the `target/` folder, to ensure that everything is working properly.

4. (Optional) Manually test the program by navigating the menus. Get the program to load up the `logs_original.txt` file. Because the expected usage is not well-documented in the executable program, it might be needed to navigate through the source code and/or change it to get the program to executed properly.

Exercise 2. Use SonarQube to analyze the project. Take note of the number of bugs, code smells and technical debt. Explore the code smells listed in SonarQube.

Some of the listed code smells do not seem to make sense initially - for example, in the class `TrazAqui`, the variable `private Comparator<User> registo` is mentioned as unused, but it is assigned a value in the constructor. This variable, however, is never used besides this assignment. Thus, it is possible to remove the variable and its assignment with no real impact on the program.

Exercise 3. Open the project in IntelliJ. Remove some of the previously obtained code smells, either using automatic techniques or manually. Remove the occurrences of following code smells:

1. Incorrect usage of libraries and/or types, such as calling `.toString()` on a string, not using `.isEmpty()` to check if a collection is empty, unnecessarily using `Boolean` instead of `boolean`...
2. Unused variables and unused method parameters.
3. Replacement of lambdas with method references.
4. High values of Cognitive Complexity.

Exercise 4. Use SonarQube again to analyze the project. Take note of the updated values of number of bugs, code smells and technical debt. It is possible that the number of bugs, code smells and technical debt went up - why?