Reflection - Coverage tools

This Android application was created with the aid of the following development tools:

- Eclipse - development environment chosen for the project.
  - Essential tool for efficiency.
- Android SDK - Android emulator and development tools.
  - Essential tool for testing without a smartphone and for writing Android-specifik code.
- Github - central issue tracker and revision control tool of the project.
  - The issue tracker was used to:
    - Report new bugs and issues.
    - Request new features.
    - Keep track of solved bugs and issues .
    - Assign bugs, issues and features to individual team members.
  - The revision control tool was used to:
    - Get an overview of the development process (branches).
    - Revert to previous states when bad things had happened.
- **Sonar** - for finding issues and viewing statistics.
  - Issues that were wide-spread and cleared with the help of Sonar were, for example, hard-coded values ('magic numbers').
  - The statistics overview was very useful for finding out the amount of comments in the code. It also provided the comment ratio of a class so that team members easily could find classes that were lacking in comments.
- **Checkstyle** - for following code conventions.
- **FindBugs** - for finding code that is prone to bugs.
- **Metrics** - for providing an overview of certain statistics such as lines of code and complexity.

Sonar

This tool was very helpful for finding major issues and also for ensuring that the code becomes easy to follow. One of the results of using this tool was to change hard-coded values (such as 1000, the number of milliseconds in a second) to a static field called MILLISECONDS_IN_SECOND. This helped in both testing and writing utility classes.

Although Sonar pointed out things that felt neglectable (such as *if-cases must use brackets {}*) it ensured that these code conventions were followed.

Sonar was extensively used in this project, and areas such as comment coverage and certain issue categories would not have been found without the help of Sonar.

Checkstyle

Checkstyle was much less useful than Sonar, since issues pointed out here were mostly (if not fully) covered by Sonar already. There were also 'error' indicators for unuseful fixes, such as import orders and "*File contains tab characters*." The presentation of the issues was more difficult to read, which meant that this tool was soon abandoned in favor of Sonar.

FindBugs

This tool proved useful for finding unused fields. Although it was run several times during development, the only issue that came up was the same one; an unimplemented feature.

Metrics

A useful tool for getting an overview of the dependencies, the Metrics plugin also helped by displaying statistics such as complexity. This plugin was updated at every build (unlike Sonar) which helped in quickly removing bigger issues.

CodeAnalyzer

Due to the fact that the proportion of comments within the source code files of the application is taken into account when grading the project, it was of interest to be able to use a tool to calculate the value whenever larger changes were made to the application. The usage of the tool on a regular basis provided a clear overview of the current value in both individual source code files as well as the whole project, and was thus used to monitor and assert that all parts of the application were commented appropriately.