CODING RULES FOR THE SOFTWARE PROJECT COURSE

The coding rules listed below (except for the Implementation Rules) will be tested for your application in the software project course as part of the continuous integration including the static program analysis by SonarQube.
There is a variety of further rules ([1], [2]) that should be considered as well as possible. But please consider at least the following rules in your implementation!

Implementation Rules

(1) The application logic should be implemented in Java.
(2) The basic functions of the web application should be able to run without JavaScript.

Blocker Rules

(3) Source files should not have any duplicated blocks.
(4) Magic numbers should not be used (except 1, 0, -1). A magic number is the usage of a hard-coded number in the code.
(5) Class variable fields should not have public accessibility.

Critical Rules

(6) Files should contain only one top-level class or interface each.
(7) Lines should have sufficient coverage (80%) by unit tests.
(8) Public types, methods and fields (API) should be documented with Javadoc.
(9) Classes should not be coupled to too many other classes (Single Responsibility Principle, at most 10), and there should be no cycles between packages.
(10) Keep things small!
- Classes should not have too many methods (at most 10).
- Files should not have too many lines (at most 400).
- Methods should not have too many lines (at most 40).
- Inner classes should not have too many lines (at most 25).

(11) All names should follow the Java Naming Conventions with at least 2 and at most 50 characters: https://www.safaribooksonline.com/library/view/java-8-pocket/9781491901083/ch01.html

Major Rules

(12) Not too much complexity!
- Methods should not have too many parameters (at most 5).
- Control Statements (If, For, …) should not be nested too deeply (at most 3).
- McCabe Complexity for methods should be at most 10, and for classes at most 40. McCabe Complexity is explained here: https://www.leepoint.net/principles_and_practices/complexity/complexity-java-method.html

(13) Keep things clear! Child class fields should not shadow parent class fields (https://en.wikipedia.org/wiki/Variable_shadowing). That means, names of variables should be different from names used in an outer scope. For example, a local variable

should not have the same name as an instance variable, names should not differ only by capitalization, and literal suffixes (like L for Long) should be Upper Case.

(14) Be consistent in formatting!
- Open curly brace at the end of the line, close curly brace at the beginning of a line.
- Use curly braces in control structures (if, for, …)
- Consistent indentation (4 spaces)
- No tabulation characters
- Maximum line length of 120 characters
- Only one statement per line.

Minor Rules

(15) "static" members should be accessed statically.
(16) Objects should be compared with "equals()".
(17) "switch" statements should end with "default" clauses, and all cases should end with a break-statement.
(18) Useless imports should be removed, and wildcard imports should not be used.
(19) Multiple variables should not be declared on the same line. Visibility should always be declared (public, private, protected).
(20) All unused code should be removed (methods, fields, variables, parameters, …).

Info Rules

(21) "TODO" tags should be handled. Commented out code should be removed.

References

[1] Robert C. Martin: *Clean Code: A Handbook of Agile Software Craftsmanship.* Prentice Hall, 2008
[2] Google Java Style Guide: https://google.github.io/styleguide/javaguide.html