Code Review Checklist - Java

1. Specification / Design

[ ] Is the functionality described in the specification fully implemented by the code?
[ ] Is there any excess functionality in the code but not described in the specification?

2. Initialization and Declarations

[ ] Are all local and global variables initialized before use?
[ ] Are variables and class members of the correct type and appropriate mode
[ ] Are variables declared in the proper scope?
[ ] Is a constructor called when a new object is desired?
[ ] Are all needed import statements included?
[ ] Variable names are spelled correctly and consistently.
[ ] Make sure that primitive data types are not set to null or empty
[ ] Is 'static' keyword used correctly?

3. Method Calls

[ ] Are parameters presented in the correct order?
[ ] Are parameters of the proper type for the method being called?
[ ] Is the correct method being called, or should it be a different method with a similar name?
[ ] Are method return values used properly? Cast to the needed type?
[ ] When calling a method that has a return value, be sure to use the return value properly.

4. Arrays

[ ] Are there any off-by-one errors in array indexing?
[ ] Can array indexes ever go out-of-bounds?
[ ] Is a constructor called when a new array item is desired?
[ ] Are array declarations syntactically correct?
[ ] Are the row and column being indexed in the right order for a 2D array

5. Object Comparision

[ ] Are all objects (including Strings) compared with "equals" and not "=="?

6. Output Format

[ ] Are there any spelling or grammatical errors in displayed output?
[ ] Is the output formatted correctly in terms of line stepping and spacing?

7. Computation, Comparisons and Assignments

[ ] Do all statements end with a semicolon?
[ ] Check order of computation/evaluation, operator precedence and parenthesizing
[ ] Can the denominator of a division ever be zero?
[ ] Is integer arithmetic, especially division, ever used inappropriately, causing unexpected truncation/rounding?
[ ] Check each condition to be sure the proper relational and logical operators are used.
[ ] If the test is an error-check, can the error condition actually be legitimate in some cases?
[ ] Does the code rely on any implicit type conversions?

8. Exceptions

[ ] Are all relevant exceptions caught?
[ ] Is the appropriate action taken for each catch block?

9. Flow of Control

[ ] In a switch statement is every case terminated by break or return?
[ ] Do all switch statements have a default branch?
[ ] Check that nested if statements don't have “dangling else” problems.
[ ] Are all loops correctly formed, with the appropriate initialization, increment and termination expressions?
[ ] Are open-close parentheses and brace pairs properly situated and matched?
[ ] Do logical expresssions evaluate to the correct true or false value?
[ ] Do boolean functions return the correct value?

10. Files

[ ] Are all files properly declared and opened?
[ ] Are all files closed properly, even in the case of an error?
[ ] Are EOF conditions detected and handled correctly?
[ ] Are all file exceptions caught?