# Code Inspection Checklist (Java)

## Variable and constant declarations

1. Are descriptive variable and constant names used in accordance with naming conventions?

   *Descriptive names make code easier to read, understand, and maintain.*

2. Are there variables with confusingly similar names?

   *If this is the case, it would be easy to type one name when you meant another. If the variables are of the same type, the compiler won't detect your mistake.*

3. Is every variable *correctly* initialised?

   *The compiler requires that all variables are initialised, but are they initialised to the right value?*

4. Could any non-local variables be made local?

   *If variables with class scope (i.e. instance variables) are only used by a single method and this method doesn't have to remember its value between calls, the variable can be made local to the method. Unnecessary non-local variables make classes overly complex and thus more difficult to read, understand, and maintain.*

5. Are there literal constants that should be named constants?

   *Replace occurrences of literal numbers, like 0.175 for the VAT rate, with named constants (e.g.* `VAT_RATE` *where this is declared as* `final int VAT_RATE = 0.175`*). Should the VAT rate change, you only need to change the value of the constant in your program, and not every occurrence of 0.175. Using a named constant, you can be sure that the right value is being used in all cases.*

6. Are there variables that should be constants?

   *If a class has a variable whose value isn't intended to change, you should make it a constant so that it can never be changed accidentally. You can then rely on the compiler to detect an attempted modification.*

## Methods

7. Are descriptive method names used in accordance with naming conventions?

*Descriptive names make code easier to read, understand, and maintain.*

8. Is every method parameter (argument) checked before being used?

   *It is good practice for methods to check their argument's values before using them. E.g. when setting a name, check for a null String and don't change the object's state if the argument is null.*

9. For every method, does it return the correct value at every method return point?

## Operators

10. For each expression with more than one operator, are the assumptions about order of evaluation and precedence correct?

    *Use brackets to clear up ambiguities.*

11. Are comparison operators (i.e. < <= > >= ==) correct.

12. Is each boolean expression correct?

*In general, use the equality operator (and not the assignment operator) when writing boolean expressions.*

## Control flow (if, switch, for while)

13. For each loop, is the most appropriate construct used?

    *Use while for loops where you don't know in advance how many iterations are to be made; use for loops in other cases.*

14. Will all loops terminate?

15. Where there are multiple exists from a loop, is each exit point necessary and handled correctly?

16. Does each switch statement have a default case?

    *In general, provide a default label for every switch statement.*

17. Are missing break statements (in a switch construct) correct and marked with a comment?

    *In general follow cases statements with a break.*

18. Do any if structures have dangling else clauses?

    *Avoid dangling else statements.*

## Interfaces

19. Is the order of arguments in every method call in agreement with the method's declaration?

    *Where a method has two arguments of the same type, for example, the compiler can't detect when you call the method that the actual arguments are given in the correct order.*

20. Do the values in units agree?

    *Recall the Mars Climate Orbiter system mentioned in lecture 1. One component supplied Metric values to another that interpreated them (without any conversion) as Imperial units.*

## Comments

21. Does every class and method have an appropriate header comment?

22. Does every variable or constant have a comment?

23. Is the underlying behaviour of each class and method expressed in plain English?

24. Do the comments and code agree?

    *Comments that describe how the code used to work are of little value once the code has been changed.*

25. Do the comments help you to understand the code?

    *Comments should add value to the source code.*

## Input/output

26. Are the spelling or grammatical errors in any text printed or displayed?

27. Are erroneous input data checked?

**Acknowledgement**
This checklist is based on the Java Code Inspection Checklist from Ian Warren.