Java Inspection Checklist	Yes	No
1. Specification / Design		
Is the functionality described in the specification fully implemented by the code?		
Is only specified functionality implemented with no additional functionality added?		
Is the program interface implemented as described in the javadocs		
Are the javadocs complete, including DBC or Error checking specs as appropriate?		
Does the code conform to the class coding standard?		
Is the code correct? Does the code implement the detailed design provided? (Suggestion: perform a hand trace of the execution to verify correctness.		
Does the implementation avoid bonehead programming?		
Is the code free of "smells?" (Duplicate code, long methods, big classes, breaking encapsulation, etc.)		
2. Initialization and Declarations		
Are variables and class members of the correct type and appropriate mode		
Are descriptive variable and constant names used in accord with naming conventions?		
Are there variables or attributes with confusingly similar names?		
Is every variable and attribute correctly typed?		
Is every variable and attribute properly initialized?		
Could any non-local variables be made local?		
Are all for-loop control variables declared in the loop header?		
Are there literal constants that should be named constants?		
Are there variables or attributes that should be constants?		
Are there attributes that should be local variables?		
Do all attributes have appropriate access modifiers (private, protected, public)?		
Are there static attributes that should be non-static or vice-versa		
Are variables declared in the proper scope?		
Is a constructor called when a new object is desired?		
Is a constructor called when a new object is desired?		
Are all object references initialized before use?		
Are all object references initialized before use?		
3. Method Calls		
Are parameters presented in the correct order?		
Is the correct method being called, or should it be a different method with a similar name?		
Are method return values used properly?		
Are descriptive method names used in accord with naming conventions?		
Is every method parameter value checked before being used?		
For every method: Does it return the correct value at every method return point?		
Do all methods have appropriate access modifiers (private, protected, public)?		
Are there static methods that should be non-static or vice-versa?		
4. Class Definition Problems		
Does each class have appropriate constructors and destructors?		
Do any subclasses have common members that should be in the superclass?		
Can the class inheritance hierarchy be simplified?		
5. Arrays		
Are there no off-by-one errors in array indexing?		
Have all array (or other collection) indexes been prevented from going out-of-bounds?		
Is a constructor called when a new array item is desired?		
6. Object Comparison		
Are all objects (including Strings) compared with "equals" and not "=="?		

7. Output Format		
Are displayed outputs free of spelling and grammatical errors?		
Are error messages comprehensive and provide guidance as to how to correct the problem?		
Is the output formatted correctly in terms of line stepping and spacing?		
8. Computation, Comparisons and Assignments		
Check order of computation/evaluation, operator precedence and parenthesizing		
Are all denominators of a division prevented from being zero?		
Is integer arithmetic, especially division, used appropriately to avoid causing unexpected truncation/rounding?		
Are the comparison and Boolean operators correct?		
If the test is an error-check, can the error condition actually be legitimate in some cases?		
Is the code free of any implicit type conversions?		
9. Exceptions		
Are all relevant exceptions caught?		
Is the appropriate action taken for each catch block?		
10. Flow of Control		
In a switch statement, are all cases by break or return?		
Do all switch statements have a default branch?		
Are all loops correctly formed, with the appropriate initialization, increment and termination expressions?		
For each loop: Is the best choice of looping constructs used?		
Will all loops terminate?		
When there are multiple exits from a loop, is each exit necessary and handled properly?		
Does each switch statement have a default case?		
Are missing switch case break statements correct and marked with a comment?		
Do named break statements send control to the right place?		
Is the nesting of loops and branches too deep, and is it correct?		
Can any nested if statements be converted into a switch statement?		
Are all exceptions handled appropriately?		
Does every method terminate?		
11. 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 file exceptions caught?		
Have all files been opened before use?		
Are the attributes of the input object consistent with the use of the file?		
Have all files been closed after use?		
Are there spelling or grammatical errors in any text printed or displayed?		
Are all I/O exceptions handled in a reasonable way?		
12. Data Reference Defects (DR)		
For every array reference: Is each subscript value within the defined bounds?		
For every object or array reference: Is the value certain to be non-null?		
13. Computation/Numeric Defects (CN)		
Are there any computations with mixed data types?		
Is overflow or underflow possible during a computation?		
For each expression with more than one operator: Are the assumptions about order of evaluation and precedence correct?		
Are parentheses used to avoid ambiguity?		
14. Comparison/Relational Defects (CR)		

For every Boolean test: Is the correct condition checked?		
Are the comparison operators correct?		
Has each Boolean expression been simplified by driving negations inward?		
Is each Boolean expression correct?		
Are there improper and unnoticed side-effects of a comparison?		
Has an "&" inadvertently been interchanged with a "&&" or a "|" for a "||"?		
15. Module Interface Defects (MI)		
Are the number, order, types, and values of parameters in every method call in agreement with the called method's declaration?		
Do the values in units agree (e.g., inches versus yards)?		
If an object or array is passed, does it get changed, and changed correctly by the called method?		
16. Comment Defects (CM)		
Does every method, class, and file has an appropriate header comment?		
Does every attribute, variable, and constant declaration has a comment?		
Is the underlying behavior of each method and class expressed in plain language?		
Is the header comment for each method and class consistent with the behavior of the method or class?		
Do the comments and code agree?		
Do the comments help in understanding the code?		
Are there enough comments in the code?		
Are there too many comments in the code?		
17. Layout and Packaging Defects (LP)		
Is a standard indentation and layout format used consistently?		
For each method: Is it no more than about 60 lines long?		
For each compile module: Is no more than about 600 lines long?		
18. Modularity Defects (MO)		
Is there a low level of coupling between modules (methods and classes)?		
Is there a high level of cohesion within each module (methods or class)?		
Is there repetitive code that could be replaced by a call to a method that provides the behavior of the repetitive code?		
Are the Java class libraries used where and when appropriate?		
19. Storage Usage Defects (SU)		
Are arrays large enough?		
Are object and array references set to null once the object or array is no longer needed?		
20. Performance Defects (PE) [Optional]		
Can better data structures or more efficient algorithms be used?		
Are logical tests arranged such that the often successful and inexpensive tests precede the more expensive and less frequently successful tests?		
Can the cost of recompiling a value be reduced by computing it once and storing the results?		
Is every result that is computed and stored actually used?		
Can a computation be moved outside a loop?		
Are there tests within a loop that do not need to be done?		
Can a short loop be unrolled?		
Are there two loops operating on the same data that can be combined into one?		
Are frequently used variables declared register?		
Are short and commonly called methods declared inline?		

Tomado de : http://users.csc.calpoly.edu/~jdalbey/205/Resources/InspectChecklist.html

http://www.fing.edu.uy/inco/cursos/ingsoft/iis/files/JavaChk.html