**Java Code Inspection Checklist**

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](http://www.csc.calpoly.edu/%7Ejdalbey/SWE/CodeSmells/bonehead.html)?  
     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 variables declared in the proper scope?  
     Is a constructor called when a new object is desired?  
     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?

4. Arrays

     Are there no off-by-one errors in array indexing?  
     Hav all array (or other collection) indexes been prevented from going out-of-bounds?  
     Is a constructor called when a new array item is desired?

5. Object Comparision

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

7. Output Format

     Are displayed output 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 parenthesising  
     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?

8. Exceptions

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

9. 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?

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?

Would someone like to merge the items in [Java Pitfalls](http://users.csc.calpoly.edu/~jdalbey/SWE/JavaPitfalls.html) into this checklist?

Revised 2/20/05 so all questions should be answered YES  
CPE 308 [Home](http://www.csc.calpoly.edu/%7Ejdalbey/205)