Politecnico di Milano

Scuola di Ingegneria Industriale e dell'Informazione

M.Sc. in Computer Science and Engineering

GlassFish Server

Code Inspection

Authors:
Angelo Gallarello
Edoardo Longo
Giacomo Locci

December 28, 2015
Version 1.0

Contents

List of Figures

Abstract

This document is the Code Inspection.

1 Assigned Classes

2 Functional Role of Classes

3 List of Issues

3.1 preInvokeNoTx

Naming Convention

1. *Name meaningfulness*:

2. *One char for loops*:

3. *Classes are nouns in camelCase*:

4. *Interfaces are nouns in camelCase*:

5. *Methods are verbs in camelCase*:

6. *Class variables in camelCase (may be preceded by "_")*:

7. *Constants all UPPERCASE (with words separated by "_")*:

Indention

8. *Three or four spaces*:

9. *No tabs*:

Braces

10. *Consistent bracing style*:

11. *Single statements sorrounded by curly braces*:

File Organizaion

12. *Blank lines and optional comments to separate sections*:

13. *Line length ¡= 80 characters (when practical)*:

14. *If it must exceed 80 characters, then they are ¡= 120*:

Wrapping Lines

15. *Line breaks after comma or operator*:

16. *Higher-level breaks are used*:

17. *New statements aligned with same level expressions*:

Comments

18. *Comments adequately explain the purpose of the code*:

19. *Commented code has reason to exist and contains date*:

Java Source Files

20. *Each java source file contains only one public class or interface*:

21. *The public class is the first class or interface in the file*:

22. *External program interfaces are implemented consistently w.r.t. the javadoc*:

23. *Javadoc is complete*:

Package and Import Statements

24. *If any package statements are needed, they should be the first non-comment statements*:

Class and Interface Declarations

25. *Class or interface follows the declaration template:*

    (a) Class/interface documentation comment
    (b) Class/interface statement
    (c) Class/interface implementation comment, if necessary
    (d) Class (static) variables
        i. first *public* class variables
        ii. next *protected* class variables
        iii. next package level (no access modifier)
        iv. last *private* class variables
    (e) Instance variables
        i. first *public* class variables
        ii. next *protected* class variables
        iii. next package level (no access modifier)
        iv. last *private* class variables
    (f) Constructors
    (g) Methods

    :

26. *Methods grouped by functionality*:

27. *Code is freee of duplicates, long methods, big classes, breaking encapsulation, also, coupling and cohesion are adequate*:

Initialization and Declarations

28. *Correct type of variables and class memebers (right visibility)*:

29. *Variables are declared in the proper scope*:

30. *Constructors are called when a new object is desired*:

31. *All object references are initialized before use*:

32. *Variables are initialized where they are declared, unless dependent upon a computation*:

33. *Declarations appear at the beginnig of blocks*:

Method Calls

34. *Parameters are presented in the correct order*:

35. *Correct method is being called*:

36. *Method returned variables are used properly*:

Arrays

37. *Required array elements are accessed through the index*:

38. *Indexes have been prevented from going out-of-bounds*:

39. *Constructors are called when a new array item is desired*:

Object Comparison

40. *All objects are compared with "equals"*:

Output Format

41. *Displayed output is free of spelling and grammatical errors*:

42. *Error messages are comprehensive and provide guidance as to how to correct the problem*:

43. *Output is formatted correctly in terms of line stepping and spacing*:

Computation, Comparisons and Assignments

44. *Implementation avoids "brutish programming"* [1]:

45. *Order of computation/evaluation, operator precedence and parenthesizing*:

46. *Liberal use of parenthesis is used to avoid operator precedence problems.*:

47. *All denominators of a division are prevented from being zero*:

48. *Integer arithmetic, especially division, are used appropriately to avoid causing unexpected truncation/rounding*:

49. *Comparison and Boolean operators are correct*:

50. *Throw-catch expressions, and their error condition is actually legitimate*:

51. *Code is free of any implicit type conversions*:

Exceptions

52. *Relevant exceptions are caught*:

53. *Appropriate action are taken for each catch block*:

Flow of control

54. *In a switch statement, all cases are addressed by break or return*:

55. *All switch statements have a default branch*:

56. *All loops are correctly formed, with the appropriate initialization, increment and termination expressions*:

Files

57. *All files are properly declared and opened*:

58. *All files are closed properly, even in the case of an error*:

59. *EOF conditions are detected and handled correctly*:

60. *All file exceptions are caught and dealt with accordingly*:

[1] See `http://users.csc.calpoly.edu/~jdalbey/SWE/CodeSmells/bonehead.html`.

3.2 getEJBObjectStub

Naming Convention

1. *Name meaningfulness*:

2. *One char for loops*:

3. *Classes are nouns in camelCase*:

4. *Interfaces are nouns in camelCase*:

5. *Methods are verbs in camelCase*:

6. *Class variables in camelCase (may be preceded by "_")*:

7. *Constants all UPPERCASE (with words separated by "_")*:

Indention

8. *Three or four spaces*:

9. *No tabs*:

Braces

10. *Consistent bracing style*:

11. *Single statements sorrounded by curly braces*:

File Organizaion

12. *Blank lines and optional comments to separate sections*:

13. *Line length ¡= 80 characters (when practical)*:

14. *If it must exceed 80 characters, then they are ¡= 120*:

Wrapping Lines

15. *Line breaks after comma or operator*:

16. *Higher-level breaks are used*:

17. *New statements aligned with same level expressions*:

Comments

18. *Comments adequately explain the purpose of the code*:

19. *Commented code has reason to exist and contains date*:

Java Source Files

20. *Each java source file contains only one public class or interface*:
21. *The public class is the first class or interface in the file*:
22. *External program interfaces are implemented consistently w.r.t. the javadoc*:
23. *Javadoc is complete*:

Package and Import Statements

24. *If any package statements are needed, they should be the first non-comment statements*:

Class and Interface Declarations

25. *Class or interface follows the declaration template:*
    (a) Class/interface documentation comment
    (b) Class/interface statement
    (c) Class/interface implementation comment, if necessary
    (d) Class (static) variables
        i. first *public* class variables
        ii. next *protected* class variables
        iii. next package level (no access modifier)
        iv. last *private* class variables
    (e) Instance variables
        i. first *public* class variables
        ii. next *protected* class variables
        iii. next package level (no access modifier)
        iv. last *private* class variables
    (f) Constructors
    (g) Methods
26. *Methods grouped by functionality*:
27. *Code is freee of duplicates, long methods, big classes, breaking encapsulation, also, coupling and cohesion are adequate*:

Initialization and Declarations

28. *Correct type of variables and class memebers (right visibility)*:
29. *Variables are declared in the proper scope*:
30. *Constructors are called when a new object is desired*:
31. *All object references are initialized before use*:
32. *Variables are initialized where they are declared, unless dependent upon a computation*:
33. *Declarations appear at the beginnig of blocks*:

Method Calls

34. *Parameters are presented in the correct order*:
35. *Correct method is being called*:
36. *Method returned variables are used properly*:

Arrays

37. *Required array elements are accessed through the index*:
38. *Indexes have been prevented from going out-of-bounds*:
39. *Constructors are called when a new array item is desired*:

Object Comparison

40. *All objects are compared with "equals"*:

Output Format

41. *Displayed output is free of spelling and grammatical errors*:
42. *Error messages are comprehensive and provide guidance as to how to correct the problem*:
43. *Output is formatted correctly in terms of line stepping and spacing*:

Computation, Comparisons and Assignments

44. *Implementation avoids "brutish programming"* [2]:

45. *Order of computation/evaluation, operator precedence and parenthesizing*:

46. *Liberal use of parenthesis is used to avoid operator precedence problems.*:

47. *All denominators of a division are prevented from being zero*:

48. *Integer arithmetic, especially division, are used appropriately to avoid causing unexpected truncation/rounding*:

49. *Comparison and Boolean operators are correct*:

50. *Throw-catch expressions, and their error condition is actually legitimate*:

51. *Code is free of any implicit type conversions*:

Exceptions

52. *Relevant exceptions are caught*:

53. *Appropriate action are taken for each catch block*:

Flow of control

54. *In a switch statement, all cases are addressed by break or return*:

55. *All switch statements have a default branch*:

56. *All loops are correctly formed, with the appropriate initialization, increment and termination expressions*:

Files

57. *All files are properly declared and opened*:

58. *All files are closed properly, even in the case of an error*:

59. *EOF conditions are detected and handled correctly*:

60. *All file exceptions are caught and dealt with accordingly*:

[2]See `http://users.csc.calpoly.edu/~jdalbey/SWE/CodeSmells/bonehead.html`.

4 Other Problems

Appendices