Software Engineering 2 Project

Code Inspection

POLITECNICO
DI MILANO

Team: Nico Montali, Enrico Fini

Contents

Document Status

Document Title	Code Inspection
Document ID	CI/1.0
Authors	N. Montali, E. Fini
Version	1.0

Changelog

Version	Date	Changes
1.0	01/02/2017	Initial version

Class identification and role

The class we were assigned to inspect is:

../org/apache/ofbiz/base/start/Config.java

Using the Javadoc (click on the link to open the URL) we found out that the class main objective was to

OFBiz server parameters needed on system startup and retrieved from one of the properties files in the start component

So it is a simple configuration class, where many parameters are loaded from files (.properties that can be found in the directory) and can be accessed through public and final instance variables.

Found Issues

We found ****** issues, listed here (and available at the corresponding checklist entry).

Checklist

In this Chapter, every step of the Code Inspection checklist is documented.

3.1 Naming Conventions

1 Variables names are meaningful

2 One-char variables are used only as exceptions.

3 Class names are mixed case with first uppercase letter.

4 No interface is declared here

5 All method names start with verbs and have mixed case

6 All class variable have mixed with starting lower case char (no underscores either)

7 This class has no constants

3.2 Indention

8 Four space are used to indent every time

9 No tab are used in the file

3.3 Braces

10 All braces are in K&R style.

11 Every single statement block is braced correctly

3.4 File Organization

12 Section are well separated by spaces

13 Many lines exceed the 80 char limit even if not necessary. Lines: 74, 86, 114, 116, 121, 198, 201, 228, 249, 268, 270, 278, 280 can be written more elegantly. Other lines exceed 80 char limit but can be accepted.

14 Some lines also exceed 120 char limit. Lines: 74, 228, 268, 270, 280

3.5 Wrapping lines

15 Some operators are inserted after the line break. Lines: 163, 164, 165, 166

16 High-level breaks are used.

17 New statements are not aligned with the beginning of the previous line at the same level Lines: 162, 163, 164, 165, 166

3.6 Comments

18 Comments could explain more. Some method's function could be better explained even if quite simple.

19 There is no commented out code.

3.7 Java Source Files

20 This java contains only one Java Class.

21 The public class is the first class of the file

22 As in the Javadoc, this Class does not expose any external interface, only public variables to be read-only.

23 The Javadoc is practically complete.

3.8 Package and Import Statements

24 The first non comment line is the package, all the imports follows.

3.9

Class and Interface Declarations

25 Declarations follow the order provided.

26 Since methods perform very different tasks, it is difficult to find a correct order.

27 Code is free of duplicates, long methods and big classes.

3.10 Initialization and Declarations

28 All variables and class members are of the correct type. Moreover they have the right visibility.

29 All variables are declared in the proper scope.

30 When new objects are created the constructor is always called.

31 All object references are initialised before use.

32 Variables are initialized where they are declared, in rare cases (e.g. line 60 and 225) they depend on computation so they cannot be initialised before.

33 Declarations always happen at the beginning of a block.

3.11 Method Calls

34 All methods are called properly with all parameters in the correct order.

35 The correct method is always being called.

36 Method returned values are used properly.

3.12 Arrays

37 There are no off-by-one errors (for Lists and ArrayLists the safe method .add() is always used)

38 All array indexes have been prevented from going out-of-bounds.

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

3.13 Object Comparisons

40 Some objects are compared with == instead of equals(). Lines: 162, 163, 164, 167, 171, 174.

3.14 Output Format

41 The displayed output is free of spelling and grammatical errors.

42 The error messages are comprehensive but they do not provide any guidance on how to correct the problem.

43 The output is correctly formatted.

3.15 Computation, Comparisons, Assignments

44 The file is a simple configuration class, hence no expensive computation is carried out. So the implementation of course avoids "brutish programming".

45 The order of computation/evaluation is correct. there are no computations needing parenthesis.

46 Computations are really simple and done one by one, hence there is no need to use parenthesis.

47 There are no denominators in this class.

48 There are only simple integer additions and the two addends are always integers. Hence the implementation of course avoids truncation/rounding.

49 Comparisons and Boolean operations are always correct.

50 Try-Catch exceptions are fine and the error condition is always legitimate.

51 The code is free of implicit type conversions.

3.16 Exceptions

52 All the relevant exceptions are caught.

53 Appropriate actions are taken for each catch block.

3.17 Flow of Control

54 In the switch statement found in line 184, the last case (line 191) does not contain break or return.

55 The switch statement found in line 184 does not have a default branch.

56 All loops are correctly formed, but the while loop (line 246) could be written in a more elegant way.

3.18 Files

57 All files are properly declared and opened.

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

59 All EOF conditions are detected and handled correctly.

60 All file exceptions are caught and dealt with.