Politecnico di Milano

A.A. 2016-2017

Software Engineering 2

Code Inspection

Version 1.2

Prasanth Ravulapalli

Fathima B

Lipika L

February 4th 2016

Contents

Introduction

Code Inspection

Code inspection is the systematic examination (often known as peer review) of computer source code. It is intended to find mistakes overlooked during the initial development phase, with the aim of improving both the overall quality of software and the developers' skills. Reviews are done in various forms such as pair programming, informal walk throughs and

formal inspections. This document contains Code Inspection results produced by following the checklist provided by the professor. The general quality of selected code extracts from a release of the Apache OFBiz project, an open source product for the automation of enterprise processes that includes framework components and business applications for ERP (Enterprise Resource Planning), CRM (Customer Relationship Management) and other business-oriented functionalities.

Assigned JAVA class of the Apache OFBiz project

The assigned class for the purpose of this document is ***CommonWidgetModels.java***. This is found in the package ***org.apache.ofbiz.widget.model***.

Overview OFBiz

Open For Business (OFBiz) is a suite of enterprise applications built on a common architecture using common data, logic and process components.

The tools and architecture of OFBiz make it easy to efficiently develop and maintain enterprise applications. This makes it possible for the creators and maintainers of the project to quickly release new functionality and maintain existing functionality without extensive effort.

The architecture alone makes it easier for you to customize the applications to your needs, but many of the best flexibility points in the system would be meaningless and even impossible if the system was not distributed as open source software. OFBiz is licensed under the Apache License Version 2.0 which grants you the right to customize, extend, modify, repackage, resell, and many other potential uses of the system.

No restrictions are placed on these activities because they are necessary for effective use of this type of software. Unlike other open source licenses, such as the GPL, the changes do not have to be released as open source. There are obvious benefits to contributing certain improvements, fixes, and additions back to the core project, but some changes will involve proprietary or confidential information that must not be released to the public. For this reason, OFBiz uses the Apache License Version 2.0 which

does not require this. For more information on open source licenses see the Open Source Initiative (OSI) website at www.opensource.org.

Functional role of assigned set of classes

In this section, we provide the description of the purpose of the methods that were assigned. In order, to do so we had to explore the functional role of the class.

From here on we represent **CommonWidgetModel** as **CWM**.

CWM is a public class which extends the java lang object. It is a collection of *shared/reused* widget models.

CWM consists of the following classes:

Modifier and Type	**Class and Description**
static class	CommonWidgetModels.AutoEntityParameters
static class	CommonWidgetModels.AutoServiceParameters
static class	CommonWidgetModels.Image
static class	CommonWidgetModels.Link
static class	CommonWidgetModels.Parameter Models the <parameter> element.

The following methods are inherited by CMW from the java lang object are: clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait.

Issues

In this section, we describe the issues found in the assigned methods, supported by the checklist that was provided.

Naming Conventions

- Point 1:
    - Line 86 - Map<String, String> autEntityParams = new HashMap<String, String>(); A better name can be given to the variable since the name is very much closer to the class name which leads to confusion.
    - Everywhere in the code Expander is represented as Exdr, which is misleading and Exdr doesn't help the reader to think about Expander. At least Expdr is encouraged.
- Point 2:
    - There are no one character named variables. No issues found.
- Point 3:
    - No issues found.
- Point 4:
    - No interfaces found, which means no issues.
- Point 5:
    - No issues found.
- Point 6:
    - All the variable names are following the required conventions. No issues found.
- Point 7:
    - Constants are hardcoded everywhere in the code. There are around 60 constants which can be converted into variables or constants with good naming and following the convention of uppercase words separated by spaces. This helps the change a constant at all the places if necessary in one change.

Indentation

- Point 8:
    - 4 spaces are used everywhere in the code as a part of indentation.
- Point 9:
    - No tab spaces are used.

Braces

- Point 10:

    - Consistent braces are used everywhere in the code. “Kernighan and Ritchie” style of indentation is followed in the entire class.
- Point 11:
    - Lines: 94, 157 don’t follow this convention. Rest of the code is fine.

File Organization

- Point 12:
    - Line 84 and 147, the header can be separated with a blank line which makes the code more readable.
    - In all the functions, there can be a possible separation of variable declarations with the rest of the implementation with a blank line.
    - Comments placed inside the functions don’t have a blank line separation.
- Point 13:
    - There are many line in the code which have more than 80 characters in a single line. Few of them can be broken in to multiple statements. The function declarations can’t be broken which is unavoidable.
- Point 14:
    - Line 89, 99, 153, 167, 394 - The sentence can be made as a constant and reduce the size of the statement to less than 120 characters.
    - Line 484 - Unfortunately, this line can’t be reduced anymore. The only way is to give some other names to the variables which may lose the naming convention.
    - Line 597 – This can be reduced by removing ternary operator and use if else conditions.

Wrapping Lines

- Point 15:
    - Line 99, 167 – The break didn’t happen at the comma.
    - Line 484 – Line can be broken at any of the comma to make the statement shorter.
    - Line 597 – Line can be broken at ‘?’ operator.

- Point 16:
    - No issues found.
- Point 17:
    - No issues found.

Comments

- Point 18:
    - Comments are not at all defined. They are placed only at very few places.
    - Line 243, 313, 334, 564 – These lines contain comments which states some fixes in the code. These are to be removed.
- Point 19:
    - There is no code which is commented.

JAVA Source Files

- Point 20:
    - No issues found.
- Point 21:
    - No issues found.
- Point 22:
    - No issues found. https://ci.apache.org/projects/ofbiz/site/javadocs/index.html?overview-summary.html
- Point 23:
    - No issues found. https://ci.apache.org/projects/ofbiz/site/javadocs/index.html?overview-summary.html

Package and Import statements

- Point 24:
    - No issues found.

Class and Interface declarations

- Point 25:
    - A) No documentation found.
    - B) They are in order.
    - C) No comments made.

    - D) Line 62 – Private variable is declared before package class variable.
    - E) No instances.
    - F) No issues found.
    - G) No issues found.
- Point 26:
    - Line 580: getWidth function can be grouped with the getSize function.
- Point 27:
    - In classes Link and Parameter, constructors adapt function overloading.

Initialization and Declarations

- Point 28:
    - No issues found.
- Point 29:
    - No issues found.
- Point 30:
    - No issues found.
- Point 31:
    - No issues found.
- Point 32:
    - No issues found.
- Point 33:
    - No issues found.

Method calls

- Point 34:
    - Parameters are fine and no issues found.
- Point 35:
    - Correct function names are called everywhere in the program. No issues found.
- Point 36:
    - Return values are properly used and no issues found.

Arrays

- Point 37:

    - List elements are accessed using iterators which are found by using contains method. No such issue found.
- Point 38:
    - All loops are made using auto iterator, so out of bounds errors will be taken care.
- Point 39:
    - Line 323, list is not declared with any constructor.

Object Comparison

- Point 40:
    - Other than checking whether the variable is **null**, everywhere in the code compares function is used.

Output Format

- Point 41:
    - There are few minor grammatical errors which can be rectified in the code.
- Point 42:
    - It is difficult for someone to go through the code and find grammatical errors. It is better to have all the errors as constants and save them in some configurators file, so that finding such kind of errors will be more simple and effective.
- Point 43:
    - Line 90, 152 – Output statement is too long and crossed the boundary of 120 characters per line.
    - Line 99 – There should be a breaking of line after '+' operator.

Computations, Comparisons, and Assignments

- Point 44:
    - Line 113 – The variable declaration can be done outside the loop. Since it is just being used as a temporary variable, it gets declared and memory is allocated in every iteration.
    - Line 634 – Instead of having more if-else statements, just having an if statement followed by return value would have reduced number of branches in the code and more easy to debug.
- Point 45:

    - Only case where operator precedence came into scenario is ”||” and “&&” in if-else statements. But this has no issues.
- Point 46:
    - No issues found.
- Point 47:
    - There are no divisors in the program.
- Point 48:
    - There are no arithmetic operations in the program.
- Point 49:
    - Line 70, 136, 347, 348, 349 – value assignment can be improvised.
- Point 50:
    - There is only one try-catch statement in line 167 and the error statement could be little more elaborated.
- Point 51:
    - No issues found.

Exceptions

- Point 52:
    - In every iteration, the code takes the iterator’s value and uses it without any checking done. It could be a null variable which can be finely caught using exceptions.
- Point 53:
    - Only a single catch is found and it is fine. No issues.

Flow of control

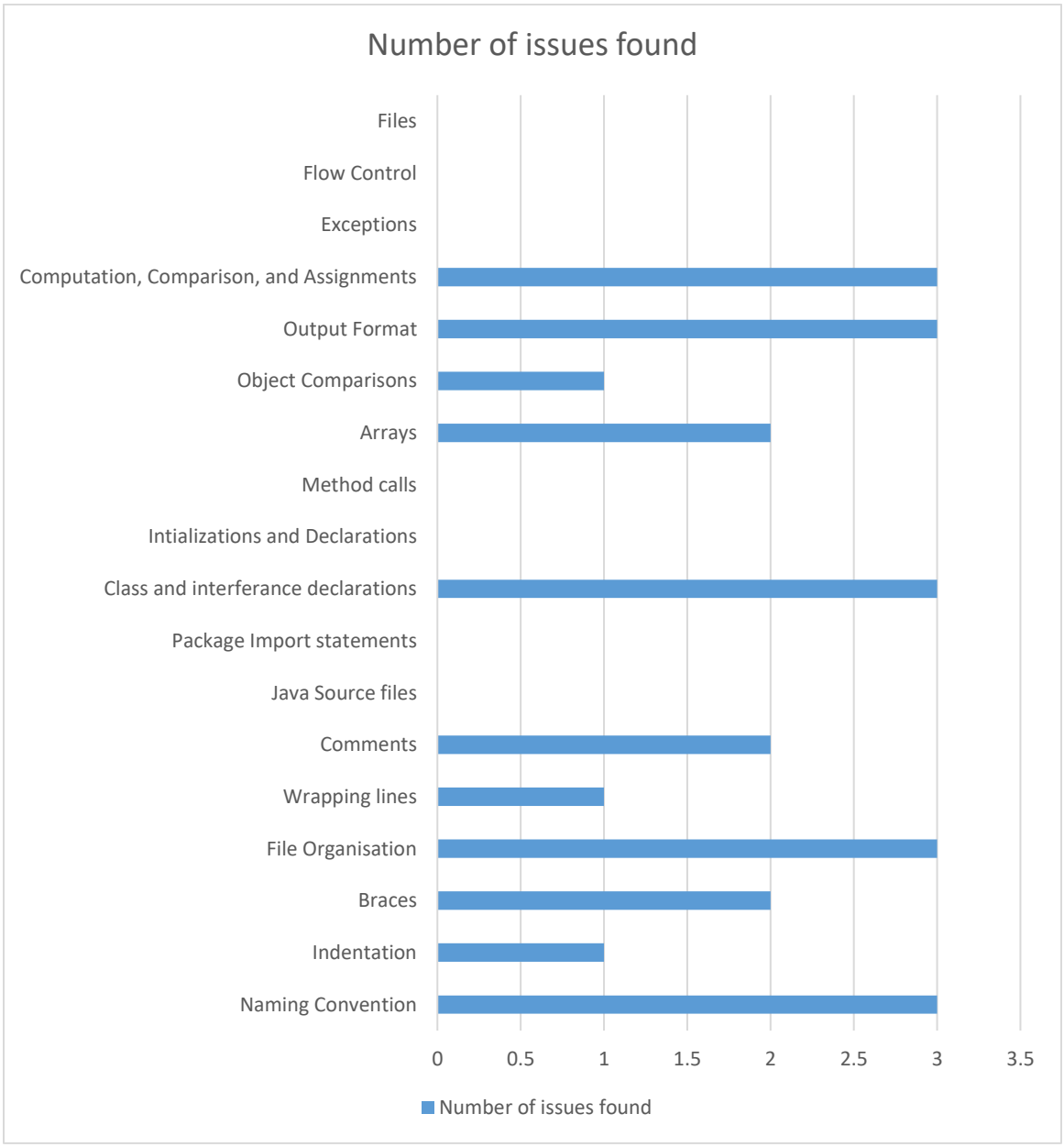
- Point 54:
    - No switch cases found in the program.
- Point 55:
    - No switch cases found in the program.
- Point 56:
    - All loops made using “for” keyword uses iterators which can’t go wrong.
    - All loops made using “while” keyword uses hasNext() function which takes care of termination of loop implicitly.

Files

- Point 57:
  - No files are used in the program to declare or open them properly. No issues found.
- Point 58:
  - No files are opened to close them. No issues found.
- Point 59:
  - No files are used in the program to handle EOF operations. No issues found.
- Point 60:
  - No files are used to create errors in the program to handle them with exceptions. No issues found.

Conclusion

The following chart shows the number of issues found per each section of the check list applied on the assigned module of the project.



Hours of Work

Prasanth R: 14 hrs

Fathima B:

Lipika L:

Change Log:

- V 1.1
    - Created basic structure of the document.
    - Added all initial basic things to the document.
    - Reviewed the source code provided and listed few issues found in the first glance.
- V 1.2
    - Reviewed the code and pointed out issues in detail.
    - Finalized the document.