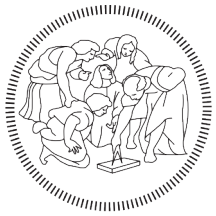

POLITECNICO
MILANO 1863

Code Inspection Document

Version: 1.0

Reda Aissaoui, Jinling Xing, Lidong Zhang

February 5, 2017

Content

1. Introduction

2. Assigned class

3. Functional role of class

4. Code inspection checklist

    1. Naming Conventions
    2. Indention
    3. Braces
    4. File Organization
    5. Wrapping Lines
    6. Comments
    7. Java Source Files
    8. Package and Import Statements
    9. Class and Interface Declarations
    10. Initialization and Declarations
    11. Method Calls
    12. Arrays
    13. Object Comparison
    14. Output Format
    15. Computation, Comparisons and Assignments
    16. Exceptions
    17. Flow of Control
    18. Files

5. Effort spent

1. Introduction

Code inspection is the most formal review type. It is led by the trained moderators. During inspection the documents are prepared and checked thoroughly by the reviewers before the meeting. It involves peers to examine the product. A separate preparation is carried out during which the product is examined and the defects are found.

The main goal of the code inspection is to have an efficient and readable code. This improves the maintainability of the code. It also increases the code-reuse.

2. Assigned Class

Requires taxcommon.class

TaxWare software

Why are they using BigDecimal. Currency calculations require precision to a specific degree, such as two digits after the decimal for most currencies. They also require a specific type of rounding behavior, such as always rounding up in the case of taxes.

Our group assigned class is '../apache-ofbiz-16.11.01/applications/order/src/main/java/org/apache/ofbiz/order
And our task is to perform the inspection and report on the quality status of selected code extracts using the checklist for Java code inspection reported. Also we are asked to deliver a document having the structure described in the code inspection assignments document.

3. Functional role of the Class

The class assigned to us is TaxWareUTL. The class, that is part of the following package **org.apache.ofbiz.order.thirdparty.taxware** . The class is part of an integration third party software, called TaxWare. This software is a solution for calculating the taxes.

The main goal of this integration is to be able to write TaxWare libraries. After exploring the other java files present with this one, we found class TaxwareServices that make use of TaxwareUTL. This class instantiates TaxwareUTL and sets a shipping address, a shipping amount and items. The process() function is then called in order to generate the file.

TaxWare software

Why are they using BigDecimal. Currency calculations require precision to a specific degree, such as two digits after the decimal for most currencies. They also require a specific type of rounding behavior, such as always rounding up in the case of taxes.

4. Code Inspection Checklist

4.1 Naming Conventions

In the following, the name of the variable is meaningless. A more relevant variable name can be chosen.

```
84  Record rec = (Record) i.next();

123 FileOutputStream fos = new FileOutputStream(outFile);

219 DataFile df = null;

272 String headStr = retBuffer.toString().substring(0, 283);
273 String itemStr = retBuffer.toString().substring(284);\end{tabular}

288 Record rec = (Record) i.next();

259 ModelField mf = (ModelField) model.fields.get(a);
```

4.2 Indention

No errors found.

4.3 Braces

The following pieces of code have missing braces for single statement IF.

```
259 if (processed)
260     throw new TaxwareException("Cannot re-process records.");

118 if (Debug.verboseOn()) Debug.logVerbose("::Out String::", module);
119 if (Debug.verboseOn()) Debug.logVerbose("\"" + outBuffer.toString() + "\"", module);

243 if (Debug.verboseOn()) Debug.logVerbose("Taxware Return: " + result, module);
244 if (result != 1)
245     throw new TaxwareException("Taxware processing failed (" + result + ")");
246
247 if (Debug.verboseOn()) Debug.logVerbose("::Return String::", module);
248 if (Debug.verboseOn()) Debug.logVerbose("\"" + inBuffer.toString() + "\"", module);

261 if (!setShipping)
262     throw new TaxwareException("Shipping amount has not been set.");
263 if (shipToAddress == null)
264     throw new TaxwareException("Shipping address has not been set.");
265 if (records.size() == 0)
266     throw new TaxwareException("No items have been defined.");

463 if (Debug.verboseOn()) Debug.logVerbose("Field: " + name + " => " + value, module);
```

4.4 File Organization

The following lines refer to blank lines that are useless: 80, 85, 93, 101, 111, 113, 136, 138, 140, 146, 220, 277, 280, 301, 312, 323, 345, 356, 367, 453, 462.

The line 222 is too long.

The code section starting at line 287 to line 456, contains multiple lines exceeding a length of 80. The lines 296, 354 and 463 exceeding 120 characters.

4.5 Wrapping Lines

4.6 Comments

4.7 Java Source Files

Satisfied

4.8 Package and Import Statements

Satisfied

4.9 Class and Interface Declarations

27.Check that the code is free of duplicates, long methods, big classes, breaking encapsulation, as well as if coupling and cohesion are ade- quate.

There are too many duplicates codes to make the adjustment lists(from numberLine 292), nesting three if-else statements and two of them also nesting a list of parallel if statements.

4.10 Initialization and Declarations

4.11 Method Calls

No significant

4.12 Arrays

```
294 List currentItem = new ArrayList();
```

A new ArrayList is desired, the constructors have not been called.

4.13 Object Comparison

```
263 if (shipToAddress == null)
264    throw new TaxwareException("Shipping address has not been set.");
```

4.14 Output Format

4.15 Computation, Comparisons and Assignments

4.16 Exceptions

The try-catch blocks quoted below do not take any action on the fact that there is a file I/O error. only the stack trace is printed.

```
124 try {
125     fos = new FileOutputStream(outFile);
126 } catch (FileNotFoundException e) {
127     e.printStackTrace();
128 }
```

```
131 try {
132     fos.close();
133 } catch (IOException e) {
134     e.printStackTrace();
135 }
```

4.17 Flow of Control

4.18 Files

5. Effort spent