Politecnico di Milano
A.A. 2015-2016
Software Engineering 2: "myTaxiService"
Code **I**nspection

Roberto Clapis (841859), Erica Stella (854443)

December 31, 2015

Contents

1 Assigned Class

2 Functional Role of Class ConnectorDeployer

3 Found Issues

3.1 Naming Conventions

3.1.1 1

Class name is meaningful;
No interfaces are in the file;
Method names are meaningful, but 2 problems emerged:

- it is suggested to change "deleteRAConfig" in "deleteResourceAdapterConfig";
- it is suggested to change the "event" method name in something more clear, or at least document it.

Class variables are meaningful.
Method variables are meaningful even if the use of names longer than 2 characters is suggested.
Constants names are meaningful but the "EAR" constant may be renamed in "ENTERPRISE_ARCHIVE" to improve readability;

3.1.2 2

Some one-character variables were found, but they were all "e" for exceptions. We cosidered acceptable to have exceptions in catch blocks named "e" since they are throwaway variables and they have a very limited scope length.

3.1.3 3

The file only contains one class and it respects the naming convenction.

3.1.4 4

No interface is declared in the assigned file.

3.1.5 5

All the methods respect the naming convenction.

3.1.6 6

The convenction is respected, but the variable "clh" has a meaningless name, because an acronym is used, but as a 3 letter lowercase word, which can be confusing. It is suggested to rename the variable clHierarchy or classLoaderHierarchy to improve readability.

3.1.7 7

The constants respect the naming convenction.

3.2 Indention

3.2.1 8

Indention is coherent, 4 or multiples of 4 spaces are used consinstently.

3.2.2 9

line 518 uses tabs to indent.

```
513                         try {
514                             if (inputStream != null) {
515                                 inputStream.close();
516                             }
517                         } catch (Exception e) {
518                             // ignore ?
519                         }
```

It is proposed to remove the line or mark it as a "TODO: check if it is adeguate to ignore this"

3.3 Braces

3.4 File Organization

3.4.1 12

A blank line is suggested before line 55 to divide imports of different domains. Remove the two blank lines 216 217 in order to have coherence in spacing. Remove line 299, it is not good to end a method with a blank line. Remove line 342 and 373

3.4.2 13

There are 74 lines longer than 80 chars. Most of them make sense as they are since splitting the line would just reduce readability. Some lines in the file are already splitted when it was practical to do so.

3.4.3 14

Only one line exceedes 120 chars and it is 121 chars long. It is line 278 and since it is a comment and exceedes by only one character two solutions are proposed:

- Leave the line as it is, since splitting it would be a low-level break in the middle of a parenthesis
- split it before the opened parentheses

3.5 Wrapping Lines

3.5.1 15

Line wrappings occour only after the following characters:

`>,=.+`

In lines 640 and 649 lines are broken before a parentheses in order to have a higher-level break so the convenction is respected.

3.5.2 16

Higher level breaks are used.

3.5.3 17

Allignment of new statements are correct.

3.6 Comments

3.6.1 18

The assigned class lacks of documentation. Most of the methods are not documented, and even if some comments to explain what code does are present, they are very few and not present in a constant way but only in some parts of the code. It is suggested to at least provide a line of comment or doc for each method.

3.6.2 19

Line 594 is the only one that has commented-out code and it has a justification to have the code, but not why it is commented out.

3.7 Java Source Files

3.7.1 20

The file ConnectorDeployer.java contains only the ConnectorDeployer public class.

3.7.2 21

The ConnectorDeployer public class is the first and only class in the file.

3.7.3 22

Since the following methods have no documentation this check is not doable.

- event (due to implementation of EventListener)
- preDestroy (due to impelementation of PreDestroy)
- logFine

The other methods are coherent with documentation.

3.7.4 23

As stated in the point above three public methods are not documented, and even if 2 are due to interface implementation at least a line of documentation to explain what the particular method does should be added. Also most of the private methods have no documentation, and it should be provided.

3.8 Package and Import Statements

3.8.1 24

The package statement package com.sun.enterprise.connectors.module; is the first non-comment statement and is followed by the import statements.

3.9 Class and Interface Declarations

3.9.1 25

3.9.1.1 A The class has a documentation comment

3.9.1.2 B The class statement follows his doc

3.9.1.3 C There is no implementation comment

3.9.1.4 D Static variables are after the instance variables, so the convenction is not respected.

3.9.1.5 E There are only private variables

3.9.1.6 F There is the empty constructor

3.9.1.7 G The methods follow the constructor

3.9.2 26

Methods are grouped by functionality.

3.9.3 27

3.10 Initialization and Declarations

3.10.1 28

Variables and class members are of the correct type and they have the right visibility.

3.10.2 29

3.10.3 30

3.10.4 31

3.10.5 32

3.10.6 33

3.11 Method Calls

3.12 Arrays

3.12.1 37

No issues were found regarding array indexing. All arrays and lists are accessed either with an enhanced for or in a while loop with an iterator that starts from the first element and scans all the other elements until there are no more.

3.12.2 38

As explained in the previous point, no issues were found.

3.12.3 39

3.13 Object Comparison

3.13.1 40

No issues were found regarding object comparisons as '==' is never used.

3.14 Output Format

3.14.1 41

The only outputs of the methods that were assigned to us are the entries logged in the logger and they're all free of spelling and grammatical errors.

3.14.2 42

3.14.3 43

3.15 Computation, Comparisons and Assignments

3.15.1 44

No examples of brutish programming have been found.

3.15.2 45

3.15.3 46

3.15.4 47

3.15.5 48

3.15.6 49

3.15.7 50

3.15.8 51

3.16 Exceptions

3.16.1 52

3.16.2 53

Two issues have been found regarding appropriate actions taken in catch blocks because, in the following extracts from two of the methods assigned to us, no actions at all are taken:

- registerBeanValidator:

```
try {
   if (inputStream != null) {
      inputStream.close();
   }
} catch (Exception e) {
   // ignore ?
}
```

- getValidationMappingDescriptors:

```
try {
   reader.close();
} catch (Exception e) {
   //ignore ?
}
```

3.17 Flow of Control

3.17.1 54 and 55

There are no switch statements in the methods that were assigned to us.

3.18 Files

3.18.1 57

All files are declared and opened properly.

3.18.2 58

All opened files are closed in a "finally" block.

3.18.3 59

Since files are read line by line with a buffered reader the condition is correctly checked with a null-check for the read line.

3.18.4 60

Exceptions are just caught and logged. If an exception is thrown while manipulating the file it is correctly closed.

4 Other Highlighted Problems

5 Appendix

Appendix for Roberto Clapis
Work hours: 10

Software Used:

Task	Software
Edit LaTeX Source	Vim
Convert LaTeX source to PDF	LaTeXMK
Spell Check	Aspell
LaTeX Check	LaCheck
Analyze brackets convenction	SonarQube
Analyze naming convenctions	sed, grep, cat, tr

Appendix for Erica Stella
Work hours: 6 @ 14:00 16/12

Software Used:

Task	Software
Edit LaTeX Source	TexStudio
Convert LaTeX source to PDF	LaTeXMK