POLITECNICO
MILANO 1863

Software Engineering 2 project:*PowerEnJoy*

A.A. 2016/2017 - Professor E. di Nitto

Code Inspection

Version **1.0** −02/02/2017

Pietro Avolio	Mat 878640
Guido Borrelli	Mat 874451

Contents

1 Assigned classes **3**

2 Functional role of assigned classes **3**

3 Result of the inspection **4**

3.1 Naming Convenctions 4
3.2 Indention 4
3.3 Braces 4
3.4 File organization 4
3.5 Wrapping Lines 4
3.6 Comments 5
3.7 Java Source Files 5
3.8 Package and Import Statements 5
3.9 Class and Interface Declarations 5
3.10 Initialization and Declarations 5
3.11 Method Calls 6
3.12 Arrays 6
3.13 Object Comparison 6
3.14 Output Format 6
3.15 Computation, Comparison and Assignments 6
3.16 Exception 6
3.17 Flow control 7
3.18 Files 7

4 Other problems **8**

5 Appendix **9**

5.1 Tools Used 9
5.2 Effort Spent 9

1 Assigned classes

The file of the project assigned to our group contains one public **ModelField** class and one public **EncryptMethod** enum, under the **org.apache.ofbiz.entity.model** package.

2 Functional role of assigned classes

As stated in a comment inside the code, this ModelField class implements the field element.

This means that this class is used to store the content of a field of a query result: the instantiated object of the class keeps track if the field is part of the primary key or if its content is null or not.

It is an immutable class: all the attributes are final and there are no setter methods, if an user needs to modify the attribute of the object needs toinstatiate a new one.

The constructor is private and the object can be created only using the overloaded *create* method (factory pattern).

The class can also represent an encrypted field: the encryption method is passed when instantiating the object. There is also a method to export the field by appending it to an XML Document.

3 Result of the inspection

3.1 Naming Convenctions

1. Almost all variables, method and class names are meaningful and their scope can be understood at a first read, except for the global variable isPk, defined on line 207, which stands for isPrimaryKey and which is not immediately understendable without reading the documentation.
2. There are no one-character variables in the classes.
3. All class names are nouns and respect the usual convenction.
4. There are no interfaces defined in the file.
5. All methods names are verbs and respect the usual conventions.
6. All class variables respect the usual conventions.
7. There are no constants defined in the file.

3.2 Indention

1. The entire file is intended using four spaces.
2. No indention is done using the TAB button.

3.3 Braces

1. A consistent bracing style is used in the file: it is the K&R style Java variant, in which the opening brace is on the same line not only for the blocks inside a function, but also for class or method declarations.
2. All if, while, do-while, try-catch, and for statments that contains only one statement uses the curly braces.

3.4 File organization

1. Blank lines are properly used to separate sections. There is an useless blank line inside a documentation fragment on line 35.
2. Many lines, especially methods params declaration and comments, exceed the 80 characters limit.
3. Many lines, especially methods params declaration, exceed the the 120 characters limit.

3.5 Wrapping Lines

1. All lines breaks occur after a comma or an operator.
2. High-level breaks are used.
3. All new statements are alligned with the beginning of the expression at the same level as the previous line.

3.6 Comments

1. There are no comments inside the code. Procedures are almost very straightforward to understand.
2. The only comments out of code are the license at the beginning of the file (line 1 to 18), a comment explaining the absence of the setter methods (line 190 to 192) and two TODO reminders (line 96, line 297).

3.7 Java Source Files

1. The file contains the declaration of a public class (line 39) and of a public enum (line 42). This violates the best practice that states that each java source file should contain a single public class or interface.
2. The public class is the first class in the file.
3. External program interfaces are implemented consistently with what is described in the JavaDoc.
4. The JavaDoc is not complete. JavaDoc is missing for methods on lines 95, 219, 249, 269, 288, 298. Some parameters are not specified (line 82, 173).

3.8 Package and Import Statements

1. The package statement is the first uncommented statement and it is followed by import statements.

3.9 Class and Interface Declarations

1. The class declaration does not follow the common order. The class implementation code is not right after the class statement but it is on line (189). Variables declaration is after the constructors.
2. Methods are grouped by functionality.
3. Code is free from duplicates. The most long method counts 32 lines.

3.10 Initialization and Declarations

1. All variables are of the correct type and have the correct visibility. Plus, they all are final since the entire class is final.
2. All variables are declared in their proper scope.
3. Constructors are called when a new object is desired.
4. All variables are intialized befor use.
5. All variables are initialized where they are declared, unless for class global variables, which are initialized using the constructor.
6. Declarations does not appear at the beginning of a block. Variables are often declared using a logic divisioning.

3.11 Method Calls

1. Parameters are presented in the right order.
2. Correct methods are always called (they mainly are overload methods).
3. Method returned values are used properly.

3.12 Arrays

1. All arrays (implemented with the ArrayList classe) are accessed using the iterator.
2. Same as the previous.
3. Constructors are always called when a new item array is desired.

3.13 Object Comparison

1. The equals method is always used.

3.14 Output Format

1. The only output is a log message (line 152) and it is free of spelling and grammatical errors.
2. Error messages are comprehensive but they do not provide information on how to fix the error.
3. Output is formatted correctly.

3.15 Computation, Comparison and Assignments

1. There are no "brutish programming" implementations.
2. Operators precedence and parenthesizing is correct.
3. There are no precedence problems.
4. There are no integers or other numeric variables.
5. Comparison and Boolean operators are correct.
6. There are no throw-catch expressions.
7. There are no implicit type conversion.

3.16 Exception

1. Not all the relevant exceptions are caught. For example, on line 298, a NullPointerException could be raised if the document param is null.
2. There are not try-catch blocks.

3.17 Flow control

1. There are no switch statements.
2. There are no switch statements.
3. All loops are correctly formed.

3.18 Files

1. No files are used.

4 Other problems

1. There is an useless null check on line 313 since the validators variable is implemented using the List class which always returns an iterator. However, this check could be useful if the implementation of the validators variable will change in the future.

5 Appendix

5.1 Tools Used

- IntelliJ Idea as Java IDE
- Sonar
- Notepad++ to verify if the indentation was realised with tab or four spaces
- Git as version controller

5.2 Effort Spent

Pietro Avolio	4
Guido Borrelli	4