Software Engineering 2: PowerEnJoy
Code Inspection
Version 1.0

Politecnico di Milano, A.A. 2016/2017

Agosti Isabella, 874835
Cattivelli Carolina, 879359

February 5, 2017

Contents

1 ASSIGNED CLASS

The class that was assigned to our group is *Paginator*. In order to find the class:

- Download the source code of Apache OFBiz from the following mirror: `http://mirror.nohup.it/apache/ofbiz/apache-ofbiz-16.11.01.zip`
- Follow this namespace pattern: *../apache-ofbiz-16.11.01/framework/widget /src/main/java/org/apache/ofbiz/widget/renderer/Paginator.java*

2 FUNCTIONAL ROLE

In this paragraph we elaborate on the functional role we have identified for the class that was assigned to us, describing how we managed to understand this role and providing the necessary evidence.

2.1 Paginator class

This class provides functionalities that handle list pagination, used for navigation between pages of a single content, which has been divided into smaller parts. It is public and does not implement any interface.

The class' JavaDoc says: *Utility methods for handling list pagination.*

3 LIST OF ISSUES

In this paragraph we report the code fragments that do not fulfill some points in the check list, explaining which point is not fulfilled and why.

3.1 Paginator class

- **L.2→8, L.10, L.13→16** do not break after a comma, nor an operator.
- **L.18** exceeds 80 characters.
- **L.38**, the comment does not adequately explain what the class does.
- **L.43**, the *module* attribute is a constant, so it should be declared using all uppercase. Also, it would be better if the *module* attribute was private and invoked using getter and setter.

3.2 getActualPageSize: Line 45 ⇒ Line 48

- This method is not commented.
- **L.46**, the *value* variable should be of type *int* instead of *Integer*.
- **L.47**, the use of Functional Java causes low readability. Also, the line exceeds 80 characters.

3.3 getHighIndex: Line 50 ⇒ Line 53

- This method is not commented.
- **L.51**, the *value* variable should be of type *int* instead of *Integer*.
- **L.52**, the use of Functional Java causes low readability.

3.4 getListLimits: Line 55 ⇒ Line 100

- This method is not commented.
- **L.55, L.75** exceed 80 characters.
- **L.61, L.65, L.76** have only one statement to execute and it is surrounded by curly braces, but the opened curly brace is on the same line of the statement (differently from what is shown in the *Code Inspection Assignment Task Description.pdf* example).

- **L.62** does not have a reason for being commented out, nor a date identifying when it can be removed from the source file if determined it is no longer needed.
- **L.68**, the error message is comprehensive (prints the exception, a description of the error and the name of the class where the error occurred) but does not provide guidance as to how to correct the problem.
- **L.74, L.76** both use the "Kernighan and Ritchie" bracing style (first brace is on the same line of the instruction that opens the new block), but there is no blank space between the if condition and the curly brace.

3.5 getListSize: Line 102 ⇒ Line 105

- This method is not commented.
- **L.103**, the *value* variable should be of type *int* instead of *Integer*.
- **L.104**, the use of Functional Java causes low readability.

3.6 getLowIndex: Line 107 ⇒ Line 110

- This method is not commented.
- **L.108**, the *value* variable should be of type *int* instead of *Integer*.
- **L.109**, the use of Functional Java causes low readability.

3.7 getViewIndex: Line 112 ⇒ Line 142

- This method is not commented.
- **L.112, L.119, L.121, L.139** exceed 80 characters.
- **L.117→127**, three nested if cause low readability.
- **L.119**, the *cast* method returns a *V* object, but the *parameters* variable is of type *Map<String, Object>*.
- **L.123, L.133, L.135, L.138** have only one statement to execute and it is surrounded by curly braces, but the opened curly brace is on the same line of the statement (differently from what is shown in the *Code Inspection Assignment Task Description.pdf* example).

- **L.139**, the error message is comprehensive (prints the exception, a description of the error and the name of the class where the error occurred) but does not provide guidance as to how to correct the problem.

3.8 getViewSize: Line 144 ⇒ Line 174

- This method is not commented.
- **L.144, L.151, L.153, L.167, L.171** exceed 80 characters.
- **L.151**, the *cast* method returns a *V* object, but the *parameters* variable is of type *Map<String, Object>*.
- **L.155, L.165, L.167, L.170** have only one statement to execute and it is surrounded by curly braces, but the open curly brace is on the same line of the statement (differently from what is shown in the *Code Inspection Assignment Task Description.pdf* example).
- **L.171**, the error message is comprehensive (prints the exception, a description of the error and the name of the class where the error occurred) but does not provide guidance as to how to correct the problem.

3.9 preparePager: Line 176 ⇒ Line 242

- This method is not commented.
- This method is too long (67 lines of code).
- **L.176, L.206, L.231, L.239** exceed 80 characters.
- **L.180, L.186, L.239**, the error message is comprehensive (prints a description of the error, the name of the class where the error occurred and sometimes the exception) but does not provide guidance as to how to correct the problem.
- **L.183, L.191, L.203, L.204, L.205, L.218, L.219**, variables are not declared at the beginning of any block, instead they are declared in the middle.
- **L.185, L.214** have only one statement to execute but it is not surrounded by curly braces.
- **L.186** exceeds 120 characters.

- **L.187**, the new statement is not aligned with the beginning of the expression at the same level as the previous line (the *module* word should be aligned with the open parenthesis of the previous line).
- **L.192, L.194, L.196, L.236, L.238** have only one statement to execute and it is surrounded by curly braces, but the open curly brace is on the same line of the statement (differently from what is shown in the *Code Inspection Assignment Task Description.pdf* example).
- **L.195**, the *listIterator* method returns a *ListIterator* but the *iter* variable is of type *Iterator.*
- **L.206, L.225** represent commented out code that contains a reason for being commented out, but it does not provide a date it can be removed from the source file if determined it is no longer needed.
- **L.231**, the use of Functional Java causes low readability.

3.10 safeNext: Line 244 ⇒ Line 250

- This method is not commented.
- **L.245, L.247** have only one statement to execute and it is surrounded by curly braces, but the open curly brace is on the same line of the statement (differently from what is shown in the *Code Inspection Assignment Task Description.pdf* example).
- **L.248**, the exception in the catch block is not logged, nor handled.

3.11 getViewIndex: Line 252 ⇒ Line 259

- This method should return *int* instead of *Integer.*
- **L.255, L.257** exceed 80 characters.

3.12 getViewIndex: Line 261 ⇒ Line 269

- This method should return *int* instead of *Integer.*
- **L.265** exceeds 80 characters.
- **L.267** exceeds 120 characters.

3.13 getViewSize: Line 271 ⇒ Line 283

- This method should return *int* instead of *Integer*.
- **L.274** does not break after a comma, nor an operator.
- **L.277, L.278** exceed 80 characters.
- **L.278**, the *getPropertyAsInteger* method returns an *Integer* but the *defaultSize* variable is of type *int*.
- **L.279** has only one statement to execute and it is surrounded by curly braces, but the open curly brace is on the same line of the statement (differently from what is shown in the *Code Inspection Assignment Task Description.pdf* example).

4 OTHER ISSUES

In this paragraph we list all the parts of code that we think create or may create a bug (or minor issues) and we explain why.

- Since the methods *getLowIndex* and *getHighIndex* do more or less the same thing, they should have been put one after the other to improve readability.
- All the *get* methods (getter) should be grouped together, instead of having some at the beginning and some others at the end.
- The method *getViewIndex* at **L.267** should be eliminated and incorporated into the method *getViewIndex* at **L.257**.
- The methods *getListSize*, *getHighIndex* and *getLowIndex* do the exact same thing and could be incorporated into a single method with the addition of a second *String* parameter to be passed to the *get* method called on the *context*.
- The methods *getViewIndex* and *getViewSize* contain duplicated code.