Politecnico Di Milano

A.A. 2015/2016

Software Engineering 2:

Inspection Document

Paramithiotti Andrea (Matr. 788794) Rompani Andrea (Matr. 854052)

Zoia Lorenzo (Matr. 852392)

v1.0

5/1/2016

Contents

[2 Assigned Methods 4](#_Toc439610947)

[3 Functional Role of the Methods 4](#_Toc439610948)

[3.1 readAlternativeRuntimeDescriptor 4](#_Toc439610949)

[3.1.1 Parameters 4](#_Toc439610950)

[3.1.2 Functional Role 4](#_Toc439610951)

[3.2 readRuntimeDeploymentDescriptor 5](#_Toc439610952)

[3.2.1 Parameters 5](#_Toc439610953)

[3.2.2 Functional Role 5](#_Toc439610954)

[3.3 getSniffersFromModule 5](#_Toc439610955)

[3.3.1 Parameters 5](#_Toc439610956)

[3.3.2 Functional Role 5](#_Toc439610957)

[3.4 getTypeFromModuleType 6](#_Toc439610958)

[3.4.1 Parameters 6](#_Toc439610959)

[3.4.2 Functional Role 6](#_Toc439610960)

[3.5 getConfigurationDeploymentDescriptorFiles 6](#_Toc439610961)

[3.5.1 Parameters 6](#_Toc439610962)

[3.5.2 Functional Role 6](#_Toc439610963)

[3.6 setElementValue 6](#_Toc439610964)

[3.6.1 Parameters 6](#_Toc439610965)

[3.6.2 Functional Role 6](#_Toc439610966)

[4 Checklist 7](#_Toc439610967)

[4.1 Naming Conventions 7](#_Toc439610968)

[4.2 Indention 7](#_Toc439610969)

[4.3 Braces 7](#_Toc439610970)

[4.4 File Organization 8](#_Toc439610971)

[4.5 Wrapping Lines 8](#_Toc439610972)

[4.6 Comments 8](#_Toc439610973)

[4.7 Java Source Files 8](#_Toc439610974)

[4.8 Package and Import Statements 9](#_Toc439610975)

[4.9 Class and Interface Declarations 9](#_Toc439610976)

[4.10 Initialization and Declarations 9](#_Toc439610977)

[4.11 Method Calls 10](#_Toc439610978)

[4.12 Arrays 10](#_Toc439610979)

[4.13 Object Comparison 10](#_Toc439610980)

[4.14 Output Format 10](#_Toc439610981)

[4.15 Computation, Comparisons and Assignments 10](#_Toc439610982)

[4.16 Exceptions 11](#_Toc439610983)

[4.17 Flow of Control 11](#_Toc439610984)

[4.18 Files 11](#_Toc439610985)

[5 List of Issues 12](#_Toc439610986)

[5.1 Class Issues 12](#_Toc439610987)

[5.2 readAlternativeRuntimeDescriptor() 13](#_Toc439610988)

[5.3 readRuntimeDeploymentDescriptor() 15](#_Toc439610989)

[5.4 getSniffersFromModule() 17](#_Toc439610990)

[5.5 getTypeFromModuleType() 19](#_Toc439610991)

[5.6 getConfigurationDeploymentDescriptorFiles() 21](#_Toc439610992)

[5.7 setElementValue() 23](#_Toc439610993)

[6 Other Problems 24](#_Toc439610994)

# Assigned Methods

All the methods assigned to the group belong to the class **DOLUtils**, in the package **com.sun.enterprise.deployment.util.**

1. readAlternativeRuntimeDescriptor()
2. readRuntimeDeploymentDescriptor()
3. getSniffersFromModule()
4. getTypeFromModuleType()
5. getConfigurationDeploymentDescriptorFiles()
6. setElementValue()

# Functional Role of the Methods

The Javadoc for the classes and the methods assigned was almost non-existent, thus the functional role of the methods was understood only by analysing the code line by line, trying to extrapolate the meaning of the actions performed also by reading the code of other classes as well.

## readAlternativeRuntimeDescriptor

### Parameters

* ReadableArchive appArchive
* ReadableArchive embeddedArchive
* Archivist archivist
* BundleDescriptor descriptor
* String altDDPath

### Functional Role

The Archivist contains the info for the deployment of determined modules. If the deployment method exists, then it is the one that is used. Otherwise this method is called, which after some control on the parameters takes an alternative descriptor from the parameters at runtime and saves it in the Archivist.

## readRuntimeDeploymentDescriptor

### Parameters

* List<ConfigurationDeploymentDescriptorFile> confDDFiles,
* ReadableArchive archive
* RootDeploymentDescriptor descriptor
* Archivist main
* final boolean warnIfMultipleDDs

### Functional Role

In the case that the parameter warnIfMultipleDDs is true, the method writes a warning on the depLogger for each element in the list confDDFiles. Then it takes the first ConfigurationDDFile in the list and sets its XML validation and its level of validation to true if confDD.isValidating() is true, otherwise it is set to false. The validation is taken from the Archivist that is passed as a parameter. In the end it calls the method Read() on the Descriptor passed as a parameter and on the InputStream that is taken from the Archivist passed as a parameter.

## getSniffersFromModule

### Parameters

* ServiceLocator habitat
* ReadableArchive archive,
* ModuleDescriptor md
* Application app

### Functional Role

The method saves in the Archive passed as a parameter the list of the sniffers that are compatible with the module passed as a parameter, then returns it.

## getTypeFromModuleType

### Parameters

* ArchiveType moduleType

### Functional Role

The method returns a String that defines the type of the document passed as a parameter. It makes the mapping between a string and a moduleType

## getConfigurationDeploymentDescriptorFiles

### Parameters

* ServiceLocator habitat
* String containerType

### Functional Role

The method returns the configuration files useful for the deployment descriptors. In the list that is returned it saves only the elements for whom the indexedType of the descriptor of each handles of the habitat is equals to the type passed as a parameter.

## setElementValue

### Parameters

* XMLElement element
* String value
* Object o

### Functional Role

Regardless of what the method does, it is never called by any function in all the project, so it is a useless dead part of the code. Sometimes another method similar to this, but with different parameters, is called in the project

# Checklist

Here is the checklist that was followed to edit this document. Some of the possible issues referred to the entire class or even to the entire project, so it was impossible to check if those issues were present or not.

The issues that couldn’t be checked are labeled with a question mark in the tables and colored in red in the checklist below.

## Naming Conventions

1. All class names, interface names, method names, class variables, method variables, and constants used should have meaningful names and do what the name suggests.
2. If one-­‐character variables are used, they are used only for temporary “throwaway” variables, such as those used in for loops.
3. Class names are nouns, in mixed case, with the first letter of each word in capitalized. Examples: class Raster; class ImageSprite;
4. Interface names should be capitalized like classes.
5. Method names should be verbs, with the first letter of each addition word capitalized. Examples: getBackground(); computeTemperature().
6. Class variables, also called attributes, are mixed case, but might begin with an underscore (‘\_’) followed by a lowercase first letter. All the remaining words in the variable name have their first letter capitalized. Examples: \_windowHeight, timeSeriesData.
7. Constants are declared using all uppercase with words separated by an underscore. Examples: MIN\_WIDTH; MAX\_HEIGHT; (ci sono errori)

## Indention

1. Three or four spaces are used for indentation and done so consistently
2. No tabs are used to indent

## Braces

1. Consistent bracing style is used, either the preferred “Allman” style (first brace goes underneath the opening block) or the “Kernighan and Ritchie” style (first brace is on the same line of the instruction that opens the new block).
2. All if, while, do-­‐while, try-­‐catch, and for statements that have only one statement to execute are surrounded by curly braces. Example: Avoid this:

if ( condition )

doThis();

Instead do this:

if ( condition )

{

doThis();

}

## File Organization

1. Blank lines and optional comments are used to separate sections (beginning comments, package/import statements, class/interface declarations which include class variable/attributes declarations, constructors, and methods).
2. Where practical, line length does not exceed 80 characters.
3. When line length must exceed 80 characters, it does NOT exceed 120 characters.

## Wrapping Lines

1. Line break occurs after a comma or an operator.
2. Higher-­‐level breaks are used.
3. A new statement is aligned with the beginning of the expression at the same level as the previous line.

## Comments

1. Comments are used to adequately explain what the class, interface, methods, and blocks of code are doing.
2. Commented out code contains a reason for being commented out and a date it can be removed from the source file if determined it is no longer needed.

## Java Source Files

1. Each Java source file contains a single public class or interface.
2. The public class is the first class or interface in the file.
3. Check that the external program interfaces are implemented consistently with what is described in the javadoc.
4. Check that the javadoc is complete (i.e., it covers all classes and files part of the set of classes assigned to you).

## Package and Import Statements

24. If any package statements are needed, they should be the first non-­comment statements. Import statements follow.

## Class and Interface Declarations

1. The class or interface declarations shall be in the following order:
   1. class/interface documentation comment
   2. class or interface statement
   3. class/interface implementation comment, if necessary
   4. class (static) variables OK
      1. first public class variables
      2. next protected class variables
      3. next package level (no access modifier)
      4. last private class variables
   5. instance variables
      1. first public instance variables
      2. next protected instance variables
      3. next package level (no access modifier)
      4. last private instance variables
   6. constructors G. methods
2. Methods are grouped by functionality rather than by scope or accessibility.
3. Check that the code is free of duplicates, long methods, big classes, breaking encapsulation, as well as if coupling and cohesion are adequate.

## Initialization and Declarations

1. Check that variables and class members are of the correct type. Check that they have the right visibility (public/private/protected)
2. Check that variables are declared in the proper scope
3. Check that constructors are called when a new object is desired
4. Check that all object references are initialized before use
5. Variables are initialized where they are declared, unless dependent upon a computation
6. Declarations appear at the beginning of blocks (A block is any code surrounded by curly braces “{“ and “}” ). The exception is a variable can be declared in a ‘for’ loop.

## Method Calls

1. Check that parameters are presented in the correct order
2. Check that the correct method is being called, or should it be a different method with a similar name
3. Check that method returned values are used properly

## Arrays

1. Check that there are no off-­‐by-­‐one errors in array indexing (that is, all required array elements are correctly accessed through the index)
2. Check that all array (or other collection) indexes have been prevented from going out-­‐of-­‐bounds
3. Check that constructors are called when a new array item is desired

## Object Comparison

40. Check that all objects (including Strings) are compared with "equals" and not with "=="

## Output Format

1. Check that displayed output is free of spelling and grammatical errors
2. Check that error messages are comprehensive and provide guidance as to how to correct the problem
3. Check that the output is formatted correctly in terms of line stepping and spacing

## Computation, Comparisons and Assignments

1. Check that the implementation avoids “brutish programming
2. Check order of computation/evaluation, operator precedence and parenthesizing
3. Check the liberal use of parenthesis is used to avoid operator precedence problems.
4. Check that all denominators of a division are prevented from being zero
5. Check that integer arithmetic, especially division, are used appropriately to avoid causing unexpected truncation/rounding
6. Check that the comparison and Boolean operators are correct
7. Check throw-­‐catch expressions, and check that the error condition is actually legitimate
8. Check that the code is free of any implicit type conversions

## Exceptions

1. Check that the relevant exceptions are caught
2. Check that the appropriate action are taken for each catch block

## Flow of Control

1. In a switch statement, check that all cases are addressed by break or return
2. Check that all switch statements have a default branch
3. Check that all loops are correctly formed, with the appropriate initialization, increment and termination expressions

## Files

1. Check that all files are properly declared and opened
2. Check that all files are closed properly, even in the case of an error
3. Check that EOF conditions are detected and handled correctly
4. Check that all file exceptions are caught and dealt with accordingly

# List of Issues

## Class Issues

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Respected | Lines | Code | Possible solution |
| 3 | button_ok.png |  |  |  |
| 4 | button_ok.png |  |  |  |
| 6 | button_ok.png |  |  |  |
| 7 | button_cancel.png | 132 | Constant is written in lower case letters |  |
| 12 | button_cancel.png | 127-130, 150-153, 155-157 | these class attributes are divided by optional comments |  |
|  | 313-315, 319-320, 324-38, 402-406, 423-425, 538-539, 616-618 | these class methods are divided by optional comments | All the optional comments should be replaced by Javadoc comments |
| 20 | button_ok.png |  |  |  |
| 21 | button_ok.png |  |  |  |
| 22 | button_ok.png |  |  |  |
| 23 | button_cancel.png |  | It cover only the class declaration and some public methods |  |
| 24 | button_ok.png |  |  |  |
| 25 | button_cancel.png |  | Subpoint D, subpoint a, lines 132,145 should be moved upper  Subpoint D, subpoint d, NO |  |
| 26 | button_cancel.png |  | * setElementValue should go below * getTypeForModuleType should go with the other methods that handle "Type" * getConfigurationDeploymentDescriptorFile should go with the other methods that handle "DD" |  |
| 27 | button_ok.png |  |  |  |
| 28 | button_ok.png |  |  |  |
| 36 | button_ok.png |  | we don't know because there is no javadoc in the methods that are called and our methods are not called in the code assigned to us |  |

## readAlternativeRuntimeDescriptor()

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Respected | Lines | Code | Possible solution |
| 1 | button_ok.png |  |  |  |
| 2 | button_ok.png |  |  |  |
| 3 | button_ok.png |  |  |  |
| 5 | button_ok.png |  |  |  |
| 7 | button_ok.png |  |  |  |
| 8 | button_ok.png |  |  |  |
| 9 | button_ok.png |  |  |  |
| 10 | button_ok.png |  | Kernighan and Ritchie style |  |
| 11 | button_ok.png |  |  |  |
| 12 | button_ok.png |  |  |  |
| 13 | button_cancel.png | 425 | Method declaration can be split on multiple lines |  |
|  | 429 | But cannot be split on multiple lines |  |
| 14 | button_cancel.png | 425  430 | These 4 lines exceed the 120 characters |  |
| 15 | button_cancel.png | 451-452 | No need to split the row |  |
| 453-454 | No need to split the row |  |
| 441-442 | The && operator must be on line 441 |  |
| 16 | button_ok.png |  |  |  |
| 17 | button_ok.png |  |  |  |
| 18 | button_cancel.png | 423-424 | Presence of a comment that isn’t in the Javadoc format but explains what the method does |  |
| 434 | Presence of a TODO |  |
| 19 | button_ok.png |  |  |  |
| 29 | button_ok.png |  |  |  |
| 30 | button_ok.png |  |  |  |
| 31 | button_ok.png |  |  |  |
| 32 | button_ok.png |  |  |  |
| 33 | button_ok.png |  |  |  |
| 34 | button_ok.png |  | It respects the order of parameters used for the other methods |  |
| 35 | button_ok.png |  |  |  |
| 37 | button_ok.png |  |  |  |
| 38 | button_ok.png |  |  |  |
| 39 | button_ok.png |  |  |  |
| 40 | button_ok.png |  |  |  |
| 41 |  |  | No output |  |
| 42 |  |  | No output |  |
| 43 |  |  | No output |  |
| 44 | button_ok.png |  |  |  |
| 45 | button_ok.png |  |  |  |
| 46 | button_ok.png |  |  |  |
| 47 | button_ok.png |  |  |  |
| 48 | button_ok.png |  |  |  |
| 49 | button_ok.png |  |  |  |
| 50 | button_ok.png |  |  |  |
| 51 | button_ok.png |  |  |  |
| 52 | button_cancel.png | 450-461 | There is no try-catch block for IOException like a similar code that executes the same confDD.read() method on a different part of the class file | Insert a try-catch-finally block |
| 53 | button_ok.png |  |  |  |
| 54 | button_ok.png |  |  |  |
| 55 | button_ok.png |  |  |  |
| 56 | button_ok.png |  |  |  |
| 57 | button_ok.png |  |  |  |
| 58 | button_ok.png |  |  |  |
| 59 | button_ok.png |  |  |  |
| 60 | button_ok.png |  |  |  |

## readRuntimeDeploymentDescriptor()

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Respected | Lines | Code | Possible solution |
| 1 | button_cancel.png | 480 | Archivist main | Rename the parameter |
| 2 | button_ok.png |  |  |  |
| 3 | button_ok.png |  |  |  |
| 4 |  |  |  |  |
| 5 | button_ok.png |  |  |  |
| 6 |  |  |  |  |
| 7 | button_ok.png |  |  |  |
| 8 | button_cancel.png | 509-511  512 |  |  |
|  | 502-504 |  |  |
| 9 | button_ok.png |  |  |  |
| 10 | button_ok.png |  | Kernighan and Ritchie style |  |
| 11 | button_ok.png |  |  |  |
| 12 | button_ok.png |  |  |  |
| 13 | button_cancel.png | 480 | Method declaration can be split on multiple lines |  |
|  | 489 | If statement condition can be split on multiple lines |  |
| 14 | button_ok.png | 480  489  502  507 | These 4 lines not exceed the 120 characters |  |
| 15 | button_ok.png |  |  |  |
| 16 | button_ok.png |  |  |  |
| 17 | button_ok.png |  |  |  |
| 18 | button_ok.png |  |  |  |
| 19 | button_ok.png |  |  |  |
| 29 | button_ok.png |  |  |  |
| 30 | button_ok.png |  |  |  |
| 31 | button_ok.png |  |  |  |
| 32 | button_ok.png |  |  |  |
| 33 | button_cancel.png | 485 |  | This declaration must be at the beginning of the block |
| 34 | button_ok.png |  | It respects the order of parameters used for the other methods |  |
| 35 | button_ok.png |  |  |  |
| 37 | button_ok.png |  |  |  |
| 38 | button_ok.png |  | If confDDFiles is an empty list this method will exit at the start  Also if confDDFiles.size() returns a value of 1 or less the for cycle of row 497-506 won’t be executed |  |
| 39 | button_ok.png |  |  |  |
| 40 | button_ok.png |  |  |  |
| 41 |  |  | No output |  |
| 42 |  |  | No output |  |
| 43 |  |  | No output |  |
| 44 | button_ok.png |  |  |  |
| 45 | button_ok.png |  |  |  |
| 46 | button_ok.png |  |  |  |
| 47 | button_ok.png |  |  |  |
| 48 | button_ok.png |  |  |  |
| 49 | button_ok.png |  |  |  |
| 50 | button_ok.png |  |  |  |
| 51 | button_ok.png |  |  |  |
| 52 | button_ok.png |  |  |  |
| 53 | button_cancel.png | 515-522 |  | Log the exception instead of leaving the catch block empty |
| 54 | button_ok.png |  |  |  |
| 55 | button_ok.png |  |  |  |
| 56 | button_ok.png |  |  |  |
| 57 | button_ok.png |  |  |  |
| 58 | button_ok.png |  |  |  |
| 59 | button_ok.png |  |  |  |
| 60 | button_ok.png |  |  |  |

## getSniffersFromModule()

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Respected | Lines | Code | Possible solution |
| 1 | button_ok.png |  |  |  |
| 2 | button_ok.png |  |  |  |
| 3 | button_ok.png |  |  |  |
| 4 |  |  |  |  |
| 5 | button_ok.png |  |  |  |
| 6 |  |  |  |  |
| 7 | button_ok.png |  |  |  |
| 8 | button_ok.png |  |  |  |
| 9 | button_ok.png |  |  |  |
| 10 | button_ok.png |  | Kernighan and Ritchie style |  |
| 11 | button_ok.png |  |  |  |
| 12 | button_ok.png |  |  | Other blank lines could have been used |
| 13 | button_cancel.png | 540  545  546  547  553  572  593 |  | Some are necessary due to the multiple parameters requested by the constructors |
| 14 | button_cancel.png | 545  547 |  | Line 545 has 151 characters  Line 547 has 153 characters |
| 15 | button_ok.png |  |  | Always respected, even if from line 578 to 582 it is unnecessary |
| 16 | button_ok.png |  |  |  |
| 17 | button_ok.png |  |  |  |
| 18 | button_ok.png |  |  | Always respected, although the comment in line 538 should have been a Javadoc |
| 19 | button_ok.png |  |  | No commented out code |
| 29 | button_ok.png |  |  |  |
| 30 | button_ok.png |  |  |  |
| 31 | button_ok.png |  |  |  |
| 32 | button_ok.png |  |  |  |
| 33 | button_cancel.png | 556  574 | Sniffer mainSniffer = null;  List<Sniffer> sniffersToRemove = new ArrayList<Sniffer>(); | They should be declared at the beginning together with the others |
| 34 | button_ok.png |  |  |  |
| 35 | button_ok.png |  |  |  |
| 37 | button_ok.png |  |  |  |
| 38 | button_ok.png |  |  |  |
| 39 | button_ok.png |  |  |  |
| 40 | button_ok.png |  |  |  |
| 41 |  |  | No output |  |
| 42 |  |  | No output |  |
| 43 |  |  | No output |  |
| 44 | button_ok.png |  |  |  |
| 45 | button_ok.png |  |  |  |
| 46 | button_ok.png |  |  |  |
| 47 | button_ok.png |  |  |  |
| 48 | button_ok.png |  |  |  |
| 49 | button_ok.png |  |  |  |
| 50 | button_ok.png |  | Not present |  |
| 51 | button_cancel.png | 547 | ExtendedDeploymentContext context = new DeploymentContextImpl(null, archive, parameters, habitat.<ServerEnvironment>getService(ServerEnvironment.class)); | Remove the implicit type conversion |
| 52 | button_ok.png |  | Not present |  |
| 53 | button_ok.png |  | Not present |  |
| 54 | button_ok.png |  | Not present |  |
| 55 | button_ok.png |  | Not present |  |
| 56 | button_ok.png |  |  |  |
| 57 | button_ok.png |  | Not present |  |
| 58 | button_ok.png |  | Not present |  |
| 59 | button_ok.png |  | Not present |  |
| 60 | button_ok.png |  | Not present |  |

## getTypeFromModuleType()

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Respected | Lines | Code or annotation | Possible solution |
| 1 | button_ok.png |  |  |  |
| 2 | button_ok.png |  |  |  |
| 3 | button_ok.png |  |  |  |
| 4 |  |  |  |  |
| 5 | button_ok.png |  |  |  |
| 6 |  |  |  |  |
| 7 | button_ok.png |  |  |  |
| 8 | button_ok.png |  |  |  |
| 9 | button_ok.png |  |  |  |
| 10 | button_ok.png |  | Kernighan and Ritchie style |  |
| 11 | button_ok.png |  |  |  |
| 12 | button_ok.png |  | No blank lines or commented out lines |  |
| 13 | button_ok.png |  |  |  |
| 14 | button_ok.png |  |  |  |
| 15 | button_ok.png |  |  |  |
| 16 | button_ok.png |  |  |  |
| 17 | button_cancel.png |  |  | The “else if” statements should be on a new line, aligned with the “if”. Align correctly |
| 18 | button_ok.png |  | No comments |  |
| 19 | button_ok.png |  | No comments |  |
| 29 | button_ok.png |  |  |  |
| 30 | button_ok.png |  |  |  |
| 31 | button_ok.png |  |  |  |
| 32 | button_ok.png |  |  |  |
| 33 | button_ok.png |  |  |  |
| 34 | button_ok.png |  |  |  |
| 35 | button_ok.png |  |  |  |
| 37 | button_ok.png |  | No arrays |  |
| 38 | button_ok.png |  | No arrays |  |
| 39 | button_ok.png |  | No arrays |  |
| 40 | button_ok.png |  |  |  |
| 41 |  |  | No output |  |
| 42 |  |  | No output |  |
| 43 |  |  | No output |  |
| 44 | button_cancel.png |  | The method is made of an “if” statement followed by three “else if” statements | Change the implementation of ModuleType in order to avoid the chained “else if” |
| 45 | button_ok.png |  |  |  |
| 46 | button_ok.png |  |  |  |
| 47 | button_ok.png |  |  |  |
| 48 | button_ok.png |  |  |  |
| 49 | button_ok.png |  |  |  |
| 50 | button_ok.png |  |  |  |
| 51 | button_ok.png |  |  |  |
| 52 | button_ok.png |  | No exceptions |  |
| 53 | button_ok.png |  | No exceptions |  |
| 54 | button_ok.png |  | No switch statements |  |
| 55 | button_ok.png |  | No switch statements |  |
| 56 | button_ok.png |  | No loops |  |
| 57 | button_ok.png |  | No files |  |
| 58 | button_ok.png |  | No files |  |
| 59 | button_ok.png |  | No files |  |
| 60 | button_ok.png |  | No files |  |

## getConfigurationDeploymentDescriptorFiles()

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Respected | Lines | Code | Possible solution |
| 1 | button_ok.png |  |  |  |
| 2 | button_ok.png |  |  |  |
| 3 | button_ok.png |  |  |  |
| 4 |  |  |  |  |
| 5 | button_ok.png |  |  |  |
| 6 |  |  |  |  |
| 7 | button_ok.png |  |  |  |
| 8 | button_ok.png |  |  |  |
| 9 | button_ok.png |  |  |  |
| 10 | button_ok.png |  | Kernighan and Ritchie style |  |
| 11 | button_ok.png |  |  |  |
| 12 | button_ok.png |  |  |  |
| 13 | button_cancel.png | 632 |  | Declaration can be split on multiple lines |
|  | 635 |  | Statement condition can be split on multiple lines |
| 14 | button_cancel.png | 632  633  635  637 | These lines exceed the 120 characters |  |
| 15 | button_ok.png |  |  |  |
| 16 | button_ok.png |  |  |  |
| 17 | button_ok.png |  |  |  |
| 18 | button_ok.png |  |  |  |
| 19 | button_ok.png |  |  |  |
| 29 | button_ok.png |  |  |  |
| 30 | button_ok.png |  |  |  |
| 31 | button_ok.png |  |  |  |
| 32 | button_ok.png |  |  |  |
| 33 | button_ok.png |  |  |  |
| 34 | button_ok.png |  | It respects the order of parameters used for the other methods |  |
| 35 | button_ok.png |  |  |  |
| 37 | button_ok.png |  |  |  |
| 38 | button_ok.png | 635 | No check for the get(0) call, the call can throw an out of bound exception |  |
| 39 | button_ok.png |  |  |  |
| 40 | button_ok.png |  |  |  |
| 41 |  |  | No output |  |
| 42 |  |  | No output |  |
| 43 |  |  | No output |  |
| 44 | button_ok.png |  |  |  |
| 45 | button_ok.png |  |  |  |
| 46 | button_ok.png |  |  |  |
| 47 | button_ok.png |  |  |  |
| 48 | button_ok.png |  |  |  |
| 49 | button_ok.png |  |  |  |
| 50 | button_ok.png |  |  |  |
| 51 | button_ok.png |  |  |  |
| 52 | button_ok.png |  |  |  |
| 53 | button_ok.png |  |  |  |
| 54 | button_ok.png |  |  |  |
| 55 | button_ok.png |  |  |  |
| 56 | button_ok.png |  |  |  |
| 57 | button_ok.png |  |  |  |
| 58 | button_ok.png |  |  |  |
| 59 | button_ok.png |  |  |  |
| 60 | button_ok.png |  |  |  |

## setElementValue()

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Respected | Lines | Code | Possible solution |
| 1 | button_cancel.png | 650-652 659 | Object o, String Value String schema | Rename the variable |
| 2 | button_cancel.png | 650-652 | Object o | Rename the variable |
| 3 | button_ok.png |  |  |  |
| 4 |  |  |  |  |
| 5 | button_ok.png |  |  |  |
| 6 |  |  |  |  |
| 7 | button_ok.png |  |  |  |
| 8 | button_ok.png |  | There have been used always 4 spaces |  |
| 9 | button_cancel.png | 659-665 |  | Remove TABS |
| 10 | button_ok.png |  | Kernighan and Ritchie style |  |
| 11 | button_cancel.png | 667 669 671 |  | Add missing braces |
| 12 | button_ok.png |  |  |  |
| 13 | button_cancel.png | 677 679 | These 2 lines exceeds the 80 characters |  |
| 14 | button_ok.png | 677 679 | These 2 lines not exceeds the 120 characters |  |
| 15 | button_ok.png |  |  |  |
| 16 | button_ok.png |  |  |  |
| 17 | button_cancel.png | 659-665 |  | Align with the previous statement |
| 18 | button_ok.png |  |  |  |
| 19 | button_ok.png |  |  |  |
| 29 | button_ok.png |  |  |  |
| 30 | button_ok.png |  |  |  |
| 31 | button_ok.png |  |  |  |
| 32 | button_ok.png |  |  |  |
| 33 | button_cancel.png | 676 |  | This declaration must be at the beginning of the block |
| 34 | button_ok.png |  |  |  |
| 35 | button_ok.png |  |  |  |
| 37 | button_ok.png |  | No array |  |
| 38 | button_ok.png |  | Usa hasmoreelemnts |  |
| 39 | button_ok.png |  |  |  |
| 40 | button_ok.png |  |  |  |
| 41 |  |  | No output |  |
| 42 |  |  | No output |  |
| 43 |  |  | No output |  |
| 44 | button_cancel.png | 667 669 671 |  | This can be done with one if statement |
| 45 | button_ok.png |  |  |  |
| 46 | button_ok.png |  |  |  |
| 47 | button_ok.png |  |  |  |
| 48 | button_ok.png |  |  |  |
| 49 | button_ok.png |  |  |  |
| 50 | button_ok.png |  |  |  |
| 51 | button_ok.png |  |  |  |
| 52 | button_ok.png |  |  |  |
| 53 | button_ok.png |  |  |  |
| 54 | button_ok.png |  |  |  |
| 55 | button_ok.png |  |  |  |
| 56 | button_ok.png |  |  |  |
| 57 | button_ok.png |  |  |  |
| 58 | button_ok.png |  |  |  |
| 59 | button_ok.png |  |  |  |
| 60 | button_ok.png |  |  |  |

# Other Problems

* As said before, the method “setElementValue” inspected is a dead code, as it is never called in all the project. It may cause a misunderstanding for a developer with the task of improving the code, and it would be necessary at least to underline this issue with a proper Javadoc or commented out lines of code
* About the length of the lines (points 13 and 14 of the checklist), in order to remain in the maximum of 80 lines, it was decided to count only the blank spaces that occur after the first character and before the last one, so the blank spaces used at the beginning of the line to make the indentation are not considered