|  |  |
| --- | --- |
|  | *Code Inspection Report*  *‘Bom Dia Academia’ Software Development Project*  BSc in LEI  Academic Year 2018/2019 - 1º Semester  Software Engineering I  ES1-2018-EIC2-03 ...  78178, Alexandre Mendes, EIC2  77812, João Tiago Aparício da Costa, EIC2  77689, Rostislav Andreev, EIC2  78643, Diogo Delfim Sarmento, EIC2  ISCTE-IUL, Instituto Universitário de Lisboa  1649-026 Lisbon  Portugal  November 2018 |

**Table of Contents**

[Introduction 3](#__RefHeading___Toc525_3986950589)

[Code inspection – GUI 3](#__RefHeading___Toc527_3986950589)

[Code inspection checklist 3](#__RefHeading___Toc529_3986950589)

[Found defects 7](#__RefHeading___Toc531_3986950589)

[Corrective measures 7](#__RefHeading___Toc533_3986950589)

[Conclusions of the inspection process 7](#__RefHeading___Toc535_3986950589)

# [Introduction](#__RefHeading___Toc2530_1503482439)

*This project is a Java based contentment aggregator. The expected outcome is a java app that collects information from facebook, twitter and email.*

# Code inspection – GUI

*The component being analyzed is the graphical interface of the application. The core components of the GUI are contained within the gui package.*

|  |  |
| --- | --- |
| *Meeting date:*  *Meeting duration:*  *Moderator:*  *Producer:*  *Inspector:*  *Recorder:* | *05/11/2017*  *40 minutes*  *Rostislav Andreev Alexandre Mendes João Tiago Aparício Diogo Delfim Sarmento* |
| *Component name (Package/Class/Method):* | *gui/Login, LoginController, MainController, MainWindow, PostBox, TwitterAccountBox.* |
| *Component was compiled:* | *Yes* |
| *Component was executed:* | *Yes* |
| *Component was tested without errors:* | *No* |
| *Testing coverage achieved:* | *60,0% (Instruction Coverage) ;  56,7% (Line Counters);* |

# Code inspection checklist

1 Variable, Attribute, and Constant Declaration Defects (VC)

1) Are descriptive variable and constant names used in accord with naming conventions?

R: Yes

2) Are there variables or attributes with confusingly similar names?

R: No

3) Is every variable and attribute correctly typed?

R: Yes

4) Is every variable and attribute properly initialized?

R: Yes

5) Could any non-local variables be made local?

R: No

6) Are all for-loop control variables declared in the loop header?

R: Yes

7) Are there literal constants that should be named constants?

R: No

8) Are there variables or attributes that should be constants?

R: No

9) Are there attributes that should be local variables?

R: No

10) Do all attributes have appropriate access modifiers (private, protected, public)?

R: Yes

11) Are there static attributes that should be non-static or vice-versa?

R: No

2. Method Definition Defects (FD)

1) Are descriptive method names used in accord with naming conventions?

R: Yes

2) Is every method parameter value checked before being used?

R: No

3) For every method: Does it return the correct value at every method return point?

R: Yes

4) Do all methods have appropriate access modifiers (private, protected, public)?

R: Yes

5) Are there static methods that should be non-static or vice-versa?

R: No

3. Class Definition Defects (CD)

1) Does each class have appropriate constructors and destructors?

R: No

2) Do any subclasses have common members that should be in the superclass?

R: No

3) Can the class inheritance hierarchy be simplified?

R: No

4. Data Reference Defects (DR)

1) For every array reference: Is each subscript value within the defined bounds?

R: Yes

2) For every object or array reference: Is the value certain to be non-null?

R: Yes

5. Computation/Numeric Defects (CN)

1) Are there any computations with mixed data types?

R: No

2) Is overflow or underflow possible during a computation?

R: No

3) For each expressions with more than one operator: Are the assumptions about order of evaluation and precedence correct?

R: Yes

4) Are parentheses used to avoid ambiguity?

R: Yes

6. Comparison/Relational Defects (CR)

1) For every boolean test: Is the correct condition checked?

R: Yes

2) Are the comparison operators correct?

R: Yes

3) Has each boolean expression been simplified by driving negations inward?

R: Doesnt apply

4) Is each boolean expression correct?

R: Yes

5) Are there improper and unnoticed side-effects of a comparison?

R: No

6) Has an "&" inadvertently been interchanged with a "&&" or a "|" for a "||"?

R: No

7. Control Flow Defects (CF)

1) For each loop: Is the best choice of looping constructs used?

R: Yes

2) Will all loops terminate?

R: Yes

3) When there are multiple exits from a loop, is each exit necessary and handled properly?

R: Yes

4) Does each switch statement have a default case?

R: Doesnt apply

5) Are missing switch case break statements correct and marked with a comment?

R: Doesnt apply

6) Do named break statements send control to the right place?

R: Yes

7) Is the nesting of loops and branches too deep, and is it correct?

R: Not deep but correct

8) Can any nested if statements be converted into a switch statement?

R: No

9) Are null bodied control structures correct and marked with braces or comments?

R: Doesnt Apply

10) Are all exceptions handled appropriately?

R: Yes

11) Does every method terminate?

R: Yes

8. Input-Output Defects (IO)

1) Have all files been opened before use?

R: Doesnt Apply

2) Are the attributes of the input object consistent with the use of the file?

R: Doesnt Apply

3) Have all files been closed after use?

R: Doesnt Apply

4) Are there spelling or grammatical errors in any text printed or displayed?

R: Doesnt Apply

5) Are all I/O exceptions handled in a reasonable way?

R: Doesnt Apply

9. Module Interface Defects (MI)

1) Are the number, order, types, and values of parameters in every method call in agreement with the called method's declaration?

R: Yes

2) Do the values in units agree (e.g., inches versus yards)?

R: Doesnt Apply

3) If an object or array is passed, does it get changed, and changed correctly by the called method?

R: Doesnt Apply

10. Comment Defects (CM)

1) Does every method, class, and file have an appropriate header comment?

R: Yes

2) Does every attribute, variable, and constant declaration have a comment?

R: Yes

3) Is the underlying behavior of each method and class expressed in plain language?

R: No

4) Is the header comment for each method and class consistent with the behavior of the method or class?

R: Yes

5) Do the comments and code agree?

R: Yes

6) Do the comments help in understanding the code?

R: Yes

7) Are there enough comments in the code?

R: Yes

8) Are there too many comments in the code?

R: No

11. Layout and Packaging Defects (LP)

1) Is a standard indentation and layout format used consistently?

R: No

2) For each method: Is it no more than about 60 lines long?

R: Yes

3) For each compile module: Is no more than about 600 lines long?

R: Yes

12. Modularity Defects (MO)

1) Is there a low level of coupling between modules (methods and classes)?

R: Doesnt Apply

2) Is there a high level of cohesion within each module (methods or class)?

R: Doesnt Apply

3) Is there repetitive code that could be replaced by a call to a method that provides the behavior of the repetitive code?

R: No

4) Are the Java class libraries used where and when appropriate?

R: Yes

13. Storage Usage Defects (SU)

1) Are arrays large enough?

R: Yes

2) Are object and array references set to null once the object or array is no longer needed?

R: No

14. Performance Defects (PE)

1) Can better data structures or more efficient algorithms be used?

R: Maybe

2) Are logical tests arranged such that the often successful and inexpensive tests precede the more expensive and less frequently successful tests?

R: Yes

3) Can the cost of recomputing a value be reduced by computing it once and storing the results?

R: No

4) Is every result that is computed and stored actually used?

R: Yes

5) Can a computation be moved outside a loop?

R: No

6) Are there tests within a loop that do not need to be done?

R: No

7) Can a short loop be unrolled?

R: No

8) Are there two loops operating on the same data that can be combined into one?

R: No

9) Are frequently used variables declared register?

R: Yes

10) Are short and commonly called methods declared inline?

R: Yes

# Found defects

Identify and describe found defects, opinions and suggestions.

|  |  |  |  |
| --- | --- | --- | --- |
| **Found defect Id** | **Package, Class, Method, Line** | **Defect category** | **Description** |
| 1 | Gui, Login Class, main, 46 | VC | The title should be available in several languages, an as such the information printed in to the APP GUI should be based on variable strings. The primaryStage title should be a variable |

# Corrective measures

*Instead of primaryStage.setTitle("Bom Dia Academia");*

*The input given to this function should be a variable.*

# Code inspection – TWITTER

*The component being analyzed is the classes that gather information from Twitter. The core components of the TWITTER are contained within the DBA.twitter package.*

|  |  |
| --- | --- |
| *Meeting date:*  *Meeting duration:*  *Moderator:*  *Producer:*  *Inspector:*  *Recorder:* | *05/12/2018*  *30 minutes*  *Rostislav Andreev Diogo Delfim Sarmento Alexandre Mendes João Tiago Aparício* |
| *Component name (Package/Class/Method):* | *BDA.twitter/TwitterAuth, BDA.twitter/TwitterConnection* |
| *Component was compiled:* | *Yes* |
| *Component was executed:* | *Yes* |
| *Component was tested without errors:* | *Yes* |
| *Testing coverage achieved:* | *60,0% (Instruction Coverage) ;  56,7% (Line Counters);* |

# Code inspection checklist

1 Variable, Attribute, and Constant Declaration Defects (VC)

1) Are descriptive variable and constant names used in accord with naming conventions?

R: Yes

2) Are there variables or attributes with confusingly similar names?

R: No

3) Is every variable and attribute correctly typed?

R: Yes

4) Is every variable and attribute properly initialized?

R: Yes

5) Could any non-local variables be made local?

R: No

6) Are all for-loop control variables declared in the loop header?

R: Yes

7) Are there literal constants that should be named constants?

R: No

8) Are there variables or attributes that should be constants?

R: No

9) Are there attributes that should be local variables?

R: No

10) Do all attributes have appropriate access modifiers (private, protected, public)?

R: Yes

11) Are there static attributes that should be non-static or vice-versa?

R: No

2. Method Definition Defects (FD)

1) Are descriptive method names used in accord with naming conventions?

R: Yes

2) Is every method parameter value checked before being used?

R: No

3) For every method: Does it return the correct value at every method return point?

R: Yes

4) Do all methods have appropriate access modifiers (private, protected, public)?

R: No

5) Are there static methods that should be non-static or vice-versa?

R: No

3. Class Definition Defects (CD)

1) Does each class have appropriate constructors and destructors?

R: No

2) Do any subclasses have common members that should be in the superclass?

R: No

3) Can the class inheritance hierarchy be simplified?

R: No

4. Data Reference Defects (DR)

1) For every array reference: Is each subscript value within the defined bounds?

R: Yes

2) For every object or array reference: Is the value certain to be non-null?

R: Yes

5. Computation/Numeric Defects (CN)

1) Are there any computations with mixed data types?

R: No

2) Is overflow or underflow possible during a computation?

R: No

3) For each expression with more than one operator: Are the assumptions about order of evaluation and precedence correct?

R: Yes

4) Are parentheses used to avoid ambiguity?

R: Yes

6. Comparison/Relational Defects (CR)

1) For every boolean test: Is the correct condition checked?

R: Yes

2) Are the comparison operators correct?

R: Yes

3) Has each boolean expression been simplified by driving negations inward?

R: Doesn’t apply

4) Is each boolean expression correct?

R: Yes

5) Are there improper and unnoticed side-effects of a comparison?

R: No

6) Has an "&" inadvertently been interchanged with a "&&" or a "|" for a "||"?

R: No

7. Control Flow Defects (CF)

1) For each loop: Is the best choice of looping constructs used?

R: Yes

2) Will all loops terminate?

R: Yes

3) When there are multiple exits from a loop, is each exit necessary and handled properly?

R: Yes

4) Does each switch statement have a default case?

R: Doesn’t apply

5) Are missing switch case break statements correct and marked with a comment?

R: Doesn’t apply

6) Do named break statements send control to the right place?

R: Yes

7) Is the nesting of loops and branches too deep, and is it correct?

R: Not deep but correct

8) Can any nested if statements be converted into a switch statement?

R: No

9) Are null bodied control structures correct and marked with braces or comments?

R: Doesn’t Apply

10) Are all exceptions handled appropriately?

R: Yes

11) Does every method terminate?

R: Yes

8. Input-Output Defects (IO)

1) Have all files been opened before use?

R: Doesn’t Apply

2) Are the attributes of the input object consistent with the use of the file?

R: Doesn’t Apply

3) Have all files been closed after use?

R: Doesn’t Apply

4) Are there spelling or grammatical errors in any text printed or displayed?

R: Doesn’t Apply

5) Are all I/O exceptions handled in a reasonable way?

R: Doesn’t Apply

9. Module Interface Defects (MI)

1) Are the number, order, types, and values of parameters in every method call in agreement with the called method's declaration?

R: Yes

2) Do the values in units agree (e.g., inches versus yards)?

R: Doesn’t Apply

3) If an object or array is passed, does it get changed, and changed correctly by the called method?

R: Doesn’t Apply

10. Comment Defects (CM)

1) Does every method, class, and file have an appropriate header comment?

R: Yes

2) Does every attribute, variable, and constant declaration have a comment?

R: Yes

3) Is the underlying behavior of each method and class expressed in plain language?

R: No

4) Is the header comment for each method and class consistent with the behavior of the method or class?

R: Yes

5) Do the comments and code agree?

R: Yes

6) Do the comments help in understanding the code?

R: Yes

7) Are there enough comments in the code?

R: Yes

8) Are there too many comments in the code?

R: No

11. Layout and Packaging Defects (LP)

1) Is a standard indentation and layout format used consistently?

R: No

2) For each method: Is it no more than about 60 lines long?

R: Yes

3) For each compile module: Is no more than about 600 lines long?

R: Yes

12. Modularity Defects (MO)

1) Is there a low level of coupling between modules (methods and classes)?

R: Doesnt Apply

2) Is there a high level of cohesion within each module (methods or class)?

R: Doesnt Apply

3) Is there repetitive code that could be replaced by a call to a method that provides the behavior of the repetitive code?

R: No

4) Are the Java class libraries used where and when appropriate?

R: Yes

13. Storage Usage Defects (SU)

1) Are arrays large enough?

R: Yes

2) Are object and array references set to null once the object or array is no longer needed?

R: No

14. Performance Defects (PE)

1) Can better data structures or more efficient algorithms be used?

R: Maybe

2) Are logical tests arranged such that the often successful and inexpensive tests precede the more expensive and less frequently successful tests?

R: Yes

3) Can the cost of recomputing a value be reduced by computing it once and storing the results?

R: No

4) Is every result that is computed and stored actually used?

R: Yes

5) Can a computation be moved outside a loop?

R: No

6) Are there tests within a loop that do not need to be done?

R: No

7) Can a short loop be unrolled?

R: No

8) Are there two loops operating on the same data that can be combined into one?

R: No

9) Are frequently used variables declared register?

R: Yes

10) Are short and commonly called methods declared inline?

R: Yes

# Found defects

Identify and describe found defects, opinions and suggestions.

|  |  |  |  |
| --- | --- | --- | --- |
| **Found defect Id** | **Package, Class, Method, Line** | **Defect category** | **Description** |
| 1 | Class TwitterAuth | CD | This Class doesn’t have a constructor, it’s using the default one |
| 2 | Class TwitterAuth | FD | This class has some methods that use the default access modifier, instead of making it public, private or protected |

# Corrective measures

*To correct the defect Id 1:*

*Create the constructor for TwitterAuth, like:*

*public TwitterAuth() {*

*}*

*To correct the defect Id 2:*

*Add modifiers to every function being it public, private or protected, my suggestion for the classes that don’t have one access modifier, is to make them protected.*

# Code inspection – Threads and Facebook

|  |  |
| --- | --- |
| *Meeting date:*  *Meeting duration:*  *Moderator:*  *Producer:*  *Inspector:*  *Recorder:* | *06/12/2018*  *40 minutes*  *João Tiago Aparício Alexandre Mendes, João Tiago Aparício*  *Rostislav Andreev Diogo Delfim Sarmento* |
| *Component name (Package/Class/Method):* | *Information EntryGatherer, SingleBarrier, TaskTable, ThreadPool, Worker, Facebook* |
| *Component was compiled:* | *Yes* |
| *Component was executed:* | *Yes* |
| *Component was tested without errors:* | *Yes* |
| *Testing coverage achieved:* | *60,0% (Instruction Coverage) ;  56,7% (Line Counters);* |

# Code inspection checklist

1 Variable, Attribute, and Constant Declaration Defects (VC)

1) Are descriptive variable and constant names used in accord with naming conventions?

R: Yes

2) Are there variables or attributes with confusingly similar names?

R: No

3) Is every variable and attribute correctly typed?

R: Yes

4) Is every variable and attribute properly initialized?

R: Yes

5) Could any non-local variables be made local?

R: No

6) Are all for-loop control variables declared in the loop header?

R: Yes

7) Are there literal constants that should be named constants?

R: No

8) Are there variables or attributes that should be constants?

R: No

9) Are there attributes that should be local variables?

R: No

10) Do all attributes have appropriate access modifiers (private, protected, public)?

R: Yes

11) Are there static attributes that should be non-static or vice-versa?

R: No

2. Method Definition Defects (FD)

1) Are descriptive method names used in accord with naming conventions?

R: Yes

2) Is every method parameter value checked before being used?

R: No

3) For every method: Does it return the correct value at every method return point?

R: Yes

4) Do all methods have appropriate access modifiers (private, protected, public)?

R: No

5) Are there static methods that should be non-static or vice-versa?

R: No

3. Class Definition Defects (CD)

1) Does each class have appropriate constructors and destructors?

R: No

2) Do any subclasses have common members that should be in the superclass?

R: No

3) Can the class inheritance hierarchy be simplified?

R: No

4. Data Reference Defects (DR)

1) For every array reference: Is each subscript value within the defined bounds?

R: Yes

2) For every object or array reference: Is the value certain to be non-null?

R: Yes

5. Computation/Numeric Defects (CN)

1) Are there any computations with mixed data types?

R: No

2) Is overflow or underflow possible during a computation?

R: No

3) For each expression with more than one operator: Are the assumptions about order of evaluation and precedence correct?

R: Yes

4) Are parentheses used to avoid ambiguity?

R: Yes

6. Comparison/Relational Defects (CR)

1) For every boolean test: Is the correct condition checked?

R: Yes

2) Are the comparison operators correct?

R: Yes

3) Has each boolean expression been simplified by driving negations inward?

R: Doesn’t apply

4) Is each boolean expression correct?

R: Yes

5) Are there improper and unnoticed side-effects of a comparison?

R: No

6) Has an "&" inadvertently been interchanged with a "&&" or a "|" for a "||"?

R: No

7. Control Flow Defects (CF)

1) For each loop: Is the best choice of looping constructs used?

R: Yes

2) Will all loops terminate?

R: Yes

3) When there are multiple exits from a loop, is each exit necessary and handled properly?

R: Yes

4) Does each switch statement have a default case?

R: Doesn’t apply

5) Are missing switch case break statements correct and marked with a comment?

R: Doesn’t apply

6) Do named break statements send control to the right place?

R: Yes

7) Is the nesting of loops and branches too deep, and is it correct?

R: Not deep but correct

8) Can any nested if statements be converted into a switch statement?

R: No

9) Are null bodied control structures correct and marked with braces or comments?

R: Doesn’t Apply

10) Are all exceptions handled appropriately?

R: Yes

11) Does every method terminate?

R: Yes

8. Input-Output Defects (IO)

1) Have all files been opened before use?

R: Doesn’t Apply

2) Are the attributes of the input object consistent with the use of the file?

R: Doesn’t Apply

3) Have all files been closed after use?

R: Doesn’t Apply

4) Are there spelling or grammatical errors in any text printed or displayed?

R: Doesn’t Apply

5) Are all I/O exceptions handled in a reasonable way?

R: Doesn’t Apply

9. Module Interface Defects (MI)

1) Are the number, order, types, and values of parameters in every method call in agreement with the called method's declaration?

R: Yes

2) Do the values in units agree (e.g., inches versus yards)?

R: Doesn’t Apply

3) If an object or array is passed, does it get changed, and changed correctly by the called method?

R: Doesn’t Apply

10. Comment Defects (CM)

1) Does every method, class, and file have an appropriate header comment?

R: Yes

2) Does every attribute, variable, and constant declaration have a comment?

R: Yes

3) Is the underlying behavior of each method and class expressed in plain language?

R: No

4) Is the header comment for each method and class consistent with the behavior of the method or class?

R: Yes

5) Do the comments and code agree?

R: Yes

6) Do the comments help in understanding the code?

R: Yes

7) Are there enough comments in the code?

R: Yes

8) Are there too many comments in the code?

R: No

11. Layout and Packaging Defects (LP)

1) Is a standard indentation and layout format used consistently?

R: No

2) For each method: Is it no more than about 60 lines long?

R: Yes

3) For each compile module: Is no more than about 600 lines long?

R: Yes

12. Modularity Defects (MO)

1) Is there a low level of coupling between modules (methods and classes)?

R: Doesnt Apply

2) Is there a high level of cohesion within each module (methods or class)?

R: Doesnt Apply

3) Is there repetitive code that could be replaced by a call to a method that provides the behavior of the repetitive code?

R: No

4) Are the Java class libraries used where and when appropriate?

R: Yes

13. Storage Usage Defects (SU)

1) Are arrays large enough?

R: Yes

2) Are object and array references set to null once the object or array is no longer needed?

R: No

14. Performance Defects (PE)

1) Can better data structures or more efficient algorithms be used?

R: Maybe

2) Are logical tests arranged such that the often successful and inexpensive tests precede the more expensive and less frequently successful tests?

R: Yes

3) Can the cost of recomputing a value be reduced by computing it once and storing the results?

R: No

4) Is every result that is computed and stored actually used?

R: Yes

5) Can a computation be moved outside a loop?

R: No

6) Are there tests within a loop that do not need to be done?

R: No

7) Can a short loop be unrolled?

R: No

8) Are there two loops operating on the same data that can be combined into one?

R: No

9) Are frequently used variables declared register?

R: Yes

10) Are short and commonly called methods declared inline?

R: Yes

# Found defects

No defects were found

# Code inspection –Email, Files and Other

*The components being analyzed are the email ,files and other packages.*

|  |  |
| --- | --- |
| *Meeting date:*  *Meeting duration:*  *Moderator:*  *Producer:*  *Inspector:*  *Recorder:* | *05/12/2018*  *30 minutes*  *Rostislav Andreev Alexandre Mendes Diogo Delfim Sarmento João Tiago Aparício* |
| *Component name (Package/Class/Method):* | *BDA.email/EmailConnection ,SimpleMailAuthenticatior*  *BDA.files/ReadandWriteFile ,ReadandWriteXMLFile*  *BDA.other/ControlCenter,Filter,OtherStaticFunctions,Service,*  *XMLUserConfigurations* |
| *Component was compiled:* | *Yes* |
| *Component was executed:* | *Yes* |
| *Component was tested without errors:* | *No* |
| *Testing coverage achieved:* | *70,0% (Instruction Coverage) ;  67,3% (Line Counters);* |

# Code inspection checklist

1 Variable, Attribute, and Constant Declaration Defects (VC)

1) Are descriptive variable and constant names used in accord with naming conventions?

R: No

2) Are there variables or attributes with confusingly similar names?

R: No

3) Is every variable and attribute correctly typed?

R: Yes

4) Is every variable and attribute properly initialized?

R: Yes

5) Could any non-local variables be made local?

R: No

6) Are all for-loop control variables declared in the loop header?

R: Yes

7) Are there literal constants that should be named constants?

R: No

8) Are there variables or attributes that should be constants?

R: No

9) Are there attributes that should be local variables?

R: No

10) Do all attributes have appropriate access modifiers (private, protected, public)?

R: Yes

11) Are there static attributes that should be non-static or vice-versa?

R: No

2. Method Definition Defects (FD)

1) Are descriptive method names used in accord with naming conventions?

R: Yes

2) Is every method parameter value checked before being used?

R: No

3) For every method: Does it return the correct value at every method return point?

R: Yes

4) Do all methods have appropriate access modifiers (private, protected, public)?

R: Yes

5) Are there static methods that should be non-static or vice-versa?

R: No

3. Class Definition Defects (CD)

1) Does each class have appropriate constructors and destructors?

R: No

2) Do any subclasses have common members that should be in the superclass?

R: No

3) Can the class inheritance hierarchy be simplified?

R: No

4. Data Reference Defects (DR)

1) For every array reference: Is each subscript value within the defined bounds?

R: Yes

2) For every object or array reference: Is the value certain to be non-null?

R: Yes

5. Computation/Numeric Defects (CN)

1) Are there any computations with mixed data types?

R: No

2) Is overflow or underflow possible during a computation?

R: No

3) For each expressions with more than one operator: Are the assumptions about order of evaluation and precedence correct?

R: Yes

4) Are parentheses used to avoid ambiguity?

R: Yes

6. Comparison/Relational Defects (CR)

1) For every boolean test: Is the correct condition checked?

R: Yes

2) Are the comparison operators correct?

R: Yes

3) Has each boolean expression been simplified by driving negations inward?

R: Doesnt apply

4) Is each boolean expression correct?

R: Yes

5) Are there improper and unnoticed side-effects of a comparison?

R: No

6) Has an "&" inadvertently been interchanged with a "&&" or a "|" for a "||"?

R: No

7. Control Flow Defects (CF)

1) For each loop: Is the best choice of looping constructs used?

R: Yes

2) Will all loops terminate?

R: Yes

3) When there are multiple exits from a loop, is each exit necessary and handled properly?

R: Yes

4) Does each switch statement have a default case?

R: Yes

5) Are missing switch case break statements correct and marked with a comment?

R: Yes

6) Do named break statements send control to the right place?

R: Yes

7) Is the nesting of loops and branches too deep, and is it correct?

R: Not deep but correct

8) Can any nested if statements be converted into a switch statement?

R: No

9) Are null bodied control structures correct and marked with braces or comments?

R: Doesnt Apply

10) Are all exceptions handled appropriately?

R: No

11) Does every method terminate?

R: Yes

8. Input-Output Defects (IO)

1) Have all files been opened before use?

R: Yes

2) Are the attributes of the input object consistent with the use of the file?

R: Yes

3) Have all files been closed after use?

R: Yes

4) Are there spelling or grammatical errors in any text printed or displayed?

R: No

5) Are all I/O exceptions handled in a reasonable way?

R: No

9. Module Interface Defects (MI)

1) Are the number, order, types, and values of parameters in every method call in agreement with the called method's declaration?

R: Yes

2) Do the values in units agree (e.g., inches versus yards)?

R: Doesnt Apply

3) If an object or array is passed, does it get changed, and changed correctly by the called method?

R: It Doesnt Get Changed

10. Comment Defects (CM)

1) Does every method, class, and file have an appropriate header comment?

R: Yes

2) Does every attribute, variable, and constant declaration have a comment?

R: Yes

3) Is the underlying behavior of each method and class expressed in plain language?

R: No

4) Is the header comment for each method and class consistent with the behavior of the method or class?

R: Yes

5) Do the comments and code agree?

R: Yes

6) Do the comments help in understanding the code?

R: Yes

7) Are there enough comments in the code?

R: Yes

8) Are there too many comments in the code?

R: No

11. Layout and Packaging Defects (LP)

1) Is a standard indentation and layout format used consistently?

R: Yes

2) For each method: Is it no more than about 60 lines long?

R: No

3) For each compile module: Is no more than about 600 lines long?

R: Yes

12. Modularity Defects (MO)

1) Is there a low level of coupling between modules (methods and classes)?

R: Doesnt Apply

2) Is there a high level of cohesion within each module (methods or class)?

R: Doesnt Apply

3) Is there repetitive code that could be replaced by a call to a method that provides the behavior of the repetitive code?

R: No

4) Are the Java class libraries used where and when appropriate?

R: Yes

13. Storage Usage Defects (SU)

1) Are arrays large enough?

R: Yes

2) Are object and array references set to null once the object or array is no longer needed?

R: No

14. Performance Defects (PE)

1) Can better data structures or more efficient algorithms be used?

R: Maybe

2) Are logical tests arranged such that the often successful and inexpensive tests precede the more expensive and less frequently successful tests?

R: Yes

3) Can the cost of recomputing a value be reduced by computing it once and storing the results?

R: No

4) Is every result that is computed and stored actually used?

R: Yes

5) Can a computation be moved outside a loop?

R: No

6) Are there tests within a loop that do not need to be done?

R: No

7) Can a short loop be unrolled?

R: No

8) Are there two loops operating on the same data that can be combined into one?

R: No

9) Are frequently used variables declared register?

R: Yes

10) Are short and commonly called methods declared inline?

R: Yes

# Found defects

Identify and describe found defects, opinions and suggestions.

|  |  |  |  |
| --- | --- | --- | --- |
| **Found defect Id** | **Package, Class, Method, Line** | **Defect category** | **Description** |
| 1 | Class BDA.email.EmailConnection, | CF | This Class doesn’t have proper handling to the exceptions, relying on printStackTrace. |
| 2 | Class BDA.other.ControlCenter | CF | This Class doesn’t have proper handling to the exceptions, relying on printStackTrace. |

# Corrective measures

*To correct the defects there should be different handling to the exceptions that does not reveal the user the way the software works, this can be done by avoiding printStackTrace as a way to handle the exceptions.*

# Conclusions of the inspection process

*The changes neededQuality assessment of the component inspected for the purpose of integration/delivery the software (does it need no changes, minor/major changes/corrections, build from scratch).*