|  |  |
| --- | --- |
|  | *Code Inspection Report*  *‘Bom Dia Academia’ Software Development Project*  BSc/MSc in [LEI | LIGE | METI]  Academic Year 2018/2019 - 1º Semester  Software Engineering I  78179, Daniel Freitas  79001, Hugo Cruz  78009, Luis Fernandes  77577, Miguel Figueiredo  ISCTE-IUL, Instituto Universitário de Lisboa  1649-026 Lisbon  Portugal  November 2018 |

**Table of Contents**

[Introduction 3](#_Toc498465002)

[Code inspection – Twitter](#_Toc498465003) 4

[Code inspection checklist](#_Toc498465004) 4

[Found defects](#_Toc498465005) 7

[Corrective measures](#_Toc498465006) 7

[Code inspection – Gmail](#_Toc498465003) 7

[Code inspection checklist](#_Toc498465004) 8

[Code inspection – Facebook](#_Toc498465003) 11

[Found defects](#_Toc498465005) 11

[Corrective measures](#_Toc498465006) 12

[Code inspection – GUI](#_Toc498465003) 12

[Found defects](#_Toc498465005) 12

[Corrective measures](#_Toc498465006) 12

[Conclusions of the inspection process](#_Toc498465007) 12

# [Introduction](#__RefHeading___Toc2530_1503482439)

*The Software produced consists of an app for the integration of academic information  
with origin in several systems (Gmail, Twitter, Facebook)*

# Code inspection – Twitter\_Class.java

*Twitter component of the app*

|  |  |
| --- | --- |
| *Meeting date:*  *Meeting duration:*  *Moderator:*  *Producer:*  *Inspector:*  *Recorder:* | *06/12/2017*  *50 minutes*  *Hugo Cruz*  *Hugo Cruz*  *Luís Fernandes*  *Luís Fernandes* |
| *Component name (Package/Class/Method):* | *Twitter\_Class.java* |
| *Component was compiled:* | *yes* |
| *Component was executed:* | *yes* |
| *Component was tested without errors:* | yes |
| *Testing coverage achieved:* | *90,1 %* |

# Code inspection checklist

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

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

R:.Yes

**Are there variables or attributes with confusingly similar names?**

R:.No

**Is every variable and attribute correctly typed?**

R:.Yes

**Is every variable and attribute properly initialized?**

R:.Yes

**Could any non-local variables be made local?**

R:.No

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

R:.Yes

**Are there literal constants that should be named constants?**

R:.No

**Are there variables or attributes that should be constants?**

R:.No

**Are there attributes that should be local variables?**

R:.No

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

R:.Yes

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

R:.No

**2. Method Definition Defects (****FD)**

**Are descriptive method names used in accord with naming conventions?**

R:.Yes

**Is every method parameter value checked before being used?**

R:.No

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

R:.Yes

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

R:.Yes

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

R:.No

**3. Class Definition Defects (CD)**

**Does each class have appropriate constructors and destructors?**

R:.Yes for constructors and No for destructors

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

R:.No

**Can the class inheritance hierarchy be simplified?**

R:.No

**4. Data Reference Defects (DR)**

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

R:.Yes

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

R:.Yes

**5. Computation/Numeric Defects (CN)**

**Are there any computations with mixed data types?**

R:.No

**Is overflow or underflow possible during a computation?**

R:.No

**For each expressions with more than one operator: Are the assumptions about order of**

**evaluation and precedence correct?**

R:.Yes

**Are parentheses used to avoid ambiguity?**

R:.Yes

**6. Comparison/Relational Defects (CR)**

**For every boolean test: Is the correct condition checked?**

R:.Yes

**Are the comparison operators correct?**

R:.Yes

**Has each boolean expression been simplified by driving negations inward?**

R:.No

**Is each boolean expression correct?**

R:.Yes

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

R:.No

**Has an "&" inadvertently been interchanged with a "&&" or a "|" for a "||"**

R:.No

**7. Control Flow Defects (CF)**

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

R:.Yes

**Will all loops terminate?**

R:.Yes

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

R:.Yes

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

R:.No

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

R:.No

**Are all exceptions handled appropriately?**

R:.Yes

**Does every method terminate?**

R:.Yes

**9. Module Interface Defects (MI)**

**Are the number, order, types, and values of parameters in every method call in agreement**

**with the called method's declaration?**

R:.Yes

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

R:.Yes

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

R:.Yes

**10. Comment Defects (CM)**

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

R:.Yes

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

R:.Yes

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

R:.Yes

**Is the header comment for each method and class consistent with the behavior of the method**

**or class?**

R:.Yes

**Do the comments and code agree?**

R:.Yes

**Do the comments help in understanding the code?**

R:.Yes

**Are there enough comments in the code?**

R:.Yes

**Are there too many comments in the code?**

R:.No

**11. Layout and Packaging Defects (LP)**

**Is a standard indentation and layout format used consistently?**

R:.Yes

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

R:.Yes

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

R:.Yes

**12. Modularity Defects (MO)**

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

R:.Yes

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

R:.No

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

R:.No

**Are the Java class libraries used where and when appropriate?**

R:.Yes

**14. Performance Defects (PE)**

**Can better data structures or more efficient algorithms be used?**

R:.No

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

R:.No

**Is every result that is computed and stored actually used?**

R:.Yes

**Can a computation be moved outside a loop?**

R:.No

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

R:.Yes

**Are frequently used variables declared register?**

R:.No

**Are short and commonly called methods declared inline?**

R:.No

# Found defects

Identify and describe found defects, opinions and suggestions.

|  |  |  |  |
| --- | --- | --- | --- |
| **Found defect Id** | **Package, Class, Method, Line** | **Defect category** | **Description** |
| 1 | *Twitter\_Class.java*  *line 40,57,84,87,90,98* | FD | This class has methods with public access |

# Corrective measures

*Found defect Id 1, it will be corrected by adjusting the methods accessibility*

# Code inspection – Gmail.package

*Gmail component of the app*

|  |  |
| --- | --- |
| *Meeting date:*  *Meeting duration:*  *Moderator:*  *Producer:*  *Inspector:*  *Recorder:* | *07/12/2018*  *35 minutes*  *Luís Fernandes*  *Luis Fernandes*  *Miguel Figueiredo*  *Miguel Figueiredo* |
| *Component name (Package/Class/Method):* | *Gmail.package* |
| *Component was compiled:* | *Yes* |
| *Component was executed:* | *Yes* |
| *Component was tested without errors:* | *Yes* |
| *Testing coverage achieved:* | *75%* |

# Code inspection checklist

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

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

R:.Yes

**Are there variables or attributes with confusingly similar names?**

R:.No

**Is every variable and attribute correctly typed?**

R:.Yes

**Is every variable and attribute properly initialized?**

R:.Yes

**Could any non-local variables be made local?**

R:.Yes

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

R:.Yes

**Are there literal constants that should be named constants?**

R:.No

**Are there variables or attributes that should be constants?**

R:.No

**Are there attributes that should be local variables?**

R:.No

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

R:.Yes

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

R:.No

**2. Method Definition Defects (FD)**

**Are descriptive method names used in accord with naming conventions?**

R:.Yes

**Is every method parameter value checked before being used?**

R:.Yes

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

R:.Yes

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

R:.Yes

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

R:.No

**3. Class Definition Defects (CD)**

**Does each class have appropriate constructors and destructors?**

R:.Mail sender has a constructor but ReceiveEmailsUsing IMAP no has constructor

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

R:.Don’t have any subclasses

**Can the class inheritance hierarchy be simplified?**

R:.It doesn’t have subclasses so the inheritance not exist an couldn’t be simplified

**4. Data Reference Defects (DR)**

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

R:.Don’t have array reference.

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

R:.The array doesn’t exist so it value certain can’t be defined.

**5. Computation/Numeric Defects (CN)**

**Are there any computations with mixed data types?**

R:.No

**For each expressions with more than one operator: Are the assumptions about order of**

**evaluation and precedence correct?**

R:.Yes

**Are parentheses used to avoid ambiguity?**

R:.Yes

**6. Comparison/Relational Defects (CR)**

**For every boolean test: Is the correct condition checked?**

R:.Yes

**Are the comparison operators correct?**

R:.Yes

**Has each boolean expression been simplified by driving negations inward?**

R:.Yes

**Is each boolean expression correct?**

R:.Yes

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

R:.No

**Has an "&" inadvertently been interchanged with a "&&" or a "|" for a "||"**

R:.No

**7. Control Flow Defects (CF)**

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

R:.Yes

**Will all loops terminate?**

R:.Yes

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

R:.Yes

**Does each switch statement have a default case?**

R:.Doesn’t have switch statement

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

R:.No

**Do named break statements send control to the right place?**

R:.Yes

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

R:.Yes

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

R:.No

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

R:.Yes

**Are all exceptions handled appropriately?**

R:.Yes

**Does every method terminate?**

R:.Yes

**8. Input-Output Defects (IO)**

**Have all files been opened before use?**

R:.Yes

**Are the attributes of the input object consistent with the use of the file?**

R:.Yes

**Have all files been closed after use?**

R:.Yes

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

R:.No

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

R:.Yes

**9. Module Interface Defects (MI)**

**Are the number, order, types, and values of parameters in every method call in agreement**

**with the called method's declaration?**

R:.Yes

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

R:.Yes

**If an object or array is passed, does it get changed, and changed correctly by the called**

**method?**

R:.It changed correctly according to the purpose of that method

**10. Comment Defects (CM)**

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

R:.Yes

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

R:.Yes

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

R:.Yes

**Is the header comment for each method and class consistent with the behavior of the method**

**or class?**

R:.Yes

**Do the comments and code agree?**

R:.Yes

**Do the comments help in understanding the code?**

R:.Yes

**Are there enough comments in the code?**

R:.Yes

**Are there too many comments in the code?**

R:.No

**11. Layout and Packaging Defects (LP)**

**Is a standard indentation and layout format used consistently?**

R:.Yes

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

R:.One method has more than 60 lines long

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

R:.Yes

**12. Modularity Defects (MO)**

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

R:.Yes

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

R:.Yes

**Is there repetitive code that could be replaced by a call to a method that provides the behavior**

**of the repetitive code?**

R:.Maybe

**Are the Java class libraries used where and when appropriate?**

R:.Yes

**13. Storage Usage Defects (SU)**

**Are arrays large enough?**

R:.Yes

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

R:.Yes

**14. Performance Defects (PE)**

**Can better data structures or more efficient algorithms be used?**

R:.No

**Are logical tests arranged such that the often successful and inexpensive tests precede the**

**more expensive and less frequently successful tests?**

R:.Yes

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

R:.No

**Is every result that is computed and stored actually used?**

R:.Yes

**Can a computation be moved outside a loop?**

R:.Yes

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

R:.No

**Can a short loop be unrolled?**

R:.No

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

R:.Exists two loops in some methods but they can’t be combined in one

**Are frequently used variables declared register?**

R:.Yes

**Are short and commonly called methods declared inline?**

R:.Yes

# Code inspection – Facebook

*Facebook component of the app*

|  |  |
| --- | --- |
| *Meeting date:*  *Meeting duration:*  *Moderator:*  *Producer:*  *Inspector:*  *Recorder:* | *07/12/2017*  *25 minutes*  *Miguel Figueiredo*  *Miguel Figueiredo*  *Hugo Cruz*  *Hugo Cruz* |
| *Component name (Package/Class/Method):* | *Facebook* |
| *Component was compiled:* | *Yes* |
| *Component was executed:* | *Yes* |
| *Component was tested without errors:* | *Yes* |
| *Testing coverage achieved:* | *85%* |

# Found defects

Identify and describe found defects, opinions and suggestions.

|  |  |  |  |
| --- | --- | --- | --- |
| **Found defect Id** | **Package, Class, Method, Line** | **Defect category** | **Description** |
| 1 | Facebook | 13 | Array not set to null |

# Corrective measures

*Found defect Id, how/when/who will correct the identified defect.*

*Only one defect found due to java not needing to set to null an array*

# Code inspection – GUI

*Graphic User Interface for the app*

|  |  |
| --- | --- |
| *Meeting date:*  *Meeting duration:*  *Moderator:*  *Producer:*  *Inspector:*  *Recorder:* | *7/12/2018*  *40 minutes*  *Daniel Freitas*  *Daniel Freitas*  *Daniel Freitas*  *Hugo Cruz* |
| *Component name (Package/Class/Method):* | *GUI* |
| *Component was compiled:* | *Yes* |
| *Component was executed:* | *Yes* |
| *Component was tested without errors:* | *Yes* |
| *Testing coverage achieved:* | *82%* |

# Found defects

*Identify and describe found defects, opinions and suggestions.*

|  |  |  |  |
| --- | --- | --- | --- |
| **Found defect Id** | **Package, Class, Method, Line** | **Defect category** | **Description** |
| 1 | gui | 2 | Every method is public |
| 2 | Gui,BDAGui | 11 | There are some methods with more than 60 lines |
| 3 | gui | 13 | An array is never references set to null once the array is no longer needed because java takes care of it once we close the program |

# Corrective measures

*Found defect Id, how/when/who will correct the identified defect.*

*3 defects were identified but only one can be corrected. Defect number 1 can be corrected by identifying what methods are needed as public, which means that they are used outside of this package, which are needed to become protected, which means that they are used inside the package, and the ones that need to be corrected into private.*

# Conclusions of the inspection process

*Quality 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).*