Code Inspection Report

‘Bom Dia Academia’ Software Development Project

BSc/MSc in [LEI | LIGE | METI]
Academic Year 2018/2019 - 1º Semester
Software Engineering I

Group 29
78002, José Raimundo, IC1
78302, Mariana Teixeira, IC1
77978, Pedro Rocha, IC1

ISCTE-IUL, Instituto Universitário de Lisboa
1649-026 Lisbon
Portugal

November 2018

Table of Contents

Introduction

Desenvolvimento de software para integração de informação académica com origem em vários sistemas.

Code inspection – ES1-2018-IC1-29

Meeting date: *Meeting duration:* *Moderator:* *Producer:* *Inspector:* *Recorder:*	*07/12/2017* *45 minutes* *José Raimundo* *Pedro Rocha*

Component name (Package/Class/Method):	*bda/GUI;*
	bda/DateComparator;
	ContentHandlers/FetchEmails

Component was compiled:	*Sim*
Component was executed:	*Sim*
Component was tested without errors:	*Sim*

Testing coverage achieved:	*GUI(60,5%);*
	DateComparator(2,1%)
	FetchEmails(80,7%);

Code inspection checklist

The checklist for Java code inspection used in this project is available at http://www.cs.toronto.edu/~sme/CSC444F/handouts/java_checklist.pdf and in blackboard ES1 page.

Found defects

Found defect Id	**Package, Class, Method, Line**	**Defect category**	**Description**
1	bda/DateComparator/moreThanDay/85	1	Nome da função escrito incorretamente
2	bda/GUI/getLog()/289	7	Método com múltiplos "if's"
3	bda/GUI/getFilteredContent()/201	11	Método com mais de 60 linhas
4	ContentHandlers/FetchEmails/3	12	Imports não necessários

Corrective measures

Defeito 1, corrigido por Pedro Rocha, correção de "morrThanDay" para "moreThanDay".
Defeito 2, corrigido por Pedro Rocha, através da implementação de um switch.
Defeito 3, não corrigido, uma vez que o código já se encontra demasiado desenvolvido e implicaria demasiadas alterações no projeto.
Defeito 4, corrigido por Pedro Rocha, remoção de "imports" não utilizadas na Classe.

Conclusions of the inspection process

Código simples e compreensível, apenas com um método extenso e imports não necessários. Foram efetuadas poucas alterações, sendo uma delas a correção do nome de um método.

Java Inspection Checklist

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

- ✓ Are there variables or attributes with confusingly similar names?
- ❑ Is every variable and attribute correctly typed?
- ✓ Are there literal constants that should be named constants?

2. Method Definition Defects (FD)

- ✓ Are descriptive method names used in accord with naming conventions?
- ✓ For every method: Does it return the correct value at every method return point?
- ✓ Do all methods have appropriate access modifiers (private, protected, public)?

3. Class Definition Defects (CD)

- ✓ Does each class have appropriate constructors and destructors?

4. Data Reference Defects (DR)

- ✓ For every array reference: Is each subscript value within the defined bounds?
- ✓ For every object or array reference: Is the value certain to be non-null?

5. Computation/Numeric Defects (CN)

- ✓ Are there any computations with mixed data types?

6. Comparison/Relational Defects (CR)

- ✓ For every boolean test: Is the correct condition checked?
- ✓ Are the comparison operators correct?
- ✓ Has each boolean expression been simplified by driving negations inward?

7. Control Flow Defects (CF)

- ✓ Will all loops terminate?
- ✓ When there are multiple exits from a loop, is each exit necessary and handled properly?
- ❑ Can any nested if statements be converted into a switch statement?
- ✓ Does every method terminate?

8. Input-Output Defects (IO)

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

9. Module Interface Defects (MI)

- ✓ Do the values in units agree (e.g., inches versus yards)?
- ✓ If an object or array is passed, does it get changed, and changed correctly by the called method?

10. Comment Defects (CM)

- ✓ Does every method, class, and file have an appropriate header comment?
- ✓ Does every attribute, variable, and constant declaration have a comment?
- ✓ Do the comments and code agree?
- ✓ Do the comments help in understanding the code?

11. Layout and Packaging Defects (LP)

- ❑ For each method: Is it no more than about 60 lines long?
- ✓ For each compile module: Is no more than about 600 lines long?

12. Modularity Defects (MO)

- ✓ Is there repetitive code that could be replaced by a call to a method that provides the behavior of the repetitive code?
- ❑ Are the Java class libraries used where and when appropriate?

13. Storage Usage Defects (SU)

- ✓ Are arrays large enough?
- ✓ Are object and array references set to null once the object or array is no longer needed?

14. Performance Defects (PE)

- ✓ Can better data structures or more efficient algorithms be used?
- ✓ Are frequently used variables declared register?

Code inspection – ES1-2018-IC1-29

Meeting date: *Meeting duration:* *Moderator:* *Producer:* *Inspector:* *Recorder:*	*07/12/2017* *30 minutes* *Mariana Teixeira* *José Raimundo*

Component name (Package/Class/Method):	*bda/BDATableModel;*
	bda/ContentGUI;
	ContentHandlers/FetchPosts;
	ContentHandlers/FetchTweets;

Component was compiled:	*Sim*
Component was executed:	*Sim*
Component was tested without errors:	*Sim*

Testing coverage achieved:	*BDATableModel(41%);*
	ContentGUI(6,4%);
	FetchPosts(66,9%);
	FetchTweets(81,9%)

Found defects

Found defect Id	Package, Class, Method, Line	Defect category	Description
1	Bda/ContentGui/72;294	11	Construtor e método com mais de 60 linhas
2	ContentHandlers/FetchPosts/3	12	Imports não necessários
3	ContentHandlers/FetchTweets/3	12	Imports não necessários

Corrective measures

Defeito 1 não corrigido, uma vez que o código já se encontra demasiado desenvolvido e implicaria demasiadas alterações no projeto.
Defeitos 2 e 3, corrigidos por José Raimundo, remoção de “imports” não utilizadas na Classe.

Conclusions of the inspection process

Neste processo de inspeção não foi necessário efetuar qualquer alteração a não ser apenas, retirar imports que não possuíam utilidade. O código inspecionado era sucinto e de qualidade, existindo apenas um método que ultrapassa as 60 linhas.

Java Inspection Checklist

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

- ✓ Are there variables or attributes with confusingly similar names?
- ✓ Is every variable and attribute correctly typed?
- ✓ Are there literal constants that should be named constants?

2. Method Definition Defects (FD)

- ✓ Are descriptive method names used in accord with naming conventions?
- ✓ For every method: Does it return the correct value at every method return point?
- ✓ Do all methods have appropriate access modifiers (private, protected, public)?

3. Class Definition Defects (CD)

- ✓ Does each class have appropriate constructors and destructors?

4. Data Reference Defects (DR)

- ✓ For every array reference: Is each subscript value within the defined bounds?
- ✓ For every object or array reference: Is the value certain to be non-null?

5. Computation/Numeric Defects (CN)

- ✓ Are there any computations with mixed data types?

6. Comparison/Relational Defects (CR)

- ✓ For every boolean test: Is the correct condition checked?
- ✓ Are the comparison operators correct?
- ✓ Has each boolean expression been simplified by driving negations inward?

7. Control Flow Defects (CF)

- ✓ Will all loops terminate?
- ✓ When there are multiple exits from a loop, is each exit necessary and handled properly?
- ✓ Can any nested if statements be converted into a switch statement?
- ✓ Does every method terminate?

8. Input-Output Defects (IO)

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

9. Module Interface Defects (MI)

- ✓ Do the values in units agree (e.g., inches versus yards)?
- ✓ If an object or array is passed, does it get changed, and changed correctly by the called method?

10. Comment Defects (CM)

- ✓ Does every method, class, and file have an appropriate header comment?
- ✓ Does every attribute, variable, and constant declaration have a comment?
- ✓ Do the comments and code agree?

- ✓ Do the comments help in understanding the code?

11. Layout and Packaging Defects (LP)

- ❑ For each method: Is it no more than about 60 lines long?
- ✓ For each compile module: Is no more than about 600 lines long?

12. Modularity Defects (MO)

- ✓ Is there repetitive code that could be replaced by a call to a method that provides the behavior of the repetitive code?
- ❑ Are the Java class libraries used where and when appropriate?

13. Storage Usage Defects (SU)

- ✓ Are arrays large enough?
- ✓ Are object and array references set to null once the object or array is no longer needed?

14. Performance Defects (PE)

- ✓ Can better data structures or more efficient algorithms be used?
- ✓ Are frequently used variables declared register?

Code inspection – ES1-2018-IC1-29

Meeting date: *Meeting duration:* *Moderator:* *Producer:* *Inspector:* *Recorder:*	*07/12/2017* *30 minutes* *Pedro Rocha* *Mariana Teixeira*

Component name (Package/Class/Method):	*bda/Login;*
	bda/BDAButton;
	bda/Content;

Component was compiled:	*Sim*
Component was executed:	*Sim*
Component was tested without errors:	*Sim*

Testing coverage achieved:	*Login(24%);*
	BDAButton(34,4%);
	Content(27,7%);

Found defects

Found defect Id	**Package, Class, Method, Line**	**Defect category**	**Description**
1	bda/Content/26;27;28	2	Atributos a “public”
2	bda/Login/login()/205	7	Método com múltiplos “if’s”
3	bda/Login/login()/201	11	Método com mais de 60 linhas
4	bda/BDAButton/addLogOperations()	11	Método com mais de 60 linhas

Corrective measures

Defeito 1, corrigido por Mariana Teixeira, atributos postos a “private”, uma vez que não era necessário estarem a “public”.
Defeito 2, corrigido por Mariana Teixeira, através da implementação de um switch.
Defeito 3 e 4 não corrigidos, uma vez que o código já se encontra demasiado desenvolvido e implicaria demasiadas alterações no projeto.

Conclusions of the inspection process

Após a inspeção de código foram detetados dois métodos com mais de 60 linhas, impossíveis de reduzir tendo em conta o nível global de desenvolvimento do projeto. Alterou-se o método login(), mais propriamente a substituição de sucessivos if’s por um switch. Modificou-se a visibilidade dos atributos (Message,Post,Status) uma vez que não era necessários encontrarem-se em public. Em resumo, foram efetuadas pequenas alterações no código, que se apresentava organizado e com métodos bem construídos.

Java Inspection Checklist

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

- ✓ Are there variables or attributes with confusingly similar names?
- ✓ Is every variable and attribute correctly typed?
- ✓ Are there literal constants that should be named constants?

2. Method Definition Defects (FD)

- ✓ Are descriptive method names used in accord with naming conventions?
- ✓ For every method: Does it return the correct value at every method return point?
- ❑ Do all methods have appropriate access modifiers (private, protected, public)?

3. Class Definition Defects (CD)

- ✓ Does each class have appropriate constructors and destructors?

4. Data Reference Defects (DR)

- ✓ For every array reference: Is each subscript value within the defined bounds?
- ✓ For every object or array reference: Is the value certain to be non-null?

5. Computation/Numeric Defects (CN)

- ✓ Are there any computations with mixed data types?

6. Comparison/Relational Defects (CR)

- ✓ For every boolean test: Is the correct condition checked?
- ✓ Are the comparison operators correct?
- ✓ Has each boolean expression been simplified by driving negations inward?

7. Control Flow Defects (CF)

- ✓ Will all loops terminate?
- ✓ When there are multiple exits from a loop, is each exit necessary and handled properly?
- ❑ Can any nested if statements be converted into a switch statement?
- ✓ Does every method terminate?

8. Input-Output Defects (IO)

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

9. Module Interface Defects (MI)

- ✓ Do the values in units agree (e.g., inches versus yards)?
- ✓ If an object or array is passed, does it get changed, and changed correctly by the called method?

10. Comment Defects (CM)

- ✓ Does every method, class, and file have an appropriate header comment?
- ✓ Does every attribute, variable, and constant declaration have a comment?
- ✓ Do the comments and code agree?

- ✓ Do the comments help in understanding the code?

11. Layout and Packaging Defects (LP)

- ❑ For each method: Is it no more than about 60 lines long?
- ✓ For each compile module: Is no more than about 600 lines long?

12. Modularity Defects (MO)

- ✓ Is there repetitive code that could be replaced by a call to a method that provides the behavior of the repetitive code?
- ❑ Are the Java class libraries used where and when appropriate?

13. Storage Usage Defects (SU)

- ✓ Are arrays large enough?
- ✓ Are object and array references set to null once the object or array is no longer needed?

14. Performance Defects (PE)

- ✓ Can better data structures or more efficient algorithms be used?
- ✓ Are frequently used variables declared register?