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 Id 22
78410, Adil Abreu IC2
77584, Francisco Pereira, IC1
78090, João Pimenta, IC2
77637, Luís Casqueiro, IC1

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

November 2018

Table of Contents

Introduction

A presente aplicação visa permitir o acesso a várias plataformas online de informação relevante.
Permite aceder ao e-mail do aluno, apresentando os e-mails presentes na caixa de entrada, bem como responder aos mesmos.
Quanto às redes sociais, recolhe Tweets relacionados com o ISCTE, permite retweetar, fazer like (favorite) e publicar tweets próprios.
Já com o Facebook, permite as mesmas funcionalidades: recolhe posts relevantes e permite ainda comentar, fazer like e criar posts próprios.
O utilizador pode ordenar a informação por ordem alfabética ou por data. Tem ainda à sua disposição filtros, que permitem que só apareça informação que respeite determinados critérios (por exemplo, apresentar apenas posts das últimas 24h).
Com o nome Bom Dia Academia, esta aplicação permite aos estudantes do ISCTE-IUL ter acesso a várias plataformas e à mais variada informação tudo num só espaço.

Code inspection – Name of the component being inspected

Description of the software component being inspected

Component name (Package/Class/Method):	*Package BDA*
Component was compiled:	*Yes*
Component was executed:	*Yes*
Component was tested without errors:	*Yes, but with warnings.*
Testing coverage achieved:	*75.6%*

Code inspection checklist

Java Inspection Checklist
Copyright © 1999 by Christopher Fox. Used with permission.
1. Variable, Attribute, and Constant Declaration Defects (VC)

☑ Are descriptive variable and constant names used in accord with naming conventions?
☐ Are there variables or attributes with confusingly similar names?
☑ Is every variable and attribute correctly typed?
☑ Is every variable and attribute properly initialized?
☑ Could any non-local variables be made local?
☑ Are all for-loop control variables declared in the loop header?
☐ Are there literal constants that should be named constants?
☐ Are there variables or attributes that should be constants?
☐ Are there attributes that should be local variables?
☑ Do all attributes have appropriate access modifiers (private, protected, public)?
☑ Are there static attributes that should be non-static or vice-versa?

2. Method Definition Defects (FD)

☑ Are descriptive method names used in accord with naming conventions?
☐ Is every method parameter value checked before being used?
☑ For every method: Does it return the correct value at every method return point?
☑ Do all methods have appropriate access modifiers (private, protected, public)?
☑ Are there static methods that should be non-static or vice-versa?

3. Class Definition Defects (CD)
☑ Does each class have appropriate constructors?
☐ Do any subclasses have common members that should be in the superclass?
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?
☑ Is overflow or underflow possible during a computation?
☑ For each expressions with more than one operator: Are the assumptions about order of evaluation and precedence correct?
☑ Are parentheses used to avoid ambiguity?
6. Comparison/Relational Defects (CR)
☑ For every boolean test: Is the correct condition checked?
☑ Is each boolean expression correct?
☐ Are there improper and unnoticed side-effects of a comparison?
☐ Has an "&" inadvertently been int
7. Control Flow Defects (CF)
☑ For each loop: Is the best choice of looping constructs used?
☑ Will all loops terminate?
☑ When there are multiple exits from a loop, is each exit necessary and handled properly?
☐ Does each switch statement have a default case?
☑ Is the nesting of loops and branches too deep, and is it correct?
☐ Can any nested if statements be converted into a switch statement?
☐ Are all exceptions handled appropriately?
☑ Does every method terminate?
8. Input-Output Defects (IO)
☑ Have all files been opened before use?
☑ Are the attributes of the input object consistent with the use of the file?
☑ Have all files been closed after use?
☑ Are all I/O exceptions handled in a reasonable way?
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?
☑ 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?
☑ Is the underlying behavior of each method and class expressed in plain language?
☑ Is the header comment for each method and class consistent with the behavior of the method or class?
☑ Do the comments and code agree?
☑ Do the comments help in understanding the code?
☐ Are there enough comments in the code?
☐ Are there too many comments in the code?
11. Layout and Packaging Defects (LP)
☑ Is a standard indentation and layout format used consistently?
☐ 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 a high level of cohesion within each module (methods or class)?
☐ 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?
Java Inspection Checklist, Page 3
13. Storage Usage Defects (SU)
☑ Are arrays large enough?
14. Performance Defects (PE)
☑ Can better data structures or more efficient algorithms be used?
☑ Are logical tests arranged such that the often successful and inexpensive tests precede the more expensive and less frequently successful tests?
☑ Can the cost of recomputing a value be reduced by computing it once and storing the results?
☑ Is every result that is computed and stored actually used?

Found defects

Found defect Id	**Package, Class, Method, Line**	**Defect category**	**Description**
1	Package BDA; GUI; guardaServicos()	IO	Dado que o objeto Message não é serializable, não conseguimos guardar em ficheiro a lista com os objetos do tipo E-Mail, para uso offline

Corrective measures

1, Para resolver este problema será necessário guardar toda a informação dos objetos Message em Strings, sendo assim possível guardar em ficheiro. Depois da leitura será necessário voltar a converter essas Strings para objetos Message.

Conclusions of the inspection process

No geral, considera-se o trabalho bem concebido, com implementação funcional de todas as características pedidas e com algumas adicionais. Existem, claro, aspetos a ser melhorados, como por exemplo:
- Terem sido resolvidos atempadamente todos os problemas encontrados.
- Melhor qualidade de codificação, por exemplo, estáticos que não deviam ser estáticos e vice-versa.

No entanto a aplicação é perfeitamente funcional e poderia ser entregue ao cliente.