|  |  |
| --- | --- |
|  | *Code Inspection Report*  *Anti-Spam Configuration Software Development Project*  BSc/MSc in LIGE  Academic Year 2017/2018 – 1st Semester  Software Engineering I  Group Id: 113  73553, Rui Tomé, LIGE-PL  ISCTE-IUL, Instituto Universitário de Lisboa  1649-026 Lisbon  Portugal  December 19th 2017 |
|  |  |

**Table of Contents**

[Introduction 3](#_Toc498465002)

[Code inspection – Name of the component being inspected 3](#_Toc498465003)

[Code inspection checklist 3](#_Toc498465004)

[Found defects 3](#_Toc498465005)

[Corrective measures 3](#_Toc498465006)

[Conclusions of the inspection process 3](#_Toc498465007)

# Introduction

*To implement software that will use a genetic algorithm (NSGA-II) to create a value vector to aid anti filtering software weight their rules. The rules will be associated with emails, depending on the plugins, and through examples, we can deduce the optimal values for all the rules, making the filtering software as reliable as possible. In the making of the software, we aimed for the best possible implementation methods that help performance more than anything. The compilation times can be huge, depending on the amount of conditions being tested, but with a smarter implementation, I could drastically reduce this computing time, being one of my main objectives.*

# Code inspection – Name of the component being inspected

*Description of the software component being inspected*

|  |  |
| --- | --- |
| *Inspection date:*  *Inspection duration:*  *Inspector:* | *18/12/2017*  *20 minutes*  *Rui Tomé* |
| *Package name:* | *“antiSpamFilter”* |
| *Package was compiled:* | *18/12/2017* |
| *Package was executed:* | *18/12/2017* |
| *Errors found:* | *0* |
| *Testing coverage achieved:* | *10.7%* |

# Code inspection checklist

1. Variable, Attribute, and Constant Declaration Defects
   1. No defects found in this category.
2. Method Definition Defects
   1. No defects found in this category.
3. Class Definition Defects
   1. No defects found in this category.
4. Data Reference Defects
   1. No defects found in this category.
5. Computation/Numeric Defects
   1. No defects found in this category.
6. Comparison/Relational Defects
   1. No defects found in this category.
7. **Control Flow Defects**
   1. Not all exceptions are handled appropriately.
8. **Input-Output Defects**
   1. Not all I/O exceptions are handled in a reasonable way.
9. Module Interface Defects
   1. No defects found in this category.
10. **Comment Defects**
    1. Not every attribute, variable, and constant declaration has a comment.
11. Layout and Packaging Defects
    1. There were found methods with more than 60 lines, but they are related with graphical interface creation.
12. Modularity Defects
    1. No defects found in this category.
13. Storage Usage Defects
    1. No defects found in this category.
14. Performance Defects
    1. No defects found in this category.

# Found defects

Identify and describe found defects, opinions and suggestions.

|  |  |  |  |
| --- | --- | --- | --- |
| **Found defect Id** | **Class** | **Defect category** | **Description** |
| 1 | All classes | 10. | Found variables without comments. |
| 2 | RulesUtility.class and Evaluator.class | 7. and 8. | Found not handled exceptions. |
| 3 |  |  |  |
| ... | ... | ... | ... |

# Corrective measures

*Defect 1#: Corrected the defect by adding comments to every local variable.*

*Defect 2#: Added missing exception handling*

# Conclusions of the inspection process

*Since during software development, most of the rules from the checklist were taken in consideration, the code inspection didn’t find a lot of defect, but most of them were important and were taken care of as soon as possible.*

*However, most of them were minor changes, resulting in only a bit of code added.*