**Fall**

16

Test report

For Code-Similarity Comparer

Copyright © Team HW 2012-2013

**City University of Hong Kong**

Contents

[Introduction 3](#_Toc354757224)

[Test method 3](#_Toc354757225)

[InputTest 4](#_Toc354757226)

[OutputTest 4](#_Toc354757227)

[FormatterTest 4](#_Toc354757228)

[UtilityTest 5](#_Toc354757229)

[LineByLineComparerTest 6](#_Toc354757230)

[FuncByFuncCompararTest 6](#_Toc354757231)

[CosSimCompararTest 7](#_Toc354757232)

[RegexCompararTest 7](#_Toc354757233)

[Test Result 9](#_Toc354757234)

[Test Coverage 9](#_Toc354757235)

[Bugs found 9](#_Toc354757236)

# Introduction

After 3 weeks of testing, we completed the test frame final release. Unit test are always be performed after a new function/class is implemented, and pairs who implements the function will be response on its unit test too. Integrated test are performed on each cycle. In the testing stage, bottom-up approach is used to test our release. During the testing, there are 5 bugs are being spotted and fixed. Those discovered bug and debug methods are recorded on bug report detailed. The coverage of test cases in each class is over 99% and all the test cases are 100% passed without any error occurs.

# Test method

In this project, we are using agile approach as the software development methods. Since we are based on cycle to develop the products and continuously add feature to enhance it. Under this approach, implementation and testing have to perform simultaneously to ensure each deliverable are as buggy free as possible. In fact, there are an uncountable number of unit test are performed during the implementation stage to ensure the newly implement functions are functional. Final system test are scheduled in cycle 3. As testing are constantly performed after a new function / class implemented, only few of minor bugs are expected. In the final system test, Bottom-up approach is used. Because of it, test stub is no longer needed in our test cases.

Test case

## InputTest

|  |  |
| --- | --- |
| Test Case Name | Purpose |
| testInput\_1 | Test input by predefined filename (Same files, non c code) |
| testInput\_2 | Test input by predefined filename (Same files, c code) |
| testInputWrongFileName | Test input by entering filename during execution (1 not exists filename, 2 Same files, c code) |
| testInputNA\_1 | Test input by entering filename during execution (Same files, non c code) |
| testInputNA\_2 | Test input by entering filename during execution (Same files, c code) |

## OutputTest

|  |  |
| --- | --- |
| Test Case Name | Purpose |
| testPrint1 | Test output format (long significant float) |
| testPrint2 | Test output format (integer) |
| testPrint3 | Test output format (short significant float) |

## FormatterTest

|  |  |
| --- | --- |
| Test Case Name | Purpose |
| testReference | Test formatter (Ref text, no formatting can be applied) |
| testDeleteIndent1 | Test formatter Delete Indent (with space) |
| testDeleteIndent2 | Test formatter Delete Indent (with tab) |
| testDeleteIndent3 | Test formatter Delete Indent (with multi space) |
| testDeleteIndent4 | Test formatter Delete Indent (with multi tab) |
| testDeleteIndent5 | Test formatter Delete Indent (with multi tab and space mixed) |
| testSimplyMultispace1 | Test formatter Simply Multispace (multispace) |
| testSimplyMultispace2 | Test formatter Simply Multispace (multispace + tab mixed) |
| testSimplyMultispace3 | Test formatter Simply Multispace (multispace + tab mixed 2) |
| testDeleteComment1 | Test formatter Delete Comment (singleline + multiline comment) |
| testDeleteComment2 | Test formatter Delete Comment (multi multiline comment) |
| testDeleteComment3 | Test formatter Delete Comment (multi singleline comment) |
| testDeleteComment4 | Test formatter Delete Comment (singleline + multiline comment 2) |
| testDeleteBlankLine1 | Test formatter Delete BlankLine (\n + \r\n mixed) |
| testDeleteBlankLine2 | Test formatter Delete BlankLine (\n + \r\n only) |
| testDeleteBlankLine3 | Test formatter Delete BlankLine (\n + \r\n mixed 2) |
| testVariableDeclaration1 | Test formatter simplify Variable Declaration (declaration without initial) |
| testVariableDeclaration2 | Test formatter simplify Variable Declaration (declaration with initial) |
| testVariableDeclaration3 | Test formatter simplify Variable Declaration (declaration with partially initial) |
| testVariableDeclaration4 | Test formatter simplify Variable Declaration (declaration with partially initial 2) |
| testVariableDeclaration5 | Test formatter simplify Variable Declaration (pointer and array declaration with partially initial) |
| testVariableDeclaration6 | Test formatter simplify Variable Declaration (declaration with partially logical expression initial) |

## UtilityTest

|  |  |
| --- | --- |
| Test Case Name | Purpose |
| testJoin1 | Test string join function (5 strings, ' 'connecter) |
| testJoin2 | Test string join function (6 strings, ','connecter) |
| testIsVar1 | Test isVar checker (valid variable 1) |
| testIsVar2 | Test isVar checker (valid variable 2) |
| testIsVar3 | Test isVar checker (invalid variable (? included)) |
| testIsVar4 | Test isVar checker (invalid variable (digit beginning)) |
| testExtractVarNames1 | Test ExtractVarNames (3 variables only) |
| testExtractVarNames2 | Test ExtractVarNames (3 variables + some code) |
| testExtractVarNames3 | Test ExtractVarNames (some ADT variables + variables + some code) |
| testExtractVarNames4 | Test ExtractVarNames (return statement with complex function call and logical expression) |
| testLcs1 | Test lcs algorithm function (4:5 tokens, 2 matched) |
| testLcs2 | Test lcs algorithm function (4:5 tokens, 0 matched) |
| testLcs3 | Test lcs algorithm function (4:4 tokens, 4 matched) |
| testReplace1 | Test replace function (partial match pattern) |
| testReplace2 | Test replace function (partial match pattern, reversed parameter) |
| testReplace3 | Test replace function (same pattern) |
| testReplace4 | Test replace function (same pattern + additional statement) |
| testReplace5 | Test replace function (complex pattern) |
| testSplitFunction | Test Split function (3 functions) |

## LineByLineComparerTest

|  |  |
| --- | --- |
| Test Case Name | Purpose |
| testCompare\_simple\_case1 | Test compare function by single line inputs(matched) |
| testCompare\_simple\_case2 | Test compare function by single line inputs(not matched) |
| testCompare\_simple\_case3 | Test compare function by 2 line inputs(1 matched,1 not matched) |
| testCompare\_complex \_case1 | Test compare function by multiline inputs(5:5 lines, 3 matched) |
| testCompare\_complex \_case2 | Test compare function by multiline inputs(same input) |
| testCompare\_complex \_case3 | Test compare function by multiline inputs(8:12 lines, 1 matched) |
| testCompare\_complex \_case4 | Test compare function by multiline inputs(10:8 lines, 3 matched) |

## FuncByFuncCompararTest

|  |  |
| --- | --- |
| Test Case Name | Purpose |
| testCompare\_1 | Test FuncByFuncComparar compare function (2 different function call statement) |
| testCompare\_2 | Test FuncByFuncComparar compare function (3 function call statement, different order) |
| testCompare\_3 | Test FuncByFuncComparar compare function (different function name and statements) |
| testCompare\_4 | Test FuncByFuncComparar compare function (different function name and statements, parameter reversed ) |
| testCompare\_5 | Test FuncByFuncComparar compare function (same function name ,different statements, parameter reversed ) |
| testCompare\_6 | Test FuncByFuncComparar compare function (multi functions, partial identical) |
| testCompare\_7 | Test FuncByFuncComparar compare function (multi functions, partial identical, parameter reversed ) |
| testCompare\_8 | Test FuncByFuncComparar compare function (multi functions, partial identical, parameter reversed 2) |

## CosSimCompararTest

|  |  |
| --- | --- |
| Test Case Name | Purpose |
| testCompare1 | Test CosSimComparar compare function (2 different function call statement) |
| testCompare2 | Test CosSimComparar compare function (3 function call statement, different order) |
| testCompare3 | Test CosSimComparar compare function (different function name and statements) |
| testCompare4 | Test CosSimComparar compare function (different function name and statements, parameter reversed ) |
| testCompare5 | Test CosSimComparar compare function (same function name ,different statements, parameter reversed ) |
| testCompare6 | Test CosSimComparar compare function (same function name, different statements) |
| testPubCosSimliar1 | Test PubCosSimliar function (identical string) |
| testPubCosSimliar2 | Test PubCosSimliar function (code shifted ) |
| testPubCosSimliar3 | Test PubCosSimliar function (first part removed) |
| testPubCosSimliar4 | Test PubCosSimliar function (last part removed) |

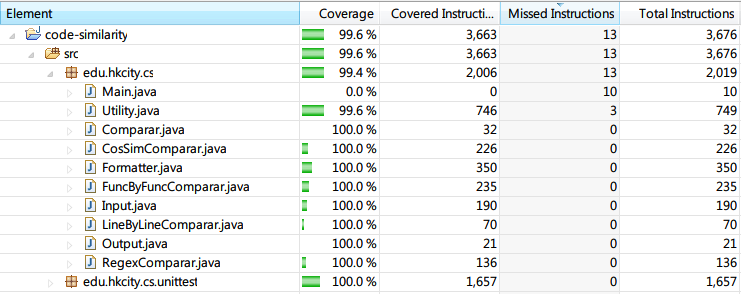
## RegexCompararTest

|  |  |
| --- | --- |
| Test Case Name | Purpose |
| testCalRegedSim1 | Test CalRegedSim function (words) |
| testCalRegedSim2 | Test CalRegedSim function (equation) |
| testCalRegedSim3 | Test CalRegedSim function (equation + words) |
| testCompare1 | Test RegexComparar compare function (words) |
| testCompare2 | Test RegexComparar compare function (words) |
| testCompare3 | Test RegexComparar compare function (equation + words) |

# 

# Test Result

## Test Coverage



All class contains more than 99.5% coverage (Main is exclude form the test).

## Bugs found

|  |  |  |  |
| --- | --- | --- | --- |
| Bug ID | Bug Description | Bug Location | Assigned to |
| Bug 000001 | java.util.regex.PatternSyntaxException | Utility.replace() | YUAN Siyan |
| Bug 000002 | java.lang.StringIndexOutOfBoundsException | Utility.replace() | YUAN Siyan |
| Bug 000003 | splitFunction() ignore first function | Utility.splitFunction() | Bill Yeung |
| Bug 000004 | compare() function may return negative number | FuncByFuncComparar.compare() | Thomas Chan |
| Bug 000005 | Lcs output not correct | Utility.lcs() | Bill Yeung |