**Winter 2017 Contest**

**Software Requirements Traceability Matrix**

**2017 February 17th**

**Team: Swifty X**

Final Software Requirements for the Winter 2017 Contest are as follows:

1. The solution shall use a minimum of Java7 and a maximum of Java 8.

The solution is developed and tested with Java 8 as Oracle strongly recommends using Java 8.

2. The solution shall test executables in the form of Java Jars via the command line.

The solution uses -jarToTestPath option to accept the path of the executable jar for execution. See command below:

|  |
| --- |
| idt.bat -jarToTestPath C:\\idt\_contest\\jars\\TesterTypeCheck.jar -jacocoOutputPath C:\\idt\_contest\\jacoco -jacocoAgentJarPath C:\\idt\_contest\\jacoco\\lib\\jacocoagent.jar -bbTests 100 -timeGoal 1 |

3. The solution shall increase code coverage of the Software Under Test (SUT) via exploratory black box testing.

We have improved code coverage across all four jars with executeBasicTests() and executeSecurityTests(). Table below shows the code coverage between Jan 23 and Feb 16 with argument bbTests set to 100 and timeGoal 1:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Jar Name | Basic Test  (1/23/17) | Basic Test  (2/16/17) | Security Test  (1/23/17) | Security Test  (2/16/17) |
| TesterTypeCheck | 65.45% | 79.09% | 84.39% | 99.39% |
| RegexPatternMatch | 4.76% | 10.48% | 13.35% | 15.61% |
| LeetConverter | 75.71% | 85.71% | 89.07% | 99.07% |
| CommandLineEncryption | 16.14% | 35.34% | 24.97% | 36.06% |

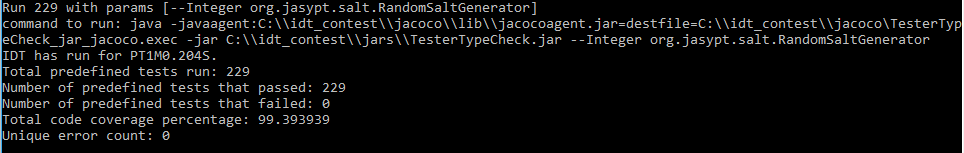
4. The solution shall increase the number of unique exceptions generated by the SUT via exploratory black box testing.

Unique exception counts will increase when number of test iterations and/or test time increase.

5. The solution shall accept the path of the executable jar to test as a required command line argument in the form of: -jarToTestPath <jar to test path here>.

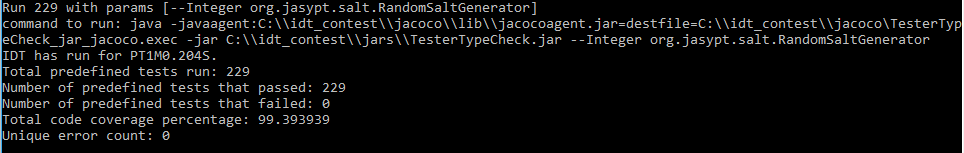
In Command Prompt, navigate to the project directory (where pom.xml is located), execute the following command (as shown in box), test results display as follows. Notice “C:\\idt\_contest\\jars\\TesterTypeCheck.jar” is passed for -jarToTestPath as jar file path.

|  |
| --- |
| idt.bat -jarToTestPath C:\\idt\_contest\\jars\\TesterTypeCheck.jar -jacocoOutputPath C:\\idt\_contest\\jacoco -jacocoAgentJarPath C:\\idt\_contest\\jacoco\\lib\\jacocoagent.jar -bbTests 100 -timeGoal 1 |



6. The solution shall accept the path to the directory where jacoco will generate output files as a required command line argument in the form of: -jacocoOutputPath <jacoco output dir path here>.

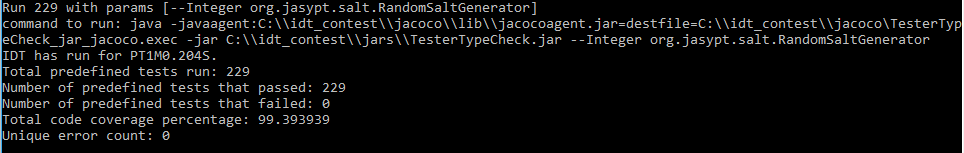
In Command Prompt, navigate to the project directory (where pom.xml is located), execute the following command (as shown in box), test results display as follows. Notice “C:\\idt\_contest\\jacoco” is passed for -jacocoOutputPath as jacoco output path.



|  |
| --- |
| idt.bat -jarToTestPath C:\\idt\_contest\\jars\\TesterTypeCheck.jar -jacocoOutputPath C:\\idt\_contest\\jacoco -jacocoAgentJarPath C:\\idt\_contest\\jacoco\\lib\\jacocoagent.jar -bbTests 100 -timeGoal 1 |

7. The solution shall accept the path of the jacoco agent jar in the form of: -jacocoAgentJarPath <jacoco agent jar path here>.

In Command Prompt, navigate to the project directory (where pom.xml is located), execute the following command (as shown in box), test results display as follows. Notice “C:\\idt\_contest\\jacoco\\lib\\jacocoagent.jar” is passed for -jacocoAgentJarPath as jacoco agent jar path.



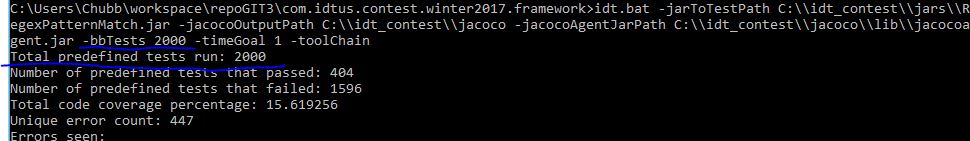
|  |
| --- |
| idt.bat -jarToTestPath C:\\idt\_contest\\jars\\TesterTypeCheck.jar -jacocoOutputPath C:\\idt\_contest\\jacoco -jacocoAgentJarPath C:\\idt\_contest\\jacoco\\lib\\jacocoagent.jar -bbTests 100 -timeGoal 1 |

8. The solution shall execute the specified number of exploratory black box test iterations (default 1000)

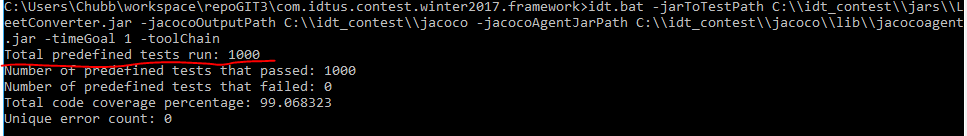
against the SUT.

8.1 Execute the solution with -bbTests option set to 2000 with the following command. The output below has Total predefined tests run: 2000 is returned, which indicates 2000 test iterations were executed.

|  |
| --- |
| idt.bat -jarToTestPath C:\\idt\_contest\\jars\\RegexPatternMatch.jar -jacocoOutputPath C:\\idt\_contest\\jacoco -jacocoAgentJarPath C:\\idt\_contest\\jacoco\\lib\\jacocoagent.jar –bbTests 2000 -timeGoal 1 |



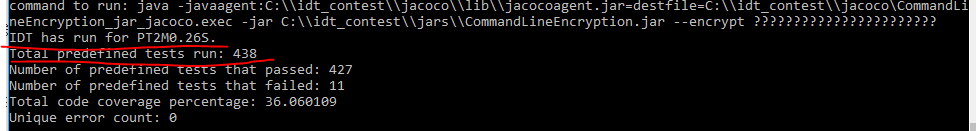
8.2 Execute the solution without -bbTests. The screenshot below displays the command and output Total predefined tests run: 1000, which indicates 1000 test iterations were executed when -bbTests option is omitted.



9. The solution shall run additional exploratory black box test iterations, if the specified number of exploratory black box test iterations completes within a specified test time goal (default 5 minutes).

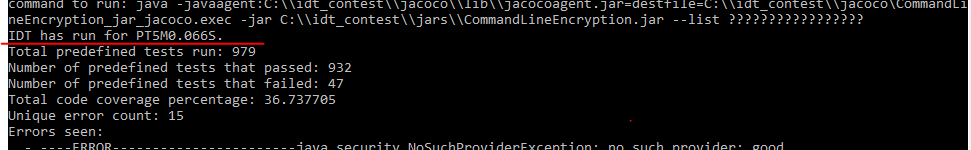
9.1 Execute the solution with -bbTests set to 10 and -timeGoal option set to 2 using the following command. The output showed total of 438 tests were executed, that is 428 tests more than specified 10.

|  |
| --- |
| idt.bat -jarToTestPath C:\\idt\_contest\\jars\\CommandLineEncryption.jar -jacocoOutputPath C:\\idt\_contest\\jacoco -jacocoAgentJarPath C:\\idt\_contest\\jacoco\\lib\\jacocoagent.jar –bbTests 10 -timeGoal 2 |



9.2 Execute the solution without -timeGoal using the following command. The output displays IDT has run for PT5M0.066S, which indicates the test ran for 5 minutes and 0.06 second.

|  |
| --- |
| idt.bat -jarToTestPath C:\\idt\_contest\\jars\\CommandLineEncryption.jar -jacocoOutputPath C:\\idt\_contest\\jacoco -jacocoAgentJarPath C:\\idt\_contest\\jacoco\\lib\\jacocoagent.jar –bbTests 100 |



10. The solution shall accept the number of exploratory black box tests to run as an optional command line argument in the form of: -bbTests <number of tests here>.

Execute the solution in Command Prompt with and without -bbTests option respectively. See section 8 above.

11. The solution shall accept the test time goal as an optional command line argument in the form of: - timeGoal <number of minutes here>.

Execute the solution in Command Prompt with and without - timeGoal option respectively. See section 9 above.

12. The solution shall support applications that take a fixed number of arguments.

Execute the solution to test against “RegexPatternMatch.jar” that takes a fixed number of arguments.

13. The solution shall support applications that take a variable number of arguments.

Execute the solution to test against “CommandLineEncryption.jar” that takes a variable number of arguments.

14. The solution shall support integer arguments.

Execute the solution to test against “TesterTypeCheck.jar” and check that “--Integer” option is tested as expected

15. The solution shall support double arguments.

Execute the solution to test against “TesterTypeCheck.jar” and check that “--Double” option is tested as expected.

16. The solution shall support bounded String argument

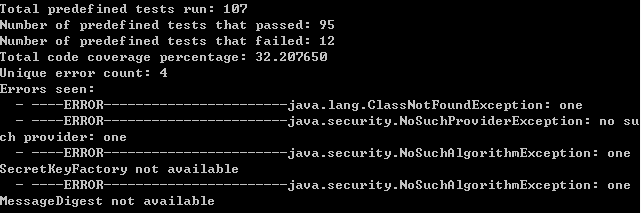
Execute the solution to test against “TesterTypeCheck.jar” and check that “--NextOneIsFormatted” option is tested as expected.

17. The solution shall support unbounded String arguments.

Execute the solution to test against “TesterTypeCheck.jar” and check that “--String” option is tested as expected.

18. The solution shall record unique exceptions or crashes seen during exploratory testing.

Execute the solution to test against “CommandLineEncryption.jar” and check that summary at end of test record unique exceptions or crashes under “Errors seen” section.



19. The solution shall print to stdout a YAML report at the end of testing that is in the following format: Total predefined tests run: <count here>

Number of predefined tests that passed: <count here> Number of predefined tests that failed: <count here>

Total code coverage percentage: <percentage as a double here> Unique error count: <count here>

Errors seen:

- <Exception or crash information here>

- <Exception or crash information here>

- <Exception or crash information here>

Execute the solution and verify that a YAML report is printed as output. Please see section 18 above.

20. The solution shall accept an optional command line argument of the form "-toolChain" that will limit the output to contain only the parseable YAML indicated above.

Execute the solution with and without “-toolChain” option respectively and verify that the output contains only YAML report when running with “-toolChain” option.

21. The solution shall be delivered as an executable jar with the name com.idtus.contest.winter2017.framework.jar.

Compile the project with gradle, the executable jar “com.idtus.contest.winter2017.framework.jar” is generated under *build\libs* directory.

Compile the project with maven, the executable jar “com.idtus.contest.winter2017.framework-0.0.1-SNAPSHOT-jar-with-dependencies.jar” is generated under *target* directory.

22. The solution shall also be delivered as an Maven Project in Eclipse (zipped up) with the name com.idtus.contest.winter2017.framework.zip.

The solution is developed in Eclipse. which can be imported as a maven project into Eclipse and can be compressed as a zip file.