Winter 2017 Contest

Black-box Testing Framework Version 2.0 Documentation

2017 February 17th

Team: Swifty X

Change History

Version	Date	Change	Author
1.0	December 15, 2016	Initial Version	IDT
2.0	February 17, 2017	1. Changed the tone of the documentation and repurposed it for developers and testers in general. 2. Section 1: Added more source files and unit test files. 3. Section 2: Added information about dependencies, Gradle, and Maven. 4. Section 4: Added usage information for three new optional arguments. Replaced Run Configuration screenshot. 5. Section 5: Removed Extending the Framework section. Added Adding Test Cases to the Framework section. 6. Section 6: Added Building and Executing the Framework JAR section. 7. Section 7: Added Launching the Framework GUI section. 8. Section 8: Added Output File section.	Swifty X

Introduction

IDT has developed a testing framework that will be used by developers and testers to test executable jars (Java). This document contains all of the documentation necessary for executing and extending the framework.

We would like to welcome you to the project. We are excited to work with your team to extend our testing framework to expose security vulnerabilities or weaknesses in the software under test.

Contents

1. Framework and Supporting File Archives

IDT has two .zip files that contain our framework codebase and supporting files.

com.idtus.contest.winter2017.framework.zip – This is a compressed archive of the framework codebase. You will need to unzip this archive on your development machines. **It does not matter where you ultimately put this codebase on your computer because you will be importing it into Eclipse and interacting with it though an IDE.** Expanding compressed framework should reveal the following files that are part of a Maven Project for Eclipse:

Program Source Files

com.idtus.contest.winter2017.framework\.classpath
com.idtus.contest.winter2017.framework\.project
com.idtus.contest.winter2017.framework\build.gradle
com.idtus.contest.winter2017.framework\idt.bat
com.idtus.contest.winter2017.framework\idt.sh
com.idtus.contest.winter2017.framework\idt-ui.bat
com.idtus.contest.winter2017.framework\idt-ui.shcom.idtus.contest.winter2017.framework\pom.xml
com.idtus.contest.winter2017.framework\README.md
com.idtus.contest.winter2017.framework\src\main\java\contest\winter2017\Config.java
com.idtus.contest.winter2017.framework\src\main\java\contest\winter2017\ContinuousParameter.java
com.idtus.contest.winter2017.framework\src\main\java\contest\winter2017\DoubleParameter.java
com.idtus.contest.winter2017.framework\src\main\java\contest\winter2017\EnumerationParameter.java
com.idtus.contest.winter2017.framework\src\main\java\contest\winter2017\FormattedStringParameter.java
com.idtus.contest.winter2017.framework\src\main\java\contest\winter2017\Gui.java
com.idtus.contest.winter2017.framework\src\main\java\contest\winter2017\GuiLauncher.java
com.idtus.contest.winter2017.framework\src\main\java\contest\winter2017\IntegerParameter.java
com.idtus.contest.winter2017.framework\src\main\java\contest\winter2017\JsonParser.java
com.idtus.contest.winter2017.framework\src\main\java\contest\winter2017\Main.java
com.idtus.contest.winter2017.framework\src\main\java\contest\winter2017\Output.java
com.idtus.contest.winter2017.framework\src\main\java\contest\winter2017\Parameter.java
com.idtus.contest.winter2017.framework\src\main\java\contest\winter2017\ParameterFactory.java
com.idtus.contest.winter2017.framework\src\main\java\contest\winter2017\Parameters.java
com.idtus.contest.winter2017.framework\src\main\java\contest\winter2017\StdoutWrapper.java
com.idtus.contest.winter2017.framework\src\main\java\contest\winter2017\StringParameter.java
com.idtus.contest.winter2017.framework\src\main\java\contest\winter2017\Test.java
com.idtus.contest.winter2017.framework\src\main\java\contest\winter2017\Tester.java
com.idtus.contest.winter2017.framework\src\main\java\contest\winter2017\YamlReport.java
com.idtus.contest.winter2017.framework\src\main\java\contest\winter2017\parameter\BinaryAccessingIterator.java
com.idtus.contest.winter2017.framework\src\main\java\contest\winter2017\parameter\CharacterGenerator.java
com.idtus.contest.winter2017.framework\src\main\java\contest\winter2017\parameter\CharacterRangeIterator.java
com.idtus.contest.winter2017.framework\src\main\java\contest\winter2017\parameter\CharacterSet.java
com.idtus.contest.winter2017.framework\src\main\java\contest\winter2017\parameter\DoubleGenerator.java
com.idtus.contest.winter2017.framework\src\main\java\contest\winter2017\parameter\DoubleRangeIterator.java
com.idtus.contest.winter2017.framework\src\main\java\contest\winter2017\parameter\FormattedStringGenerator.java
com.idtus.contest.winter2017.framework\src\main\java\contest\winter2017\parameter\Generator.java
com.idtus.contest.winter2017.framework\src\main\java\contest\winter2017\parameter\InfiniteGenerator.java
com.idtus.contest.winter2017.framework\src\main\java\contest\winter2017\parameter\IntegerGenerator.java
com.idtus.contest.winter2017.framework\src\main\java\contest\winter2017\parameter\IntegerRangeIterator.java
com.idtus.contest.winter2017.framework\src\main\java\contest\winter2017\parameter\RegExprs.java
com.idtus.contest.winter2017.framework\src\main\java\contest\winter2017\parameter\StringGenerator.java

com.idtus.contest.winter2017.framework\src\main\java\contest\winter2017\util\NumberUtil.java
com.idtus.contest.winter2017.framework\src\main\java\contest\winter2017\util\PatternRecognizer.java
com.idtus.contest.winter2017.framework\src\main\resources\config.yaml
com.idtus.contest.winter2017.framework\src\main\resources\log4j.properties
com.idtus.contest.winter2017.framework\src\main\resources\icon\checkbox_off_blue.png
com.idtus.contest.winter2017.framework\src\main\resources\icon\checkbox_on_blue.png
com.idtus.contest.winter2017.framework\src\main\resources\icon\icon.pgn
com.idtus.contest.winter2017.framework\src\main\resources\icon\logo.png
com.idtus.contest.winter2017.framework\src\main\resources\json\CommandLineEncryption.json
com.idtus.contest.winter2017.framework\src\main\resources\json\LeetConverter.json
com.idtus.contest.winter2017.framework\src\main\resources\json\RegexPatternMatch.json
com.idtus.contest.winter2017.framework\src\main\resources\json\TesterTypeCheck.json
com.idtus.contest.winter2017.framework\src\main\resources\parameter\validReg.txt

Unit Test Source Files

com.idtus.contest.winter2017.framework\src\test\java\contest\winter2017\ConfigTest.java
com.idtus.contest.winter2017.framework\src\test\java\contest\winter2017\DoubleGeneratorTest.java
com.idtus.contest.winter2017.framework\src\test\java\contest\winter2017\FormattedStringGeneratorTest.java
com.idtus.contest.winter2017.framework\src\test\java\contest\winter2017\GeneratorsTest.java
com.idtus.contest.winter2017.framework\src\test\java\contest\winter2017\MainTest.java
com.idtus.contest.winter2017.framework\src\test\java\contest\winter2017\OutputTest.java
com.idtus.contest.winter2017.framework\src\test\java\contest\winter2017\StringGeneratorTest.java
com.idtus.contest.winter2017.framework\src\test\java\contest\winter2017\TesterTest.java
com.idtus.contest.winter2017.framework\src\test\java\contest\winter2017\YamlReportTest.java
com.idtus.contest.winter2017.framework\src\test\java\contest\winter2017\parameter\RegExprsTest.java
com.idtus.contest.winter2017.framework\src\test\java\contest\winter2017\util\NumberUtilTest.java
com.idtus.contest.winter2017.framework\src\test\java\contest\winter2017\util\PatternRecognizerTest.java

com.idtus.contest.winter2017.supportingfiles.zip – This is a compressed archive of the supporting files that you will need to execute the framework. **The archive contains a directory called 'idt_contest' that we recommend putting directly into your C:\ directory (if you are using windows), the root directory (if you are using macOS), or your user's directory (if you are using Linux).** You will ultimately be referencing multiple files/directories as arguments for the framework, so we recommend you keep the paths short for convenience sake. This expanded structure should contain the following files:

idt_contest\jacoco\lib\jacocoagent.jar	- this is the jacoco agent jar
idt_contest\jars\CommandLineEncryption.jar	- this is an executable black-box jar that you will test
idt_contest\jars\LeetConverter.jar	- this is an executable black-box jar that you will test
idt_contest\jars\RegexPatternMatch.jar	- this is an executable black-box jar that you will test
idt_contest\jars\TesterTypeCheck.jar	- this is an executable **white-box** jar that you will test

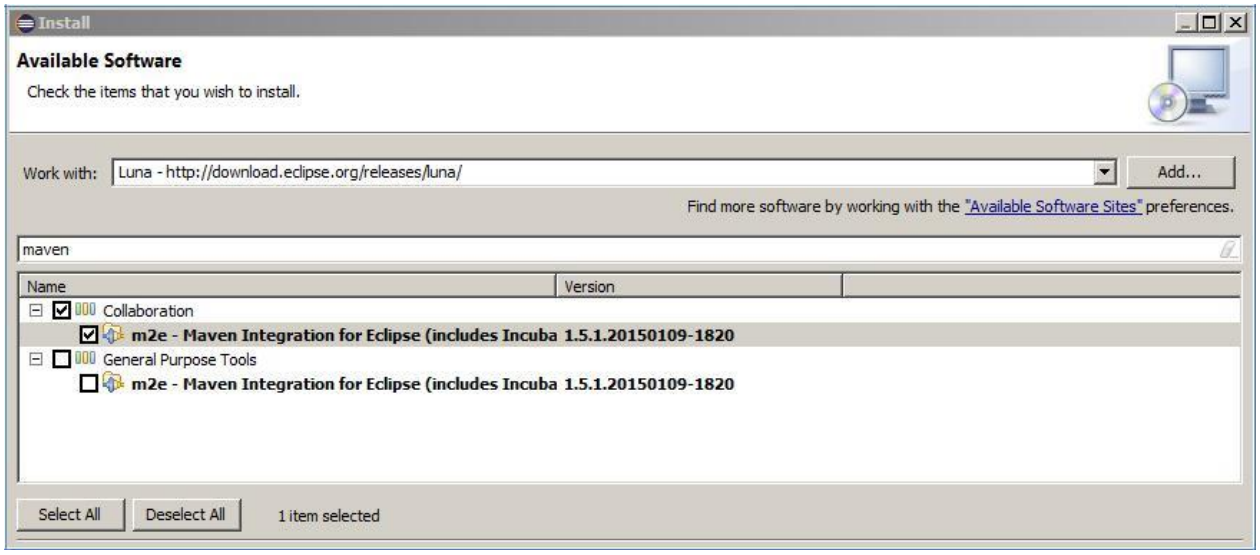
2. Framework Dependencies

We intend for you to develop and execute the framework through Eclipse. There are a minimal set of dependencies that you will need to install in order to get the framework up and running.

A. **Java SE 1.7 or greater** - You will need to have Java installed and available as a command on the command line (added to your path). **The minimum acceptable Java version is 1.7**. The framework will be executing external Java processes, and will require the command 'java –jar' to resolve correctly from the command line.

B. **Eclipse 4.4 or greater**- You will need to have Eclipse installed because the framework was created as a Maven Project for Eclipse. The project was originally created using Eclipse 4.4. (Luna), but there do not appear to be any requirements on a specific version of Eclipse. If you have a version prior to 4.4, you should be okay. Our recommendation is to use Luna (4.4), Mars (4.5), or Neon (4.6). http://www.eclipse.org/downloads/

C. **Maven2Eclipse (m2e)** – You will need to make sure that you have Maven support in Eclipse installed to import and resolve dependencies associated with the framework project. One way to test if you already have this, is to go to File > Import, and look for Maven > Existing Maven Projects. If you do not have that option, you will need to install m2e from Help > Install New Software. Look for the site that uses the name of the version of Eclipse that you have installed (e.g. Luna, Mars, or Neon). You will find 'm2e' under the Collaboration entry. Install m2e and verify that you see the Maven options under File > Import.


Install
Available Software
Check the items that you wish to install.
Work with: Luna - http://download.eclipse.org/releases/luna/
Add...
Find more software by working with the "Available Software Sites" preferences.
maven
Name
Version
Collaboration
m2e - Maven Integration for Eclipse (includes Incuba 1.5.1.20150109-1820
General Purpose Tools
m2e - Maven Integration for Eclipse (includes Incuba 1.5.1.20150109-1820
Select All
Deselect All
1 item selected

Installing Maven from Help > Install New Software in Eclipse Luna

Eclipse will use Maven to resolve the library dependencies for the framework automatically using the pom.xml file that is included with the framework project:

(GROUP)	(ARTIFACT IDs)
org.jacoco	jacoco-maven-plugin, org.jacoco.core, org.jacoco.agent, org.jacoco.report
org.ow2.asm	asm-all

commons-cli	commons-cli
com.google.guava	guava
log4j	log4j
org.slf4j	slf4j-api, slf4j-log4j12
org.yaml	snakeyaml
junit	junit
com.google.truth	truth
org.apache.commons	commons-lang3
commons-collections	commons-collections
com.fasterxml.jackson.core	jackson-databind

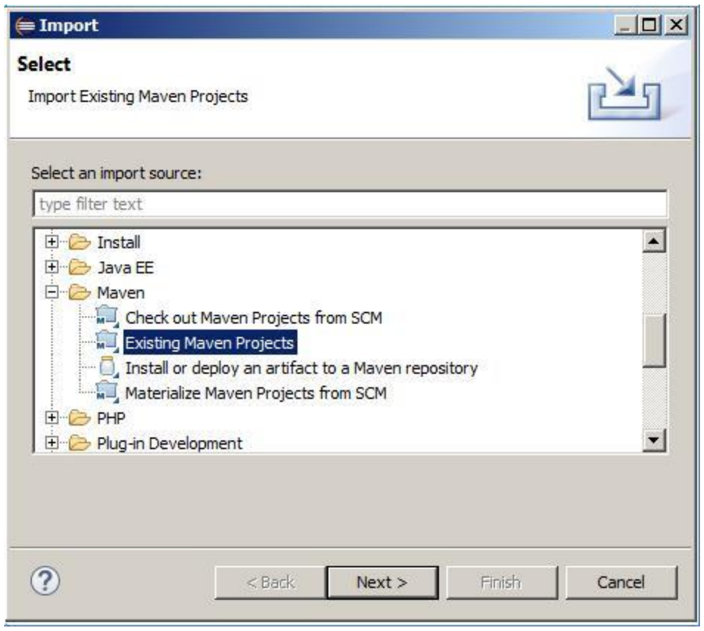
D. **Gradle** – You will need to have Gradle for java build tool. It supports automatic download and configuration of dependencies and libraries. It compiles Java source code, runs unit tests, and create a JAR file, which can be distributed and run at command line. The current version is 3.3. You can download Gradle at https://gradle.org/install.

E. **Maven 3.x or greater** – Alternatively you can use Maven for java build tool. You can download Maven at http://maven.apache.org.

3. Importing the framework into Eclipse

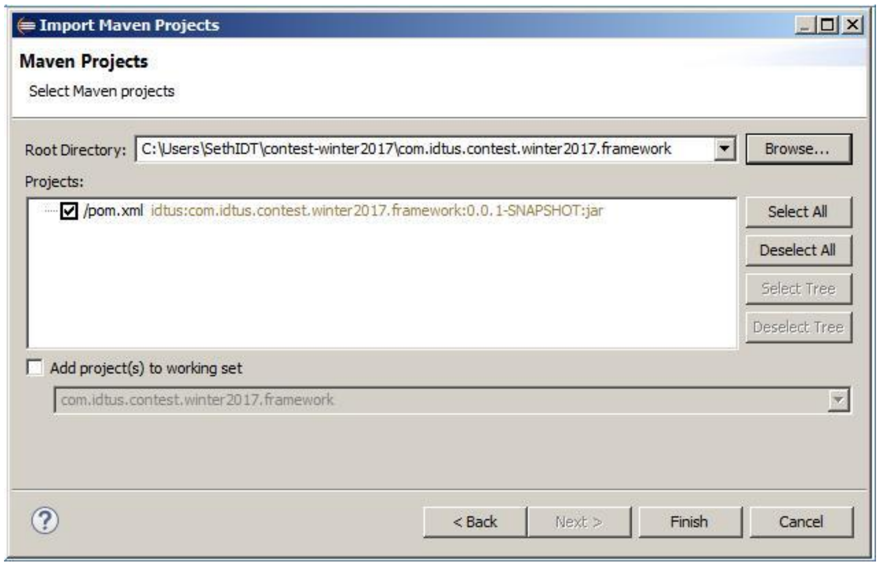
Once you have Java, Eclipse, and Maven all installed, you can import the framework project into Eclipse.

A. **File** > **Import** will bring up a window with several options, select **Maven** > **Existing Maven Projects** and click on the Next button at the bottom.


Import
Select
Import Existing Maven Projects
Select an import source:
type filter text
Install
Java EE
Maven
Check out Maven Projects from SCM
Existing Maven Projects
Install or deploy an artifact to a Maven repository
Materialize Maven Projects from SCM
PHP
Plug-in Development
< Back
Next >
Finish
Cancel

Selecting 'Existing Maven Projects' in Eclipse Import window

B. Use the Browse button on the Maven Projects page to select the path to the framework on your machine (it was unzipped from **com.idtus.contest.winter2017.framework.zip** somewhere on your system as part of step 1). The framework should be a folder called **com.idtus.contest.winter2017.framework**. When the framework is selected, click on the Finish button at the bottom.



Root Directory: C:\Users\SethIDT\contest-winter2017\com.idtus.contest.winter2017.framework

Projects:

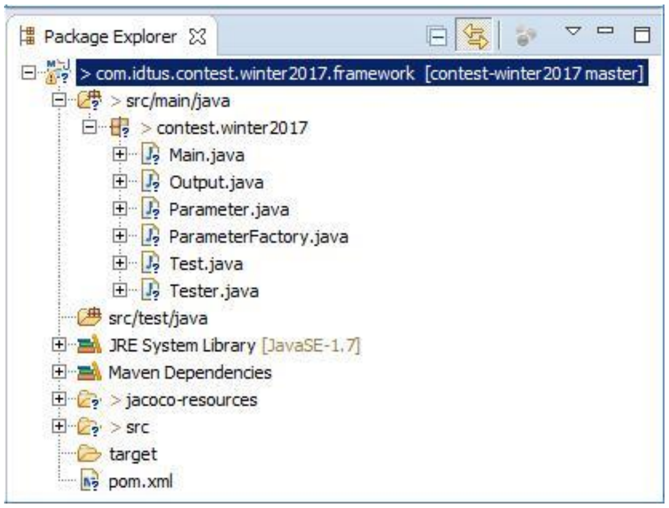
☑ /pom.xml idtus:com.idtus.contest.winter2017.framework:0.0.1-SNAPSHOT:jar

☐ Add project(s) to working set

com.idtus.contest.winter2017.framework

Browse to location of the unzipped framework code on your system and click on the Finish button

C. If you have successfully imported the framework project into Eclipse, you should see framework project appear in the Package Explorer on the left hand side of Eclipse.



Successful import of framework project into Eclipse

D. If you have an error in your project due to the fact that the JRE for the project is set to something other than 1.7 or 1.8, simply switch the version of JRE on the project build path by right clicking on 'JRE System Library [JSE-1.5]' under the project contents, and selecting 'Properties'. Under the properties, you can set the proper JRE (1.7 – 1.8).

4. Executing the framework in Eclipse

The framework, at this time, uses three **required** arguments (-jacocoAgentJarPath, -jacocoOutputPath, and - jarToTestPath) and three optional arguments (-bbTests, - timeGoal, and -toolChain) in order to execute and test a black-box executable jar. Additionally, the framework currently accepts a –h or –help switch to trigger the help menu.

```
usage: com.idtus.contest.winter2017.framework [-h] [-help]
       [-jacocoAgentJarPath <arg>] [-jacocoOutputPath
       <arg>] [-jarToTestPath <arg>]
 -h                           help
 -help                        help
 -jacocoAgentJarPath <arg>     path to the jacoco agent jar
 -jacocoOutputPath <arg>      path to directory for jacoco output
 -jarToTestPath <arg>         path to the executable jar to test
 -bbTests <arg>               number of iterations to test, default to 1000 if omitted
 -timeGoal <arg>              number of minutes to test, default to five minutes if omitted
 -toolChain <arg>             flag to output in YAML format, default to false if omitted
```

The entry-point for the framework is found in Main.java.

To execute the framework through Eclipse, right click on the Main.java file in the Package Explorer and select Run As > Java Application. If the three required arguments have not been set, you should see the usage print to stdout.

To set the three required arguments for the framework, you will need to open the Run Configuration for the project. This can be done by clicking on Run > Run Configurations at the top of Eclipse. Make sure that the 'Main' configuration is selected in the left side of the Run Configuration window and then select the 'Arguments' tab on the right hand side. You will need to add program arguments for the three required arguments.

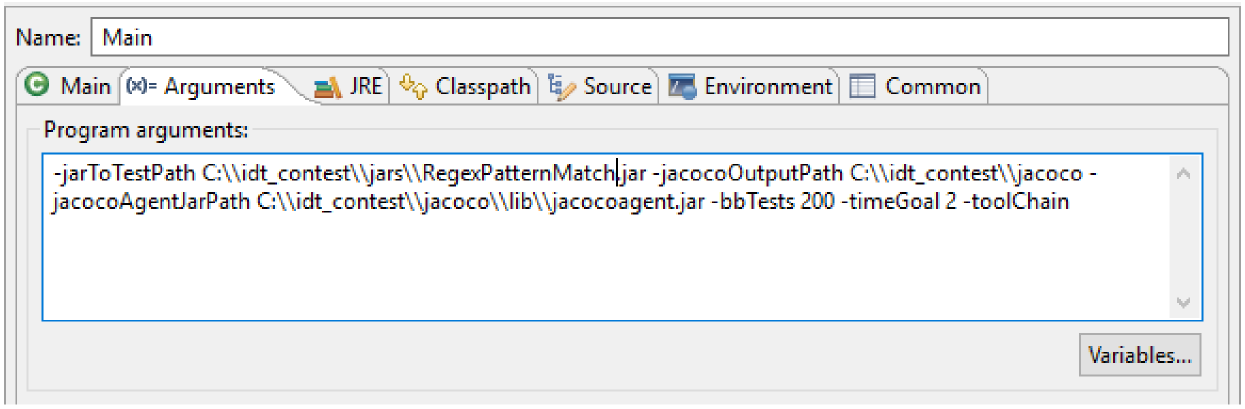
jacocoAgentJarPath – We distributed the jacoco agent jar file as part of the supporting files in com.idtus.contest.winter2017.supportingfiles.zip. Inside the expanded directory, the relative path to the jacoco agent jar should be: idt_contest\jacoco\lib\jacocoagent.jar. If you put the idt_contest directory directly into the C:\ directory, your jacocoAgentJarPath (absolute) should be **C:\ idt_contest\jacoco\lib\jacocoagent.jar.**

jacocoOutputPath – This could be any directory on your system - it is not important that you use a specific one. When we were running locally, we used idt_contest\jacoco as our jacoco output directory. If you put the idt_contest directory directly into the C:\ directory, your jacocoOutputPath might be **C:\ idt_contest\jacoco.**

jarToTestPath – This is the path to the black-box executable jar that you intend to test. We distributed all of the sample black-box executable jars as part of the supporting files in com.idtus.contest.winter2017.supportingfiles.zip. Inside the expanded directory, the relative path to the jars to test should be: idt_contest\jars. If you put the idt_contest directory directly into the C:\ directory and intended to test RegexPatternMatch.jar, your jarToTestPath might be **C:\idt_contest\jars\RegexPatternMatch.jar**.

All arguments are entered with a hyphen preceding the argument name and a space separating the argument name and value:

```
-jarToTestPath C:\\idt_contest\\jars\\RegexPatternMatch.jar
-jacocoOutputPath C:\\idt_contest\\jacoco
-jacocoAgentJarPath C:\\idt_contest\\jacoco\\lib\\jacocoagent.jar
-bbTests 200
-timeGoal 2
-toolChain
```



Program arguments:

```
-jarToTestPath C:\\idt_contest\\jars\\RegexPatternMatch.jar -jacocoOutputPath C:\\idt_contest\\jacoco -
jacocoAgentJarPath C:\\idt_contest\\jacoco\\lib\\jacocoagent.jar -bbTests 200 -timeGoal 2 -toolChain
```

Example of program arguments filled out completely to execute framework against RegexPatternMatch.jar

After updating Arguments > Program Arguments and executing the framework again, the Console in Eclipse should show you results from the testing.

5. Adding Test Cases to the Framework

You can define more test cases in a JSON file that will be executed by this test framework. The JSON file should be put on $CLASSPATH and have the same name as the jar under test. Please see the JSON files under src/main/resources/json/ source directory for more information.

RegexPatternMatch.json

```
[
    {
        "parameters":[
            "^a",
            "aboard"
        ],
        "stdOutExpectedResultRegex":".*matches.*",
        "stdErrExpectedResultRegex":""
    },
    {
        "parameters":[
            "sha$",
            "sha"
        ],
        "stdOutExpectedResultRegex":".*matches.*",
        "stdErrExpectedResultRegex":""
       },
       ……
       ……
       ……
       ……

]
```

6. Building and Executing the Framework JAR

When you want to build JAR file for the framework source code, there are two options. You can choose from Gradle and Maven. Both are widely adopted build tools by industries.

Build with Gradle

Use *build.gradle* file to configure dependencies and tasks for build. Run following steps at command line under com.idtus.contest.winter2017.framework directory:

```
>gradle tasks
>gradle build -x test
```

Executable jar "com.idtus.contest.winter2017.framework.jar" is generated under build\libs directory. If you are using Windows, execute the jar through the batch file:

```
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
```

If your machine runs on macOS, Unix, or Linux, execute the jar through the bash file.

```
sh idt.sh -jarToTestPath /idt_contest/jars/TesterTypeCheck.jar -jacocoOutputPath
/idt_contest/jacoco -jacocoAgentJarPath /idt_contest/jacoco/lib/jacocoagent.jar -
bbTests 100 -timeGoal 1 -toolChain
```

Build with Maven

Maven uses *pom.xml* to manage dependencies and build configurations. Run following steps at command line under com.idtus.contest.winter2017.framework directory:

```
>mvn clean compile assembly:single
```

Executable jar "com.idtus.contest.winter2017.framework-0.0.1-SNAPSHOT-jar-with-dependencies.jar" is generated under *target* directory. Execute the jar at command line:

```
java -jar target/com.idtus.contest.winter2017.framework-0.0.1-SNAPSHOT-jar-with-
dependencies.jar -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. Launching the Framework GUI

After executable jar is generated from Gradle, the framework GUI can be launched at command line if using windows:

```
>idt-ui.bat
```

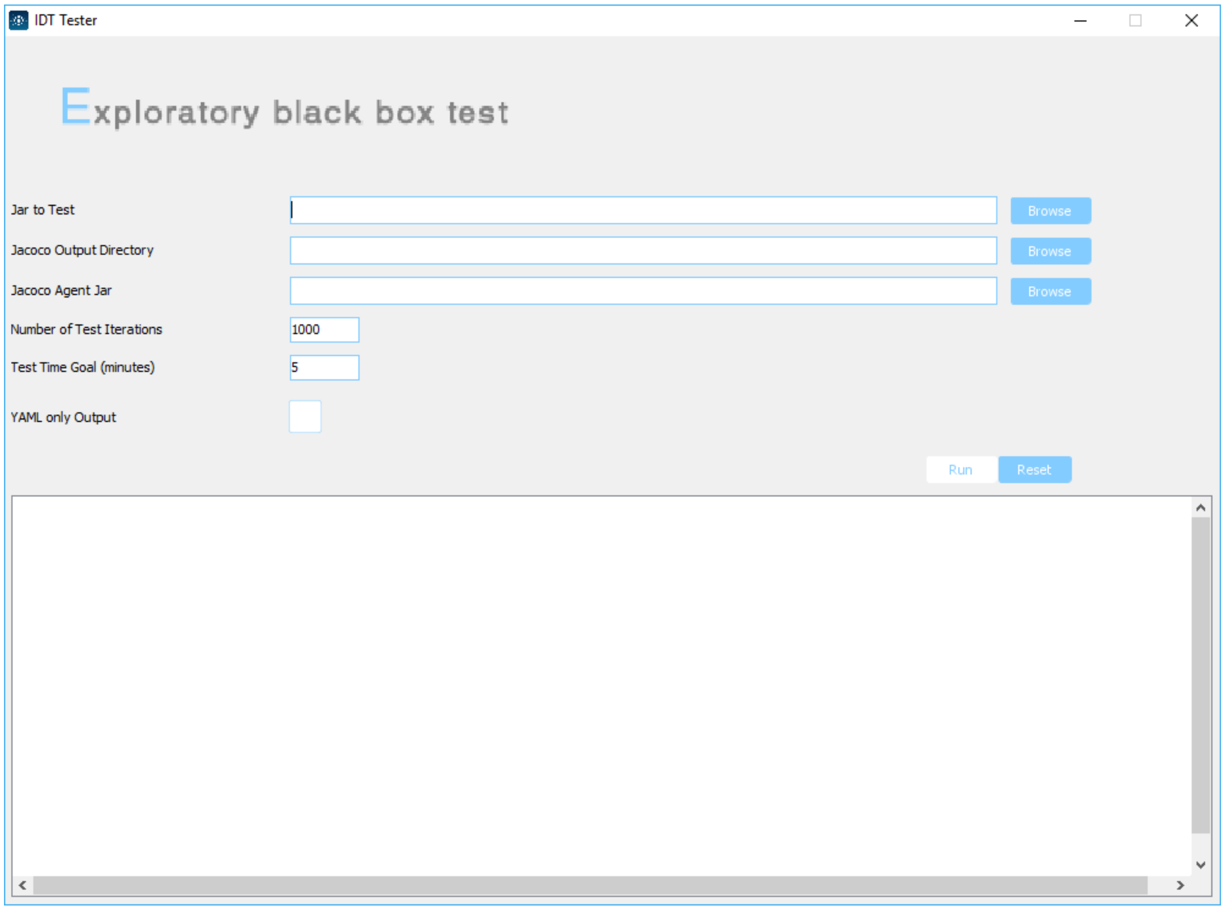
If your machine runs on macOS, Unix, or Linux, GUI can be launched through the bash file.

```
>sh idt-ui.sh
```

If jar is generated from Maven, the framework GUI can be launched at command line with the following line:

```
>java -cp target/com.idtus.contest.winter2017.framework-0.0.1-SNAPSHOT-jar-with-
dependencies.jar contest.winter2017.GuiLauncher
```

IDT Tester window is in display.

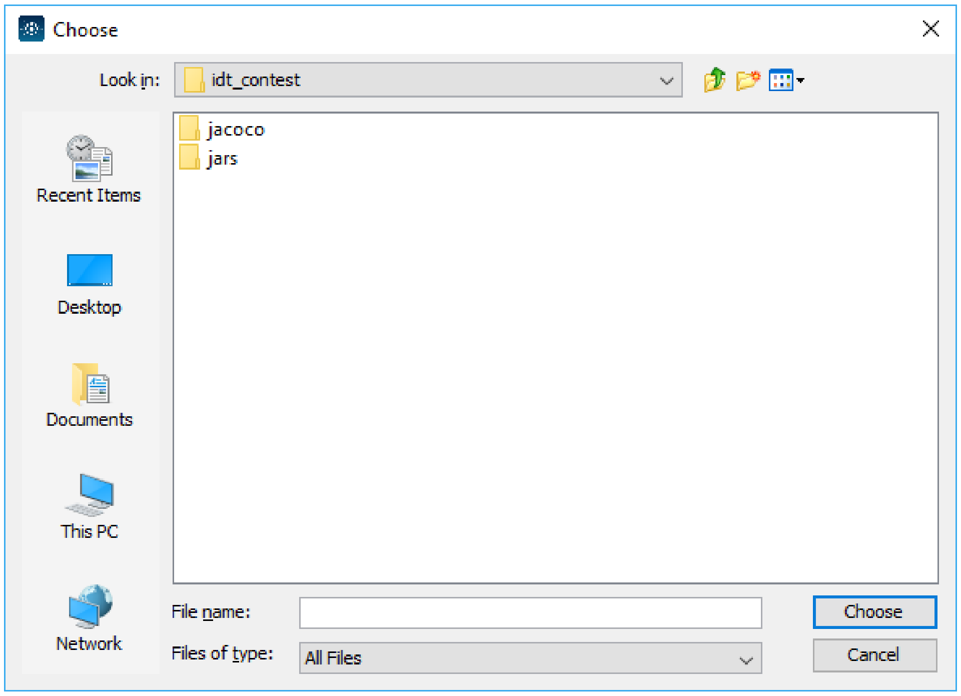

IDT Tester

Exploratory black box test

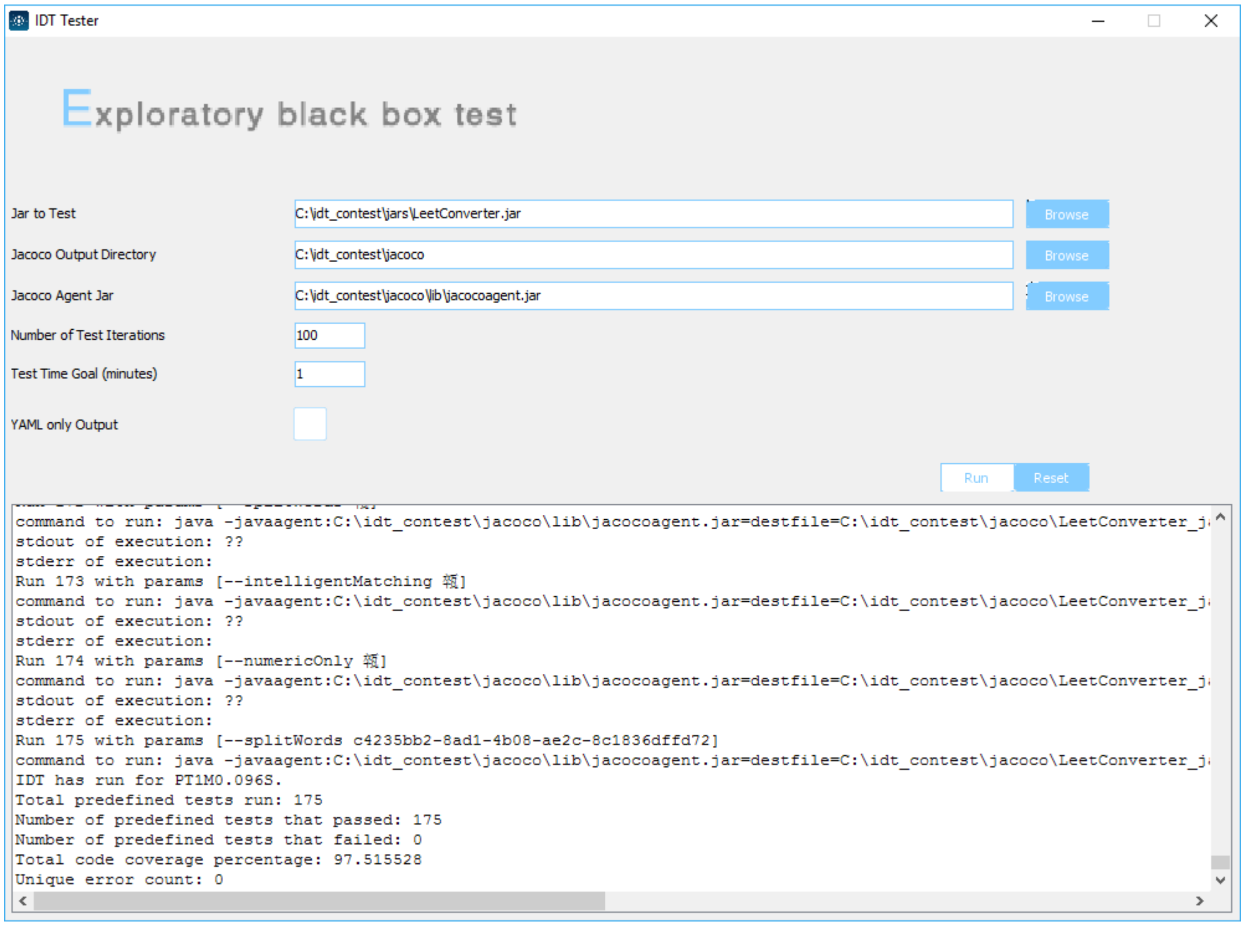
Jar to Test		Browse
Jacoco Output Directory		Browse
Jacoco Agent Jar		Browse
Number of Test Iterations	1000	
Test Time Goal (minutes)	5	
YAML only Output		

IDT Tester GUI Window

The user interface offers same functionality as command line tool. The Browse button opens up dialog window to choose the file or path needed for test. Run button executes the test. The output field displays test result. Reset button cleans up the fields and output.



File Choose Window when Clicking Browse Button



Jar to Test: C:\idt_contest\jars\LeetConverter.jar
Jacoco Output Directory: C:\idt_contest\jacoco
Jacoco Agent Jar: C:\idt_contest\jacoco\lib\jacocoagent.jar
Number of Test Iterations: 100
Test Time Goal (minutes): 1
YAML only Output

```
command to run: java -javaagent:C:\idt_contest\jacoco\lib\jacocoagent.jar=destfile=C:\idt_contest\jacoco\LeetConverter_j
stdout of execution: ??
stderr of execution:
Run 173 with params [--intelligentMatching 䋘]
command to run: java -javaagent:C:\idt_contest\jacoco\lib\jacocoagent.jar=destfile=C:\idt_contest\jacoco\LeetConverter_j
stdout of execution: ??
stderr of execution:
Run 174 with params [--numericOnly 䋘]
command to run: java -javaagent:C:\idt_contest\jacoco\lib\jacocoagent.jar=destfile=C:\idt_contest\jacoco\LeetConverter_j
stdout of execution: ??
stderr of execution:
Run 175 with params [--splitWords c4235bb2-8ad1-4b08-ae2c-8c1836dffd72]
command to run: java -javaagent:C:\idt_contest\jacoco\lib\jacocoagent.jar=destfile=C:\idt_contest\jacoco\LeetConverter_j
IDT has run for PT1M0.096S.
Total predefined tests run: 175
Number of predefined tests that passed: 175
Number of predefined tests that failed: 0
Total code coverage percentage: 97.515528
Unique error count: 0
```

Running a Test in IDT Tester GUI Window

8. Output File

Test results are automatically saved as `idt-yyyy-MM-ddTHH-mm-ss.out` in the source code folder.

A glimpse on a non YAML output file

```
IDT options:
  -jacocoAgentJarPath C:\idt_contest\jacoco\lib\jacocoagent.jar
  -jacocoOutputPath C:\idt_contest\jacoco
  -jarToTestPath C:\idt_contest\jars\LeetConverter.jar
  -bbTests 100
  -timeGoal 60
  -toolChain
Running basic tests...
command to run: java -
javaagent:C:\idt_contest\jacoco\lib\jacocoagent.jar=destfile=C:\idt_contest\jacoc
o\LeetConverter_jar_jacoco.exec -jar C:\idt_contest\jars\LeetConverter.jar "fun"
stdout of execution: |=|_|/\/
stderr of execution:
basic test result: PASS
--------------------------------------------------------------------------------
……………
……………
……………
……………
IDT has run for 00:01:0.096.
Total predefined tests run: 175
Number of predefined tests that passed: 175
Number of predefined tests that failed: 0
Total code coverage percentage: 97.515528
Unique error count: 0
```