# Assignment 1

## Introduction

In this assignment, you will learn the basics of EvoSuite to automatically generate test suites for Java programs. The assignment has the following prerequisites:

* IntellijIDEA or any other IDE
* Maven
* Git

## Part 1

Visit [this repository](https://github.com/satbekmyrza/repo-d4j-se) and download (or clone) the contents. The repository contains 3 packages - each having (b)uggy and (f)ixed versions. The naming convention is as follows <PackageName><BugID><f|b>.

(Optional) These packages were extracted from [Defects4J](https://github.com/rjust/defects4j). You can install Defects4J on your machine and get detailed information about the packages and the bugs that they contain.

These packages are Maven projects:

* Lang: <https://mvnrepository.com/artifact/org.apache.commons/commons-lang3>
* Math: <https://mvnrepository.com/artifact/org.apache.commons/commons-math3>
* Time: <https://mvnrepository.com/artifact/joda-time/joda-time>

Lang7b is a snapshot of the Lang package with a bug (bug id 7 in Defects4J). Lang7f is a snapshot of the Lang package after fixing that bug. Same applies for Math15b, Math15f, and Time8b, Time8f.

## Part 2

### Introduction

This part of the assignment will help you get started. Let’s begin with the Lang package and get some information about the bug. Running this command:

$ defects4j info -p Lang -b 7

Produces the following output:

Summary for Bug: Lang-7

--------------------------------------------------------------------------------

Revision ID (fixed version):

e71f6dd3f2f70c640ae73d28b432b3a69ffcab4b

--------------------------------------------------------------------------------

Revision date (fixed version):

2012-11-11 13:16:22 +0000

--------------------------------------------------------------------------------

Root cause in triggering tests:

- org.apache.commons.lang3.math.NumberUtilsTest::testCreateNumber

--> junit.framework.AssertionFailedError: Expected NumberFormatException

--------------------------------------------------------------------------------

List of modified sources:

- org.apache.commons.lang3.math.NumberUtils

--------------------------------------------------------------------------------

We can see that the bug is in the NumberUtils.java file and was revealed by the testCreateNumber method of NumberUtilsTest class. More about the bug [here](https://issues.apache.org/jira/browse/LANG-822).

Note: Installing Defects4J is optional. You can also obtain the information about the bug [here](https://program-repair.org/defects4j-dissection/#!/). For example, [this link](https://program-repair.org/defects4j-dissection/#!/bug/Lang/7) shows you the information about the Lang-7 bug.

(Optional) You can find NumberUtils.java in Lang7b and Lang7f, and then compare the source codes using <https://www.diffchecker.com/diff>. You will see how Lang7b was modified to get the bug fixed. NumberUtils.java is located at src/main/java/org/apache/commons/lang3/math/ directory.

We are going to follow this scenario:

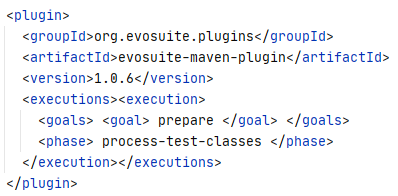
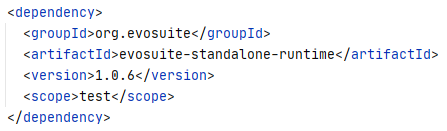
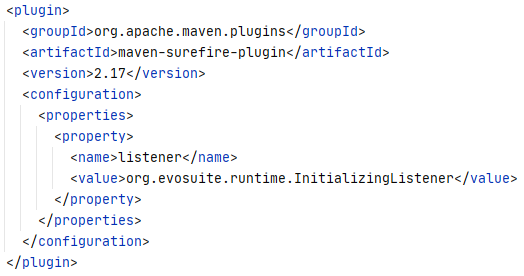
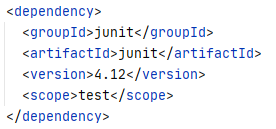
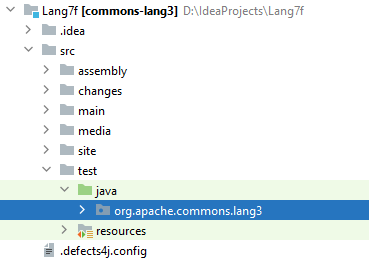
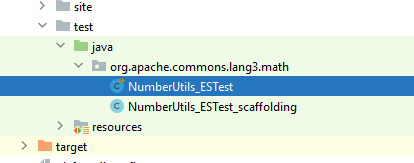
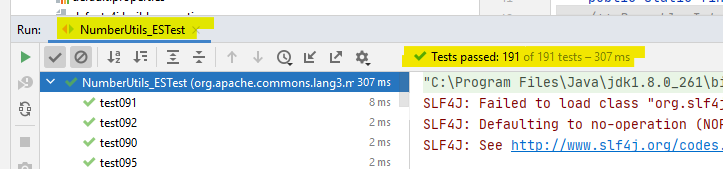
1. Generate tests using EvoSuite for Lang7f package.
2. Substitute a fixed NumberUtils class with a buggy version as if introducing a bug into the code.
3. See if the tests generated by EvoSuite can detect the introduced bug. (Is code coverage of tests generated by EvoSuite good enough to find a bug?)

Environment:

* Apache Maven 3.6.3
* Oracle JDK 1.8.0\_261

### Test Generation

Let’s generate tests for Lang7f using EvoSuite:

1. Open the project using any IDE.
2. Maven projects come with a configuration file named pom.xml. In order to run EvoSuite on Maven projects, complete the steps described here <https://www.evosuite.org/documentation/maven-plugin/>:
   1. Add EvoSuite plugin:  
      
   2. Add EvoSuite runtime dependency:  
      
   3. If pom.xml already contains maven-surefire-plugin, substitute it with this. Otherwise, add it:  
      
   4. Change the junit version to 4.12:  
      
3. Remove all tests from the project (the contents of test/java/), we will generate new tests using EvoSuite instead.   
   
4. From the project’s root directory, run the following commands:
   1. $ mvn clean
   2. $ mvn compile
   3. $ mvn "-DmemoryInMB=8000" "-Dcores=4" evosuite:generate  
      Tune the parameters based on your needs. Note that double quotes are not needed for Linux/Mac.
5. Running the last command starts EvoSuite. It will now generate tests for all classes in the package. With the parameters specified above, it took approximately 45 minutes to generate tests.
6. By default, EvoSuite places the generated tests into .evosuite\ directory. Move the generated tests for NumberUtils class from .evosuite\best-tests\ to test\java\ (default test location for Maven projects):  
   
7. Run the tests using   
   $ mvn test  
   

Now you can substitute fixed NumberUtils with buggy NumberUtils to see if all the tests pass. In our case, EvoSuite found the bug (Since Evosuite randomly generates tests, you may not obtain exactly the same test):



The meaning of the bug is as follows:  
createBigDecimal method of NumberUtils class should have thrown a NumberFormatException, because the argument string starts with “--”. If you check the fixed version of NumberUtils, you will see that this bug is fixed with an if check moved from createNumber to createBigDecimal. The fixed version throws an exception for the input starting with “--” inside the createBigDecimal method.

## Part 3

In this part of the assignment, you will run EvoSuite on the remaining packages on your own.

**Math15f and Math15b**

Math15b contains a bug in org.apache.commons.math3.util.FastMath. The bug was revealed by org.apache.commons.math3.util.FastMathTest::testMath904 and the error was junit.framework.AssertionFailedError: expected:<-1.0> but was:<1.0>.

**Time8f and Time8b**

Time8b contains a bug in org.joda.time.DateTimeZone. The bug was revealed by org.joda.time.TestDateTimeZone::testForOffsetHoursMinutes\_int\_int and the error was java.lang.IllegalArgumentException: Minutes out of range: -15.

## FAQ

This part of the assignment contains some useful links that might be helpful and will be updated without notice with answers to repeating questions.

[Correct way to add external jars (lib/\*.jar) to an IntelliJ IDEA project](https://stackoverflow.com/questions/1051640/correct-way-to-add-external-jars-lib-jar-to-an-intellij-idea-project)

### How to generate tests for specific classes?

By default, EvoSuite generates tests for all classes. Obviously, we do not want to do that, because for our scenario it is enough to generate tests only for classes that contain bugs. EvoSuite has an option for the purpose of generating tests for specific classes. Here is an example command to generate tests for NumberUtils class:

mvn "-Dcuts=org.apache.commons.lang3.math.NumberUtils" evosuite:generate

Where “cuts” stands for Class Under Test. You can specify more classes using a comma-separated list.

### How to increase time for test generation?

By default, EvoSuite will run for two minutes per class. You can change this by using an option timeInMinutesPerClass. For example, the command below will run for up to 60 minutes per class:

mvn "-DtimeInMinutesPerClass=60" evosuite:generate

### How to get code coverage information?

Code coverage is a useful metric to assess the quality of a test suite. In order to collect code coverage information for the subjects you are working on, you can use Intellij IDEA. It has a built-in code coverage tool and is easy to use. Follow these steps:

1. Open the project in Intellij IDEA.
2. Right click on the project name or click on the “Run” menu and then select “Run ‘All tests’ with Coverage”.
3. You should see coverage information shown to the right of each class. If you open the class in the editor, you will see that some lines are denoted with green and some are denoted with red. Green means that the line was covered, and red means that the line was not covered.

You can find more information on code coverage using Intellij IDEA [here](https://www.jetbrains.com/help/idea/code-coverage.html).

## Submission

You should submit a single zip file containing the following files:

1. A generated test file for each of the three subjects (Lang 7, Math 15, and Time 8). Use the following directory structure:

Subjects

Lang\_7

Math\_15

Time\_8

For example, for Lang 7, you would have the following two files.

* Subjects/Lang\_7/org/apache/commons/lang3/math/NumberUtils\_ESTest
* Subjects/Lang\_7/org/apache/commons/lang3/math/NumberUtils\_ESTest\_scaffolding

1. You should submit a *single* report that contains the following
2. A table that shows the branch coverage of the target class (the class that contains the bug). You can get this information from evosuite-report/statistics.csv. (20 points)
3. A copy of a generated test that reveals the bug. In case generated tests do not reveal the bug (for Math 15, Evosuite is more likely to fail to generate a bug-revealing test), describe how you modify one of the generated tests so that the bug can be revealed by the modified test. (20 points)
4. One paragraph that explains the bug. (20 points)
5. One paragraph that explains the bug-revealing test. Explain how the generated test (or the modified test) reveals the bug. (20 points)
6. Your thoughts on generated tests. Describe the aspects you find useful or interesting. Also describe the aspects you find problematic. In your description, use generated tests (and also modified tests if you want) as concrete examples. In particular, describe why you think Evosuite does not always generate a bug-revealing test. In your description, use the test cases you obtained as an example. (20 points)

The first 4 items should be described for each of the three programs you work on. The last one should be described in a separate section where you describe your overall thoughts.

## How to submit

Submit your file via BlackBoard. The submission should be made by April 11th, 11:59 pm.