Assignment #2 Software Testing

Com S/SE 417 Spring 2024

Handed out Feb 13th, 2024

Due at 9 PM, Feb 22rd as a pdf uploaded to Canvas

Homework Policy

Homework Policy: The homework assignment should be done individually. You may talk to classmates about the problems in general, and you can help each other with getting the tools running, but you must complete the homework on your own. You are not permitted to use published answers from websites, etc. Assistance by others must be specifically credited in the solution to the problem that is turned in, describing what the contribution was (e.g., "Thanks to [name] for explaining X in Problem Y", not "Thanks to [name] for help with HW1."). The Dean of Students Office offers several good resources on how to avoid plagiarism.

The goal is for each of you to learn the material sufficiently well to use it productively, to think innovatively, and to develop confidence in your problem-solving abilities. Feel free to talk to me individually about this if you have any questions.

Late policy: *10% penalty per day* (or part of day) for late homework. ***Assignments will not be accepted after Feb 26th***th, unless otherwise arranged/discussed with me.

Late Policy One Slack Period for all Homework
During the semester you have one "slack period". This means that you can hand in one homework late – up until the late acceptance date –(e.g. Feb 14th for this assignment) without penalty. No questions asked. Once you use up your slack period late penalties will apply as per the late policy

Some homework problems are adapted from the course textbook, "Introduction to Software Testing", 2nd edition, Ammann & Offutt, 2017. There is a Student Solution Manuals available online with answers to other practice questions https://cs.gmu.edu/~offutt/softwaretest/exer-student.pdf.

Using Evosuite 1.2.0

https://www.evosuite.org/

I have put a version of Evosuite that can be untarred and used for this exercise. It runs on pyrite. It has the minimal parts of Evosuite (the .jar files and two example programs we will use for this assignment). You are welcome to get the full distribution of Evosuite. It has a maven version and there are two docker versions you can use.

You can run these locally on other machines (if you want to set it up). These are directions for running on pyrite.

For each numbered item answer the questions, and provide screen shots where asked. Create a .pdf of the final report and submit via canvas. Make sure to clearly label the question #s. I have put ** ** around the text and bolded parts that need to be handed in.

There is nothing to submit for #1.

1. **Download the evosuite program** from Canvas and unzip/untar this (pyrite is a good place for this).

I have given you two programs that you can use for generating test suites

One is a stack class (Tutorial_Stack/Stack.java) and the other is a variant of our triangle program:

(Triangle/TriangleType.java and Triangle/Triangle.java)

Read the evosuite tutorial to get a sense of what you will be doing (evosuite.org).

2. Generate test cases for both programs using the following steps:

Note that we need to use Java 8 or Java 11 for this exercise. Pyrite has an alias for Java11 (java11 and javac11) hence we will use that. There will be a few warnings using this version of Java, but the tool will work correctly.

First compile both programs (*if you are working on your home machine you will use just 'java' and 'javac' if you have set Java to be the correct version – or you may need to use the full path to the correct java*).

> javac11 Tutorial_Stack/tutorial/*.java

> javac11 Triangle/triangle/*.java

3. **Now run evosuite on each program** (note you use the -projectCP flag to tell it which directory

and the -class to tell it which class you want to generate tests for)

java11 -jar evosuite-1.2.0.jar -projectCP Tutorial_Stack -class tutorial.Stack

java11 -jar evosuite-1.2.0.jar -projectCP Triangle -class triangle.TriangleType

Take a look at the tests (under evosuite-tests) and the reports (under evosuite-reports)

Handin

** (a) Copy the contents of the statistics.csv** file to this report

Open the test files and answer the following questions:

Handin

**(b) How many tests are there for each program? **

(c) Show an example of a test for EQUILATERAL

(d) Show an example of a stack test **that maximizes the stack**

4. **Now export your class path**. You need to include the evosuite-tests, the testing jars and the paths to the programs. (note – you will need to type this in on a single line. All of the libraries are in the "lib: folder if you want to see their names)

export CLASSPATH=.:evosuite-tests:lib/junit-4.13.jar:lib/hamcrest-core-1.3.jar:lib:Triangle:Tutorial_Stack:evosuite-standalone-runtime-1.2.0.jar

Compile the tests

javac11 evosuite-tests/triangle/*.java

javac11 evosuite-tests/tutorial/*.java

(If you have not set your classpath correctly this will give you an error)

And run them:

java11 org.junit.runner.JUnitCore tutorial.Stack_ESTest

java11 org.junit.runner.JUnitCore triangle.TriangleType_ESTest

Answer the following question.

** (a) **Save the TriangleType_EST.java** file to another location so you can copy it back to the tests directory alter. Now modify the TriangleType.java program by adding the fault into the isosceles branch (there is a comment that shows you how to do this). Repeat the above steps and look at the new TriangleType_EST.java tests and note the results of running these (i.e. do you see any faults).

** (b) Was the fault found by the generated tests? Find the isosceles test that covers the faulty branch and provide a screenshot. State why this test passes.

**(c) Now save the TriangleType_EST.java file again (in case you need it) and copy the first one back into the evosuite_tests/triangle directory. Compile the tests and run them. You should see a failure now. Take a screenshot of the result of running the original tests on the faulty program.

5. **Now copy the "PrimeNumberFinder.java"** which is a mostly corrected version from our last assignment. It needs to be in package hierarchy. Create a folder called "Prime" inside of the main evosuite directory and then create a directory called 'prime' inside of that. Put this java file in the prime directory.

Use evosuite to **generate tests for PrimeNumberFinder.java**

Add Prime to the classpath and generate test cases and run them using evosuite.

Handin

** (a) Take a screenshot of the 'statistics.csv' file

** (b) Provide a screenshot(s) of the test file

** (c) Provide a screenshot of the test results

** (d) Look through the test cases that were generated and see if there are any that 'surprise' you. Find **at least one test case you would not have created yourself during the first homework** and discuss (note – I am not looking for an odd input value, but what the test is actually testing –e.g. its intent/oracle).

6. Now **manually copy the test inputs from evosuite** into a Junit format that can be used by the jacoco tool. To do this, create a new **PrimeNumberFinderTest.java** file inside of the jaccoco test directory (you can save the olde one) and populate the tests using the inputs and oracles from the evosuite tests. Run the jacoco tool and get a code coverage report.

Handin

** (a) compare the code coverage report with the one you got for assignment one (the coverage after you fixed the faults and after you added new coverage). **Include a screen shot** of the coverage report for the evosuite tests and discuss the difference from your original, manually created coverage (if any).

(b) the exception in computeSumOfPrimes was not fixed in this version of the program. Is it found when you run **mvn test? If not, see if there is a test case for this and discuss why it is not being exposed.