CS 731 SOFTWARE TESTING

PROJECT REPORT

MUTATION TESTING ON SOURCE CODE OF VARIOUS MATHEMATICAL CONVERTERS

SUBMITTED BY:

MT2022036 DEEPAK

MT2022145 KUSHAGRA AGRAWAL

Project Aim:

To apply concepts of mutation testing on a real world software project with the help of open source tools.

Code tested:

Conversions between units for different types is needed everywhere and, in any field, to achieve proper representation of data. For example, S.I. unit system we studied in 11^{th} standard is necessary to compute various equations in physics. Be it conversion between mass, or be in temperature, units are needed. Given the importance, it is necessary to make sure conversion system is precise to the point and produce precise output. Hence, we choose to perform mutation testing on open-source program and extracted various converters in a separate project.

Resources used: https://github.com/mono832/Calculator-FX

TESTING STRATEGY AND TOOLS USED

Mutation testing is a form of testing where small modifications are made to the source code (and each modification is called a mutant). The aim of mutation testing is to "kill" these mutants – that is show that making small changes to the code can alter the execution of the program and hence the result that it produces. There are 2 ways to "kill" a mutant:

1. Kill a mutant **weakly**: Here, the memory state of the program after the execution of the mutated statement is different from the memory state of the program when the statement was not mutated and executed. Notice that in this case, the output of the program on a test case can remain the same, irrespective of whether a program statement was mutated or not.
2. Kill a mutant **strongly**: Here, the output of the program on a test case, when a statement was mutated and not mutated, must change. Notice that when we kill a mutant strongly, the error propagates through the program and we notice this by seeing different outputs in the presence and absence of the mutant.

We chose **mutation testing** as our testing strategy, with the aim to kill the mutants **strongly**.

The tools that we used for mutation testing are as follows:

1. Eclipse IDE for Java: A fantastic Java IDE, which allows adding plugins to add extra features to the IDE.
2. Junit5: A basic tool for testing in java. It doesn't generate reports, but works faster.
3. PIT Mutation Testing Tool: An easy-to-use mutation testing tool, that works for Java. We used the PITclipse plugin for Eclipse to integrate this tool into the Eclipse IDE. It's slower, but generates report as required.

Mutation Operators

1. Arithmetic Operator Replacement (AOR)

This mutator replaces binary arithmetic operations with another operation for either integer or floating-point arithmetic. Each occurrence of the following operators like +, -, *, **, /, and % is replaced by one of the other operators listed above.

The table below shows how an operator can be replaced:

Operator	Replace +	Replace -	Replace *	Replace %	Replace /
+	✗	✔	✔	✔	✔
-	✔	✗	✔	✔	✔
*	✔	✔	✗	✔	✔
%	✔	✔	✔	✗	✔
/	✔	✔	✔	✔	✗

2. Relational Operator Replacement (ROR)

This mutator replaces relational arithmetic operations with another relational arithmetic. Each occurrence of the following operators like ==, !=, <, <=, >, >= is replaced by one of the other operators listed above.

The below table shows how a relational operator can be replaced:

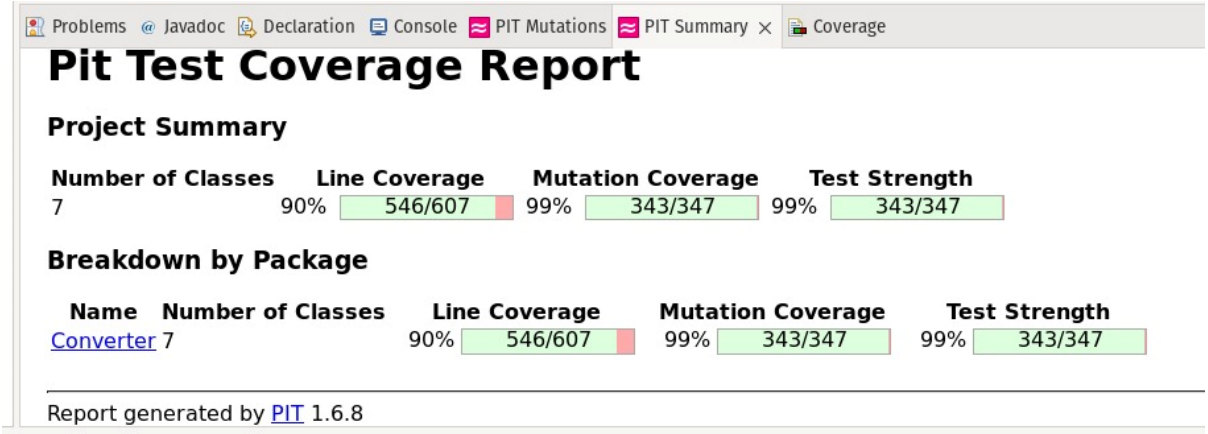
Op	Replace ==	Replace !=	Replace <	Replace <=	Replace >	Replace >=
==	✗	✔	✔	✔	✔	✔
!=	✔	✗	✔	✔	✔	✔
<	✔	✔	✗	✔	✔	✔
<=	✔	✔	✔	✗	✔	✔
>	✔	✔	✔	✔	✗	✔
>=	✔	✔	✔	✔	✔	✗

3. Unary Operator Insertion (UOI)

This mutator replaces a variable call with a unary operator (increment or decrement). Local variables, array variables, fields, and parameters are all affected. It is made up of the following
sub-mutators (UOI1 to UOI4) that insert operators in the order listed below.

Variable	UOI1	UOI2	UOI3	UOI4
x	x++	x-	++x	-x

Results:



Category	Count
Mutants Survived	4
Mutants Killed	343
Mutants with No coverage	0

How to execute the code?

1. Import the “Converter” folder as a project in eclipse.
2. Install PITest. Google for instructions.
3. Right click on any test and “Run as” -> “PIT mutation test”.

After clicking on run, the console starts displaying all the running status and after all the running is completed.
A report and summary will be generated in other tabs.