

Download and install the Java mutation tool: muJava. Use muJava to test cal().

Use all the operators. Design tests to kill all non-equivalent mutants. Note that a test case is a method call to cal().

(a) How many mutants are there?

The screenshot shows the muJava TestCase Runner interface. The 'Traditional Mutants Viewer' tab is active. The 'Class' is set to 'Cal', the 'Method' is 'int_cal(int,int,int,int,int)', and the 'TestCase' is 'CalTest'. The 'Time-Out' is set to '3 seconds'. The 'Execute all mutants' radio button is selected. The 'RUN' button is highlighted in yellow.

On the left, there are two tables of operators and their counts:

Op	#
AORB	36
AORS	1
AOIU	12
AOIS	74
AODU	0
AODS	0
ROR	34
COR	4
COD	0
COI	7
SOR	0
LOR	0
LOI	19
LOD	0
ASRS	0
SDL	10
VDL	13
CDL	5
ODL	22

Total : 237

Op	#
IHI	0
IHD	0
IOD	0
IOP	0
IOR	0
ISI	0
ISD	0
IPC	0
PNC	0
PMD	0
PPD	0
PCI	0
PCC	0
PCD	0
PRV	0
OMR	0
OMD	0
OAN	0
JTI	0
JTD	0
JSI	0
JSD	0
JID	0
JDC	0
EOA	0
EOC	0
EAM	0
EMM	0

Total : 0

The 'Traditional Mutants Result' section shows:

Traditional Mutants Result	
Live Mutants #	30
Killed Mutants #	207
Total Mutants #	237
Mutant Score	87.0%

The 'Class Mutants Result' section shows:

Class Mutants Result	
Live Mutants #	0
Killed Mutants #	0
Total Mutants #	0
Mutant Score	- %

The 'Live' and 'Killed' lists for Traditional Mutants are:

Live	Killed
ROR_4	ROR_2
AOIS_55	AOIU_1
AOIS_56	AOIS_61
AOIS_72	ROR_11
AOIS_44	ROR_15
AOIS_43	VDL_5
AOIU_6	AOIS_8
ROR_22	LOI_11
AOIS_28	LOI_2
AOIS_52	AOIS_65
AOIS_73	AOIS_37
AOIS_12	ROR_14
AOIS_31	ODL_6
AOIS_15	AORB_1
ROR_18	AOIU_10
AOIS_36	AOIU_12
ROR_8	LOI_7
AOIS_32	ODL_19
AOIS_74	VDL_13

237 mutants in total

(b) How many test cases do you need to kill the non-equivalent mutants?

I design **8** test cases to kill the non-equivalent mutants:

1. `Assert.assertEquals(151, Cal.cal(5, 23, 10, 21, 2022));`
2. `Assert.assertEquals(121, Cal.cal(1, 1, 5, 1, 2020));`
3. `Assert.assertEquals(181, Cal.cal(1, 1, 7, 1, 2100));`
4. `Assert.assertEquals(182, Cal.cal(1, 1, 7, 1, 2000));`
5. `Assert.assertEquals(14, Cal.cal(1, 7, 1, 21, 2100));`
6. `Assert.assertEquals(59, Cal.cal(1, 1, 3, 1, 2022));`
7. `Assert.assertEquals(-6, Cal.cal(8, 7, 8, 1, 2022));`
8. `Assert.assertEquals(0, Cal.cal(7, 7, 7, 7, 2022));`

(c) What mutation score were you able to achieve before analyzing for equivalent mutants?

I achieved **85%** mutant killed before I started to analysis the remaining mutant one by one

And achieved **87%** before submit my homework

(d) How many equivalent mutants are there?

In my test, I found **30** equivalent mutants. Since I didn't check all the 30 remaining mutants, I can't sure there don't exist non-equivalent mutants anymore.

Most of equivalent mutants are in following type:

1. make postfix add (a++) on variables that won't be used in the remaining program (e.g. AOIS_51)
2. change logic condition mark that won't influence the outcome (e.g. COR_4)
3. switch the sign of a variable that is used for modular 100 (e.g. AOIU_5)
4. cases that are already covered by program (e.g. ROR_4)

Since these faults may produce an error state, they won't produce a failure outcome.

Github: