MuJava (µJava)

A mutation system for Java programs

Useful links

- MuJava Homepage
http://cs.gmu.edu/~offutt/mujava/

- MuClipse: Eclipse plugin for MuJava
http://muclipse.sourceforge.net/

- Mutation Testing Online
http://www.mutationtest.net/
http://cs.gmu.edu/~offutt/rsrch/mut.html

Preliminary concepts

- **Mutation analysis** is the process of measuring how good a test set is (how many mutations it kills).
- **Mutation testing** is the process of generating tests to improve the mutation analysis score - so mutation testing is a broader topic.

Preliminary concepts

- **Mutation operator** is a rule that is applied to a program to create mutants. Typical mutation operators, for example, replace each operand by every other syntactically legal operand, or modify expressions by replacing operators and inserting new operators, or delete entire statements.

Traditional mutants: method-level

- AORB - Arithmetic Operator Replacement (binary) + - * / %
- AORU - Arithmetic Operator Replacement (unary) + -
- AORS - Arithmetic Operator Replacement (short-cut) ++ --
- AOIU - Arithmetic Operator Insertion (unary)
- AOIS - Arithmetic Operator Insertion (short-cut)
- AODU - Arithmetic Operator Deletion (unary)
- AODS - Arithmetic Operator Deletion (short-cut)

Traditional mutants: method-level

- ROR -Relational Operator Replacement > >= < <= == !=
- COR -Conditional Operator Replacement && || & | ^
- COI -Conditional Operator Insertion
- COD -Conditional Operator Deletion
- SOR -Shift Operator Replacement >> << >>>
- LOR -Logical Operator Replacement
- LOI -Logical Operator Insertion
- LOD -Logical Operator Deletion
- ASRS -Assignment Operator Replacement (short-cut) += -= *= /= %= &= >>= |=

Class level mutants

- AMC -Access modifier change
- IHD -Hiding variable deletion
- IHI -Hiding variable insertion
- IOD -Overriding method deletion
- IOP -Overriding method calling position change
- IOR -Overriding method rename
- ISI -Super keyword insertion
- ISD -Super keyword deletion
- IPC -Explicit call to parent’s constructor deletion

Class level mutants

- PNC -New method call with child class type
- PMD -Member variable declaration with parent class type
- PPD-Member variable declaration with child class type
- PCI -Type cast operator insertion
- PCC -Cast type change
- PCD -Type cast operator deletion
- PRV -Reference assignment with other compatible variable
- OMR -Overloading method contents replace
- OMD -Overloading method deletion
- OAN -Arguments of overloading method call change

Class level mutants

- JTI -*this* keyword insertion
- JTD -*this* keyword deletion
- JSI -*static* modifier insertion
- JSD -*static* modifier deletion
- JID -Member variable initialisation deletion
- JDC -Java-supported default constructor creation
- EOA -Reference assignment & content assignment replacement
- EOC -Reference comparison & content assignment replacement
- EAM -Accessor method change
- EMM -Modifier method change

Using MuJava to test classes

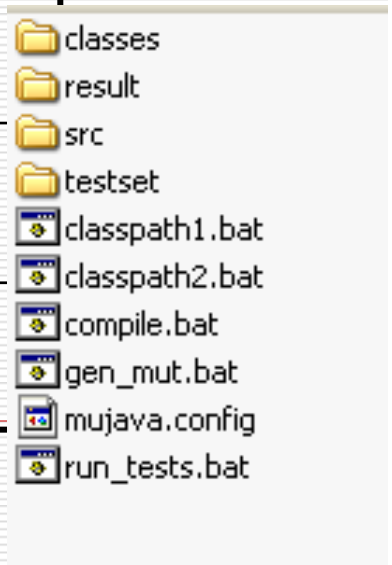
- Update the **PATH** Variable, to be able to conveniently run the executables (javac.exe, java.exe) without typing the full path of the command:

- My Computer → Properties → Advanced → Environment Variables → System Variables → **Path** → Edit → add for example "C:\Program Files\Java\jdk1.6.0_07\bin;"

Directory structure for MuJava

In the **mujava.config** file is stored the:
MuJava_HOME=D:\mujava_prog_master

MuJava_HOME\src	directory for Java files to be tested
MuJava_HOME\classes	directory for compiled classes of Java files from *MuJava_HOME*\src
*MuJava_HOME**testset*	directory for test sets
MuJava_HOME\result	directory for generated mutants



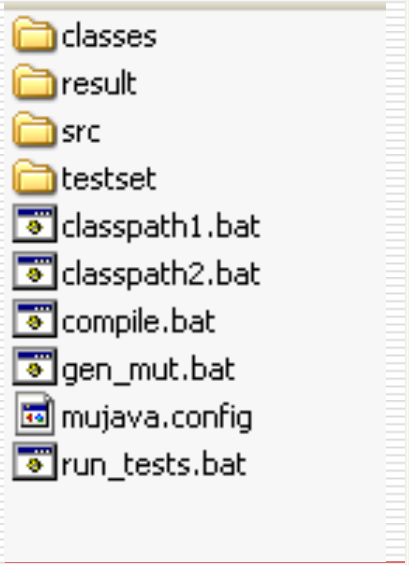
Using MuJava to test classes

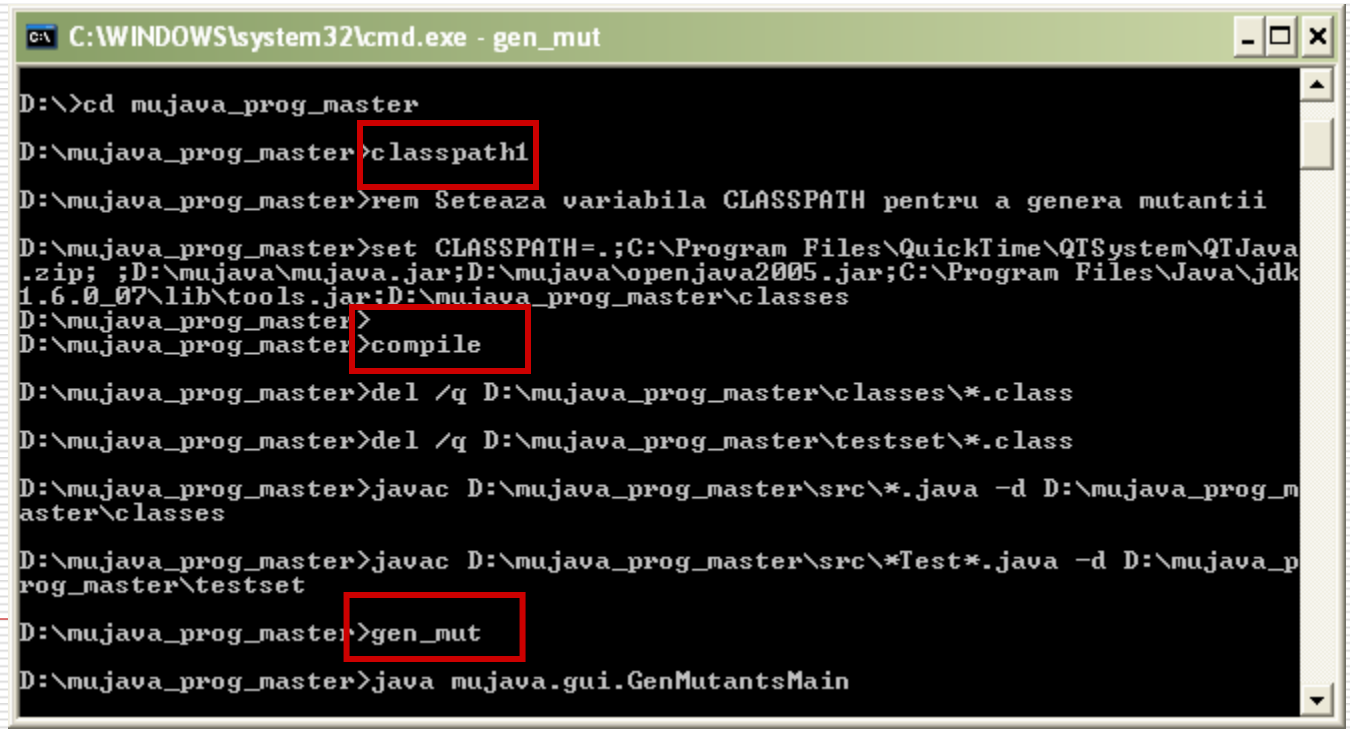
- Follow the instruction from http://cs.gmu.edu/~offutt/mujava/#Install

- Or modify the **mujava.config** file and batch files provided in the class:
  - **classpath1** – classpath for compiling and generating mutants
  - **classpath2** – classpath for running the test sets
  - **compile** – to compile and store adequately the resulted .class files (in classes or testset folders)
  - **gen_mut** – to generate the mutants
  - **run_tests** – to run the test sets

Generating mutants

1. Set the classpath, to include paths to: mujava.jar, openjava2005.jar, tools.jar, *MuJava_HOME*\classes)
2. Compile the existing Java classes, from src folder and
3. Call the mutants generator





Mutants generator

Mutants Generator | Traditional Mutants Viewer | Class Mutants Viewer

Usage :
[1] Select files to test
[2] Select mutation operators to apply
[3] Push "RUN" button
[4] Wait with endurance. ^^;

	File
☐	Book.java
☐	Book2.java
☐	Book2_Test_AT_Minim.java
☐	Book2_Test_AT_Oracol_Abstract.java
☐	Book2_Test_AT_Oracol_Precis.java
☐	Book_Test_AT_Minim.java
☐	Book_Test_AT_Oracol_Abstract.java
☐	Book_Test_AT_Oracol_Precis.java
☐	Complex.java
☐	Complex_Test.java
☑	Hello.java
☐	HelloTest.java
☐	MyClass.java
☑	Stack.java
☐	StackTest.java

Log level 1

None | All | Generate

Java Mutation Operator

Method-level

	Operator
☑	AORB
☑	AORS
☑	AOIU
☑	AOIS
☑	AODU
☑	AODS
☑	ROR
☑	COR
☑	COD
☑	COI
☑	SOR
☑	LOR
☑	LOI
☑	LOD
☑	ASRS

None | All

Class-level

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

None | All

Generating mutants

Note that the log messages go to the command window, not the GUI.

C:\WINDOWS\system32\cmd.exe - gen_mut

```
D:\mujava_prog_master>gen_mut

D:\mujava_prog_master>java mujava.gui.GenMutantsMain
0 : Hello.java
 -- Hello.java class contains 'static void main()' method.
    Pleas note that mutants are not generated for the 'static void main()' metho
d
----------------------------------------------------------------------
 All files are handled
0 : Book.java
----------------------------------------------------------------------
 All files are handled
0 : Stack.java
 -- Stack.java class contains 'static void main()' method.
    Pleas note that mutants are not generated for the 'static void main()' metho
d
D:\mujava_prog_master\result\Stack\traditional_mutants\void_add(int)\AODS_1\Stac
k.java:22: not a statement
            poz;
            ^
1 error
D:\mujava_prog_master\result\Stack\traditional_mutants\void_add(int)\AOIS_10\Sta
ck.java:20: cannot assign a value to final variable length
        if (poz < --v.length) {
                  ^
1 error
D:\mujava_prog_master\result\Stack\traditional_mutants\void_add(int)\AOIS_11\Sta
ck.java:20: cannot assign a value to final variable length
        if (poz < v.length++) {
                  ^
1 error
D:\mujava_prog_master\result\Stack\traditional_mutants\void_add(int)\AOIS_12\Sta
ck.java:20: cannot assign a value to final variable length
        if (poz < v.length--) {
                  ^
1 error
D:\mujava_prog_master\result\Stack\traditional_mutants\void_add(int)\AOIS_9\Stac
k.java:20: cannot assign a value to final variable length
        if (poz < ++v.length) {
                  ^
1 error
D:\mujava_prog_master\result\Stack\traditional_mutants\void_add(int)\LOI_6\Stack
.java:22: not a statement
            ~poz++;
            ^
1 error
D:\mujava_prog_master\result\Stack\traditional_mutants\int_remove()\AODS_2\Stack
.java:32: not a statement
            poz;
            ^
1 error
D:\mujava_prog_master\result\Stack\traditional_mutants\int_remove()\LOI_8\Stack.
java:32: not a statement
            ~poz--;
            ^
1 error
----------------------------------------------------------------------
 All files are handled
```

Traditional mutants viewer

Mutants Generator | Traditional Mutants Viewer | Class Mutants Viewer

Select a class : Hello

Select a method : int_min(int,int)

* Summary *

Op	#
AO...	0
AO...	0
AOIU	2
AOIS	12
AO...	0
AO...	0
ROR	0
COR	0
COD	0
COI	1
SOR	0
LOR	0
LOI	4
LOD	0
ASRS	0

Total : 19

AOIS_1
AOIS_10
AOIS_11
AOIS_12
AOIS_2
AOIS_3
AOIS_4
AOIS_5
AOIS_6
AOIS_7
AOIS_8
AOIS_9
AOIU_1
AOIU_2
COI_1
LOI_1
LOI_2
LOI_3
LOI_4

(line 9) a => ~a

Original

```
1  // This is mutant program.
2  // Author : ysma
3
4  class Hello
5  {
6
7     public static int min( int a, int b )
8     {
9        if (a < b) {
10          return a;
11       } else {
12          return b;
13       }
14    }
15
```

Mutant

```
1  // This is mutant program.
2  // Author : ysma
3
4  class Hello
5  {
6
7     public static int min( int a, int b )
8     {
9        if (~a < b) {
10          return a;
11       } else {
12          return b;
13       }
14    }
15
```

Class mutants viewer

Mutants Generator | Traditional Mutants Viewer | Class Mutants Viewer

Select a class : Complex

* Summary *

Op	#
IHI	0
IHD	0
IOD	2
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	4
JTI	0
JTD	0
JSI	2
JSD	0
JID	0
JDC	0
EOA	0
EOC	0
EAM	0
EMM	0

Total : 8

IOD_1
IOD_2
JSI_1
JSI_2
OAN_1
OAN_2
OAN_3
OAN_4

(line 7) static is inserted

Original

```
7   private int r;
8
9   private int i;
10
11   Complex( int rr, int ii )
12   {
13      r = rr;
14      i = ii;
15   }
16
17   public java.lang.String toString()
18   {
19      java.lang.StringBuffer sb = (new java.lang.StringBuffer()).append( r );
20      if (i >= 0) {
21         sb.append( '+' );
```

Mutant

```
7   private static int r;
8
9   private int i;
10
11   Complex( int rr, int ii )
12   {
13      r = rr;
14      i = ii;
15   }
16
17   public java.lang.String toString()
18   {
19      java.lang.StringBuffer sb = (new java.lang.StringBuffer()).append( r );
20      if (i >= 0) {
21         sb.append( '+' );
```

Creating a test set

- A **testset** in MuJava is a Java file that contains executable test scripts.
- Each test is a method that contains a sequence of calls to methods in the class under test.
- Each test method returns a **String** result that is used to compare outputs of mutants with outputs of the original class.
- Each test method should start with the string "**test**". The test methods and the test class should have public access.

Test set example

Java Class

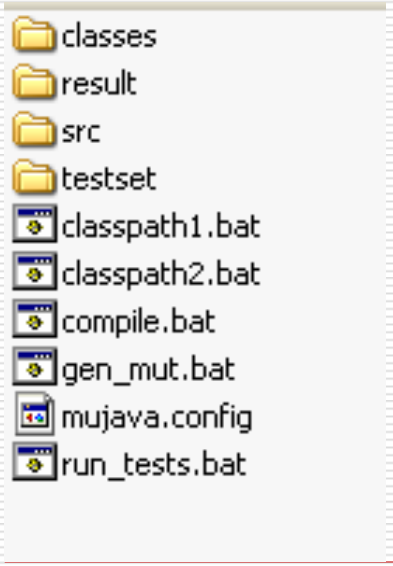
```java
/* @author: Raluca */
class Hello{

    public static int min(int a, int b)
    {
        if (a<b)
            return a;
        else
            return b;
    }
    public static void main (String args[]){
        System.out.println("Hello world!");

    }
}
```

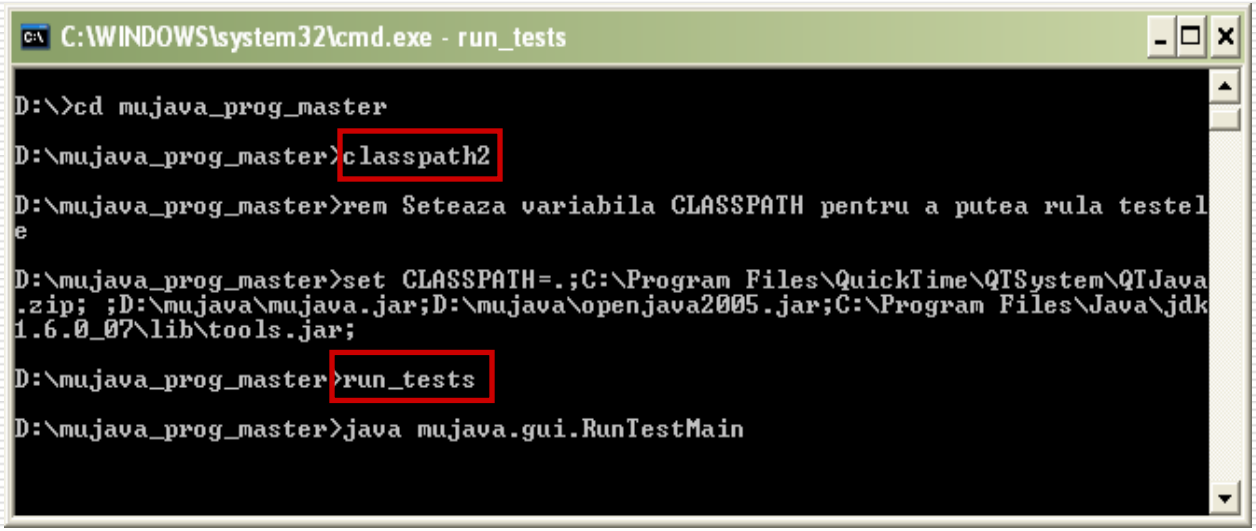
Test set

```java
/* @author: Raluca */
public class HelloTest{
    public String test1(){
        String result = "";
        result = result + Hello.min(1, 2);
        return result;
    }
    public String test2(){
        String result = "";
        result = result + Hello.min(2, 1);
        return result;
    }
    public String test3(){
        String result = "";
        result = result + Hello.min(3, 3);
        return result;
    }
}
```

Running the tests

1. Set the corresponding classpath to include mujava.jar, openjava2005.jar, tools.jar; should **not** include the path to *MuJava_HOME*\classes\
2. Call the tests runner (the test sets were previously compiled and the results are in *MuJava_HOME*\testset)





TestCase runner

TestCase Runner | Traditional Mutants Viewer | Class Mutants Viewer

- Execute only class mutants
- Execute only traditional mutants
- Execute all mutants

Class : Hello

Method : int_min(int,int)

TestCase: HelloTest RUN

Time-Out : 3 seconds

Op	#
AORB	0
AORS	0
AOIU	2
AOIS	12
AODU	0
AODS	0
ROR	0
COR	0
COD	0
COI	1
SOR	0
LOR	0
LOI	4
LOD	0
ASRS	0

Total : 19

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

Traditional Mutants Result

Live Mutants #	4
Killed Mutants #	15
Total Mutants #	19
Mutant Score	78.0%

Live

AOIS_10
AOIS_11
AOIS_12
AOIS_9

Killed

AOIS_1
AOIS_2
AOIS_3
AOIS_4
AOIS_5
AOIS_6
AOIS_7
AOIS_8
AOIU_1
AOIU_2
COI_1
LOI_1
LOI_2
LOI_3
LOI_4

Running the tests

Note that the log messages go to the command window, but the GUI displays the statistics!

C:\WINDOWS\system32\cmd.exe - run_tests

```
==== Generating original results ========================================
Result for test1  :  1
Result for test2  :  1
Result for test3  :  3


*Executing Mutants
  AOIS_1 - test1 = 2
  AOIS_10 - test1 = 1
 - test2 = 1
 - test3 = 3
  AOIS_11 - test1 = 1
 - test2 = 1
 - test3 = 3
  AOIS_12 - test1 = 1
 - test2 = 1
 - test3 = 3
  AOIS_2 - test1 = 0
  AOIS_3 - test1 = 2
  AOIS_4 - test1 = 0
  AOIS_5 - test1 = 1
 - test2 = 2
  AOIS_6 - test1 = 1
 - test2 = 0
  AOIS_7 - test1 = 1
 - test2 = 2
  AOIS_8 - test1 = 1
 - test2 = 0
  AOIS_9 - test1 = 1
 - test2 = 1
 - test3 = 3
  AOIU_1 - test1 = -1
  AOIU_2 - test1 = 1
 - test2 = -1
  COI_1 - test1 = 2
  LOI_1 - test1 = 1
 - test2 = 2
  LOI_2 - test1 = 2
  LOI_3 - test1 = -2
  LOI_4 - test1 = 1
 - test2 = -2
 Analysis of testcases
  test 1  kill  ==> AOIS_1 AOIS_2 AOIS_3 AOIS_4 AOIU_1 COI_1 LOI_2 LOI_3
  test 2  kill  ==> AOIS_5 AOIS_6 AOIS_7 AOIS_8 AOIU_2 LOI_1 LOI_4
  test 3  kill  ==>
```

Test results

- **Mutant score** =

  Killed mutants # / Total mutants #
- You can design tests to kill mutants by finding a live mutant, then analyzing the program to decide what input will kill it.
- Remember that between 5% to 20% of the mutants are typically **equivalent**!
- Check the AOIS live mutants!

Thank you!