IV. Using µJava to Test Classes

The muJava system requires that the Java CLASSPATH be modified to include the muJava jar and the Java tools.jar files. Then one GUI (Java applet) is used to generate mutants, the tester must create tests, and another GUI is used to run mutants.

1. Environment Settings for the muJava System

There are three steps to setting up the environment for muJava, (1) CLASSPATH, (2) setting the config file, and (3) creating subdirectories.

i. The Java CLASSPATH must include two µJava jar files and one standard Java jar file. **tools.jar** is standard with Java compilers and is probably located in the "lib/" directory. The two µJava files are **mujava.jar** and **adaptedOJ.jar**, which are downloaded from this site. One slightly awkward requirement is that the CLASSPATH must include the location of the classes under test when generating mutants, but **NOT** when running mutants. (If it does, no mutants can be killed.) This location is the "classes/" directory under where the mujava.config is stored.

   In a DOS window, use the following command (assuming that classes is under C:\mujava):

```
set
CLASSPATH=%CLASSPATH%;C:\mujava\mujava.jar;C:\mujava\adaptedOJ.jar;C:\j
2sdk1.4.0_01\lib\tools.jar;C:\mujava\classes
```

   Used as CLASSPATH:
   C:\JMutation\mujava.jar;C:\JMutation\adaptedOJ.jar;C:\Program Files\Java\jdk1.5.0_06\lib\tools.jar

   Add this in when generating mutants: ;C:\JMutation\classes

   In a Cygwin window, use the following command:

```
CLASSPATH="$CLASSPATH;C:\mujava\mujava.jar;C:\mujava\adaptedOJ.jar;C:\j2
sdk1.4.0_01\lib\tools.jar;C:\mujava\classes" ; export CLASSPATH
```

   To change your CLASSPATH permanently in Win2000 and WinXP, go to Start-settings-Control Panel. Double-click System, go to the Advanced tab, and choose Environment Variables. Edit the CLASSPATH variable or create a new variable if there is none. Add the full path to mujava.jar and adaptedOJ.jar to the CLASSPATH.

   In Unix, set the CLASSPATH environment variable. Assuming the jar files are stored in the home directory of user `gpd`:

```
CLASSPATH=$CLASSPATH:/home/gpd/mujava.jar:/home/gpd/adaptedOJ.jar:/ja
va1.4/j2sdk1.4.0_01/lib/tools.jar ; export CLASSPATH
```

Note that the syntax will vary under different shells, and it will be more convenient to put the command in your setup file such as `.login`, or `.bashrc`.

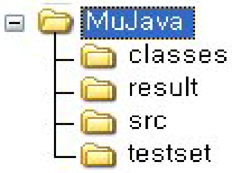
ii. Next, modify the mujava.config file to point to a directory where you wish to store source Java files and muJava temporary files. The directory must be the complete path (either Windows or Unix). For example, the config file may contain the line:

```
MuJava_HOME=C:\home\gpd\exp.
```

Config file called mujava.config contains this: MuJava_HOME=C:\JMutation. Store the config file as this C:\JMutation\mujava.config.

IMPORTANT: It is necessary to copy the config file to the directory you run the muJava system in.

iii. Finally, create a directory structure for the muJava system in the $MuJava_HOME directory. Assuming your *MuJava_HOME* directory is called `MuJava`, the subdirectories should look like: The home directory must be called JMutation.



There should be four subdirectories, used in the following ways.

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

You can create these subdirectories by hand or by using the muJava class "mujava.makeMuJavaStructure".

```
java mujava.makeMuJavaStructure
```

Potential problems: We have identified several potential problems with installing µJava.

- It is important that the MuJava_HOME variable **NOT** have a trailing slash. This will confuse µJava.
- If you have a different version of the java compiler and the JVM, µJava may get confused. This happens sometimes when a new application on your computer updates the JVM. If you have problems compiling or killing mutants, we suggest deleting all Java components and reinstalling the latest version.
- If your tools.jar file is out of date (pre Java 1.4, we think), parts of µJava may not work.

2. Generating Mutants with muJava

Important note: You should run all commands in a directory that contains "mujava.config"

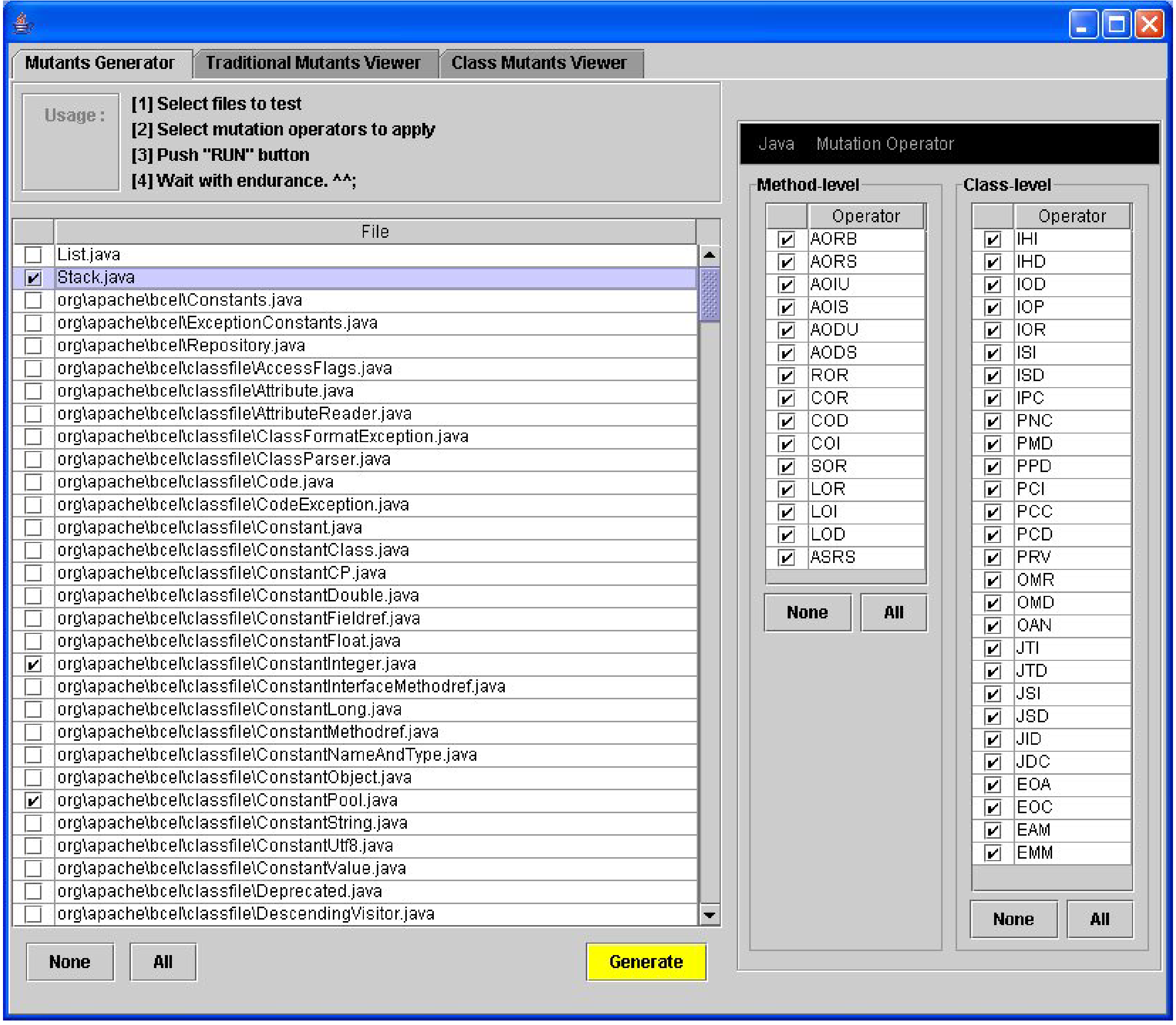
i. Put the source files to test to *MuJava_HOME*\src directory. muJava does not check for compilation errors, so all Java files should compile correctly. If the Java file to test needs other Java files or class files, they should also be placed in *MuJava_HOME*\src. For example, suppose you want to test B, which is a child class of A. Then, you should put both A.java and B.java into *MuJava_HOME*\src. If the file has a package structure, you should store the entire package underneath *MuJava_HOME*\src.

ii. Compile all the Java files in *MuJava_HOME*\src and copy the `.class` files into the *MuJava_HOME*\classes\ directory.

iii. Start the GUI from the command line. Use it to generate mutants:

```
java mujava.gui.GenMutantsMain
```

iv. This command should bring a up a screen similar to the following:



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

	File
☐	List.java
☑	Stack.java
☐	org\apache\bcel\Constants.java
☐	org\apache\bcel\ExceptionConstants.java
☐	org\apache\bcel\Repository.java
☐	org\apache\bcel\classfile\AccessFlags.java
☐	org\apache\bcel\classfile\Attribute.java
☐	org\apache\bcel\classfile\AttributeReader.java
☐	org\apache\bcel\classfile\ClassFormatException.java
☐	org\apache\bcel\classfile\ClassParser.java
☐	org\apache\bcel\classfile\Code.java
☐	org\apache\bcel\classfile\CodeException.java
☐	org\apache\bcel\classfile\Constant.java
☐	org\apache\bcel\classfile\ConstantClass.java
☐	org\apache\bcel\classfile\ConstantCP.java
☐	org\apache\bcel\classfile\ConstantDouble.java
☐	org\apache\bcel\classfile\ConstantFieldref.java
☐	org\apache\bcel\classfile\ConstantFloat.java
☑	org\apache\bcel\classfile\ConstantInteger.java
☐	org\apache\bcel\classfile\ConstantInterfaceMethodref.java
☐	org\apache\bcel\classfile\ConstantLong.java
☐	org\apache\bcel\classfile\ConstantMethodref.java
☐	org\apache\bcel\classfile\ConstantNameAndType.java
☐	org\apache\bcel\classfile\ConstantObject.java
☑	org\apache\bcel\classfile\ConstantPool.java
☐	org\apache\bcel\classfile\ConstantString.java
☐	org\apache\bcel\classfile\ConstantUtf8.java
☐	org\apache\bcel\classfile\ConstantValue.java
☐	org\apache\bcel\classfile\Deprecated.java
☐	org\apache\bcel\classfile\DescendingVisitor.java

Method-level

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

Class-level

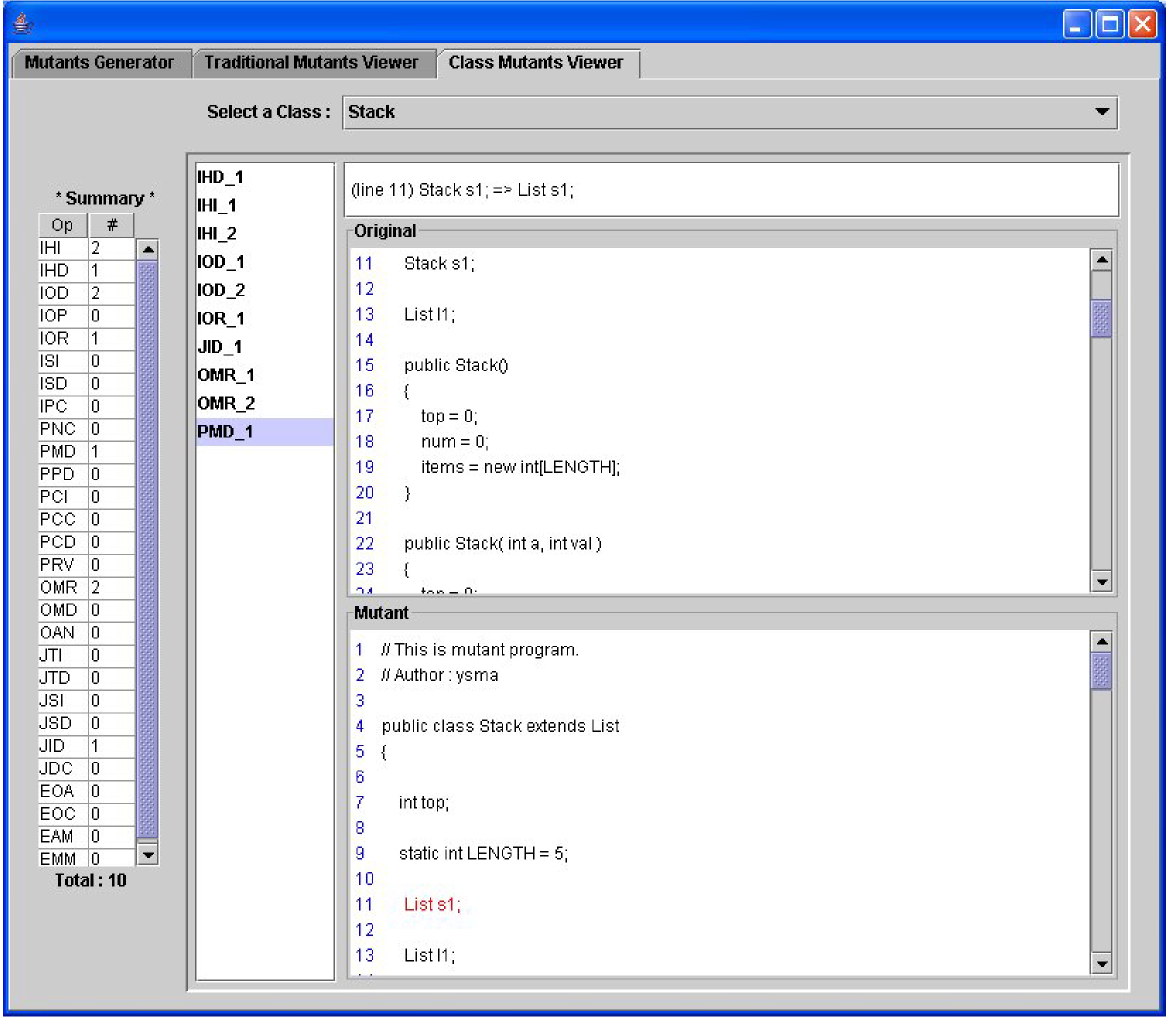
	Operator
☑	IHI
☑	IHD
☑	IOD
☑	IOP
☑	IOR
☑	ISI
☑	ISD
☑	IPC
☑	PNC
☑	PMD
☑	PPD
☑	PCI
☑	PCC
☑	PCD
☑	PRV
☑	OMR
☑	OMD
☑	OAN
☑	JTI
☑	JTD
☑	JSI
☑	JSD
☑	JID
☑	JDC
☑	EOA
☑	EOC
☑	EAM
☑	EMM

v.

vi. Select the files you want to mutate by clicking in the boxes on the left. Select the mutation operators you want to use by slecting their boxes. Then push RUN.

vii. Note: The class mutation operators produce far fewer mutants. Also note that a number of status messages go to the command window, but not the GUI.

viii. After mutants are generated, you can view the mutants in the "Class Mutants Viewer" and "Traditional Mutants Viewer" tabs, as shown in the following two figures.



Select a Class : Stack

* Summary *

Op	#
IHI	2
IHD	1
IOD	2
IOP	0
IOR	1
ISI	0
ISD	0
IPC	0
PNC	0
PMD	1
PPD	0
PCI	0
PCC	0
PCD	0
PRV	0
OMR	2
OMD	0
OAN	0
JTI	0
JTD	0
JSI	0
JSD	0
JID	1
JDC	0
EOA	0
EOC	0
EAM	0
EMM	0

Total : 10

IHD_1
IHI_1
IHI_2
IOD_1
IOD_2
IOR_1
JID_1
OMR_1
OMR_2
PMD_1

(line 11) Stack s1; => List s1;

Original

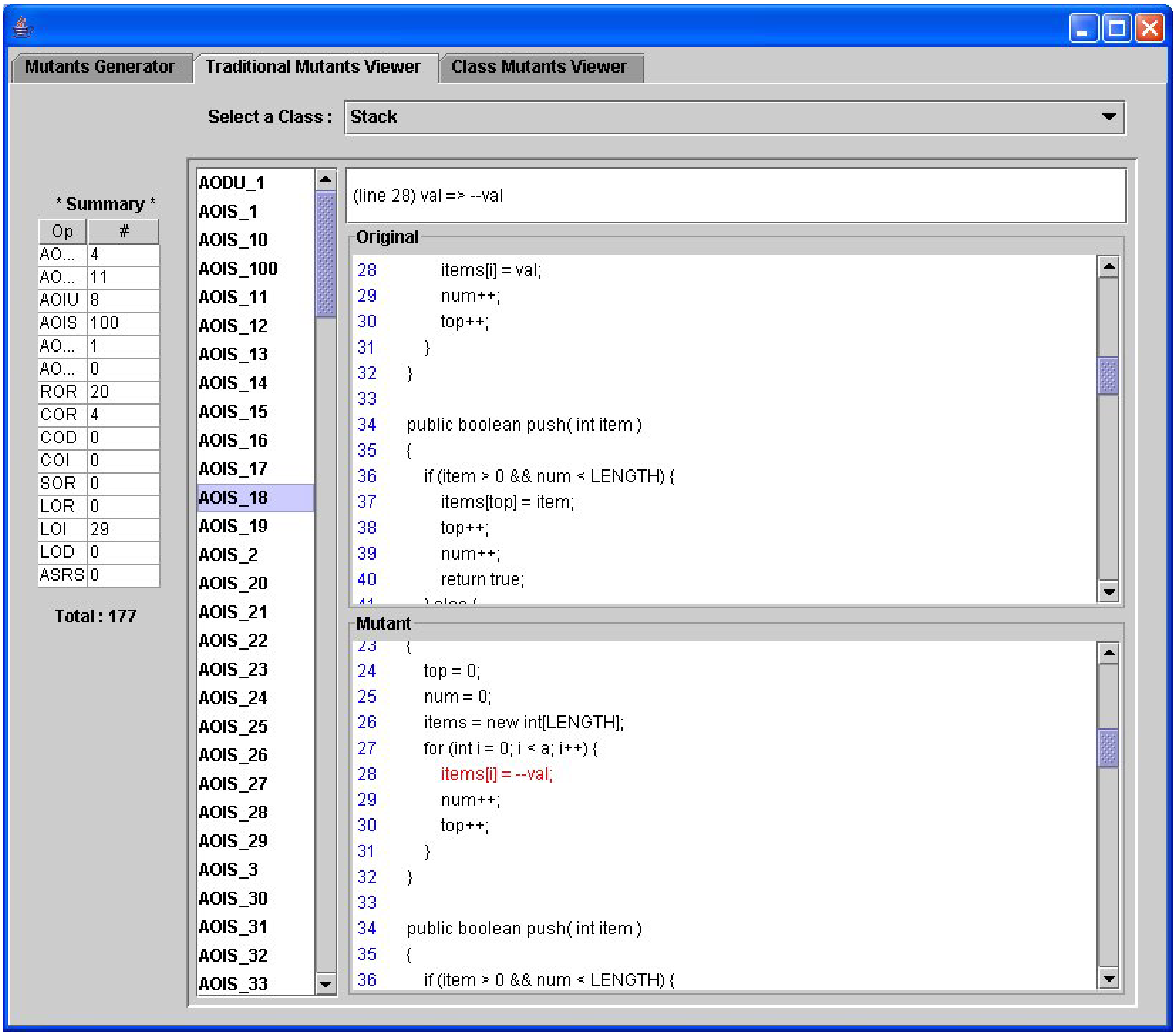
```
11      Stack s1;
12
13      List l1;
14
15      public Stack()
16      {
17          top = 0;
18          num = 0;
19          items = new int[LENGTH];
20      }
21
22      public Stack( int a, int val )
23      {
```

Mutant

```
1   // This is mutant program.
2   // Author : ysma
3
4   public class Stack extends List
5   {
6
7       int top;
8
9       static int LENGTH = 5;
10
11      List s1;
12
13      List l1;
```

ix.

x.



Select a Class : Stack

* Summary *

Op	#
AO...	4
AO...	11
AOIU	8
AOIS	100
AO...	1
AO...	0
ROR	20
COR	4
COD	0
COI	0
SOR	0
LOR	0
LOI	29
LOD	0
ASRS	0

Total : 177

```
AODU_1
AOIS_1
AOIS_10
AOIS_100
AOIS_11
AOIS_12
AOIS_13
AOIS_14
AOIS_15
AOIS_16
AOIS_17
AOIS_18
AOIS_19
AOIS_2
AOIS_20
AOIS_21
AOIS_22
AOIS_23
AOIS_24
AOIS_25
AOIS_26
AOIS_27
AOIS_28
AOIS_29
AOIS_3
AOIS_30
AOIS_31
AOIS_32
AOIS_33
```

(line 28) val => --val

Original

```
28          items[i] = val;
29          num++;
30          top++;
31       }
32    }
33
34    public boolean push( int item )
35    {
36       if (item > 0 && num < LENGTH) {
37          items[top] = item;
38          top++;
39          num++;
40          return true;
41       } else {
```

Mutant

```
23    {
24       top = 0;
25       num = 0;
26       items = new int[LENGTH];
27       for (int i = 0; i < a; i++) {
28          items[i] = --val;
29          num++;
30          top++;
31       }
32    }
33
34    public boolean push( int item )
35    {
36       if (item > 0 && num < LENGTH) {
```

xi.

xii. You may be interested in knowing where the mutant versions of the classes are stored. They are underneath the *MuJava_HOME*\result\ directory. The following example shows the directory Stack underneath result, with object-oriented mutants in class_mutants and traditional mutants in a separate directory.

3. Making 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.

Below is an example of a testset class for the class Stack. Its name is StackTest. StackTest contains two test case: test1() and test2(). The testset .class file should be in the directory *MuJava_HOME*\testset\.

```
public class StackTest
{

    public String test1()
    {
        String result = "";
        Stack obj = new Stack();
        obj.push(2);
        obj.push(4);
        result = result + obj.isFull();
        result = result + obj.pop();
        return result;
    }

    public String test2()
    {
        String result = "";
        Stack obj = new Stack();
        obj.push(5);
        obj.push(3);
        result = result + obj.pop();
        result = result + obj.pop();
        return result;
    }

}
```

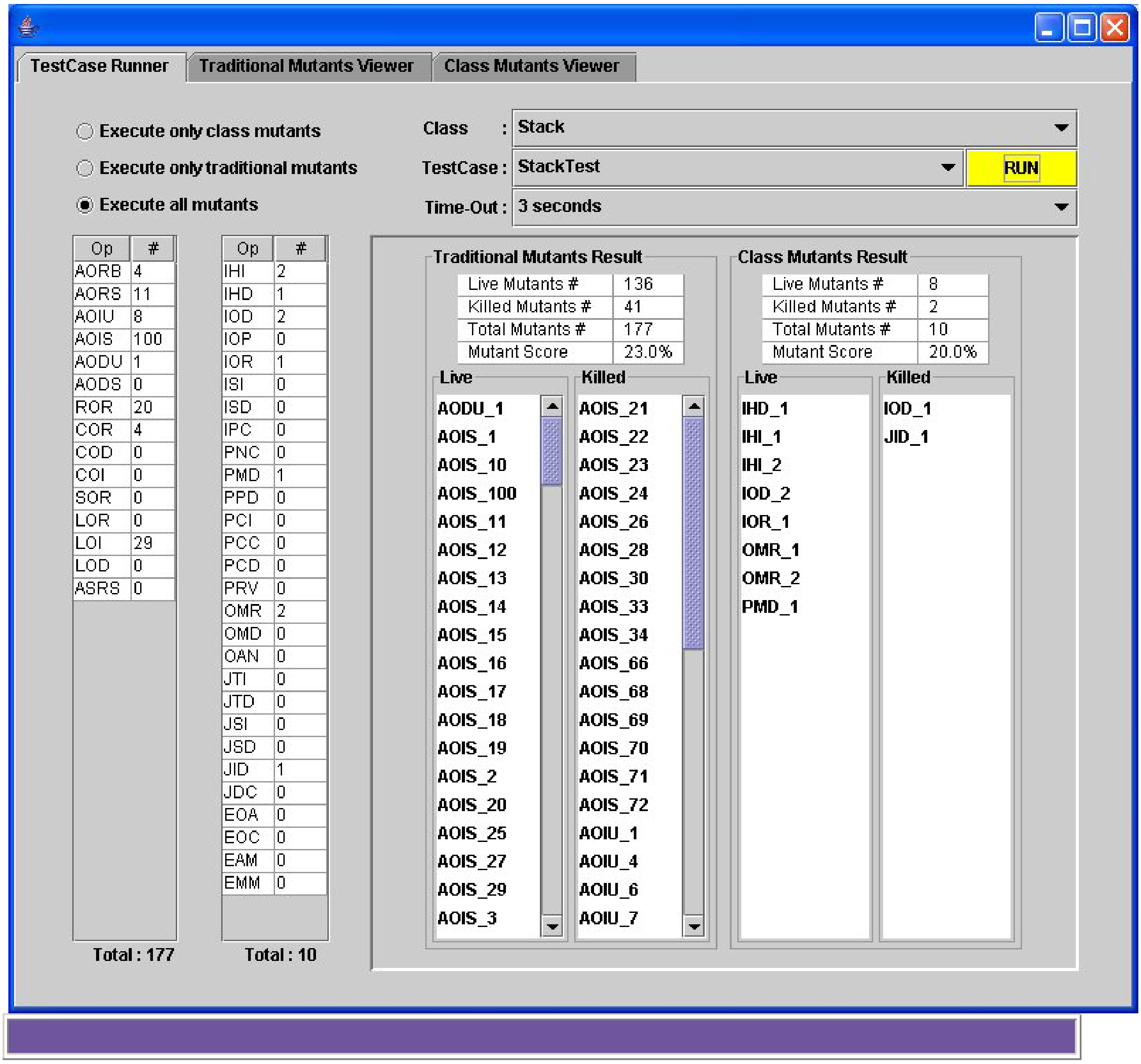
4. Running mutants.

Run the mutants from another GUI. Start it with the following command:

```
java mujava.gui.RunTestMain
```

Note: Your CLASSPATH must not include *MuJava_HOME*\classes\. If it does, no mutants can be killed.

You should see the following GUI. You can select which collection of mutants to run, and which testset to use. The "Class Mutants Viewer" and "Traditional Mutants Viewer" tabs will show the source view of the 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.



○ Execute only class mutants

○ Execute only traditional mutants

● Execute all mutants

Class : Stack

TestCase : StackTest

Time-Out : 3 seconds

Op	#
AORB	4
AORS	11
AOIU	8
AOIS	100
AODU	1
AODS	0
ROR	20
COR	4
COD	0
COI	0
SOR	0
LOR	0
LOI	29
LOD	0
ASRS	0

Total : 177

Op	#
IHI	2
IHD	1
IOD	2
IOP	0
IOR	1
ISI	0
ISD	0
IPC	0
PNC	0
PMD	1
PPD	0
PCI	0
PCC	0
PCD	0
PRV	0
OMR	2
OMD	0
OAN	0
JTI	0
JTD	0
JSI	0
JSD	0
JID	1
JDC	0
EOA	0
EOC	0
EAM	0
EMM	0

Total : 10

Traditional Mutants Result

Live Mutants #	136
Killed Mutants #	41
Total Mutants #	177
Mutant Score	23.0%

Live	Killed
AODU_1	AOIS_21
AOIS_1	AOIS_22
AOIS_10	AOIS_23
AOIS_100	AOIS_24
AOIS_11	AOIS_26
AOIS_12	AOIS_28
AOIS_13	AOIS_30
AOIS_14	AOIS_33
AOIS_15	AOIS_34
AOIS_16	AOIS_66
AOIS_17	AOIS_68
AOIS_18	AOIS_69
AOIS_19	AOIS_70
AOIS_2	AOIS_71
AOIS_20	AOIS_72
AOIS_25	AOIU_1
AOIS_27	AOIU_4
AOIS_29	AOIU_6
AOIS_3	AOIU_7

Class Mutants Result

Live Mutants #	8
Killed Mutants #	2
Total Mutants #	10
Mutant Score	20.0%

Live	Killed
IHD_1	IOD_1
IHI_1	JID_1
IHI_2	
IOD_2	
IOR_1	
OMR_1	
OMR_2	
PMD_1	

Last update: May 2006, Jeff Offutt

*Assistance from the U.S. National Science Foundation
under grant CCR-98-04111 to George Mason University
is greatly appreciated.*

Remember when using MuJava

Run from JMutation folder on C:disk.

C:\JMutation\java mujava.gui.GenMutatantsMain (to generate mutants)

C:\JMutation\java mujava.gui.RunTestMain (to kill mutants

Notes in red by Alma Ágústsdóttir.