****

**软件测试技术第四次实验报告**



**学 院 智能与计算学部**

**专 业 软件工程**

**年 级 2016级**

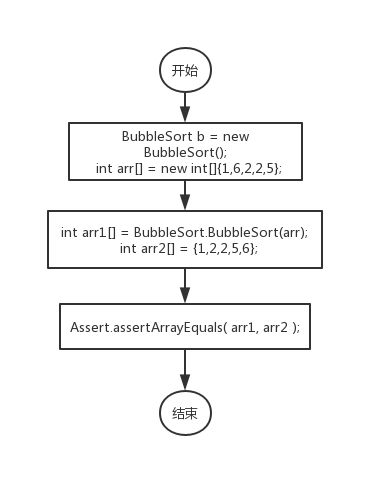
**姓 名 夏子霞**

**2019年 4 月 8 日**

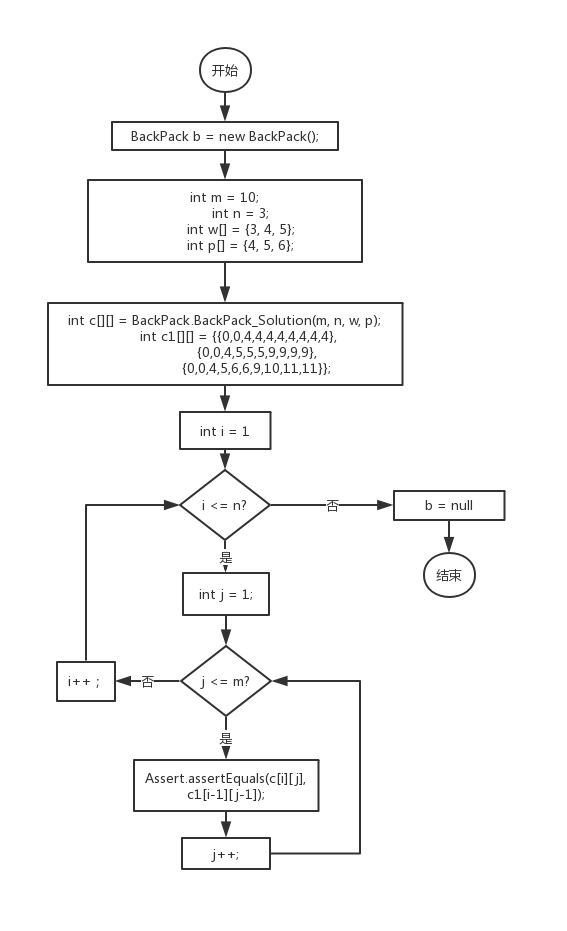
# 软件测试技术第四次实验报告

1. 需求分析（描述具体需求）
2. 安装mujava
3. 用MuJava生成两个给定程序的突变体
4. 用Junit编写2个程序的测试集，并用MuJava在测试集上运行突变体
5. 概要设计（简单描述设计思路，配合UML图）
6. 直接在命令行利用javac命令生成后缀为“.class”的文件，再用mujava生成突变体
7. 编写Junit测试，UML图如下

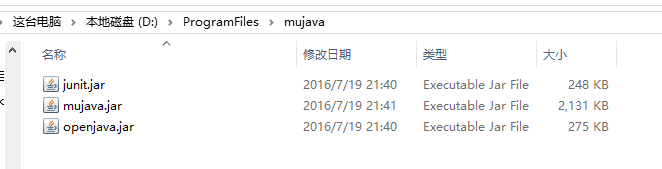
冒泡排序：



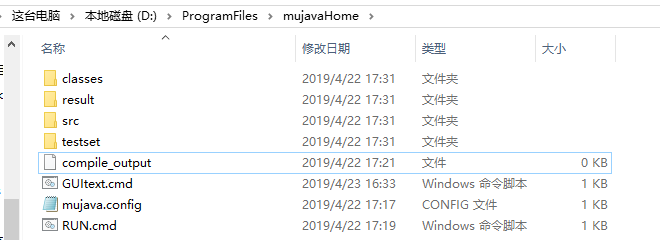
背包算法：



1. 用命令行为每个test生成后缀为“.class”的文件，再用mujava进行测试。
2. 详细设计（详细描述具体如何实现，附代码及说明）
3. 安装mujava
4. 放置配置文件，目录如下：



1. 构建运行文件，目录如下：



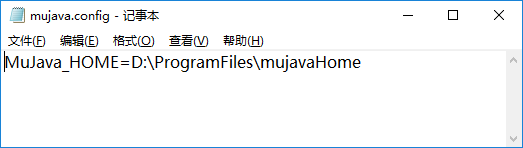
Src存放待测Java程序源代码

Classes存放src中源代码经过正确编译的.class文件

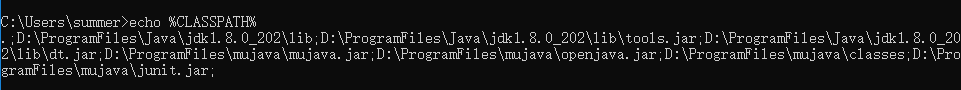
Testset存放JUnit测试用例的.class文件

Result存放生成的变体结果

其中mujava.config的内容为：



1. 更改环境变量CLASSPATH,更改后结果如下：



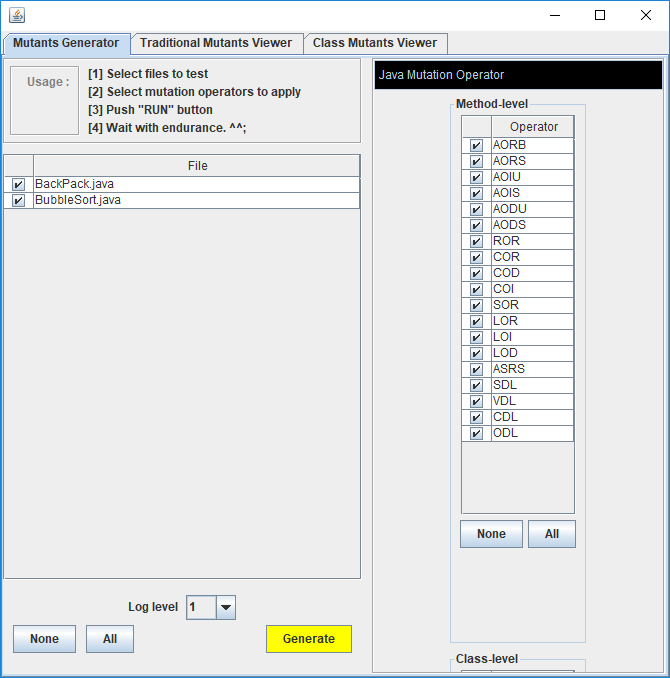
1. 用MuJava生成两个给定程序的突变体
2. 为待测试程序生成类文件，并将待测程序和类文件放在指定目录下：  
   

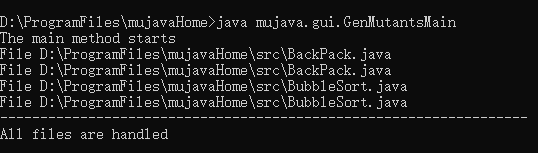


1. 新建GUI.cmd运行脚本，内容为：

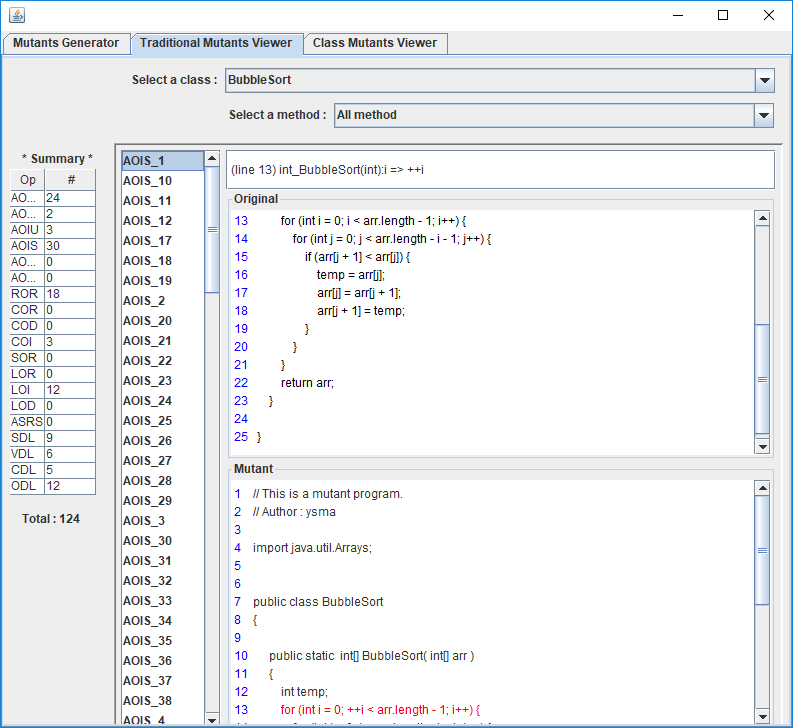
java mujava.gui.GenMutantsMain

1. 点击脚本，为两个程序生成变异体：

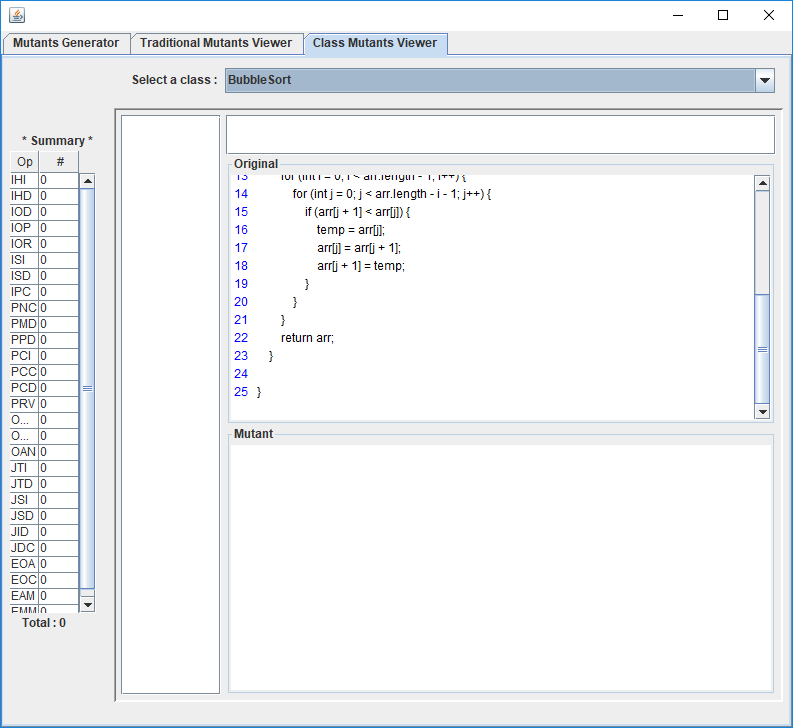


生成结果如下：  


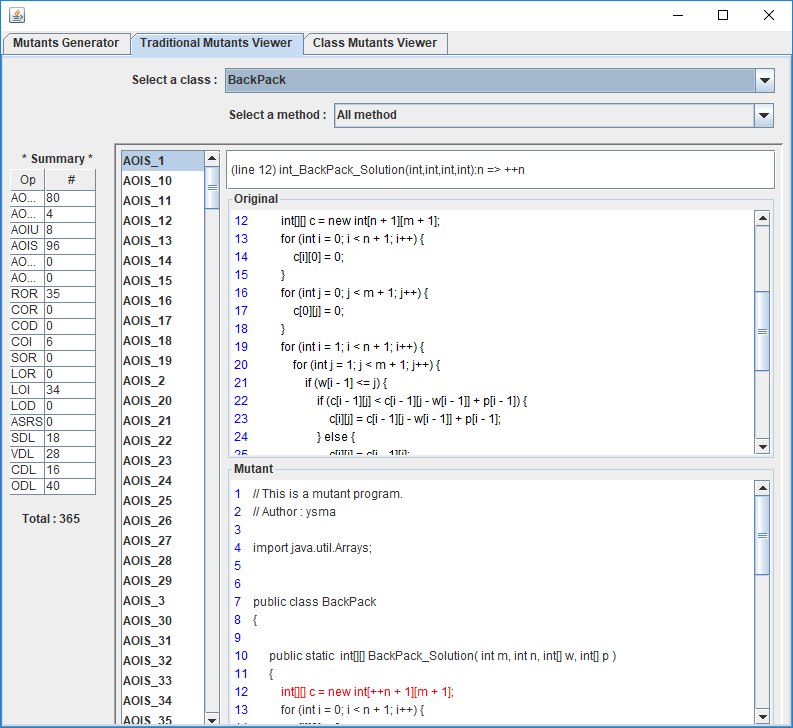
Bubblesort的运算符变异为：



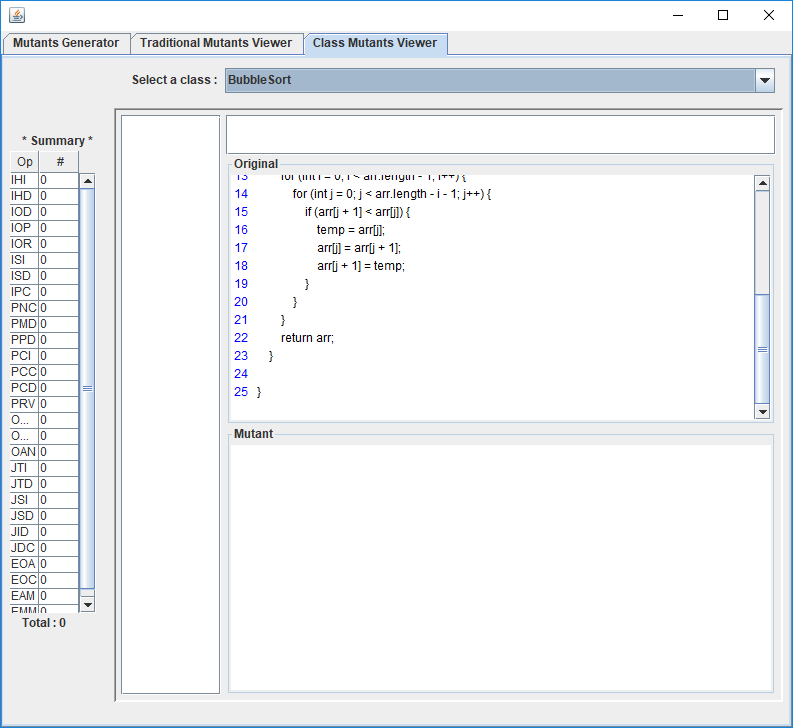
Bubblesort的类变异为空：



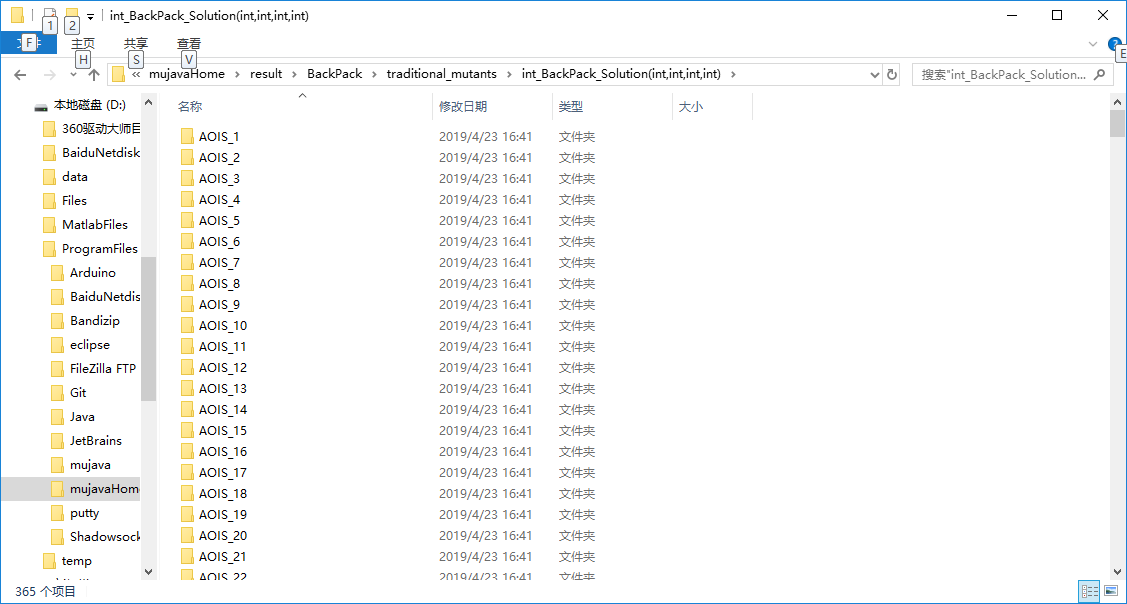
Backpack的运算符变异为：



Backpack的类变异为空：



1. 在result文件夹下查看：



1. 用Junit编写2个程序的测试集，并用MuJava在测试集上运行突变体
2. BubbleSortTest.java为：

import org.junit.After;

import org.junit.Assert;

import org.junit.Before;

import org.junit.Test;

public class BubbleSortTest {

public BubbleSort b;

@Before

public void setUp() throws Exception {

b = new BubbleSort();

}

@After

public void tearDown() throws Exception {

b = null;

}

@Test

public void test() {

int arr[] = new int[]{1,6,2,2,5};

int arr1[] = BubbleSort.BubbleSort(arr);

int arr2[] = {1,2,2,5,6};

Assert.assertArrayEquals( arr1, arr2 );

}

}

BackPackTest.java为：  
import static org.junit.Assert.\*;

import org.junit.After;

import org.junit.Assert;

import org.junit.Before;

import org.junit.Test;

public class BackPackTest {

public BackPack b;

@Before

public void setUp() throws Exception {

b = new BackPack();

}

@After

public void tearDown() throws Exception {

b = null;

}

@Test

public void test() {

int m = 10;

int n = 3;

int w[] = {3, 4, 5};

int p[] = {4, 5, 6};

int c[][] = BackPack.BackPack\_Solution(m, n, w, p);

int c1[][] = {{0,0,4,4,4,4,4,4,4,4},

{0,0,4,5,5,5,9,9,9,9},

{0,0,4,5,6,6,9,10,11,11}};

for (int i = 1; i <= n; i++) {

for (int j = 1; j <= m; j++) {

Assert.assertEquals(c[i][j], c1[i-1][j-1]);

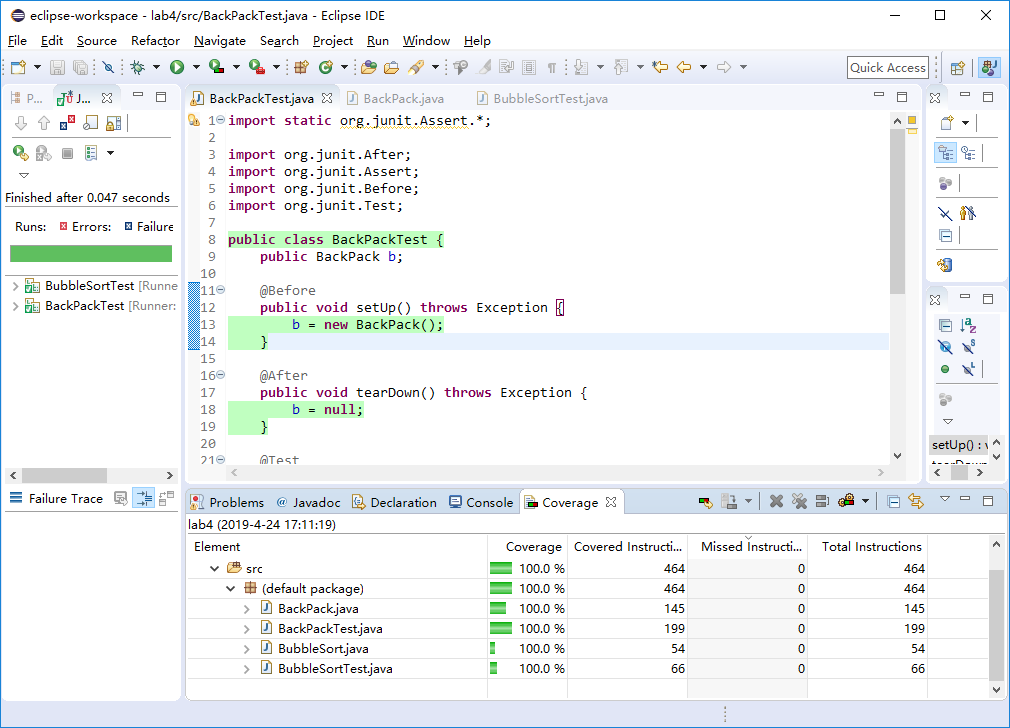
}

}

}

}

1. Junit测试的结果为：



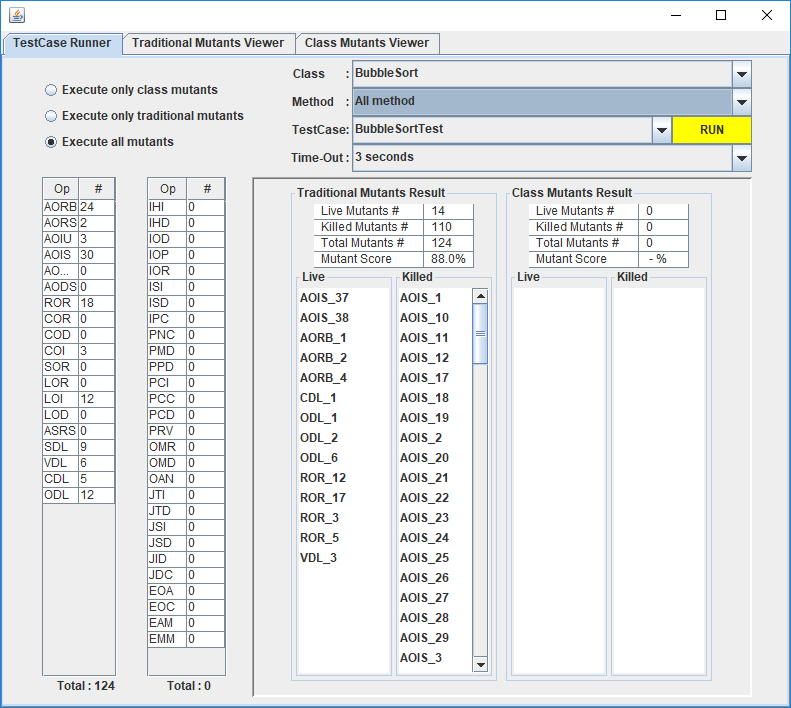
1. 将两个Junit测试的类文件放在指定目录下：  
   
2. 新建RUNbubble.cmd运行脚本，内容为：

java mujava.gui.RunTestMain > BubbleSort.txt

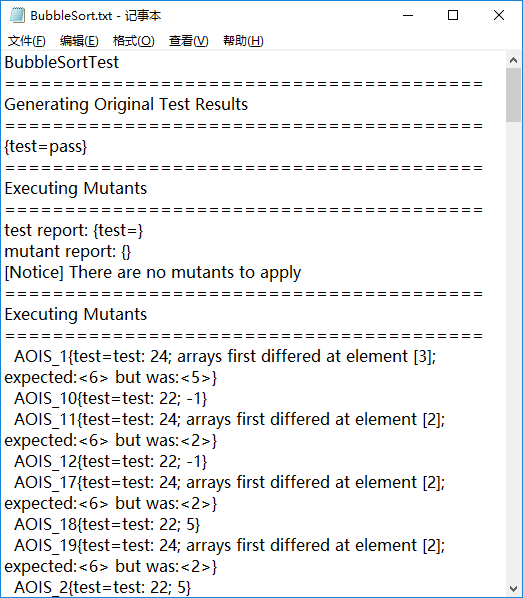
新建RUNback.cmd运行脚本，内容为：

java mujava.gui.RunTestMain > BackPack.txt

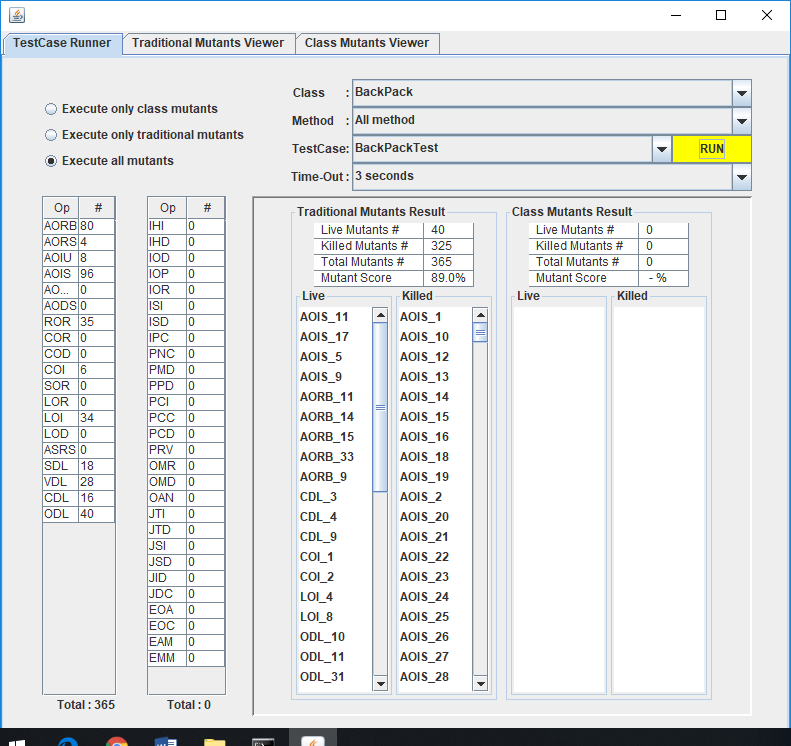
1. 运行RUNbubble.cmd，对变异体的测试结果为：



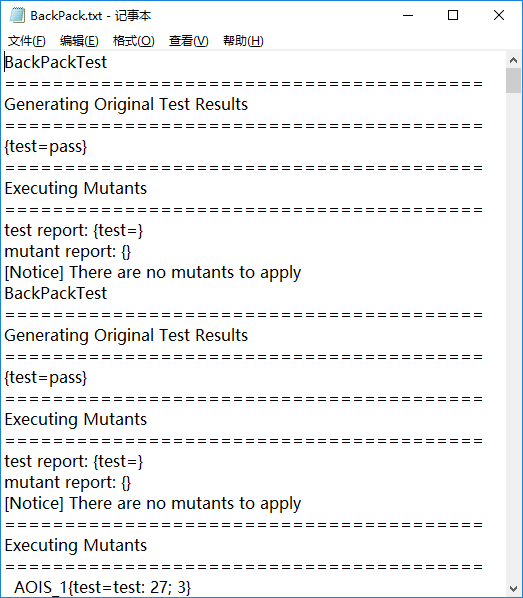
生成的BubbleSort.txt为：



运行RUNbubble.cmd，对变异体的测试结果为：



生成的BubbleSort.txt为：



1. 对测试结果产生的文本文件进行分析，以bubblesort为例。

其中,

BubbleSortTest, 测试用例集合类名；

Generating Original Test Results：对待测类BubbleSort原程序的测试结果，应该要全部正确pass，作为软件测试中的oracle。

第一个Executing Mutants：对待测类BubbleSort类级别变体 Class-Level的测试结果。即使没有进行类级别测试，也会输出该块。

第二个Executing Mutants：对待测类BubbleSort方法级别变体Method-Level的测试结果。

test report列出某个用例可以杀死的所有变体

mutant report列出某个变体可以被哪些用例杀死

若没有类级别变体或者方法级别变体，则最后一行分别出现 [Notice] There are no mutants to apply

1. 调试分析（在实验过程中遇到的问题以及如何解决）
2. 安装mujava的时候，对于JAVA\_HOME变量应该注意结尾不能带分号，否则在CLASSPATH中引用JAVAHOME就会出错变成

“D:\ProgranFiles\Java\jdk1.8.0\_202;\lib\tools.jar;”，

而所期待的正确写法是

“D:\ProgramFiles \Java\jdk1.8.0\_202\lib\tools.jar;”

1. 在配置路径的时候，应该注意路径中的文件夹不能有空格，这个错误和上一个错误都会造成不能生成变异体，变异结果为空。
2. 对于java文件，应该注意其包目录。如果包含“package XX;”，应在src目录下新建文件夹“XX”，将java文件放在此目录下运行，否则DOS端是无法正确生成class文件的。不过经过多次实验，为了方便，还是去掉包目录比较好。
3. 官网上的CLASSPATH还要求添加了路径“C:\mujava\classes”，后猜测其目的是为了在testset中运行Junit测试文件从而生成类文件的时候，能够依赖classes文件夹中的BubbleSort.class和BackPack.class文件。而我后来查阅资料添加的“D:\ProgramFiles\mujava\junit.jar”路径也是为了依赖Junit包。

看似没有问题，但在测试变异体的时候就出错了。明明的确产生了变异，但最后测试结果却是一个没有kill掉。我以为又是配置问题或是JDK的版本问题，整个过程重新做了好几遍，可是Mutant Score还是0。

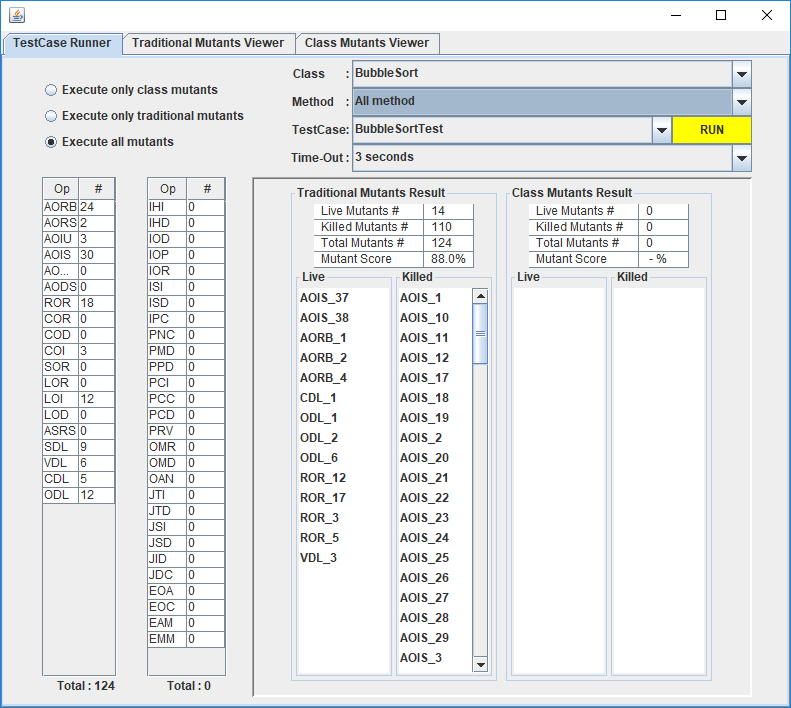
后来我以为是jar包和几个文件夹不能放在一起，就又重新建了一个文件夹“MujavaHome”,然后又修改了CLASSPATH的路径，但是此时class文件夹没有BubbleSort.class和BackPack.class文件。这时对变异体进行测试Mutant Score不再是0了！

为了保险我又重新走了一遍流程，然后Mutant Score就又变成0了。比较了一下前后差异，我推测是因为在对变异体进行测试的时候，编译器可能直接编译了CLASSPATH中的classes文件夹的相应类文件，而没有编译变异体对应的类文件，所以才会一个也没有kill掉。

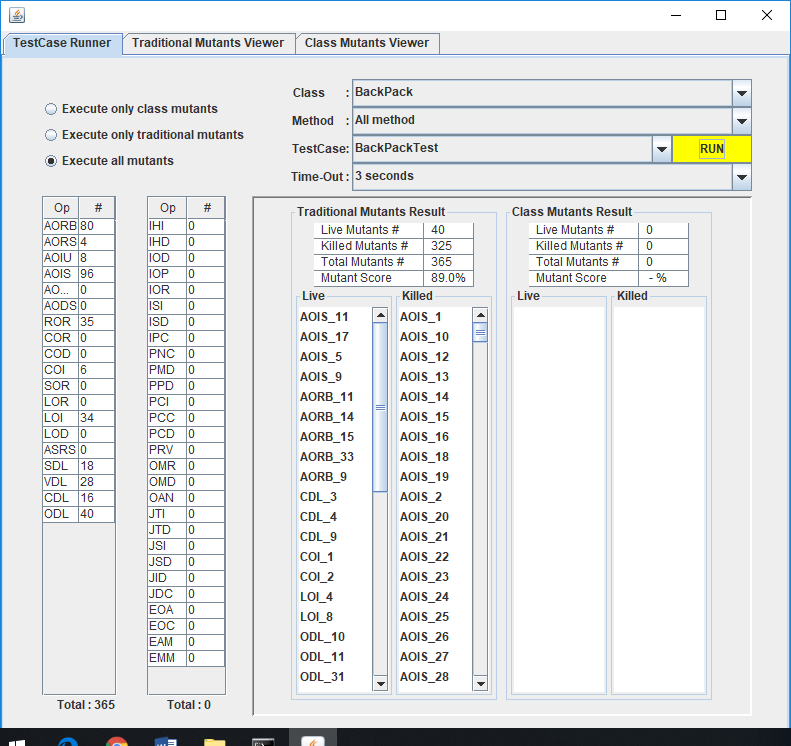
我最后采取的解决方法是去掉CLASSPATH中的“D:\ProgramFiles\mujavaHome\classes”

1. 测试结果（描述输入和输出）

BubbleSort：



BackPack:



对没有kill掉的变异体进行分析，比如BubbleSort中的AOIS\_37,它是将“arr[j + 1] = temp;”变异成了“arr[j + 1] = temp++;”，这对arr[j + 1]的值并没有产生影响，而在下一次的循环中temp变量又被重新赋值，所以对最后的输出结果没有影响。

1. 总结

在本次实验中，我学习到了一个新的软件测试工具mujava,对它的使用有了基本的了解。

同时，在利用mujava进行变异测试的时候，我遇到了很多问题，踩了很多坑，这让我对自己所学的专业有了更深刻的理解，虽然编程是有固定的框架，但是对于框架也要有深刻的理解，并且可兼容性也十分重要。

（git地址：https://github.com/XiaZiXia/lab4）