软件测试上机报告

第一次上机作业 - Lab 3 MuJava

学　　院　智能与计算学部

专　　业　软件工程

姓　　名　郎文翀

学　　号　**3019244247**

年　　级　2019 级

班　　级　软件工程 5 班

1. Experimental requirements

1. Install MuJava. The instruction of how to install and use Mujava can be seen in https://cs.gmu.edu/~offutt/mujava/ .
2. Two small programs are given for your task. BubbleSort.java is an implementation of bubble sort algorithm and BackPack.java is a solution of 01 backpack problem. Try to generate Mutants of 2 given programs with MuJava.
3. Write testing sets for 2 programs with Junit, and run mutants on the test sets with MuJava.

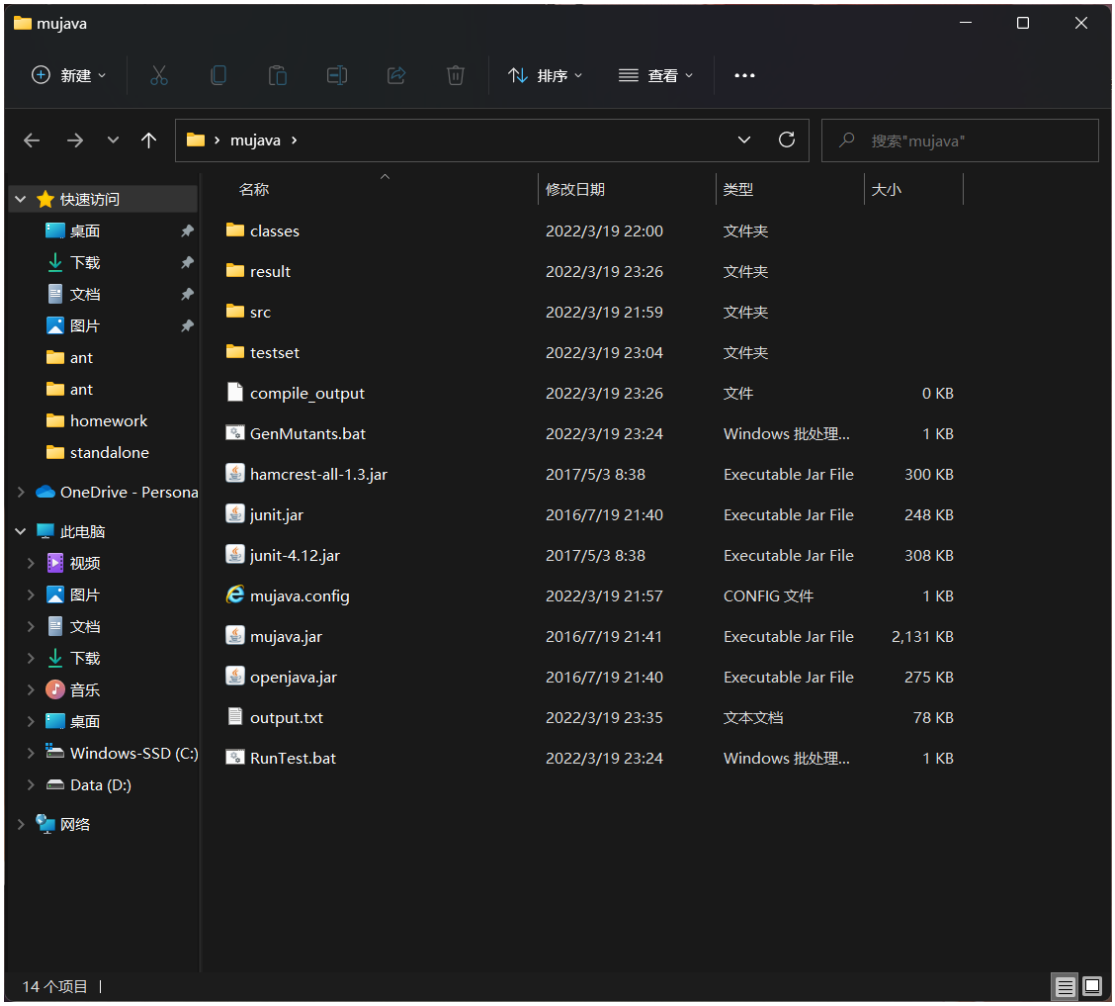
Requirements for the experiment:

1. Finish the tasks above individually.
2. Check in your java code to github or gitee.
3. Post your experiment report to “智慧树” , the following information should be included in your report:
    a) The brief description that you install MuJava
    b) Steps for generating Mutants
    c) Steps for making test sets and running mutants.
    d) Your mutants result (The number of live mutants, killed mutants, etc.)

2. Configuration

2.1 环境配置

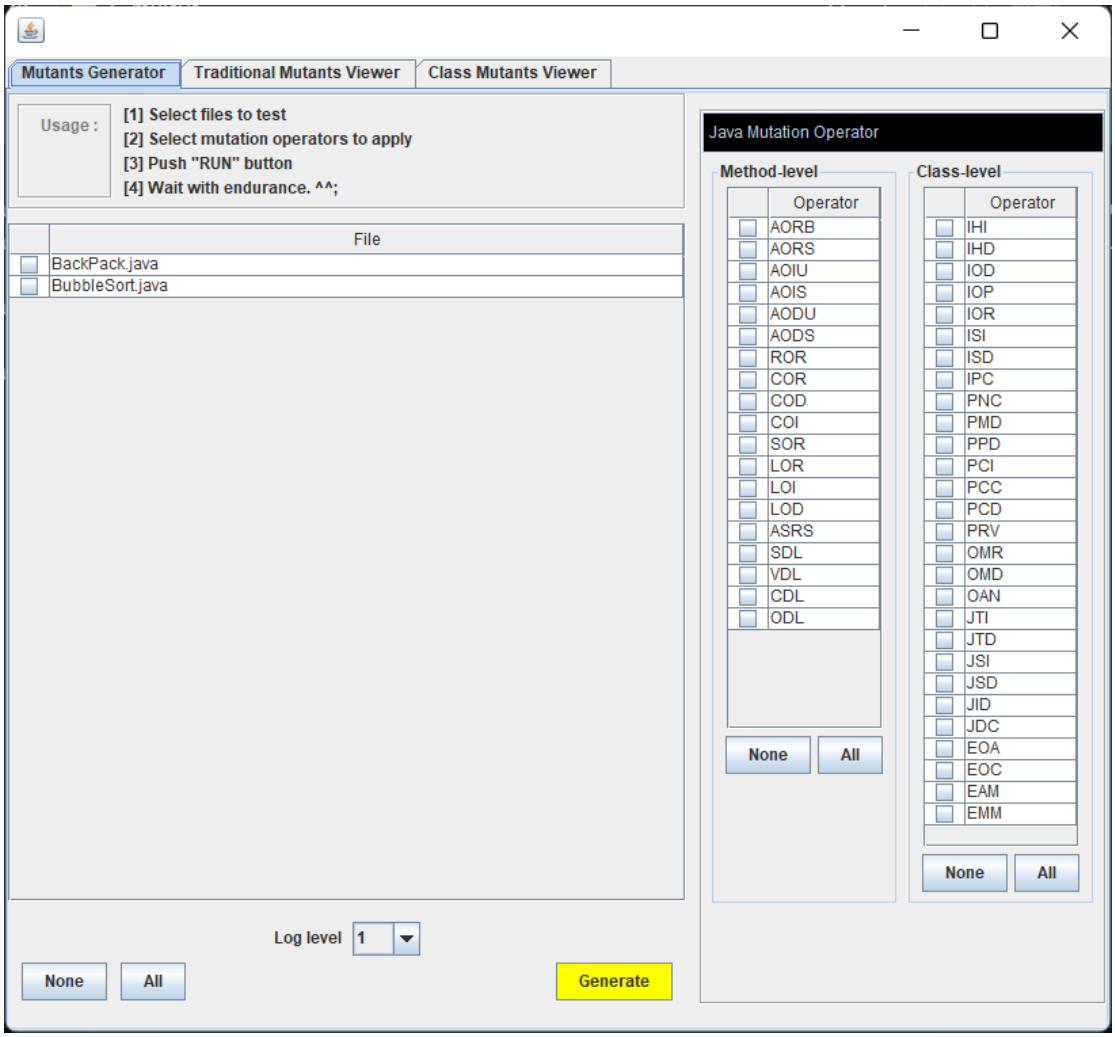
在使用 Mujava 之前首先我们要配置一下运行环境，将下载的 openjava.jar 和 mujava.jar 放置到新建的文件加 mujava 中，然后我们在这个文件夹中新建一个 mujava.config 文件，在其中写上环境根目录为 mujava 从而指向该文件夹。然后我们还需要创建 src 文件夹（存放源码文件），classes 文件夹（存放源码文件的编译结果 class 文件），result 文件夹（存放生成的变异结果），testset 文件夹（Junit 测试文件以及编译结果）。
文件结构如下图：



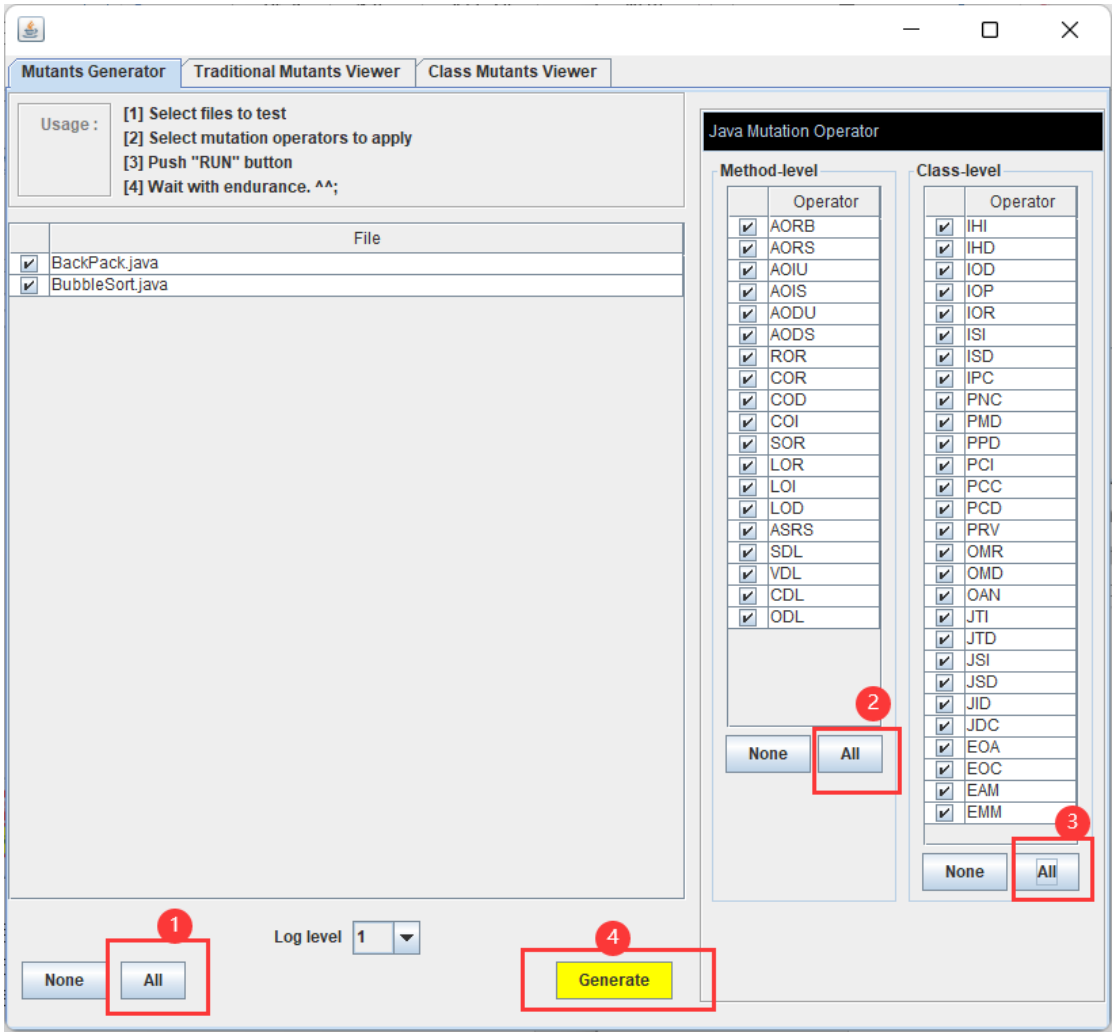
然后我们还需要编写两个脚本文件，分别命名为 GenMutants.bat（生成编译结果），RunTest.bat(对变异结果进行测试)。两个脚本文件的具体命令见第四部分。

2.2 生成变异结果

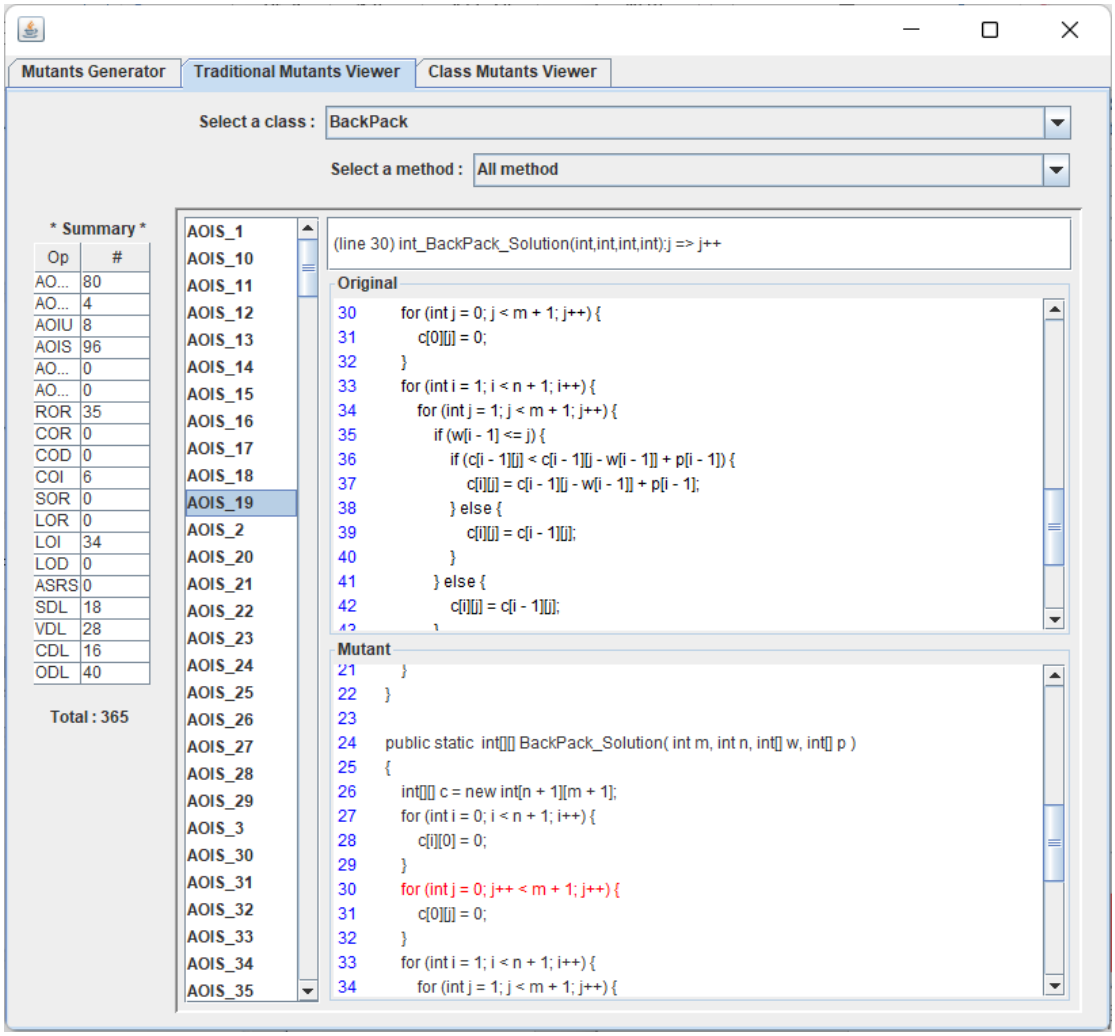
然后我们开始进行变异体的生成，首先我们执行刚刚写好的 GenMutants.bat 脚本，此时如果能正常运行，则会弹出一个 GUI 界面，如下所示：



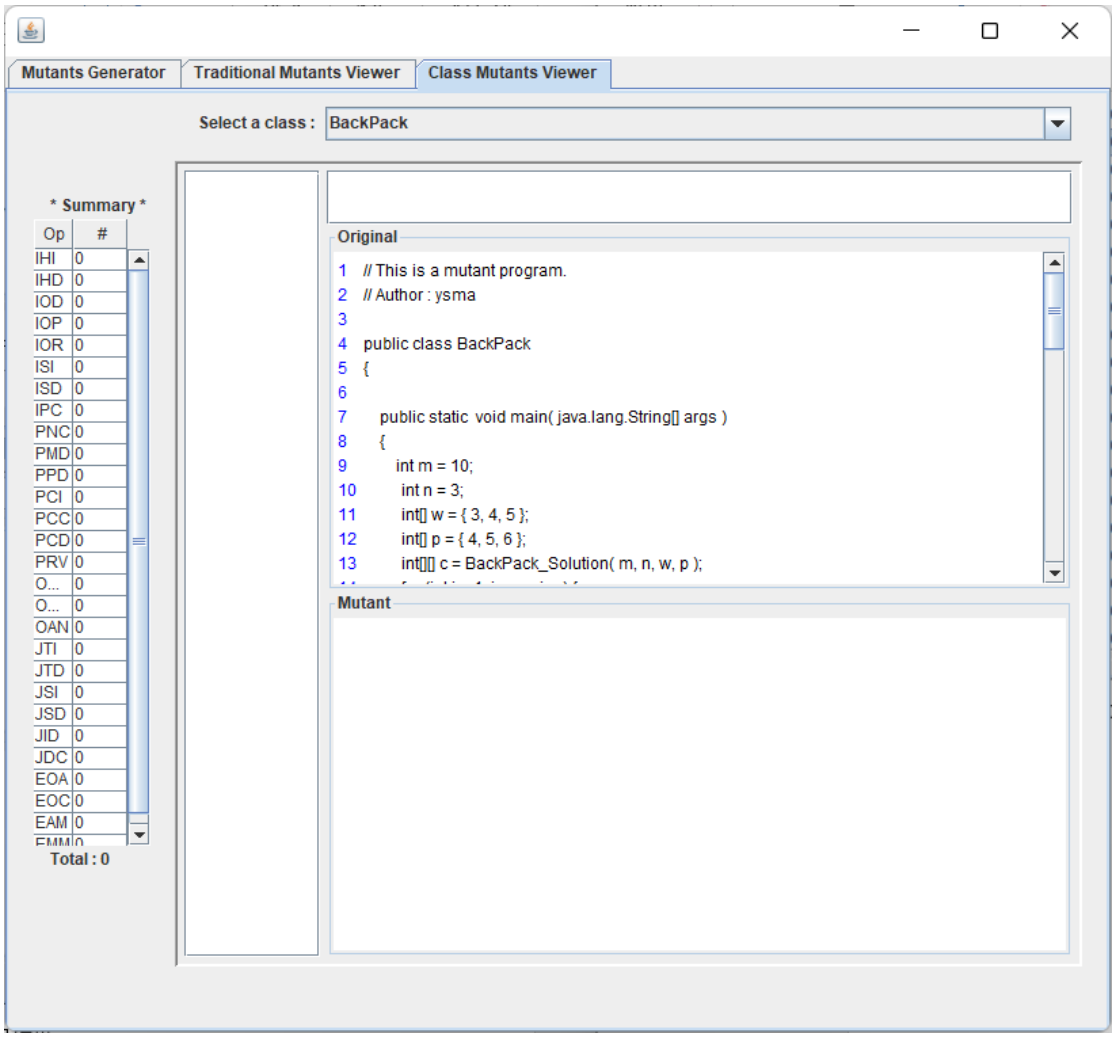
他会自动查找 classes 目录下的 class 文件，然后按照我们选择的变异算子和方法进行变异体的生成，并并且他会将生成的变异体存放到 result 目录下。我们这里选择所有文件，并且使用所有的变异算子来进行变异体的生成，操作步骤如下图：



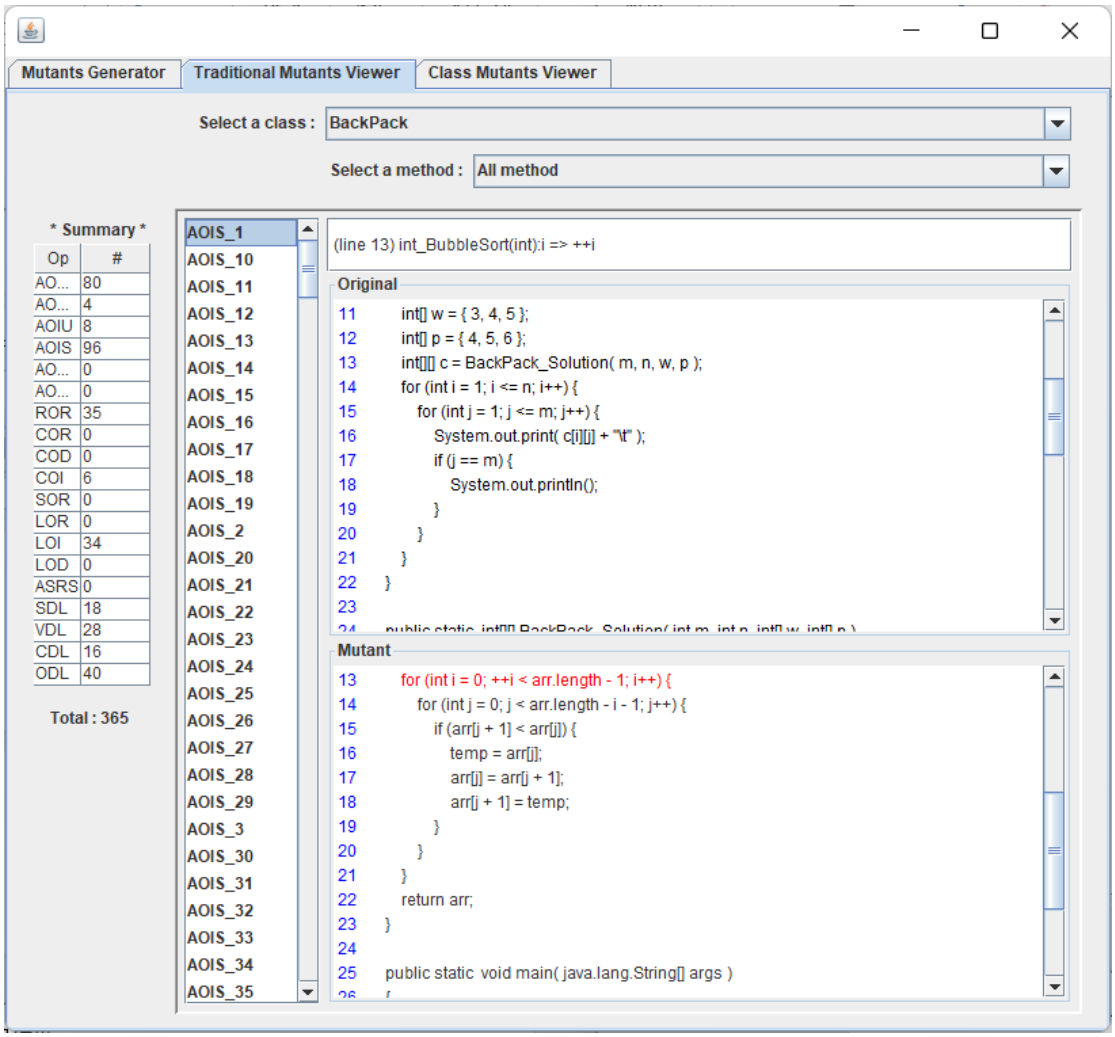
然后我们还可以再上方选择变异方法，这里我们分别选择传统的变异和类变异，两个文件的变异结果如下图所示：



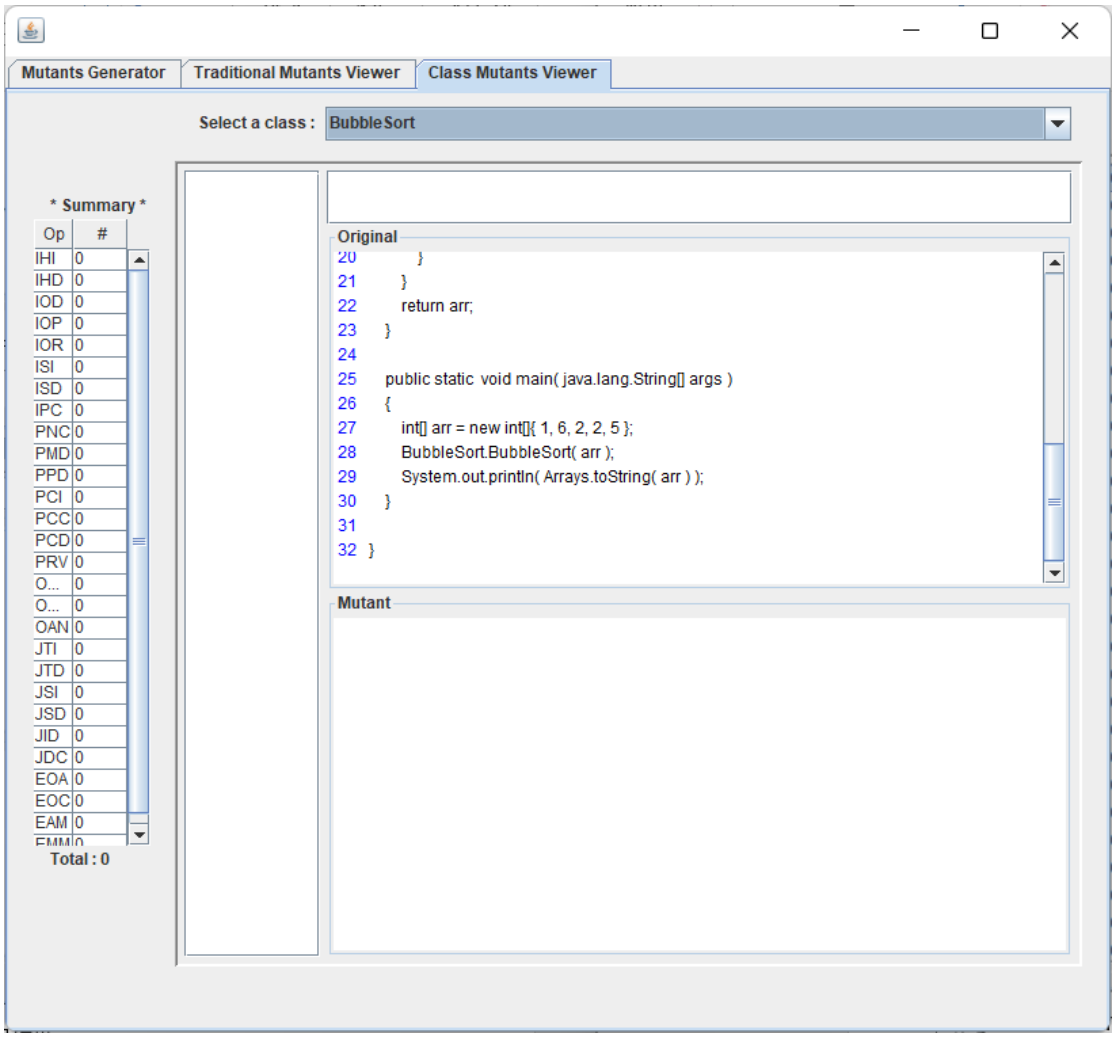
如上图是 BackPack.java 的某一个变异体，并且是传统变异结果，他将 j 改成了 j++。



上图是 BackPack.java 的类变异结果



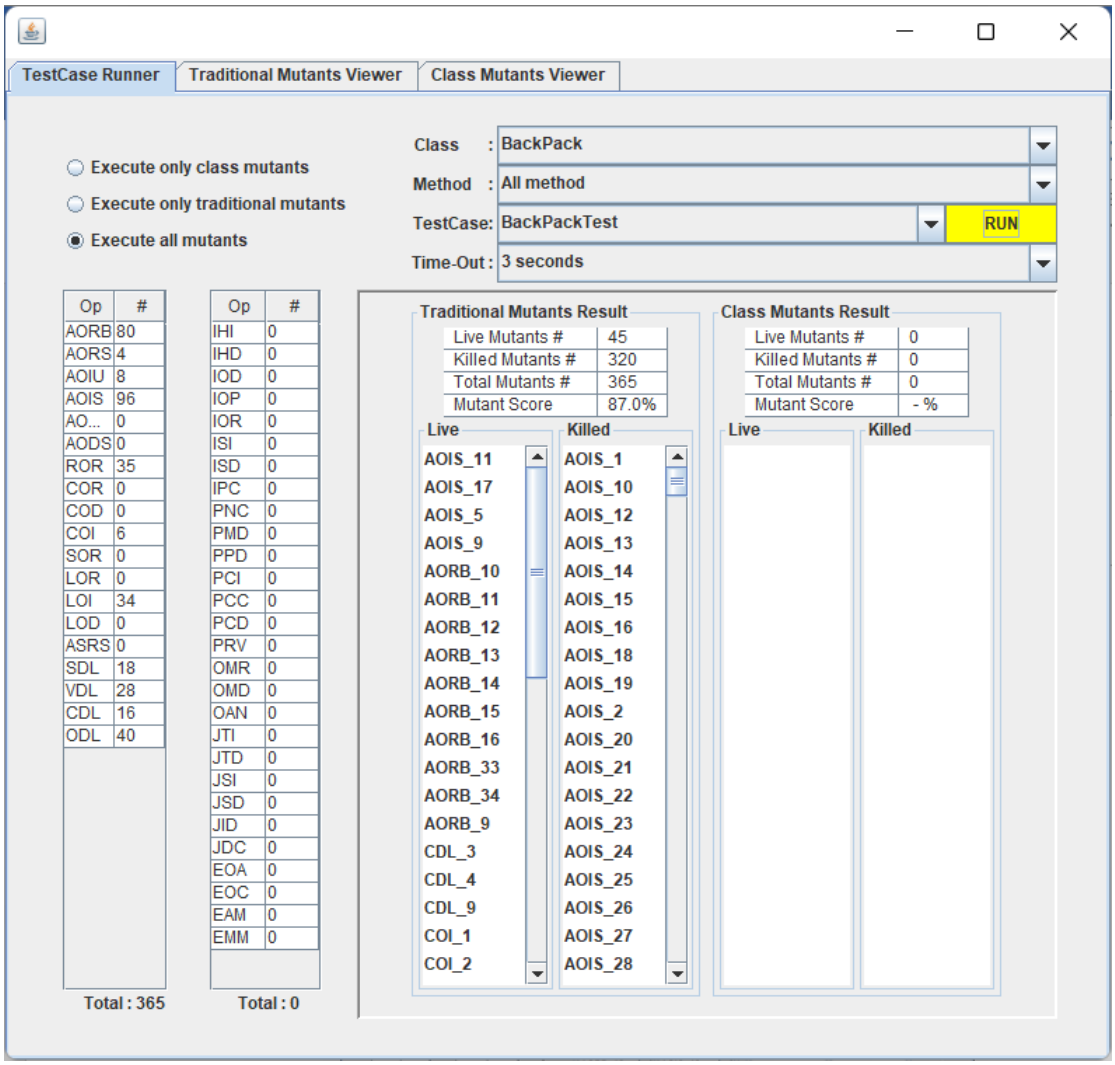
上图是 BubbleSort.java 的传统变异结果



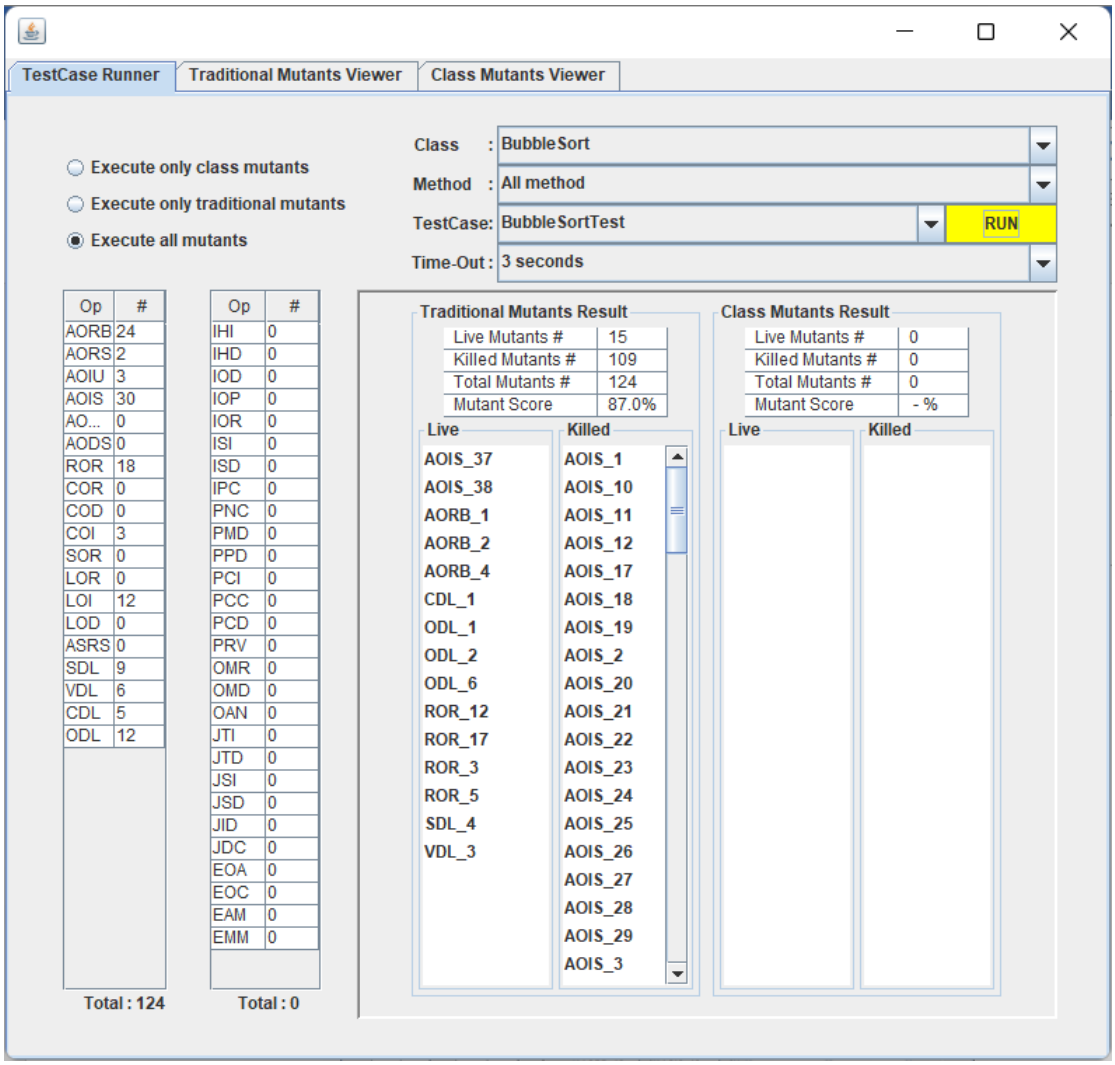
如上图是 BubbleSort.java 的类变异结果

2.3 对变异体进行测试

我们发现执行完以后 result 中已经生成了许多变异结果，然后我们接着执行 RunTest.bat 脚本文件 ，此时也会弹出一个 GUI 界面，他的作用是测试之前的变异的结果是否能够满足测试用例，我们选择要测试的类和其对应的测试类，然后点击 run 即可进行测试。这里可能需要等待较长的时间：



BckPack.java 的变异体测试，我们可以看到杀死率为 87%。



BubbleSort.java 变异体测试，我们可以看到杀死率为 87%

3. Result analysis

我们从上面的测试结果可以看到变异体的杀死率大约在 87%，不能够满足我们的覆盖要求，还需要进一步进行样例的补充增强，以保证测试样例覆盖率够高能够满足需求。

4. Source code

实验程序已存储到 github: https://github.com/Langwenchong/SoftwareTestingLabs/tree/lab3

1. GenMutants.bat

```
set CLASSPATH=%CLASSPATH%;.;C:\Users\86159\Desktop\mujava\mujava.jar;C:\Users\86159\Desktop\mujava\openjava.jar;D:\Java\jdk1.8.0_102\lib\tools.jar;
cd C:\Users\86159\Desktop\mujava
```

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

2. RunTest.bat

```
set ClASSPATH=C:\Users\86159\Desktop\mujava\mujava.jar;C:\Users\86159\Desktop\mujava\openjava.jar;D:\Java\jdk1.8.0_102\lib\tools.jar;C:\Users\86159\Desktop\mujava\junit-4.12.jar;C:\Users\86159\Desktop\mujava\hamcrest-all-1.3.jar;
cd C:\Users\86159\Desktop\mujava
java mujava.gui.RunTestMain > output.txt
```

3.BackPackTest.java

```
import static org.junit.Assert.assertEquals;

import org.junit.Before;
import org.junit.Test;

public class BackPackTest {

	BackPack bp;

	@Before
	public void setUp() throws Exception {
	  bp = new BackPack();
	}

	@Test
	public void test() {
	  int m = 10;
	  int n = 3;
	  int w[] = { 3, 4, 5 };
	  int p[] = { 4, 5, 6 };
	  int c[][] = bp.BackPack_Solution(m, n, w, p);
	  int expected[][] = { { 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++) {
			System.out.println("i: " + i + "	j: " + j + "	" + expected[i - 1][j - 1] + " " + c[i][j]);
			assertEquals(expected[i - 1][j - 1], c[i][j]);
		}
	  }

	}

}
```

4. BubbleSortTest.java

```
import static org.junit.Assert.assertEquals;

import java.util.Arrays;
```

```
import org.junit.Test;

public class BubbleSortTest {
    @Test
    public void test() {
        int arr[] = new int[] { 1, 6, 2, 2, 5 };
        int expected[] = new int[] { 1, 2, 2, 5, 6 };
        arr = BubbleSort.BubbleSort(arr);
        assertEquals(Arrays.toString(expected), Arrays.toString(arr));
    }
}
```

5. 实验心得体会

实验三和实验四实际上都是使用变异体的生成和测试工具来测试我们的测试样例是否足够强壮，发现更多的错误。在实验三中我们使用 mujava 工具来生成源程序的变异体，然后进一步又对变异体进行了测试，并且通过 GUI 界面既可以选择使用那些变异算子来进行变异体的生成，在 result 文件夹中查看自动生产的变异体，还可以直观的看到杀死率信息来进一步帮助我们判断是否需要提升补足测试样例。