****

**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**

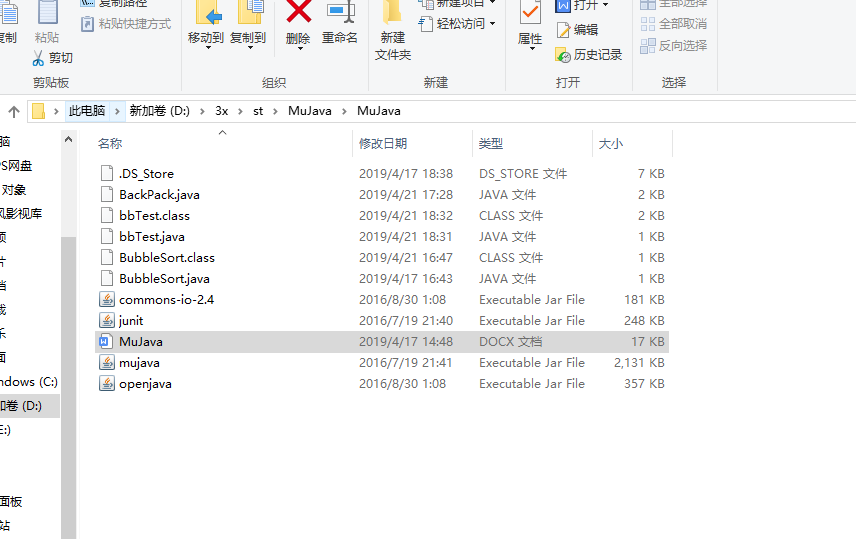
**年 级 16**

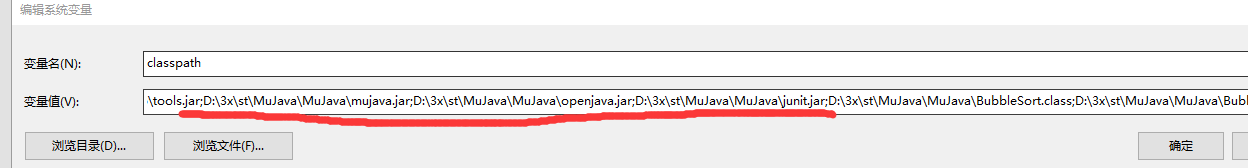
**姓 名 李沛霖**

**2019年 4 月 21 日**

# 软件测试技术第四次实验报告

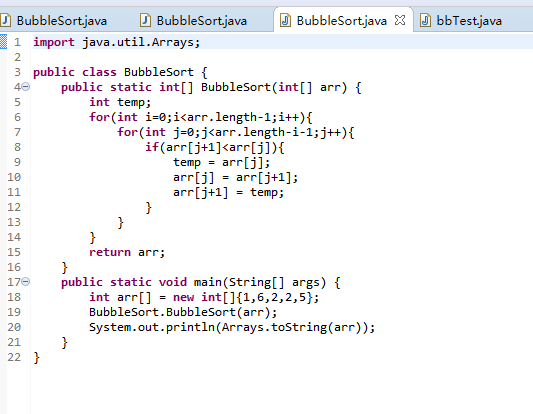
1. 需求分析（描述具体需求）
2. Install MuJava. The instruction of how to install and use Mujava can be seen in <https://cs.gmu.edu/~offutt/mujava/> .
3. Two small programs are given for your task. BubbleSort.java is an implementation of bubble sort algorithm and BackPack.java is a solution of 01 backpack problem. Try to generate Mutants of 2 given programs with MuJava.
4. Write testing sets for 2 programs with Junit, and run mutants on the test sets with MuJava.
5. 概要设计（简单描述设计思路，配合UML图）
6. 下载解压Mujava的安装包和jar包，并为jar包配置环境变量。



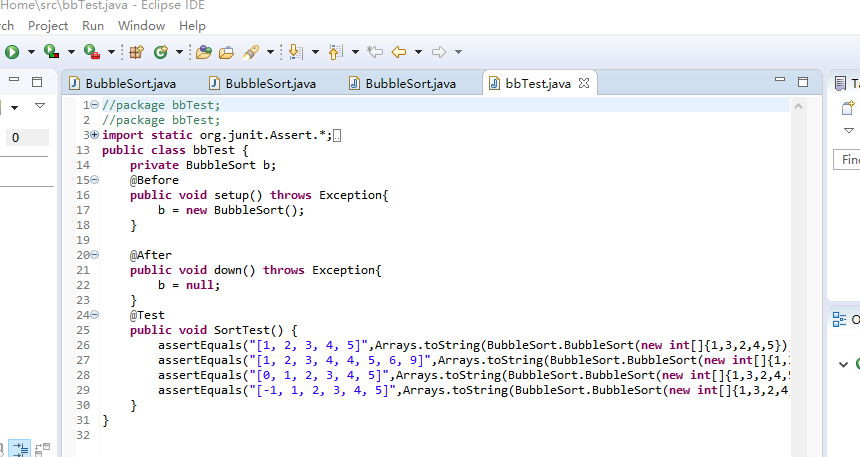


2.1

1. 首先打开并分析BubbleSort.java,并对其进行编译，产生class文件



1. 以测试为目标编写测试文件bbTest，编译通过即可。



1. 新建文件夹mujavaHome，完成相应目标文件的创建。

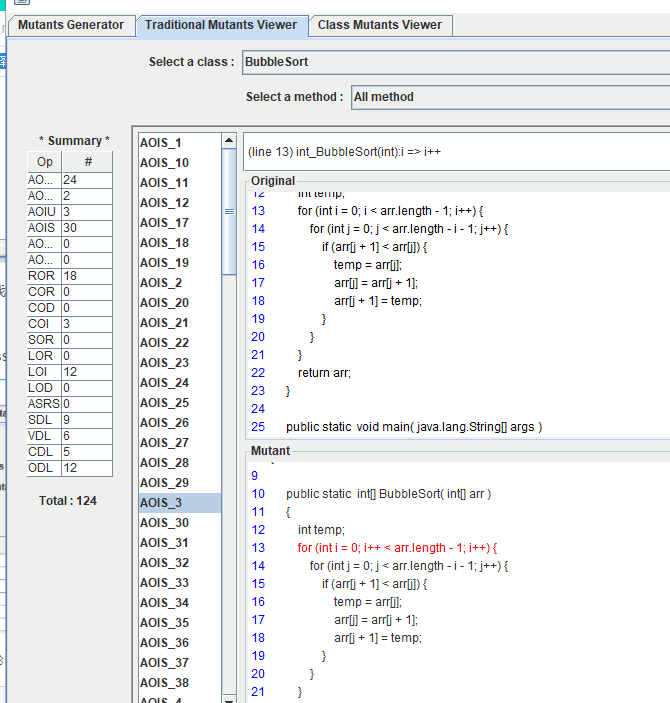


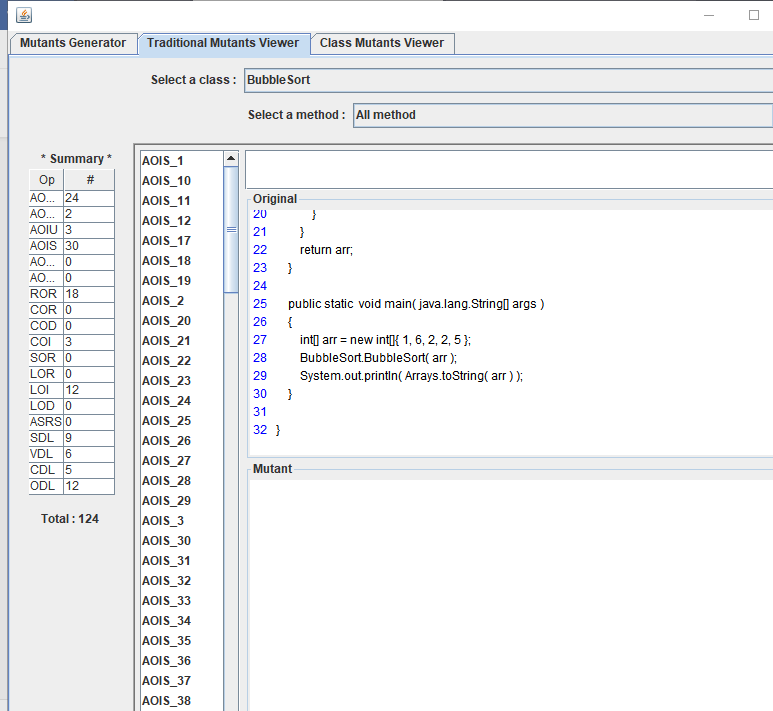
四个文件夹的用处为：

|  |  |
| --- | --- |
| Classes | 待测试源代码编译后的.class文件 |
| Result | 待测试源代码变异后产生的代码 |
| Src | 待测试的.java源文件 |
| Testset | Junit测试类.java源文件和编译后的.class文件 |

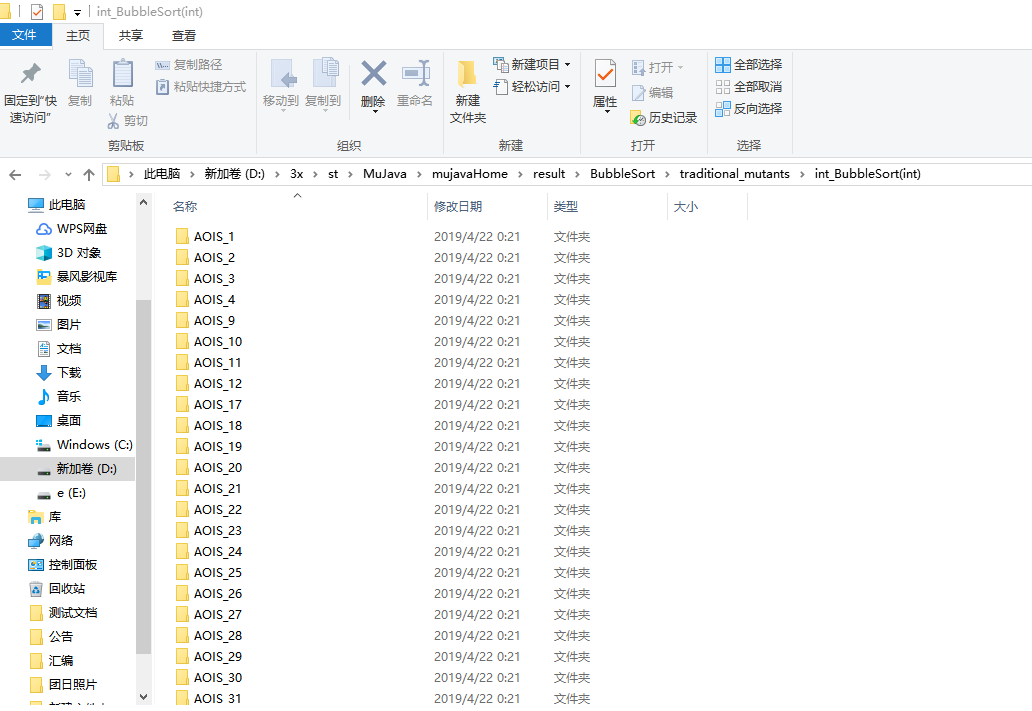
4.随后，将BubbleSort对应的java文件放入src目录，class文件放入classes目录中去。将bbTest对应的java和class文件放入Testset目录中去。

5.点击GenMutants，执行命令，出现变种执行界面，生成mutants文件。

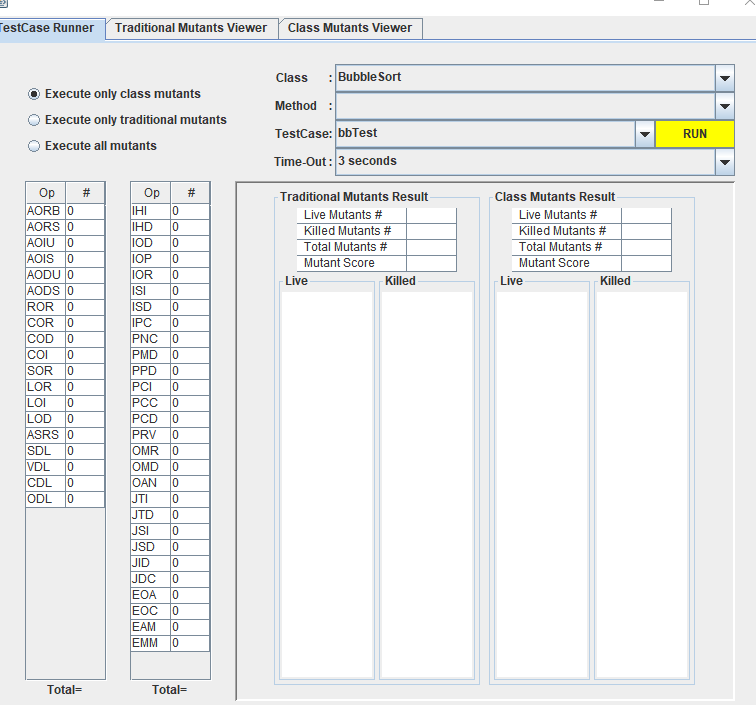


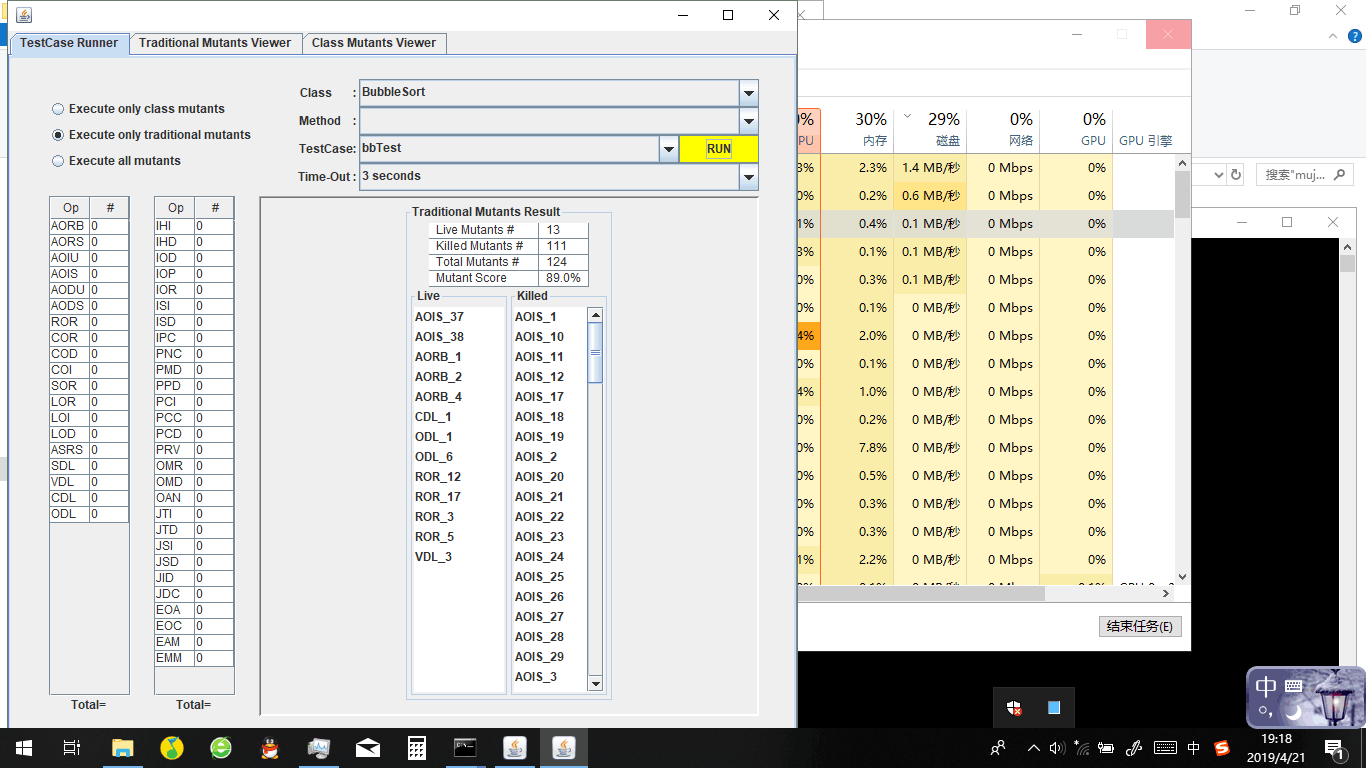


可以看到，生成的变种结果被放在result目录下。

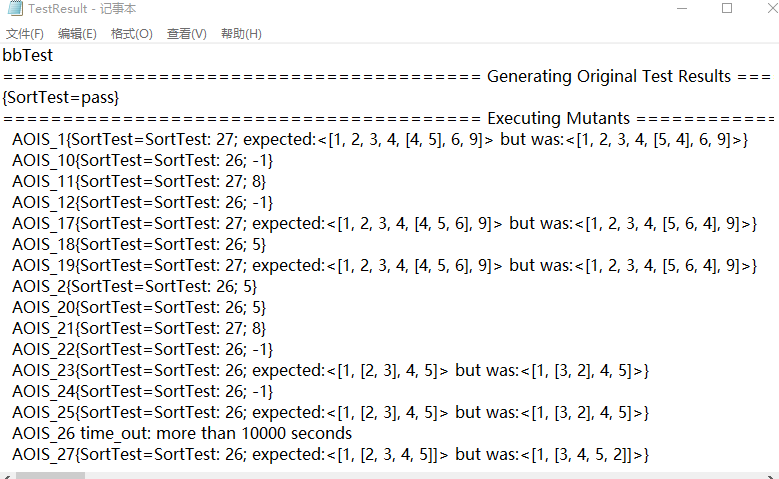


1. 同理，执行RunTest命令，生成针对变种类的测试，并记录结果数值。

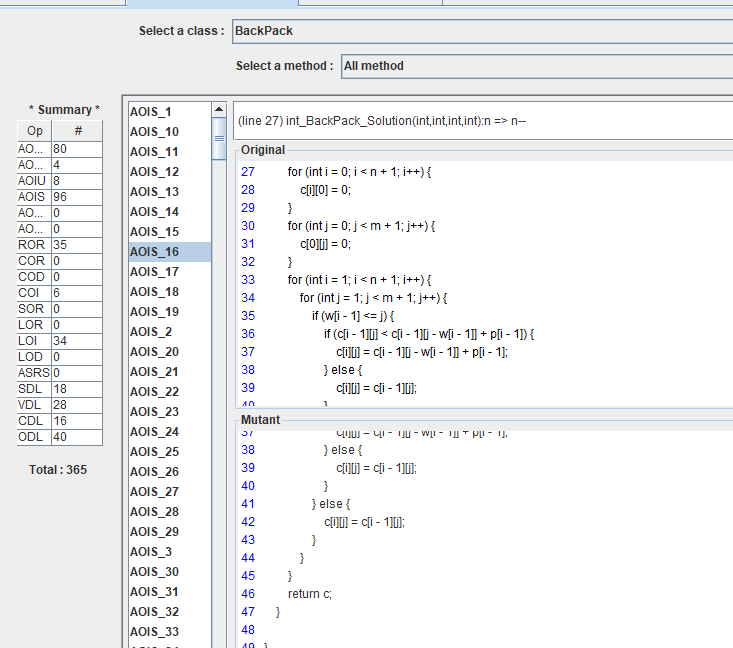


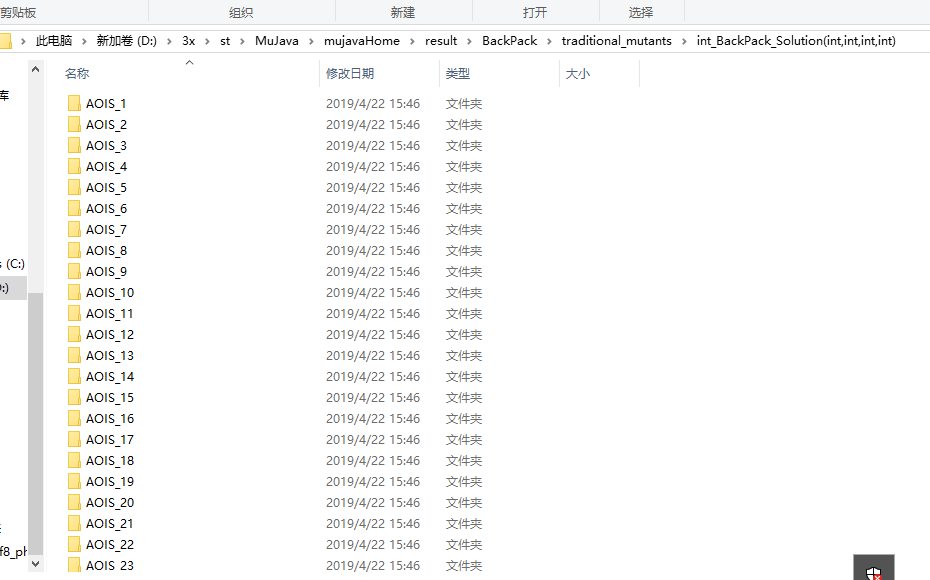


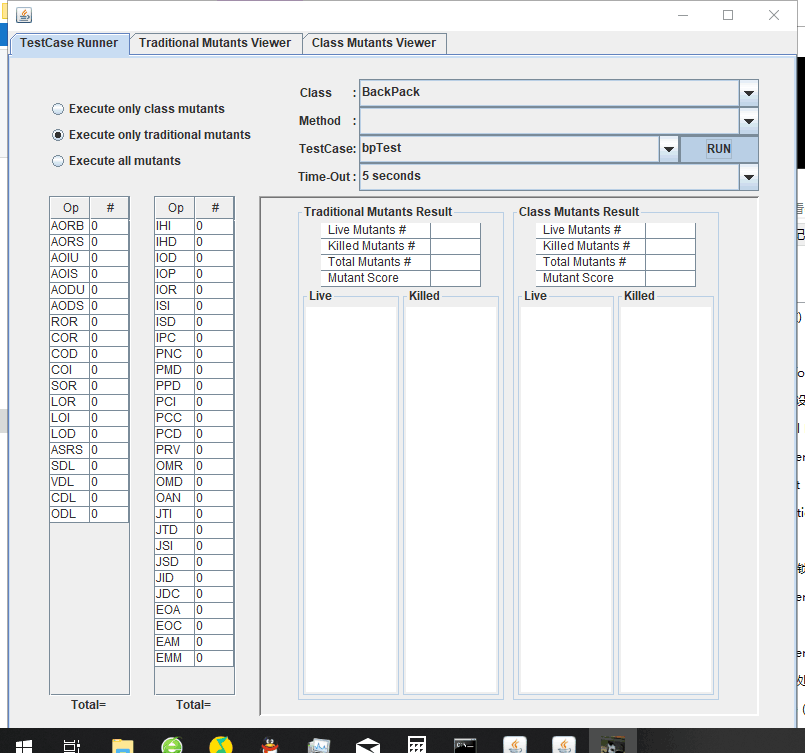
详细的测试记录保存在TestResult文档中

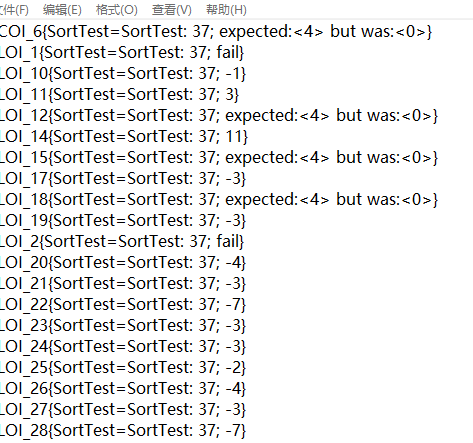


用同样的方式，对BackPack进行变异测试。结果如下：









1. 详细设计（详细描述具体如何实现，附代码及说明）

实验一（BubbleSort）：

**import** java.util.Arrays;

**public** **class** BubbleSort {

**public** **static** **int**[] BubbleSort(**int**[] arr) {

**int** temp;

**for**(**int** i=0;i<arr.length-1;i++){

**for**(**int** j=0;j<arr.length-i-1;j++){

**if**(arr[j+1]<arr[j]){

temp = arr[j];

arr[j] = arr[j+1];

arr[j+1] = temp;

}

}

}

**return** arr;

}

**public** **static** **void** main(String[] args) {

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

BubbleSort.BubbleSort(arr);

System.out.println(Arrays.toString(arr));

}

}

//package bbTest;

//package bbTest;

import static org.junit.Assert.\*;

import org.junit.After;

import org.junit.Before;

import org.junit.Test;

//import bubble.BubbleSort;

//import bubble.BubbleSort;

import java.util.Arrays;

public class bbTest {

private BubbleSort b;

@Before

public void setup() throws Exception{

b = new BubbleSort();

}

@After

public void down() throws Exception{

b = null;

}

@Test

public void SortTest() {

assertEquals("[1, 2, 3, 4, 5]",Arrays.toString(BubbleSort.BubbleSort(new int[]{1,3,2,4,5})));

assertEquals("[1, 2, 3, 4, 4, 5, 6, 9]",Arrays.toString(BubbleSort.BubbleSort(new int[]{1,3,2,4,5,6,4,9})));

assertEquals("[0, 1, 2, 3, 4, 5]",Arrays.toString(BubbleSort.BubbleSort(new int[]{1,3,2,4,5,0})));

assertEquals("[-1, 1, 2, 3, 4, 5]",Arrays.toString(BubbleSort.BubbleSort(new int[]{1,3,2,4,5,-1})));

}

}

实验二（BackPack）：

**public** **class** BackPack {

**public** **static** **void** main(String[] args) {

//最大容量为10

**int** m = 10;

**int** n = 3;

/\*

放入的三个物体重量分别为3， 4， 5；价值分别为4， 5， 6

\*/

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

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

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

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

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

System.out.print(c[i][j] + "\t");

**if** (j == m) {

System.out.println();

}

}

}

//printPack(c, w, m, n);

}

/\*\*

\* **@param** m 表示背包的最大容量

\* **@param** n 表示商品个数

\* **@param** w 表示商品重量数组

\* **@param** p 表示商品价值数组

\*/

**public** **static** **int**[][] BackPack\_Solution(**int** m, **int** n, **int**[] w, **int**[] p) {

//c[i][v]表示前i件物品恰放入一个重量为m的背包可以获得的最大价值

//递推式：c[i][m]=max{c[i-1][m-w[i]]+pi , c[i-1][m]}

**int** c[][] = **new** **int**[n + 1][m + 1];

**for** (**int** i = 0; i < n + 1; i++) {

c[i][0] = 0;

}

**for** (**int** j = 0; j < m + 1; j++) {

c[0][j] = 0;

}

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

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

// //当物品为i件重量为j时，如果第i件的重量(w[i-1])小于重量j时，c[i][j]为下列两种情况之一：

//(1)物品i不放入背包中，所以c[i][j]为c[i-1][j]的值

//(2)物品i放入背包中，则背包剩余重量为j-w[i-1],所以c[i][j]为c[i-1][j-w[i-1]]的值加上当前物品i的价值

**if** (w[i - 1] <= j) {

**if** (c[i - 1][j] < (c[i - 1][j - w[i - 1]] + p[i - 1])) {

c[i][j] = c[i - 1][j - w[i - 1]] + p[i - 1];

} **else** {

c[i][j] = c[i - 1][j];

}

} **else** {

c[i][j] = c[i - 1][j];

}

}

}

**return** c;

}

}

//package bpTest;

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

//import bubble.BubbleSort;

//import bubble.BubbleSort;

**import** java.util.Arrays;

**import** org.junit.After;

**import** org.junit.Before;

//import backPack.BackPack;

**import** org.junit.Test;

**public** **class** bpTest {

**private** BackPack b;

**int** m = 10;

**int** n =3;

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

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

**int** c[][] = {{0},{0,0,0,4,4,4,4,4,4,4,4},{0,0,0,4,5,5,5,9,9,9,9},{0,0,0,4,5,6,6,9,10,11,11}};

@Before

**public** **void** setup() **throws** Exception{

b = **new** BackPack();

}

@After

**public** **void** down() **throws** Exception{

b = **null**;

}

@Test

**public** **void** SortTest() {

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

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

assertEquals(c[i][j],BackPack.BackPack\_Solution(m,n,w,p)[i][j]);

}

}

}

}

1. 调试分析（在实验过程中遇到的问题以及如何解决）

1在一开始使用mujava时，相关的测试命令并不能直接使用，我经过查找资料发现需要添加系统环境变量。

2 编译产生class文件时无法构造关联，后来我将源文件加入系统环境变量成功编译。

1. 总结

成功完成任务，对java程序使用junit和mujava进行变异测试有了更加深入的了解。

仓库url：https://github.com/SSFDLware/software-testing.git