实验报告

课程名称：软件测试
实验名称：白盒测试
专业班级：软件工程 四班
学　　号：1611736
姓　　名：钟腾

2018 年　11 月　18 日

实验一

实验名称	白盒测试		
实验地点	泰达 5 区 105	实验时间	2018/11/13
实验目的和要求			

本次实验是基于 Junit 的白盒测试，要求找一份 50-100 行包含至少三个判断和两个循环的算法代码。搭建 Junit 环境，读懂算法代码，生成测试程序，并编写测试样例进行测试。

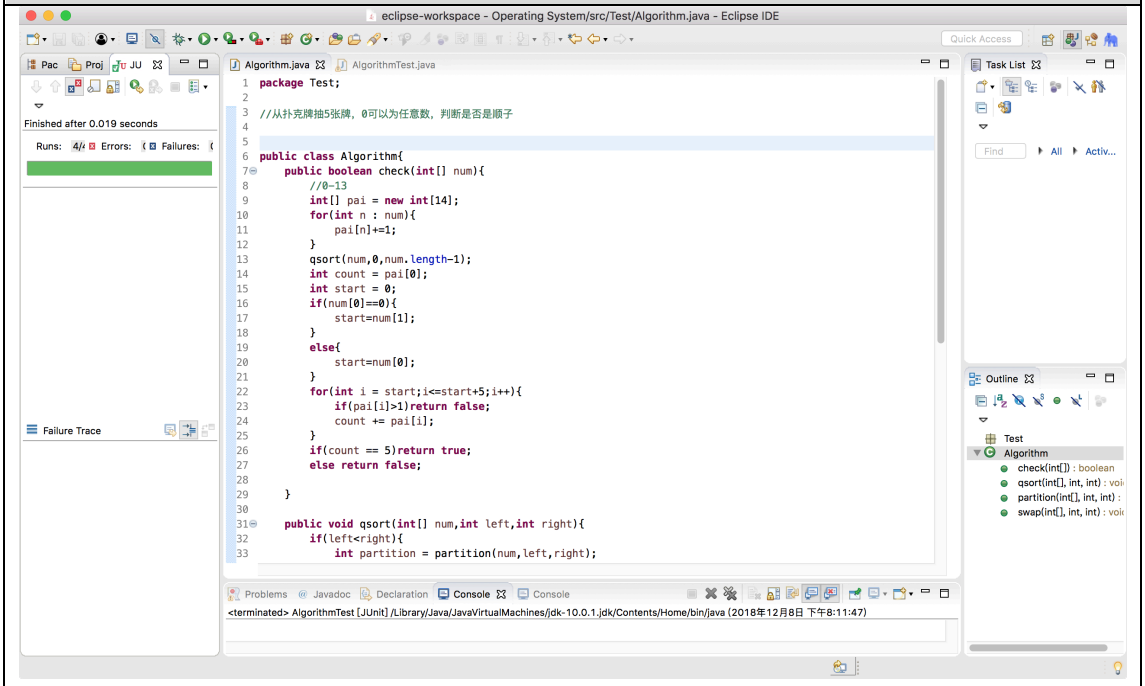
实验环境

Mac os 10.13.5

Junit 4.12.0

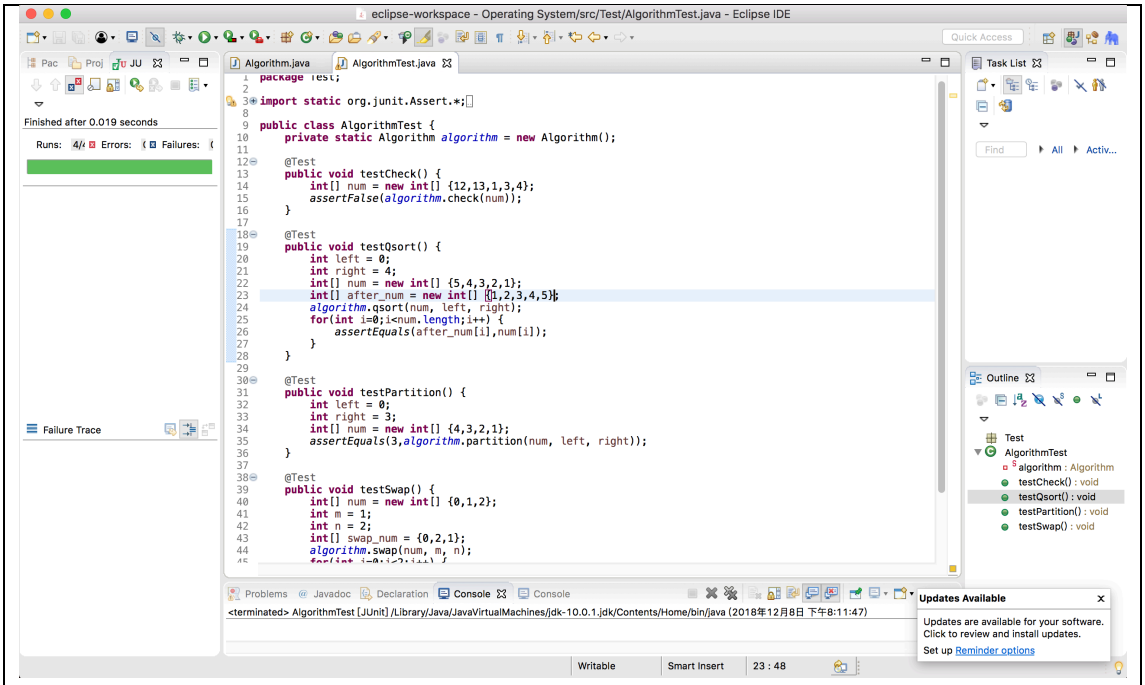
Eclipse 4.8.0

实验过程




```
package Test;

//从扑克牌抽5张牌，0可以为任意数，判断是否是顺子


public class Algorithm{
    public boolean check(int[] num){
        //0-13
        int[] pai = new int[14];
        for(int n : num){
            pai[n]+=1;
        }
        qsort(num,0,num.length-1);
        int count = pai[0];
        int start = 0;
        if(num[0]==0){
            start=num[1];
        }
        else{
            start=num[0];
        }
        for(int i = start;i<=start+5;i++){
            if(pai[i]>1)return false;
            count += pai[i];
        }
        if(count == 5)return true;
        else return false;

    }

    public void qsort(int[] num,int left,int right){
        if(left<right){
            int partition = partition(num,left,right);
```

1. 下载 junit 包，找到合适的代码，生成新的文件，将 junit 包放入引入包中，右键新建文件中放入的代码（图中为 Algorithm），生成其原始的测试代码(图中为 AlgorithmTest)。



2. 生成的测试代码会根据原代码的各个函数生成对应的测试函数，以 test 开头，接下来读懂源代码，根据源代码做好测试样例。在本例中，源代码为检测卡牌中抽五张牌是否为顺子的算法，其中用到了快排，共有四个函数，Check()检查牌是否为顺子，Qsort()为快排递归，Partition()检查数组左右大小，Swap()交换数组中的两个数，返回类型分别是 bool，void，int，void，以此生成四个测试函数
testCheck(),testQsort(),testPartition(),testSwap(),先创建源代码的类对象，
测试 Swap()函数时，构造数组 num = {0,1,2},m=1,n=2,以及推算出的结果数组 swap_num = {0,2,1} 调用 assertEquals()方法判断带入到函数后的 num 变化是否与推算出的数组一致，如果一致说明此代码正常，测试通过，进行下一阶段测试，以此类推，在判断布尔型时稍有不同，调用 assertTrue()和 assertFalse()发放判断值为对错。

测试用例：
testSwap:

测试用	num	m	n	预测结果	实际结

例					果
1	{7,6,5}	1	2	{7,5,6}	{7,5,6}
2	{0,1,2}	1	2	{0,2,1}	{0,2,1}
3	{1}	0	0	{1}	{1}

testPartition:

测试用例	num	left	right	预测结果	实际结果
1	{4,3,2,1}	0	3	3	3
2	{1,4,3,2,5}	0	3	0	0

testQsort:

测试用例	num	m	n	预测结果	实际结果
1	{5,4,3,2,1}	0	4	{1,2,3,4,5}	{1,2,3,4,5}
2	{0,2,2}	0	2	{0,2,2}	{0,2,2}
3	{1,6,0}	0	3	{0,1,6}	{0,1,6}

因为判断语句一个测试用例在递归中判断多次，因而不再列出。

testCheck:

测试用例	num	Num[0]=0	Count=5	预测结果	实际结果
1	{5,4,3,2,1}	假	真	真	真
2	{0,3,4,2,1}	真	真	真	真
3	{12,13,1,3,4}	假	假	假	假
4	{0,3,6,9,2}	真	假	假	假

在实验中我们可以得知，本次代码层层嵌套，在检验上层函数时，其实对下层函数代码都做了再次的测试。

测试时间：2018/11/13

覆盖率：98%

（ `if(pai[i]>1)return false;`无法测试到）

未发现缺陷

测试题目：**从扑克牌中随机抽 5 张，判断是不是顺子，大小王可以看成任意数字(看成 0)**

源代码和测试代码：

源代码：

```
package Test;

//从扑克牌抽 5 张牌，0 可以为任意数，判断是否是顺子


public class Algorithm{
    public boolean check(int[] num){
        //0-13
        int[] pai = new int[14];
        for(int n : num){
            pai[n]+=1;
        }
        qsort(num,0,num.length-1);
        int count = pai[0];
        int start = 0;
        if(num[0]==0){
            start=num[1];
        }
        else{
            start=num[0];
        }
        for(int i = start;i<=start+5;i++){
            if(pai[i]>1)return false;
            count += pai[i];
        }
        if(count == 5)return true;
        else return false;

    }

    public void qsort(int[] num,int left,int right){
        if(left<right){
```

```
            int partition = partition(num,left,right);
            qsort(num,left,partition-1);
            qsort(num,partition+1,right);
        }
    }

    public int partition(int[] num,int left,int right){
        int partition = num[left];
        while(left<right){
            while(left<right && num[right]>=partition){
                right--;
            }
            swap(num,left,right);
            while(left<right && num[left]<=partition){
                left++;
            }
            swap(num,left,right);
        }

        return left;
    }

    public void swap(int[] num,int m,int n){
        int temp = num[m];
        num[m]=num[n];
        num[n]=temp;
    }
}
```

测试代码：

```
package Test;

import static org.junit.Assert.*;

import java.util.Date;

import org.junit.Test;

public class AlgorithmTest {
    private static Algorithm algorithm = new Algorithm();

    @Test
    public void testCheck() {
        int[] num = new int[] {12,13,1,3,4};
        assertFalse(algorithm.check(num));
    }

    @Test
    public void testQsort() {
        int left = 0;
```

```
        int right = 4;
        int[] num = new int[] {5,4,3,2,1};
        int[] after_num = new int[] {1,2,3,4,5};
        algorithm.qsort(num, left, right);
        for(int i=0;i<num.length;i++) {
            assertEquals(after_num[i],num[i]);
        }
    }

    @Test
    public void testPartition() {
        int left = 0;
        int right = 3;
        int[] num = new int[] {4,3,2,1};
        assertEquals(3,algorithm.partition(num, left, right));
    }

    @Test
    public void testSwap() {
        int[] num = new int[] {0,1,2};
        int m = 1;
        int n = 2;
        int[] swap_num = {0,2,1};
        algorithm.swap(num, m, n);
        for(int i=0;i<2;i++) {
            assertEquals(num[i],swap_num[i]);
        }
    }

}
```

心得体会

通过本次实验掌握了白盒测试的基本方法，学会了 junit 的使用，并实践了测试样例的编写和使用，对学习新知识感到兴奋