

LAB-7

Name: Tank Nandani Jagdishbhai
Student ID:202001201

Section - A

Design the equivalence class test cases

Equivalence classes

EC1 - date ≥ 1 and date ≤ 31
EC2 - date < 1
EC3 - date > 31
EC4 - month ≥ 1 and month ≤ 12
EC5 - month < 1
EC6 - month > 12
EC7 - year ≥ 1900 and year ≤ 2015
EC8 - year < 1900
EC9 - year > 2015

The dates should not be invalid - 31-02-2020 is an invalid date since month 2 does not have 31 days.

12 - 13 - 2005 is an example of a range date.

Equivalent test cases

EC1 - day - 3, month - 3, year - 2011
output - 2-03-2011

EC2 - day - 0, month - 3, year - 2012
output - Invalid date

EC3 - day - 41, month - 3, year - 2012
output - Invalid date

EC4 - day - 1, month - 1, year - 1980
output - 31/12/1979

EC5 - day - 20, month - -3, year - 2012
output - Invalid date

EC6 - day - 20, month - 15, year - 2013
output - Invalid date

EC8 - day - 5, month - 6, year - 1899
output - Invalid date

EC9 - day - 4, month - 3, year - 2021
output - Invalid date

Valid dates - calculate previous dates

1. 14-12-2011
output - 13-12-2011

2. 1-1-2014
output - 31-12-2013

Invalid dates

1. 31-4-2016
2. 14-45-1899

Out-of-range dates

1. 15-4-1899
2. 15-5-2022

Boundary Value Analysis

1. Earliest date - 1-1-1900
2. Last possible date - 31-12-2015
3. leap year - 29-2-2000
4. invalid leap year date - 29-2-2001

5. previous of earliest date - 31-12-1899
6. a day after latest date - 1-1-2016

PROGRAMS P1

The function linear search searches for a value v in an array of integers a. If v appears in the array a, then the function returns the first index i, such that a[i] == v; otherwise, -1 is returned.

```
public class lab7_1 {
    public static int linearSearch(int v, int[] a) {
        int i = 0;
        while (i < a.length) {
            if (a[i] == v) {
                return i;
            }
            i++;
        }
        return -1;
    }
}
```

Equivalence Partitioning

V is present in an	Index v
V is not present	-1

Boundary Value Analysis

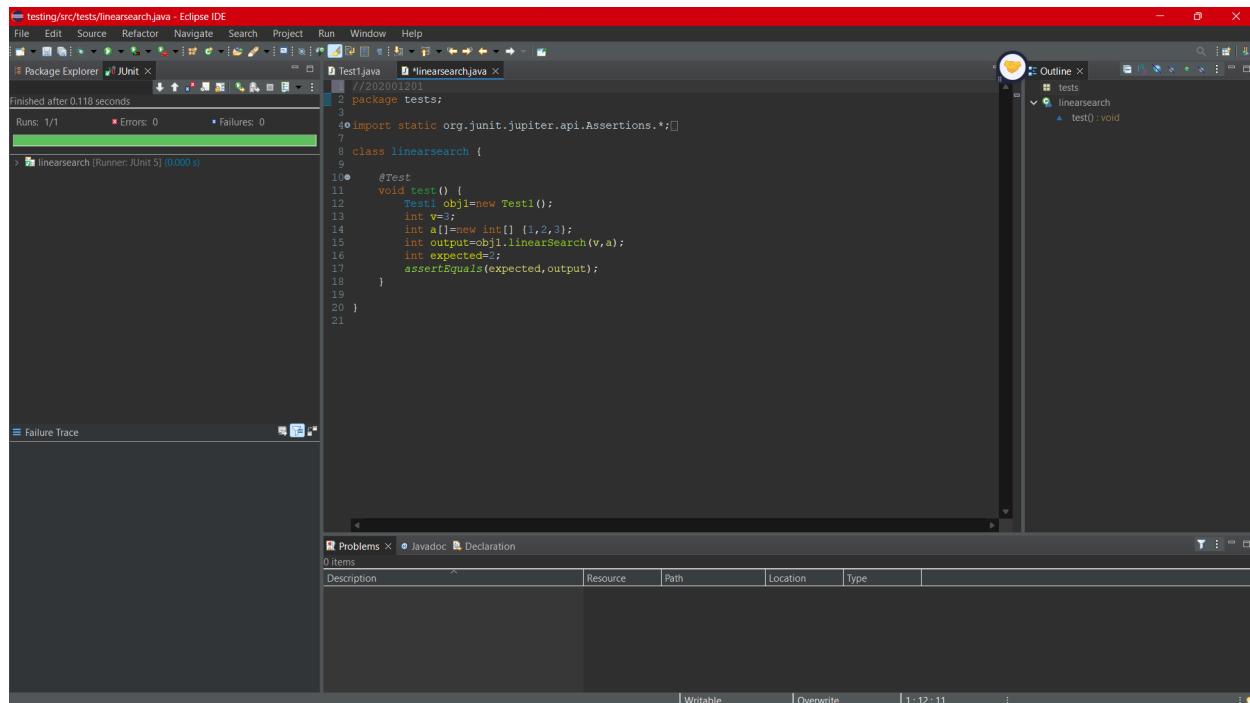
Empty array a	-1
V is present at the first index of a	0
V is present at the last index of a	-1

length of a	
V is not present in a	-1

TEST CASES WHICH WE TESTED

1. v = 3; a[] = {1,2,3}; expected = 2

Test case passed!



```

testing/src/tests/linearsearch.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer JUnit X
Testsuite tests
  Testsuite linearsearch
    Testcase test0
      Method linearSearch(v,a)
        Line 12: int output=obj.linearSearch(v,a);
        Line 15: int output=obj.linearSearch(v,a);
        Line 18: assertEquals(expected,output);
        Line 21: }

Outline X
  tests
    linearsearch
      test0: void

Failure Trace
Problems X Javadoc Declaration
0 items
Description Resource Path Location Type
Writable Overwrite 1:12:11

```

3. v = 3; a[] = {}; expected = -1;

Test case Passed!

testing/src/tests/linearsrch.java - Eclipse IDE

```

File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer JUnit x
Finished after 0.116 seconds
Runs: 1/1 Errors: 0 Failures: 0
linearsrch [Runner: JUnit 5] (0.019 s)

1 //202001201
2 package tests;
3
4 import static org.junit.jupiter.api.Assertions.*;
5
6 class linearsrch {
7
8     @Test
9     public void test() {
10         Test1 obj1=new Test1();
11         int v=3;
12         int a[]={1,2,3};
13         int output=obj1.linearSearch(v,a);
14         int expected=2;
15         assertEquals(expected,output);
16     }
17
18 }
19
20     @Test
21     public void test2() {
22         Test1 obj2=new Test1();
23         int v=3;
24         int a[]={1,2,4,5};
25         int output=obj2.linearSearch(v,a);
26         int expected=2;
27         assertEquals(expected,output);
28     }
29
30 }
31
32     @Test
33     public void test3() {
34         Test1 obj3=new Test1();
35         int v=3;
36         int a[]={1,2,3,4,5,6,7,8,9,10};
37         int output=obj3.linearSearch(v,a);
38         int expected=2;
39         assertEquals(expected,output);
40     }
41
42 }
43     @Test
44     public void test4() {
45         Test1 obj4=new Test1();
46         int v=10;
47         int a[]={10,20,30};
48         int output=obj4.linearSearch(v,a);
49         int expected=0;
50         assertEquals(expected,output);
51
52 }
53
54 }
55

```

Outline

- tests
 - linearsrch
 - test0: void
 - test2: void
 - test3: void
 - test4: void

Problems Javadoc Declaration Coverage

Element	Coverage	Defined Instructions	Unused Instructions	Total Instructions
> testing	96.4 %	107	4	111

4. $v = 10$; $a[] = \{10,20,30\}$, $expected = 0$;

Test case Passed!

testing/src/tests/linearsrch.java - Eclipse IDE

```

File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer JUnit x
Finished after 0.128 seconds
Runs: 1/1 Errors: 0 Failures: 0
linearsrch [Runner: JUnit 5] (0.021 s)

1 //202001201
2 package tests;
3
4 import static org.junit.jupiter.api.Assertions.*;
5
6 class linearsrch {
7
8     @Test
9     public void test0() {
10         Test1 obj1=new Test1();
11         int v=3;
12         int a[]={1,2,4,5};
13         int output=obj1.linearSearch(v,a);
14         int expected=2;
15         assertEquals(expected,output);
16     }
17
18 }
19
20     @Test
21     public void test2() {
22         Test1 obj2=new Test1();
23         int v=3;
24         int a[]={1,2,3};
25         int output=obj2.linearSearch(v,a);
26         int expected=2;
27         assertEquals(expected,output);
28     }
29
30 }
31
32     @Test
33     public void test3() {
34         Test1 obj3=new Test1();
35         int v=3;
36         int a[]={1,2,3,4,5,6,7,8,9,10};
37         int output=obj3.linearSearch(v,a);
38         int expected=2;
39         assertEquals(expected,output);
40     }
41
42 }
43     @Test
44     public void test4() {
45         Test1 obj4=new Test1();
46         int v=10;
47         int a[]={10,20,30};
48         int output=obj4.linearSearch(v,a);
49         int expected=0;
50         assertEquals(expected,output);
51
52 }
53
54 }
55

```

Outline

- tests
 - linearsrch
 - test0: void
 - test2: void
 - test3: void
 - test4: void

Problems Javadoc Declaration Coverage

Element	Coverage	Defined Instructions	Unused Instructions	Total Instructions
> testing	96.4 %	107	4	111

PROGRAM P2

The function counter returns the number of times a value v appears in an array of integers a.

```
public class lab7_2 {  
    public static int countItem(int v, int a[])  
    {  
        int count = 0;  
        for (int i = 0; i < a.length; i++)  
        {  
            if (a[i] == v)  
                count++;  
        }  
        return (count);  
    }  
}
```

Equivalence Partitioning

V is present in a	number of times v appears in
V is not present in a	0

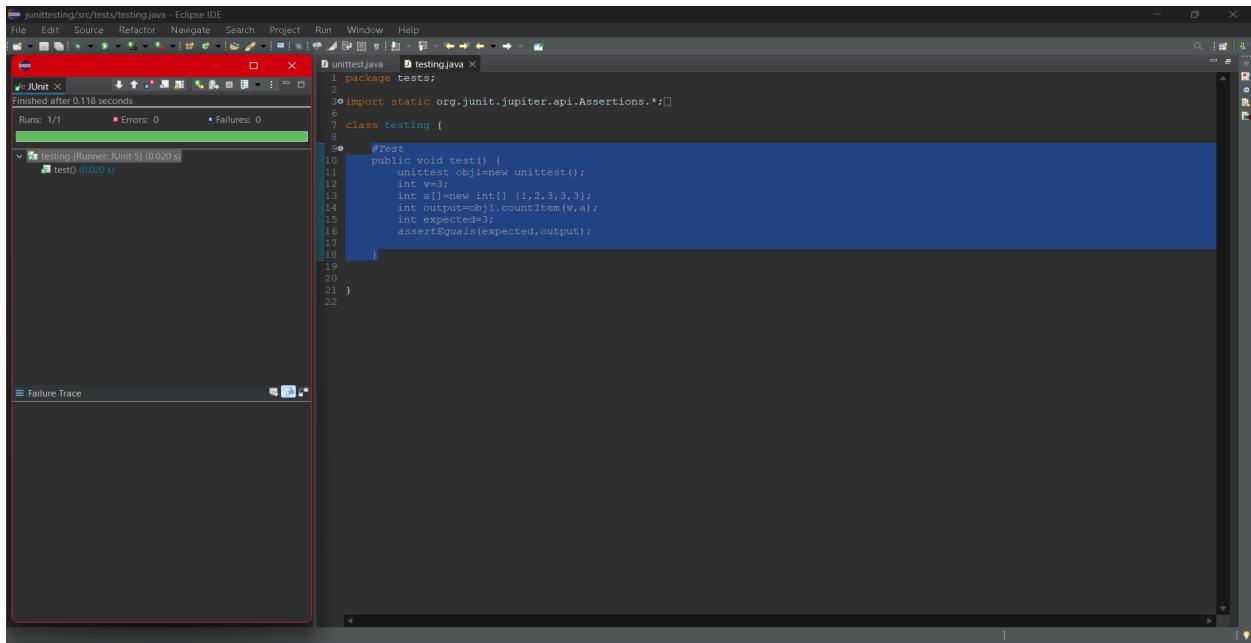
Boundary Value Analysis

Empty array a	0
V is present once in a	1
V is present multiple times in a	Number of times V appears in a
V is present at the first index of a	1
V is present at the last index of a	1
V is not present in a	0

TEST CASES

1. v = 3; a[] = {1,2,3,3,3}; expected = 3

Test case passed!

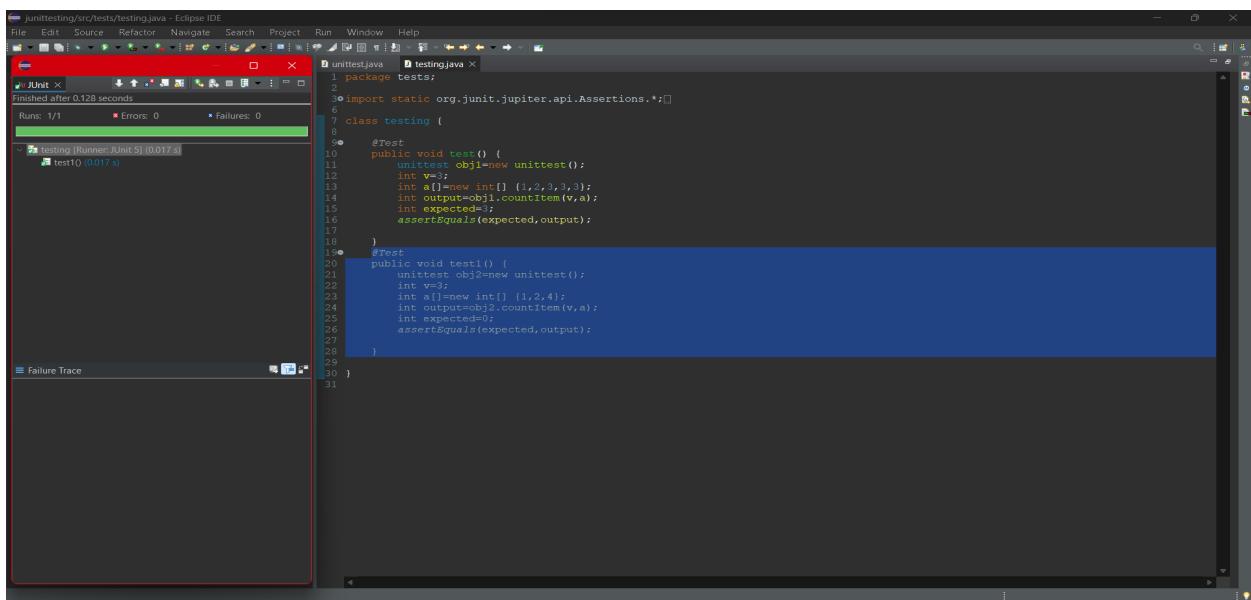


The screenshot shows the Eclipse IDE interface with the title "junittesting/src/tests/testing.java - Eclipse IDE". The JUnit view on the left displays "Runs: 1/1", "Errors: 0", and "Failures: 0", indicating a successful run. The code editor on the right contains the following Java code:

```
1 package tests;
2
3 import static org.junit.jupiter.api.Assertions.*;
4
5 class testing {
6
7     @Test
8     public void test() {
9         unittest obj1=new unittest();
10        int v=3;
11        int a[]={1,2,3,3,3};
12        int output=obj1.countItem(v,a);
13        int expected=3;
14        assertEquals(expected,output);
15    }
16
17 }
18
19
20
21 }
```

2. v = 3; a[] = {1,2,4}; expected = 0

Test case passed!

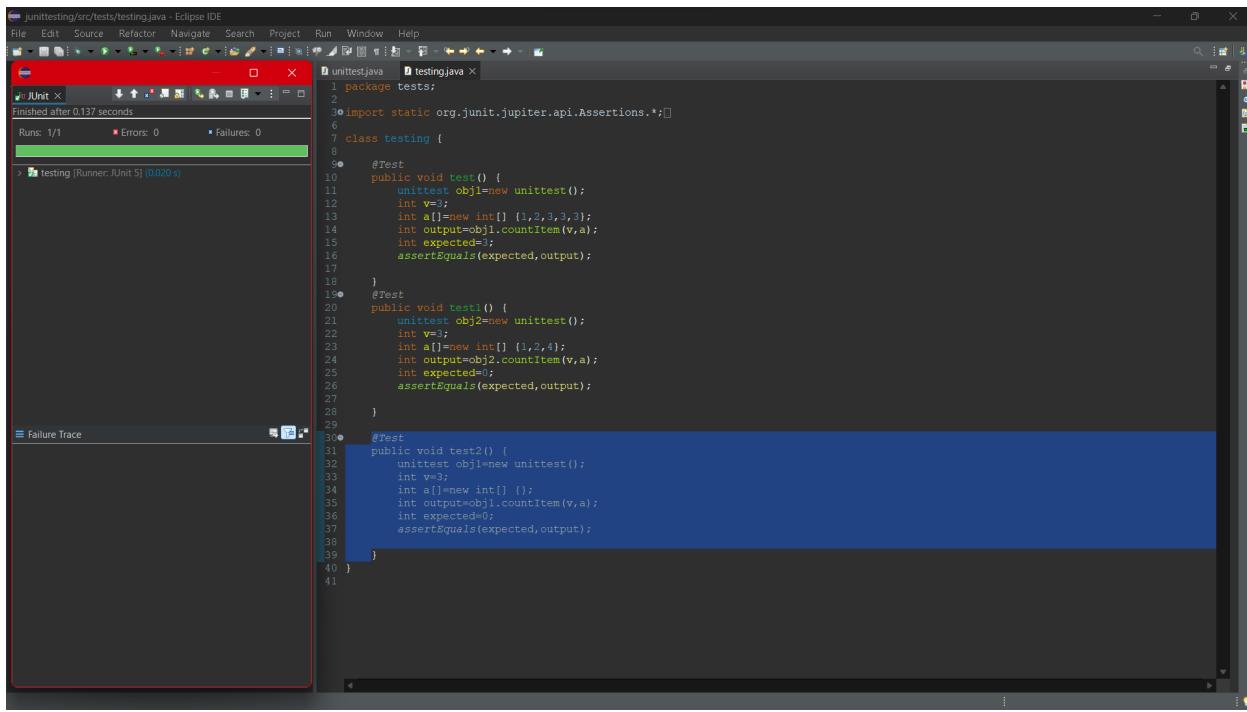


The screenshot shows the Eclipse IDE interface with the title "junittesting/src/tests/testing.java - Eclipse IDE". The JUnit view on the left displays "Runs: 1/1", "Errors: 0", and "Failures: 0", indicating a successful run. The code editor on the right contains the following Java code, showing two test methods:

```
1 package tests;
2
3 import static org.junit.jupiter.api.Assertions.*;
4
5 class testing {
6
7     @Test
8     public void test() {
9         unittest obj1=new unittest();
10        int v=3;
11        int a[]={1,2,3,3,3};
12        int output=obj1.countItem(v,a);
13        int expected=3;
14        assertEquals(expected,output);
15    }
16
17     @Test
18     public void test1() {
19         unittest obj2=new unittest();
20        int v=3;
21        int a[]={1,2,4};
22        int output=obj2.countItem(v,a);
23        int expected=0;
24        assertEquals(expected,output);
25    }
26
27 }
28
29
30 }
```

3. $v = 3$; $a[] = \{\}$; expected = 0

Test case passed!

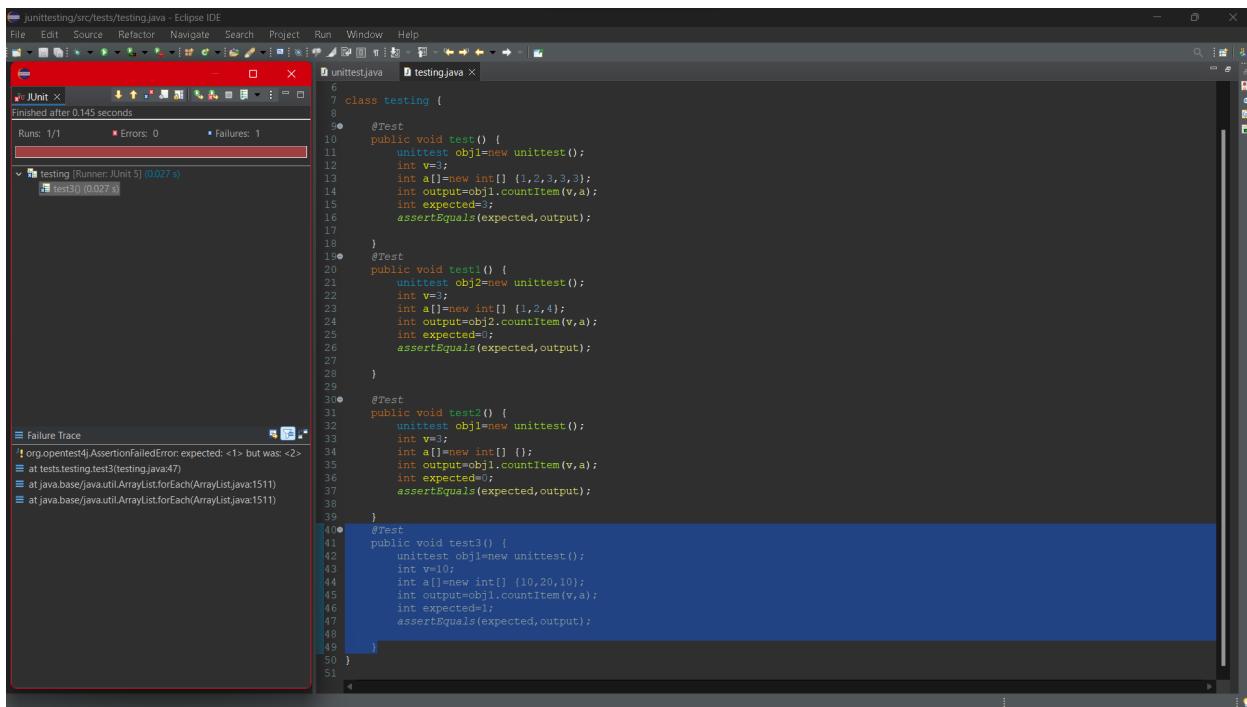


The screenshot shows the Eclipse IDE interface with the title "junittesting/src/tests/testing.java - Eclipse IDE". The JUnit view displays "Runs: 1/1", "Errors: 0", and "Failures: 0". The code editor shows a Java class named "testing" with three test methods: test0, test1, and test2. The test0 method passes, while test1 and test2 fail with a red error icon. The failure trace for test1 is visible in the bottom left.

```
1 package tests;
2 import static org.junit.jupiter.api.Assertions.*;
3
4 class testing {
5
6     @Test
7     public void test0() {
8         unittest obj1=new unittest();
9         int v=3;
10        int a[]={new int[] {1,2,3,3,3}};
11        int output=obj1.countitem(v,a);
12        int expected=0;
13        assertEquals(expected,output);
14    }
15
16    @Test
17    public void test1() {
18        unittest obj2=new unittest();
19        int v[]={new int[] {1,2,4}};
20        int a[]={new int[] {}};
21        int output=obj2.countitem(v,a);
22        int expected=0;
23        assertEquals(expected,output);
24    }
25
26    @Test
27    public void test2() {
28        unittest obj1=new unittest();
29        int v=3;
30        int a[]={new int[] {1,2,3,3,3}};
31        int output=obj1.countitem(v,a);
32        int expected=0;
33        assertEquals(expected,output);
34    }
35 }
```

4. $v = 10$; $a[] = \{10, 20, 10\}$; expected = 1

test case failed!



The screenshot shows the Eclipse IDE interface with the title "junittesting/src/tests/testing.java - Eclipse IDE". The JUnit view displays "Runs: 1/1", "Errors: 0", and "Failures: 1". The code editor shows the same Java class "testing" with three test methods. The test3 method fails with a red error icon. The failure trace for test3 is visible in the bottom left, showing an assertion failure where the expected value is 1 and the actual value is 2.

```
1 package tests;
2 import static org.junit.jupiter.api.Assertions.*;
3
4 class testing {
5
6     @Test
7     public void test0() {
8         unittest obj1=new unittest();
9         int v=3;
10        int a[]={new int[] {1,2,3,3,3}};
11        int output=obj1.countitem(v,a);
12        int expected=0;
13        assertEquals(expected,output);
14    }
15
16    @Test
17    public void test1() {
18        unittest obj2=new unittest();
19        int v=3;
20        int a[]={new int[] {1,2,4}};
21        int output=obj2.countitem(v,a);
22        int expected=0;
23        assertEquals(expected,output);
24    }
25
26    @Test
27    public void test2() {
28        unittest obj1=new unittest();
29        int v=3;
30        int a[]={new int[] {}};
31        int output=obj1.countitem(v,a);
32        int expected=0;
33        assertEquals(expected,output);
34    }
35
36    @Test
37    public void test3() {
38        unittest obj1=new unittest();
39        int v=10;
40        int a[]={new int[] {10,20,10}};
41        int output=obj1.countitem(v,a);
42        int expected=1;
43        assertEquals(expected,output);
44    }
45 }
```

PROGRAM P3

The function binarySearch searches for a value v in an ordered array of integers a. If v appears in the array a, then the function returns an index i, such that a[i] == v; otherwise, -1 is returned. Assumption: the elements in the array a are sorted in non-decreasing order.

```
public class lab7_3 {  
    public static int binarySearch(int v, int a[])  
    {  
        int lo,mid,hi;  
        lo = 0;  
        hi = a.length-1;  
        while (lo <= hi)  
        {  
            mid = (lo+hi)/2;  
            if (v == a[mid])  
                return (mid);  
            else if (v < a[mid])  
                hi = mid-1;  
            else  
                lo = mid+1;  
        }  
        return(-1);  
    }  
}
```

Equivalence Partitioning

V is present in an array	Index of v
V is not present in an array	-1

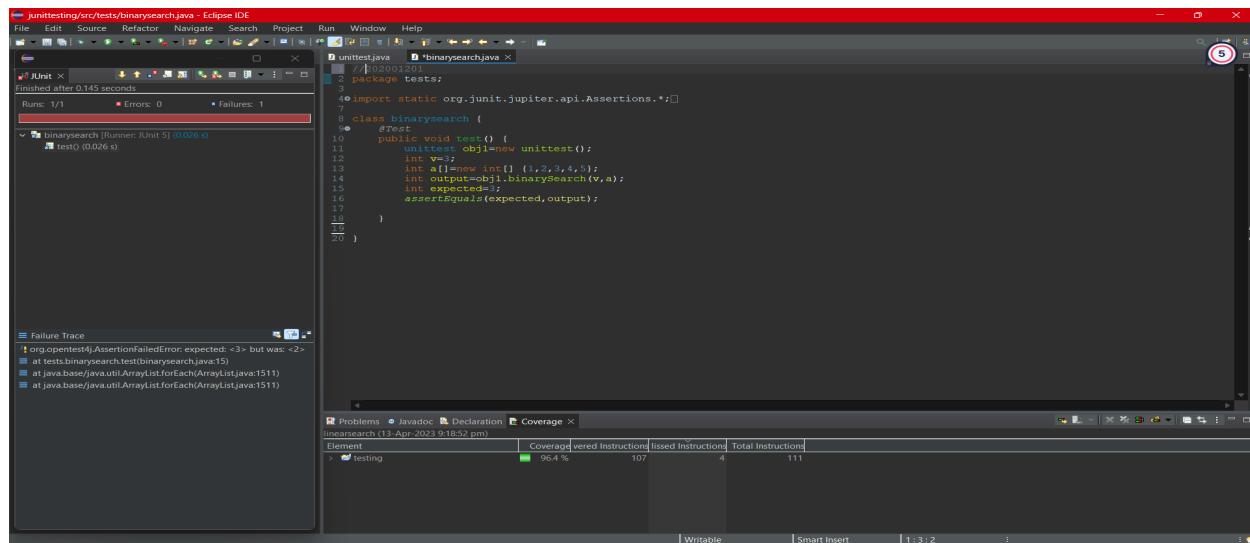
Boundary Value Analysis

Empty array a	-1
V is present at the first index of a	0
V is present at the last index of a length of a	-1
V is not present in a	-1

TEST CASES

1. v = 3; a[] = {1,2,3,4,5}; expected = 3

Test case failed!

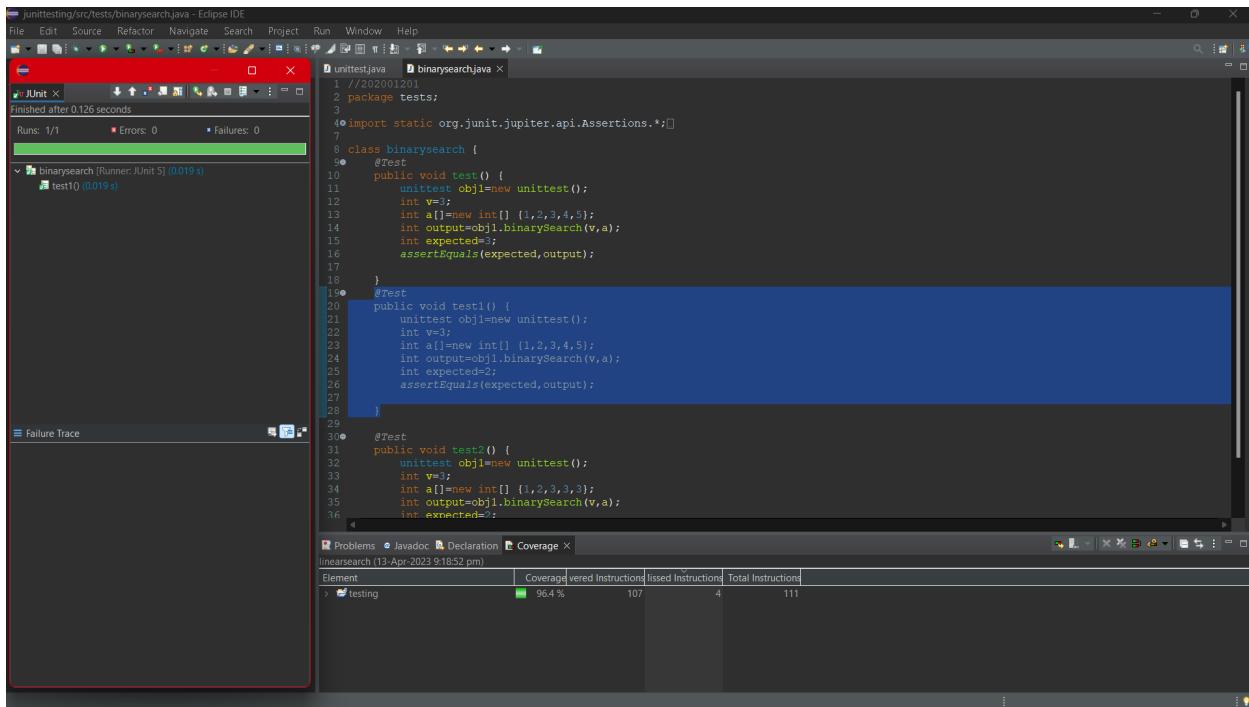


The screenshot shows the Eclipse IDE interface with the following details:

- Top Bar:** junittesting/src/tests/binarysearch.java - Eclipse IDE
- Toolbar:** File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, Help
- Left Sidebar:** JUnit (1 test), binarysearch [Runner: JUnit 5] (0.026 s), test() (0.026 s)
- Middle Area:** A code editor window titled "binarysearch.java" containing Java code for a binary search test. The code defines a class `binarysearch` with a single test method `test()`. The test creates a new instance of `binarysearch`, initializes an array `v` with values 1, 2, 3, 4, 5, and calls the `binarySearch` method on the object with `v` and `v` as arguments. It then asserts that the result equals 3.
- Bottom Area:** A Coverage tab showing coverage statistics for the test. The coverage is 96.4% with 107 covered instructions, 4 missed instructions, and a total of 111 instructions. A failure trace is also visible in the bottom left.

2. v = 3; a[] = {1,2,3,4,5}; expected = 2

Test case passed!

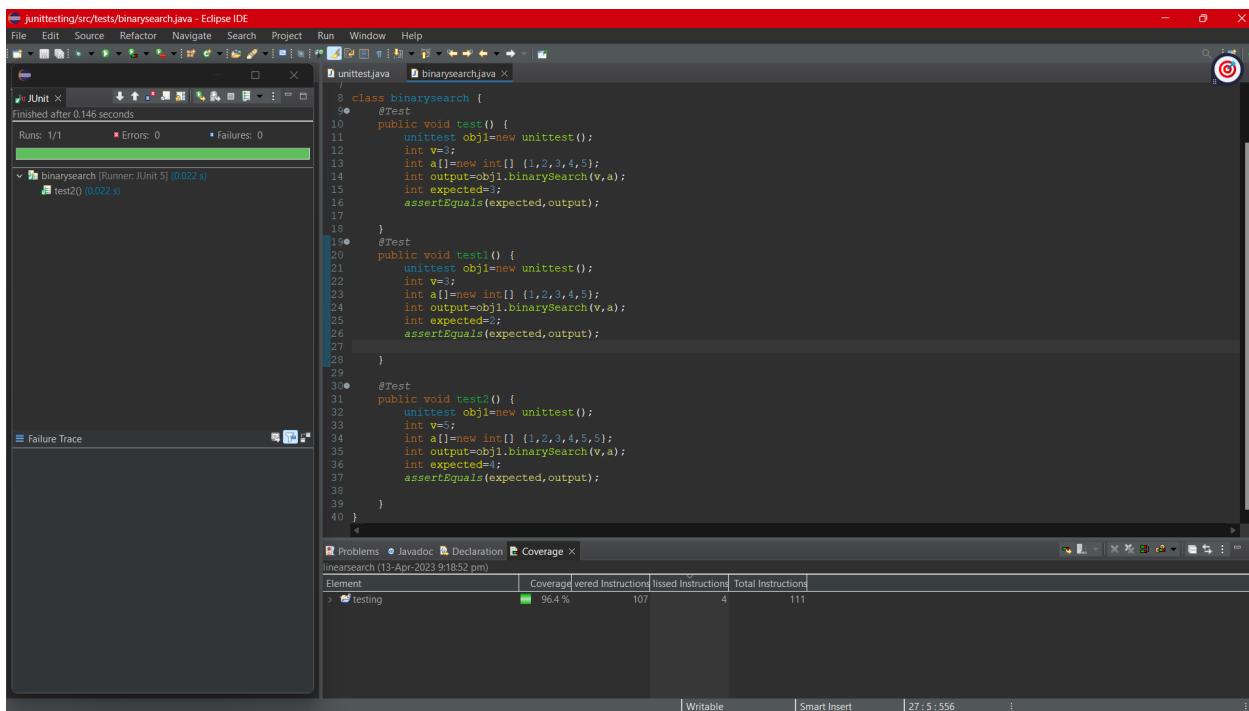


The screenshot shows the Eclipse IDE interface with the following details:

- Project Bar:** junittesting/src/tests/binarysearch.java - Eclipse IDE
- Toolbar:** File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, Help
- Left Panel:** JUnit view showing "Finished after 0.126 seconds" with 1/1 runs, 0 errors, and 0 failures.
- Code Editor:** A Java file named `binarysearch.java` containing test cases for a binary search algorithm. The code includes imports for `org.junit.jupiter.api.Assertions`, class definitions for `binarysearch`, and several `@Test` annotated methods like `test()`, `test1()`, `test2()`, and `test3()`.
- Bottom Panel:** Coverage view showing coverage statistics for the `binarysearch` class. The coverage is 96.4% with 107 covered instructions, 4 missed instructions, and a total of 111 instructions.

3. $v = 5$; $a[] = \{1,2,3,4,5,5\}$; expected = 4

Test case passed!



The screenshot shows the Eclipse IDE interface with the following details:

- Project Bar:** junittesting/src/tests/binarysearch.java - Eclipse IDE
- Toolbar:** File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, Help
- Left Panel:** JUnit view showing "Finished after 0.146 seconds" with 1/1 runs, 0 errors, and 0 failures.
- Code Editor:** A Java file named `binarysearch.java` containing test cases for a binary search algorithm. The code includes imports for `org.junit.jupiter.api.Assertions`, class definitions for `binarysearch`, and several `@Test` annotated methods like `test()`, `test1()`, `test2()`, and `test3()`.
- Bottom Panel:** Coverage view showing coverage statistics for the `binarysearch` class. The coverage is 96.4% with 107 covered instructions, 4 missed instructions, and a total of 111 instructions.

4. v = 5; a[] = {1,2,3}; expected = -1

Test case passed!

The screenshot shows an IDE interface with several windows open. The top menu bar includes 'Edit', 'Source', 'Refactor', 'Navigate', 'Search', 'Project', 'Run', 'Window', and 'Help'. On the left, there's a 'JUnit' window showing 'finished after 0.143 seconds', 'Runs: 1/1', 'Errors: 0', and 'Failures: 0'. Below it is a 'binarysearch [Runner: JUnit 5] (0.026 s)' window. The main editor area contains Java code for unit tests:

```
17
18
19● @Test
20 public void test1() {
21     unittest obj1=new unittest();
22     int v=3;
23     int a[]={1,2,3,4,5};
24     int output=obj1.binarySearch(v,a);
25     int expected=2;
26     assertEquals(expected,output);
27 }
28
29
30● @Test
31 public void test2() {
32     unittest obj1=new unittest();
33     int v=5;
34     int a[]={1,2,3,4,5,5};
35     int output=obj1.binarySearch(v,a);
36     int expected=4;
37     assertEquals(expected,output);
38 }
39
40● @Test
41 public void test3() {
42     unittest obj1=new unittest();
43     int v=5;
44     int a[]={1,2,3};
45     int output=obj1.binarySearch(v,a);
46     int expected=-1;
47     assertEquals(expected,output);
48 }
49 }
50 }
```

At the bottom, a 'Coverage' tab is selected in the 'Problems' tab bar, showing coverage statistics for the 'testing' element:

Element	Coverage	Covered Instructions	Missed Instructions	Total Instructions
: testing	96.4 %	107	4	111

5. v = 5; a[] = {1,2,3}; expected = 1

Test case failed!

```
junittesting/src/tests/binarysearch.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
JUnit X
Finished after 0.153 seconds
Runs: 1/1 Errors: 0 Failures: 1
binarysearch (Runner: JUnit 3) (0.027 s)
test1(0.027 s)

Failure Trace
! org.opentest4j.AssertionFailedError: expected: <1> but was: <1>
  at tests.binarysearch.test4(binarysearch.java:7)
  at java.base/java.util.ArrayList.forEach(ArrayList.java:151)
  at java.base/java.util.ArrayList.forEach(ArrayList.java:151)

 junittesting/src/tests/binarysearch.java
28
29
30
31     @Test
32     public void test2() {
33         unittest obj=new unittest();
34         int v=5;
35         int a[]={1,2,3,4,5,5};
36         int output=obj.binarySearch(v,a);
37         int expected=4;
38         assertEquals(expected,output);
39     }
40     @Test
41     public void test3() {
42         unittest obj=new unittest();
43         int v=5;
44         int a[]={1,2,3};
45         int output=obj.binarySearch(v,a);
46         int expected=-1;
47         assertEquals(expected,output);
48     }
49     @Test
50     public void test4() {
51         unittest obj=new unittest();
52         int v=5;
53         int a[]={1,2,3};
54         int output=obj.binarySearch(v,a);
55         int expected=1;
56         assertEquals(expected,output);
57     }
58 }
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111

Problems JavaDoc Declaration Coverage
binarysearch (13-Apr-2023 9:18:52 pm)
Element Coverage Used Instructions Missed Instructions Total Instructions
> testing 96.4 % 107 4 111
```

PROGRAM P4

- The following problem has been adapted from The Art of Software Testing, by G. Myers (1979). The function triangle takes three integer parameters that are interpreted as the lengths of the sides of a triangle. It returns whether the triangle is equilateral (three lengths equal), isosceles (two lengths equal),scalene (no lengths equal), or invalid (impossible lengths).

```
public class lab7_4 {  
    final int EQUILATERAL = 0;  
    final int ISOSCELES = 1;  
    final int SCALENE = 2;  
    final int INVALID = 3;  
    public int triangle(int a, int b, int c)  
    { if (a >= b+c || b >= a+c || c >= a+b)  
        return(INVALID);  
    if (a == b && b == c)  
        return(EQUILATERAL);  
    if (a == b || a == c || b == c)  
        return(ISOSCELES);  
    return(SCALENE);  
}  
}
```

Equivalence Partitioning

Invalid triangle($a+b \leq c$)	INVALID
Valid equilateral triangle($a=b=c$)	EQUILATERAL
Valid isosceles triangle($a=b < c$)	ISOSCELES
Valid scalene triangle($a < b < c$)	SCALENE

Boundary Value Analysis

Invalid triangle ($a+b \leq c$)	INVALID
Invalid triangle($a+c < b$)	INVALID
Invalid triangle($b+c < a$)	INVALID
Valid equilateral triangle ($a=b=c$)	EQUILATRAL
Valid isosceles triangle ($a=B<C$)	ISOSCELES
Valid isosceles triangle ($a=c < b$)	ISOSCELES
Valid isosceles triangle ($b=c < a$)	ISOSCELES
Valid scalene triable ($a < b < c$)	SCALENE

TEST CASES

1. $a = 3; b=3; c = 3$; expected = 0 (equilateral)

Test case passed!

The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** Lab-7 - Lab7/src/lab7_4Test.java - Eclipse IDE
- Toolbar:** File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, Help
- Package Explorer:** Shows files like lab7_1Test.java, lab7_2.java, lab7_2Test.java, lab7_3.java, lab7_3Test.java, lab7_4.java, and lab7_4Test.java.
- Outline View:** Shows the class structure of lab7_4Test with a single method test().
- Code Editor:** Displays the Java code for the test class:

```
7. * @test
8. * public void test() {
9. *     lab7_4 ll=new lab7_4();
10. *     int a=3;
11. *     int b=3;
12. *     int c=3;
13. *     int output=ll.triangle(a,b,c);
14. *     int expected=0;
15. *     assertEquals(expected,output);
16. * }
17. *
18. */
19. *
```
- Run View:** Shows "Runs: 1/1" with "Errors: 0" and "Failures: 0".
- Problems View:** Shows 0 errors, 11 warnings, and 0 others. A detailed list of warnings is shown in the table below.
- Bottom Status Bar:** Shows "Writable" and "Smart Insert" status.

Description	Resource	Path	Location	Type
Warnings (11 items)				
↳ The static method binarySearch(int[], int) from the type Arrays	/Lab7/src		line 12	Java Problem
↳ The static method binarySearch(int, int) from the type Arrays	/Lab7/src		line 22	Java Problem
↳ The static method binarySearch(int, int) from the type Arrays	/Lab7/src		line 32	Java Problem
↳ The static method binarySearch(int, int) from the type Arrays	/Lab7/src		line 43	Java Problem
↳ The static method count(int, int) from the type Arrays	/Lab7/src		line 12	Java Problem
↳ The static method copy(int[], int, int[], int) from the type Arrays	/Lab7/src		line 37	Java Problem
↳ The static method copy(int[], int) from the type Arrays	/Lab7/src		line 47	Java Problem
↳ The static method fill(int[], int) from the type Arrays	/Lab7/src		line 57	Java Problem
↳ The static method fill(int[], int, int) from the type Arrays	/Lab7/src		line 67	Java Problem
↳ The static method swap(int, int) from the type Arrays	/Lab7/src		line 77	Java Problem

2. a = 1, b= 2, c= 3; expected = 0

Test case failed!

```
Lab-7 - Lab7/src/lab7_4Test.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer JUnit X
finished after 0.023 seconds
Runs: 2/2 Errors: 0 Failures: 1
lab7_1Test.java lab7_2.java lab7_2Test.java lab7_3.java lab7_3Test.java lab7_4.java lab7_4Test.java
Outline X
lab7_4Test
  test0 : void
  test1 : void
  test10 : void
Failure Trace
java.lang.AssertionError: expected:<0> but was:<1>
at lab7_4Test.test1(lab7_4Test.java:26)
Problems X Javadoc Declaration
0 errors, 11 warnings, 0 others
Description Resource Path Location Type
Warnings (11 items)
  The static method binarySearch(int, int[]) from the lab7_3Test.java /Lab7/src line 12 Java Problem
  The static method binarySearch(int, int[]) from the lab7_3Test.java /Lab7/src line 22 Java Problem
  The static method binarySearch(int, int[]) from the lab7_3Test.java /Lab7/src line 32 Java Problem
  The static method binarySearch(int, int[]) from the lab7_3Test.java /Lab7/src line 43 Java Problem
  The static method binarySearch(int, int[]) from the lab7_3Test.java /Lab7/src line 12 Java Problem
  The static method countItem(int, int[]) from the type lab7_2Test.java /Lab7/src line 23 Java Problem
  The static method countItem(int, int[]) from the type lab7_2Test.java /Lab7/src line 12 Java Problem
  The static method countItem(int, int[]) from the type lab7_2Test.java /Lab7/src line 23 Java Problem
  The static method countItem(int, int[]) from the type lab7_2Test.java /Lab7/src line 12 Java Problem
  The static method countItem(int, int[]) from the type lab7_2Test.java /Lab7/src line 23 Java Problem
  The static method countItem(int, int[]) from the type lab7_2Test.java /Lab7/src line 12 Java Problem
```

3. a = -1, b = 2, c = 3, expected = 2

Test case failed!

```
Lab-7 - Lab7/src/lab7_4Test.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer JUnit X
finished after 0.025 seconds
Runs: 3/3 Errors: 0 Failures: 1
lab7_1Test.java lab7_2.java lab7_2Test.java lab7_3.java lab7_3Test.java lab7_4.java lab7_4Test.java
Outline X
lab7_4Test
  test0 : void
  test1 : void
  test2 : void
Failure Trace
java.lang.AssertionError: expected:<2> but was:<-1>
at lab7_4Test.test2(lab7_4Test.java:37)
Problems X Javadoc Declaration
0 errors, 11 warnings, 0 others
Description Resource Path Location Type
Warnings (11 items)
  The static method binarySearch(int, int[]) from the lab7_3Test.java /Lab7/src line 12 Java Problem
  The static method binarySearch(int, int[]) from the lab7_3Test.java /Lab7/src line 22 Java Problem
  The static method binarySearch(int, int[]) from the lab7_3Test.java /Lab7/src line 32 Java Problem
  The static method binarySearch(int, int[]) from the lab7_3Test.java /Lab7/src line 43 Java Problem
  The static method binarySearch(int, int[]) from the lab7_3Test.java /Lab7/src line 12 Java Problem
  The static method countItem(int, int[]) from the type lab7_2Test.java /Lab7/src line 23 Java Problem
  The static method countItem(int, int[]) from the type lab7_2Test.java /Lab7/src line 12 Java Problem
  The static method countItem(int, int[]) from the type lab7_2Test.java /Lab7/src line 23 Java Problem
  The static method countItem(int, int[]) from the type lab7_2Test.java /Lab7/src line 12 Java Problem
  The static method countItem(int, int[]) from the type lab7_2Test.java /Lab7/src line 23 Java Problem
  The static method countItem(int, int[]) from the type lab7_2Test.java /Lab7/src line 12 Java Problem
```

4. $a = 3, b = 2, c = 3$, expected = 1

Test case passed!

```
Lab-7 - Lab7/src/lab7_4Test.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer JUnit
lab7_1Test.java lab7_2.java lab7_2Test.java lab7_3.java lab7_3Test.java lab7_4.java lab7_4Test.java
Finished after 0.022 seconds
Runs: 4/4 Errors: 0 Failures: 0
@Test
public void test1() {
    lab7_4 ll=new lab7_4();
    int a=2;
    int b=2;
    int c=3;
    int output=ll.triangle(a,b,c);
    int expected=1;
    assertEquals(expected,output);
}
@Test
public void test2() {
    lab7_4 ll=new lab7_4();
    int a=-1;
    int b=2;
    int c=3;
    int output=ll.triangle(a,b,c);
    int expected=3;
    assertEquals(expected,output);
}
@Test
public void test3() {
    lab7_4 ll=new lab7_4();
    int a=1;
    int b=2;
    int c=3;
    int output=ll.triangle(a,b,c);
    int expected=1;
    assertEquals(expected,output);
}
Failure Trace
Problems X Javadoc Declaration
0 errors, 11 warnings, 0 others
Description Resource Path Location Type
Warnings (11 items)
The static method binarySearch(int, int[]) from the lab7_3Test.java /Lab7/src line 12 Java Problem
The static method binarySearch(int, int[]) from the lab7_3Test.java /Lab7/src line 22 Java Problem
The static method binarySearch(int, int[]) from the lab7_3Test.java /Lab7/src line 32 Java Problem
The static method binarySearch(int, int[]) from the lab7_3Test.java /Lab7/src line 43 Java Problem
The static method countOne(int, int[]) from the type lab7_2Test.java /Lab7/src line 12 Java Problem
The static method countOne(int, int[]) from the type lab7_2Test.java /Lab7/src line 22 Java Problem
The static method countOne(int, int[]) from the type lab7_2Test.java /Lab7/src line 32 Java Problem
The static method countOne(int, int[]) from the type lab7_2Test.java /Lab7/src line 43 Java Problem
The static method countOne(int, int[]) from the type lab7_2Test.java /Lab7/src line 53 Java Problem
The static method countOne(int, int[]) from the type lab7_2Test.java /Lab7/src line 63 Java Problem

```

4. $a = 4, b = 2, c = 3$, expected = 2

Test case passed!

```
Lab-7 - Lab7/src/lab7_4Test.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer JUnit
lab7_1Test.java lab7_2.java lab7_2Test.java lab7_3.java lab7_3Test.java lab7_4.java lab7_4Test.java
Finished after 0.02 seconds
Runs: 5/5 Errors: 0 Failures: 0
@Test
public void test1() {
    lab7_4 ll=new lab7_4();
    int a=2;
    int b=2;
    int c=3;
    int output=ll.triangle(a,b,c);
    int expected=1;
    assertEquals(expected,output);
}
@Test
public void test2() {
    lab7_4 ll=new lab7_4();
    int a=-1;
    int b=2;
    int c=3;
    int output=ll.triangle(a,b,c);
    int expected=3;
    assertEquals(expected,output);
}
@Test
public void test3() {
    lab7_4 ll=new lab7_4();
    int a=1;
    int b=2;
    int c=3;
    int output=ll.triangle(a,b,c);
    int expected=1;
    assertEquals(expected,output);
}
@Test
public void test4() {
    lab7_4 ll=new lab7_4();
    int a=4;
    int b=2;
    int c=3;
    int output=ll.triangle(a,b,c);
    int expected=2;
    assertEquals(expected,output);
}
Failure Trace
Problems X Javadoc Declaration
0 errors, 11 warnings, 0 others
Description Resource Path Location Type
Warnings (11 items)
The static method binarySearch(int, int[]) from the lab7_3Test.java /Lab7/src line 12 Java Problem
The static method binarySearch(int, int[]) from the lab7_3Test.java /Lab7/src line 22 Java Problem
The static method binarySearch(int, int[]) from the lab7_3Test.java /Lab7/src line 32 Java Problem
The static method binarySearch(int, int[]) from the lab7_3Test.java /Lab7/src line 43 Java Problem
The static method countOne(int, int[]) from the type lab7_2Test.java /Lab7/src line 12 Java Problem
The static method countOne(int, int[]) from the type lab7_2Test.java /Lab7/src line 22 Java Problem
The static method countOne(int, int[]) from the type lab7_2Test.java /Lab7/src line 32 Java Problem
The static method countOne(int, int[]) from the type lab7_2Test.java /Lab7/src line 43 Java Problem
The static method countOne(int, int[]) from the type lab7_2Test.java /Lab7/src line 53 Java Problem
The static method countOne(int, int[]) from the type lab7_2Test.java /Lab7/src line 63 Java Problem

```

PROGRAM P5

- The function prefix (String s1, String s2) returns whether or not the string s1 is a prefix of string s2 (you may assume that neither s1 nor s2 is null).

```
public static boolean prefix(String s1, String s2)
{
if (s1.length() > s2.length())
{
return false;
}
for (int i = 0; i < s1.length(); i++)
{
if (s1.charAt(i) != s2.charAt(i))
{
return false;
}
}
return true;
}
```

Equivalence Partitioning

Empty string s1 and s2	True	TRUE
Empty string s1 and non-empty s2		TRUE
Non-empty s1 is a prefix of non-empty		TRUE
Non-empty s1 is not a prefix of s2		FALSE
Non-empty s1 is longer than s2		FALSE

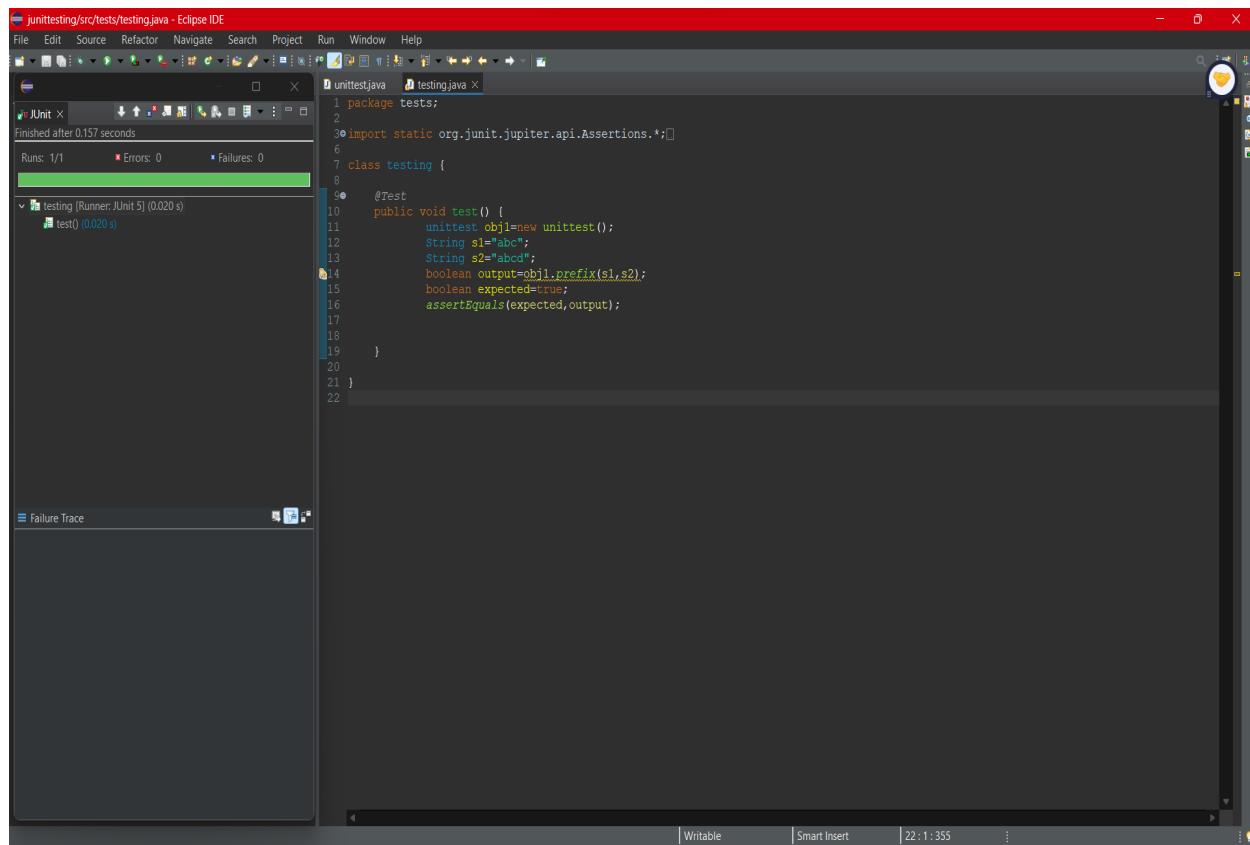
Boundary value analysis

Empty string s1 and s2	True
Empty string s1 and nonempty s2	True
S1 prefix of s2	True
S1 longer than s2	false

TEST CASES

- String s1 = "abc", s2 = "abcd"; expected = true

Test case passed!

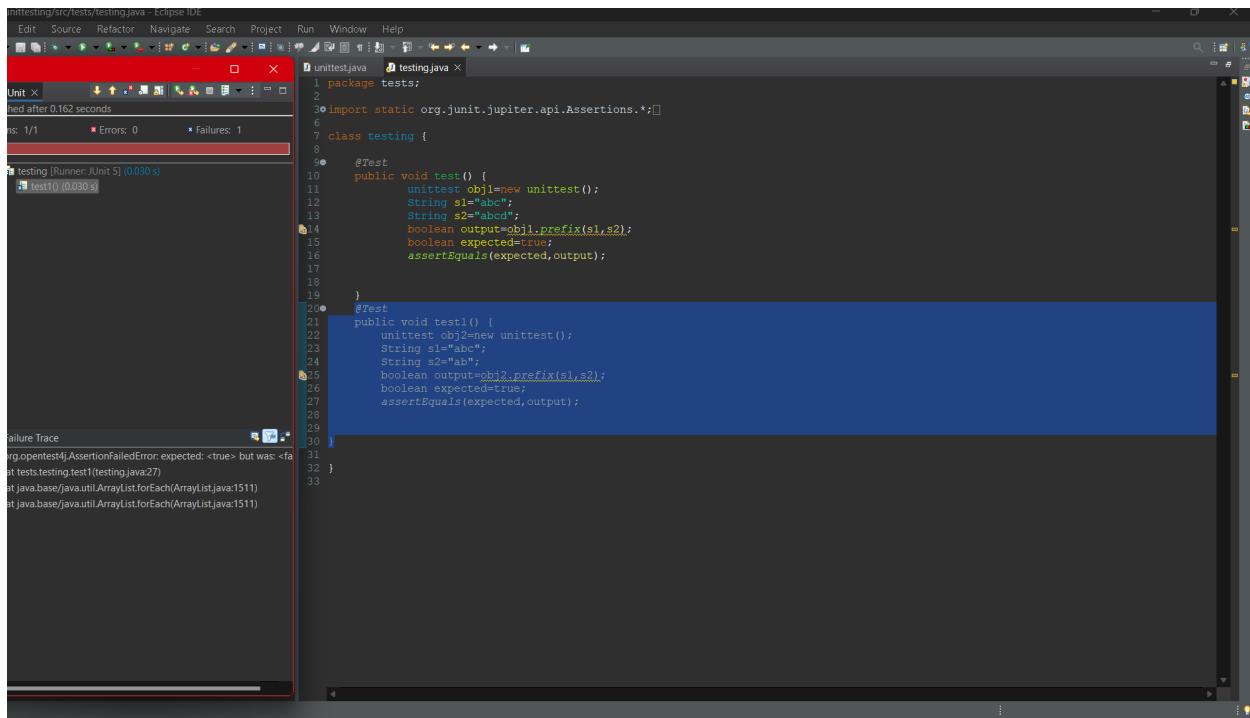


The screenshot shows the Eclipse IDE interface with the title bar "junittesting/src/tests/testing.java - Eclipse IDE". The menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, Help. The toolbar has various icons for file operations. The left sidebar shows a "JUnit" view with a green status bar indicating "Finished after 0.157 seconds", "Runs: 1/1", "Errors: 0", and "Failures: 0". Below it is a "Testing (Runner: JUnit 5) (0.020 s)" view showing a single test method "test() (0.020 s)". The main editor area contains Java code for a JUnit test:

```
1 package tests;
2
3 import static org.junit.jupiter.api.Assertions.*;
4
5 class testing {
6
7     @Test
8     public void test() {
9         unittest obj1=new unittest();
10        String s1="abc";
11        String s2="abcd";
12        boolean output=obj1.prefix(s1,s2);
13        boolean expected=true;
14        assertEquals(expected,output);
15    }
16
17
18
19    }
20
21 }
```

2. String s1 = "abc", s2 = "ab"; expected = true

Test case failed!



The screenshot shows the Eclipse IDE interface with the JUnit perspective selected. The status bar at the top indicates "junittesting/src/tests/testing.java - Eclipse IDE". The "Unit" view on the left shows "Run: 1/1", "Errors: 0", and "Failures: 1". The "Failure Trace" view displays the following stack trace:

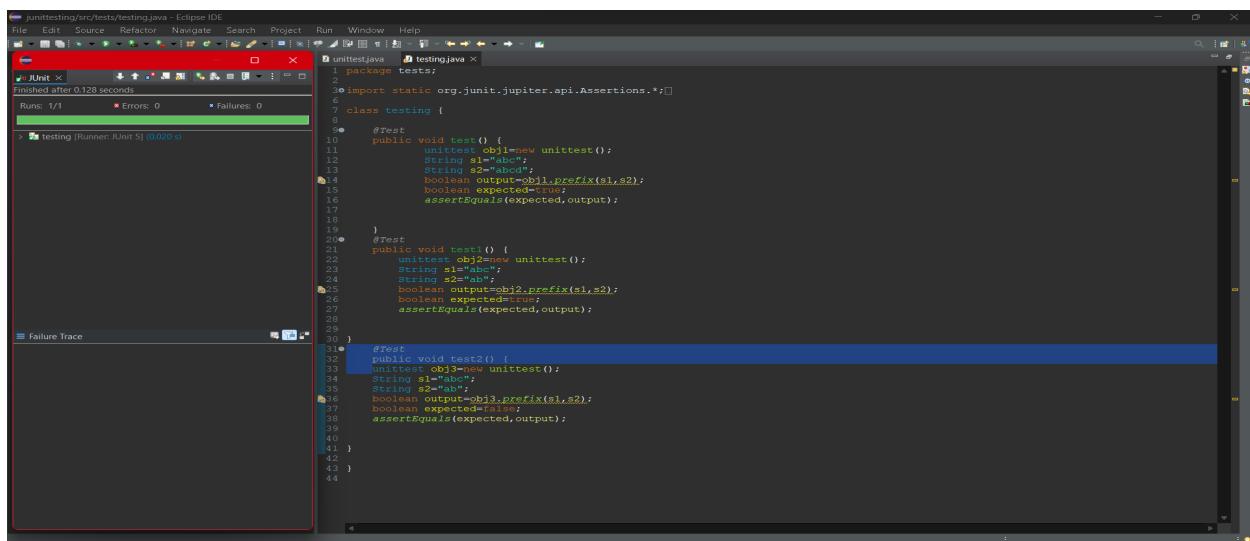
```
org.opentest4j.AssertionFailedError: expected: <true> but was: <false>
at tests.testing.test1(testing.java:27)
at java.base/java.util.ArrayList.forEach(ArrayList.java:1511)
at java.base/java.util.ArrayList.forEach(ArrayList.java:1511)
```

The code editor window contains the following Java code:

```
package tests;
import static org.junit.jupiter.api.Assertions.*;
class testing {
    @Test
    public void test0() {
        unittest obj1=new unittest();
        String s1="abc";
        String s2="bcd";
        boolean output=obj1.prefix(s1,s2);
        boolean expected=true;
        assertEquals(expected,output);
    }
    @Test
    public void test1() {
        unittest obj2=new unittest();
        String s1="abc";
        String s2="ab";
        boolean output=obj2.prefix(s1,s2);
        boolean expected=true;
        assertEquals(expected,output);
    }
}
```

3. String s1 = "abc", s2 = "ab"; expected = false

Test case passed!



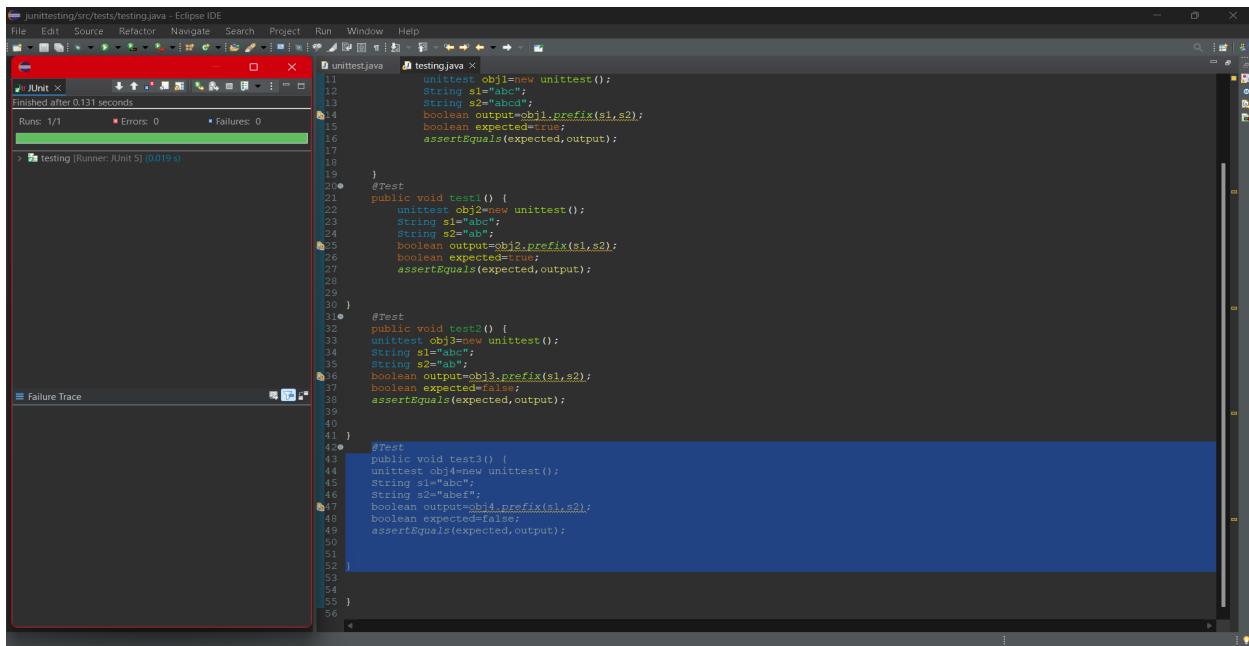
The screenshot shows the Eclipse IDE interface with the JUnit perspective selected. The status bar at the top indicates "junittesting/src/tests/testing.java - Eclipse IDE". The "Unit" view on the left shows "Run: 1/1", "Errors: 0", and "Failures: 0". The "Failure Trace" view is empty.

The code editor window contains the same Java code as the previous screenshot, but the test results are now successful:

```
package tests;
import static org.junit.jupiter.api.Assertions.*;
class testing {
    @Test
    public void test0() {
        unittest obj1=new unittest();
        String s1="abc";
        String s2="bcd";
        boolean output=obj1.prefix(s1,s2);
        boolean expected=true;
        assertEquals(expected,output);
    }
    @Test
    public void test1() {
        unittest obj2=new unittest();
        String s1="abc";
        String s2="ab";
        boolean output=obj2.prefix(s1,s2);
        boolean expected=false;
        assertEquals(expected,output);
    }
}
```

4. String s1 = "abc", s2 = "abef"; expected = false

Test case passed!

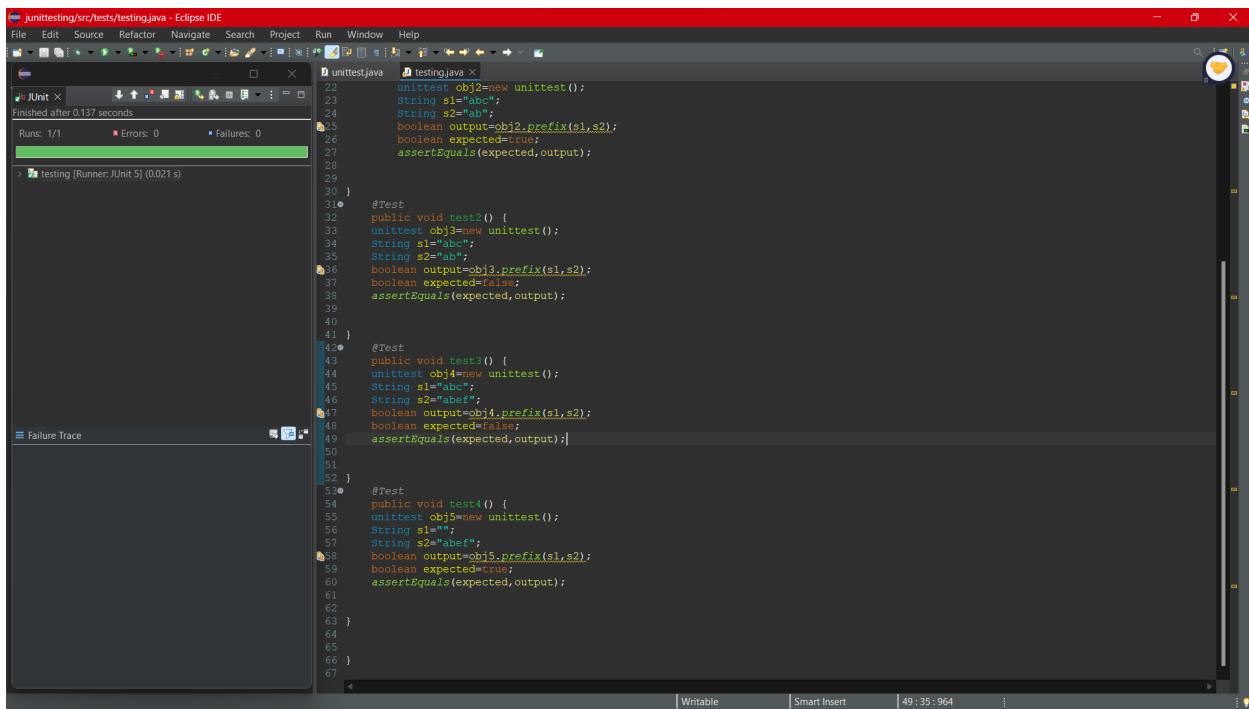


```
junittesting/src/tests/testing.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
JUnit x
Finished after 0.137 seconds
Runs: 1/1 Errors: 0 Failures: 0
testing [Runner: JUnit 5] (0.019 s)
Failure Trace

1.  junittesting.testing.Testing obj1=new unittest();
2.  String s1="abc";
3.  String s2="abcd";
4.  boolean output=obj1.prefix(s1,s2);
5.  boolean expected=true;
6.  assertEquals(expected,output);
7.
8.
9.
10. @Test
11. public void test1() {
12.     unittest obj2=new unittest();
13.     String s1="ab";
14.     String s2="ab";
15.     boolean output=obj2.prefix(s1,s2);
16.     boolean expected=true;
17.     assertEquals(expected,output);
18.
19.
20. }
21. @Test
22. public void test20() {
23.     unittest obj3=new unittest();
24.     String s1="abc";
25.     String s2="ab";
26.     boolean output=obj3.prefix(s1,s2);
27.     boolean expected=false;
28.     assertEquals(expected,output);
29.
30. }
31. @Test
32. public void test30() {
33.     unittest obj4=new unittest();
34.     String s1="abc";
35.     String s2="abef";
36.     boolean output=obj4.prefix(s1,s2);
37.     boolean expected=false;
38.     assertEquals(expected,output);
39.
40. }
41. @Test
42. public void test40() {
43.     unittest obj5=new unittest();
44.     String s1="";
45.     String s2="abef";
46.     boolean output=obj5.prefix(s1,s2);
47.     boolean expected=true;
48.     assertEquals(expected,output);
49.
50.
51.
52. }
53. @Test
54. public void test50() {
55.     unittest obj6=new unittest();
56.     String s1="";
57.     String s2="abef";
58.     boolean output=obj6.prefix(s1,s2);
59.     boolean expected=false;
60.     assertEquals(expected,output);
61.
62.
63. }
64.
65.
66. }
```

5. String s1 = "", s2 = "abef"; expected = true

Test case passed!



```
junittesting/src/tests/testing.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
JUnit x
Finished after 0.137 seconds
Runs: 1/1 Errors: 0 Failures: 0
testing [Runner: JUnit 5] (0.021 s)
Failure Trace

1.  junittesting.testing.Testing obj1=new unittest();
2.  String s1="";
3.  String s2="ab";
4.  boolean output=obj1.prefix(s1,s2);
5.  boolean expected=true;
6.  assertEquals(expected,output);
7.
8.
9.
10. @Test
11. public void test20() {
12.     unittest obj2=new unittest();
13.     String s1="abc";
14.     String s2="ab";
15.     boolean output=obj2.prefix(s1,s2);
16.     boolean expected=false;
17.     assertEquals(expected,output);
18.
19.
20. }
21. @Test
22. public void test30() {
23.     unittest obj3=new unittest();
24.     String s1="abc";
25.     String s2="ab";
26.     boolean output=obj3.prefix(s1,s2);
27.     boolean expected=false;
28.     assertEquals(expected,output);
29.
30. }
31. @Test
32. public void test40() {
33.     unittest obj4=new unittest();
34.     String s1="abc";
35.     String s2="abef";
36.     boolean output=obj4.prefix(s1,s2);
37.     boolean expected=false;
38.     assertEquals(expected,output);
39.
40. }
41. @Test
42. public void test50() {
43.     unittest obj5=new unittest();
44.     String s1="";
45.     String s2="abef";
46.     boolean output=obj5.prefix(s1,s2);
47.     boolean expected=true;
48.     assertEquals(expected,output);
49.
50.
51.
52. }
53. @Test
54. public void test60() {
55.     unittest obj6=new unittest();
56.     String s1="";
57.     String s2="abef";
58.     boolean output=obj6.prefix(s1,s2);
59.     boolean expected=false;
60.     assertEquals(expected,output);
61.
62.
63. }
64.
65.
66. }
```

PROGRAM P6

Consider again the triangle classification program (P4) with a slightly different specification: The program reads floating values from the standard input. The three values A, B, and C are interpreted as representing the lengths of the sides of a triangle. The program then prints a message to the standard output that states whether the triangle, if it can be formed, is scalene, isosceles, equilateral, or right angled. Determine the following for the above program:

- a) Identify the equivalence classes for the system

Equivalence Classes will contain:

1. All sides are positive, real numbers.
2. One or more sides are negative or zero.
3. The sum of the lengths of any two sides is less than or equal to the length of the remaining side.
4. The sum of the lengths of any two sides is greater than the length of the remaining side.

Examples

E1 : $a+b \leq c$ (point 3)

E2 : $a+c \leq b$ (point 3)

E3 : $b+c \leq a$ (point 3)

E4 : $a=b, b=c, c=a$

E5 : $a=b, a \neq c$

E6 : $a=c, a \neq b$

E7 : $b=c, b \neq a$

E8 : $a \neq b, b \neq c, c \neq a$

E9: $a^2 + b^2 = c^2$

E10: $b^2 + c^2 = a^2$

E11: $a^2 + c^2 = b^2$

E12 : $a+b \geq c$ (point 4)

E13: $a+c \geq b$ (point 4)

E14: $b+c \geq a$ (point 4)

b) Identify test cases to cover the identified equivalence classes. Also, explicitly mention which test case would cover which equivalence class.

Test cases

1. Right angled triangle (point 1 of (a)) - $A = 5, B = 12, C = 13$
2. Equilateral triangle (point 1 of (a)) - $A = 3, B = 3, C = 3$
3. Scalene triangle (point 1 of (a)) - $A = 3, B = 4, C = 3$
4. Isosceles triangle (point 1 of (a)) - $A = 3, B = 2, C = 3$
4. Invalid Input - $A = 1, B = 2, C = 3$
5. Invalid Input - $A = 0, B = 4, C = -1$

c) For the boundary condition $A + B > C$ case (scalene triangle), identify test cases to verify the boundary.

* Test cases

1. $A = 5, B = 4, C = 3$
2. $A = 5, B = 1, C = 6$
3. $A = 2, B = 3, C = 6$

d) For the boundary condition $A = C$ case (isosceles triangle), identify test cases to verify the boundary.

* Test cases

1. $A = 4, B = 3, C = 4$
2. $A = 4, B = 3, C = 3.9$
3. $A = 4, B = 3, C = 4.1$

e) For the boundary condition $A = B = C$ case (equilateral triangle), identify test cases to verify the boundary.

* Test cases

1. $A = 3, B = 3, C = 3$
2. $A = 3, B = 2.9, C = 3.1$

f) For the boundary condition $A^2 + B^2 = C^2$ case (right-angle triangle), identify test cases to verify the boundary.

* Test cases

1. $A = 3, B = 4, C = 5$
2. $A = 6, B = 8, C = 10$

g) For the non-triangle case, identify test cases to explore the boundary.

* Test cases

1. $A = 2, B = 2, C = 4$
2. $A = 2, B = 4, C = 2$

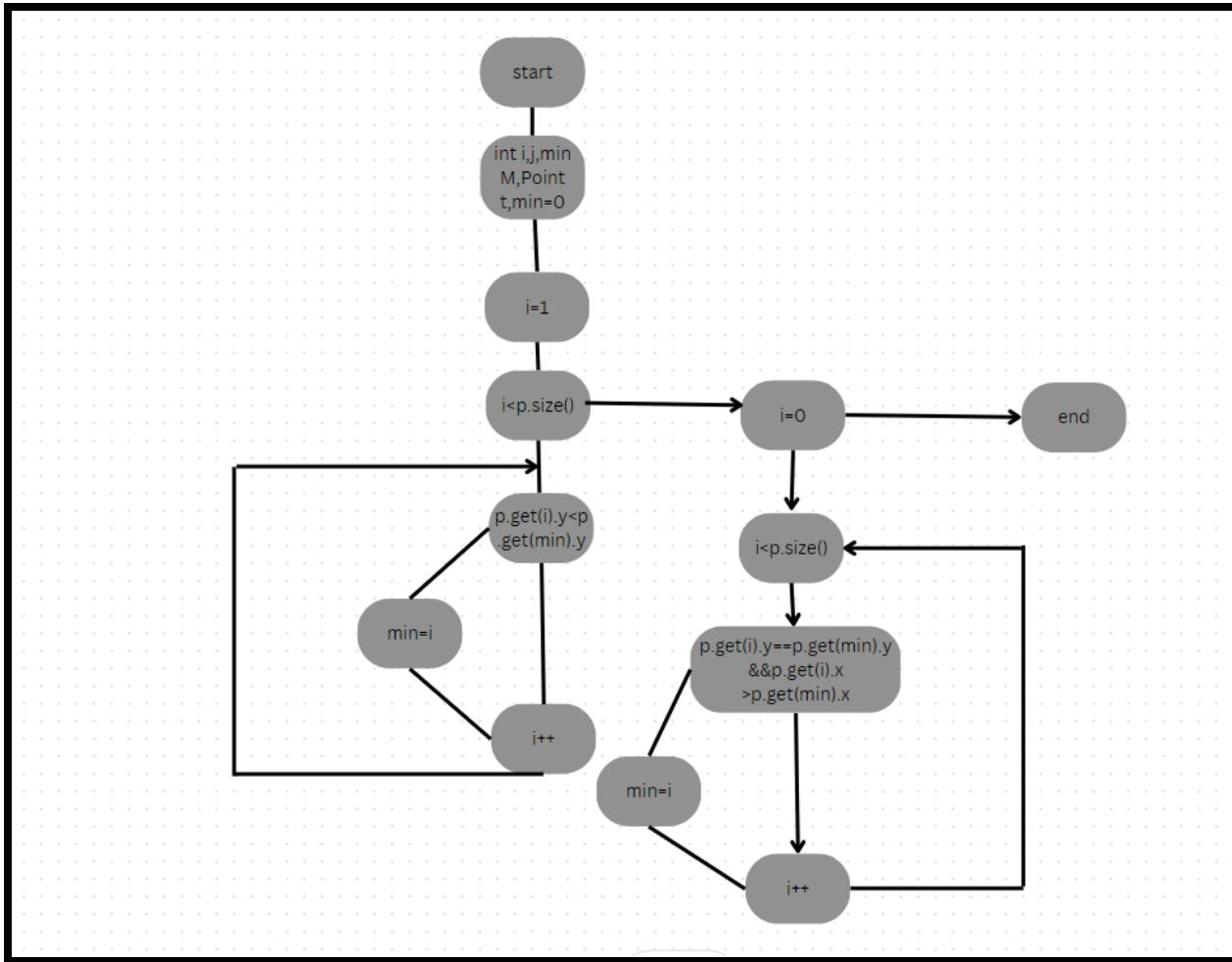
h) For non-positive input, identify test points.

* Test cases

1. $A = -3, B = 3, C = 4.5$
2. $A = 0, B = 4, C = 5$

SECTION B

1. Control Flow Graph



2. Construct test sets for your flow graph that are adequate for the following criteria:

- Statement Coverage.
- Branch Coverage.
- Basic Condition Coverage.

a) Statement coverage test sets:

* Test cases

1. p is an empty vector
2. p is a vector with one point
3. p is a vector with two points having same y component

4. p is a vector with two points having different y components
5. p is a vector with three or more different points with different points with same y components
6. p is a vector with three or more different points with different points with different y components

b) Branch coverage test sets:

***TEST CASES**

1. p is an empty vector
 2. p is a vector with one point
 3. p is a vector with two points having same y component
 4. p is a vector with two points having different y components
 5. p is a vector with three or more different points with different points with same y components and with same x components.
 6. p is a vector with three or more different points with different points with different y components and with same x components.
 7. p is a vector with three or more points with the same x and y components
- </pre>

c) Basic condition coverage test sets:

***TEST CASES**

1. p is an empty vector
2. p is a vector with one point
3. p is a vector with two points having same y component
4. p is a vector with two points having different y components
5. p is a vector with three or more different points with different points with same y components and with same x components.
6. p is a vector with three or more different points with different points with different y components and with same x components.
7. p is a vector with three or more points with the same x and y components

8. p is a vector with some of them having same x component and all of them having same y component

* **Test cases examples:**

- 1) $p=[(x=2,y=3),(x=2,y=2),(x=1,y=5),(x=1,y=4)]$
- 2) $p=[(x=5,y=6),(x=3,y=2),(x=3,y=4),(x=1,y=2)]$
- 3) $p=[(x=5,y=6),(x=3,y=5),(x=1,y=5),(x=4,y=5),(x=2,y=7)]$
- 4) $p=[(x=7,y=8)]$
- 5) $p=[]$

These 5 test cases covers all - statement coverage, branch coverage and basic condition coverage.