IT314

Patel Krunal Maheshbhai
202001445 Lab 7

Section A

Q1:
Consider a program for determining the previous date. Its input is triple of day, month and year with the following ranges 1 <= month <= 12, 1 <= day <= 31, 1900 <= year <= 2015. The possible output dates would be the previous date or invalid date. Design the equivalence class test cases?

Ans :

	Valid classes	Invalid classes
Days	1 <= day <= 31	day < 1, day > 31
Months	1 <= month <= 12	month < 1, month > 12
Years	1900 <= year <= 2015	year < 1900, year > 2015

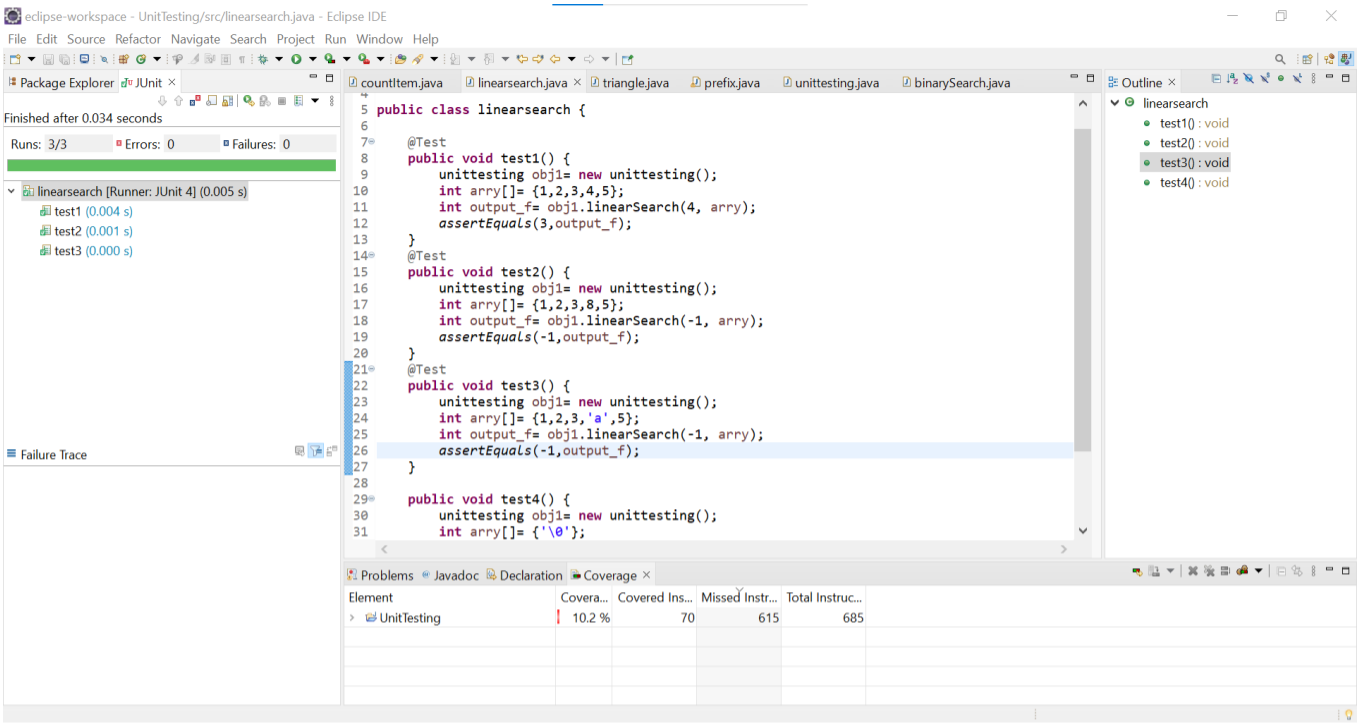
Some Test Cases:

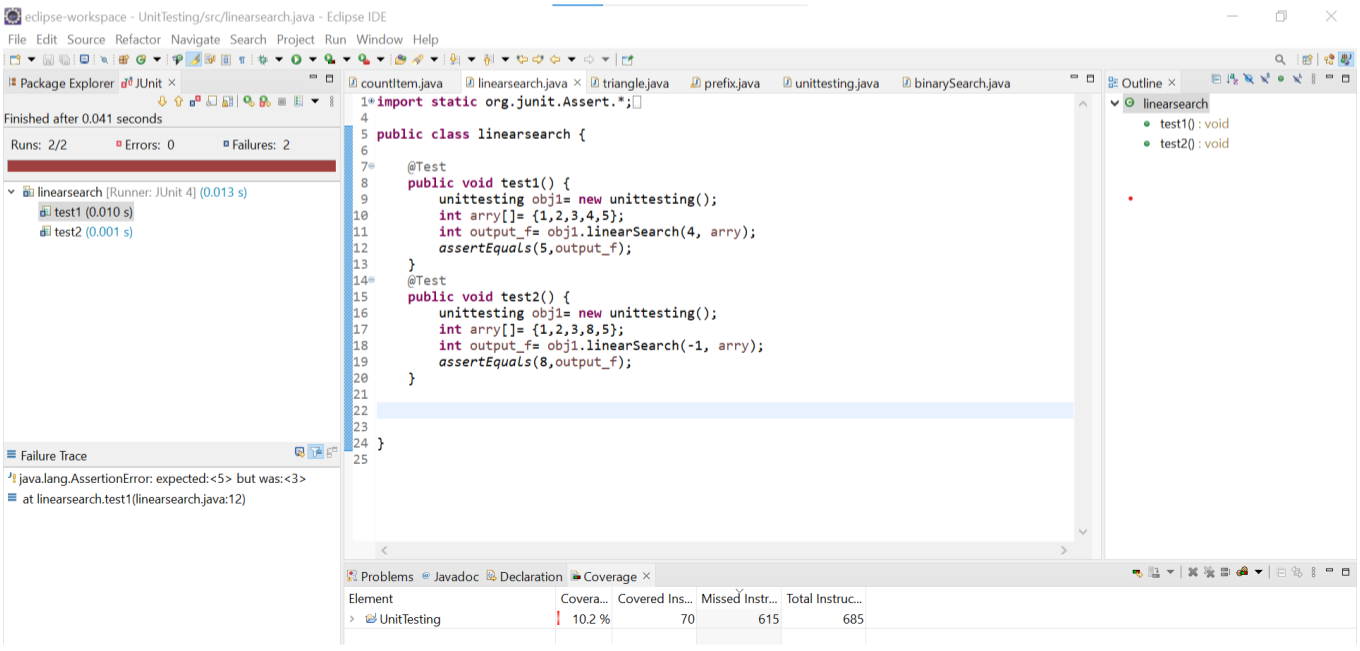
1. Valid test cases:
    a. (1, 1, 1900) - the minimum valid date
    b. (15, 5, 2002) - a random valid date
    c. (31, 12, 2015) - the maximum valid date
2. Invalid test cases:
    a. (0, 5, 1980) - day is less than 1
    b. (32, 1, 2012) - day is greater than 31
    c. (29, 2, 2002)- the year is not a leap year
    d. (3, 5, 2016) - the year is greater than 2015
    e. (5, 13, 1908) - month is greater than 12
    f. (-10, -1, 2020) - all parameters are invalid

    These test cases represent the equivalence classes and should cover all possible scenarios.

Programs:

P1:


eclipse-workspace - UnitTesting/src/linearsearch.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer JUnit
Finished after 0.034 seconds
Runs: 3/3 Errors: 0 Failures: 0
linearsearch [Runner: JUnit 4] (0.005 s)
test1 (0.004 s)
test2 (0.001 s)
test3 (0.000 s)
Failure Trace
countItem.java linearsearch.java triangle.java prefix.java unittesting.java binarySearch.java
5 public class linearsearch {
6
7 @Test
8 public void test1() {
9 unittesting obj1= new unittesting();
10 int arry[]= {1,2,3,4,5};
11 int output_f= obj1.linearSearch(4, arry);
12 assertEquals(3,output_f);
13 }
14 @Test
15 public void test2() {
16 unittesting obj1= new unittesting();
17 int arry[]= {1,2,3,8,5};
18 int output_f= obj1.linearSearch(-1, arry);
19 assertEquals(-1,output_f);
20 }
21 @Test
22 public void test3() {
23 unittesting obj1= new unittesting();
24 int arry[]= {1,2,3,'a',5};
25 int output_f= obj1.linearSearch(-1, arry);
26 assertEquals(-1,output_f);
27 }
28
29 public void test4() {
30 unittesting obj1= new unittesting();
31 int arry[]= {'\0'};
Outline
linearsearch
test1() : void
test2() : void
test3() : void
test4() : void
Problems Javadoc Declaration Coverage
Element Covera... Covered Ins... Missed Instr... Total Instruc...
UnitTesting 10.2 % 70 615 685


eclipse-workspace - UnitTesting/src/linearsearch.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer JUnit
Finished after 0.041 seconds
Runs: 2/2 Errors: 0 Failures: 2
linearsearch [Runner: JUnit 4] (0.013 s)
test1 (0.010 s)
test2 (0.001 s)
Failure Trace
java.lang.AssertionError: expected:<5> but was:<3>
at linearsearch.test1(linearsearch.java:12)
countItem.java linearsearch.java triangle.java prefix.java unittesting.java binarySearch.java
1 import static org.junit.Assert.*;
4
5 public class linearsearch {
6
7 @Test
8 public void test1() {
9 unittesting obj1= new unittesting();
10 int arry[]= {1,2,3,4,5};
11 int output_f= obj1.linearSearch(4, arry);
12 assertEquals(5,output_f);
13 }
14 @Test
15 public void test2() {
16 unittesting obj1= new unittesting();
17 int arry[]= {1,2,3,8,5};
18 int output_f= obj1.linearSearch(-1, arry);
19 assertEquals(8,output_f);
20 }
21
22
23
24 }
25
Outline
linearsearch
test1() : void
test2() : void
Problems Javadoc Declaration Coverage
Element Covera... Covered Ins... Missed Instr... Total Instruc...
UnitTesting 10.2 % 70 615 685

Equivalence Partitioning :

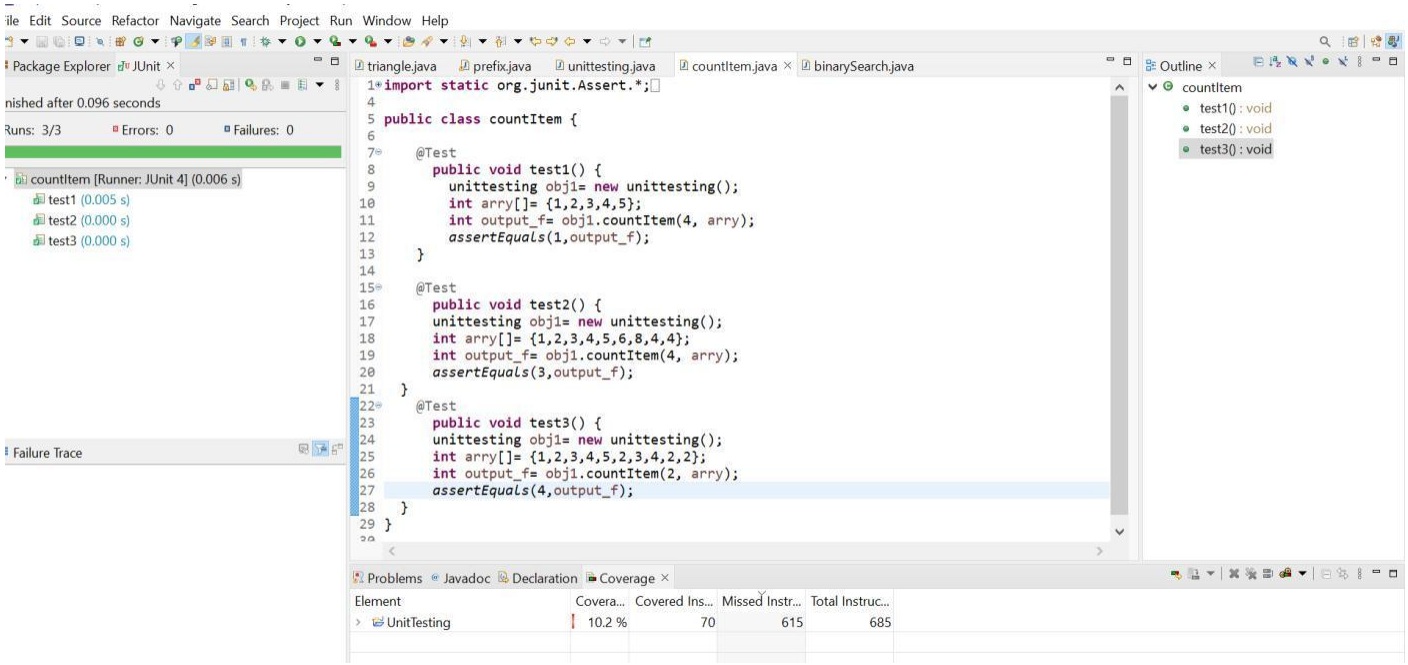
Tester Action and Input Data	Expected Outcome
Test with v as a non-existent value and an empty array a[]	-1
Test with v as a non-existent value and a non-empty array a[]	-1
Test with v as an existent value and an empty array a[]	-1
Test with v as an existent value and a non-empty array a[] the index of v in a[] where v exists	the index of v in a[]
Test with v as an existent value and a non-empty array a[] -1 where v does not exist	-1

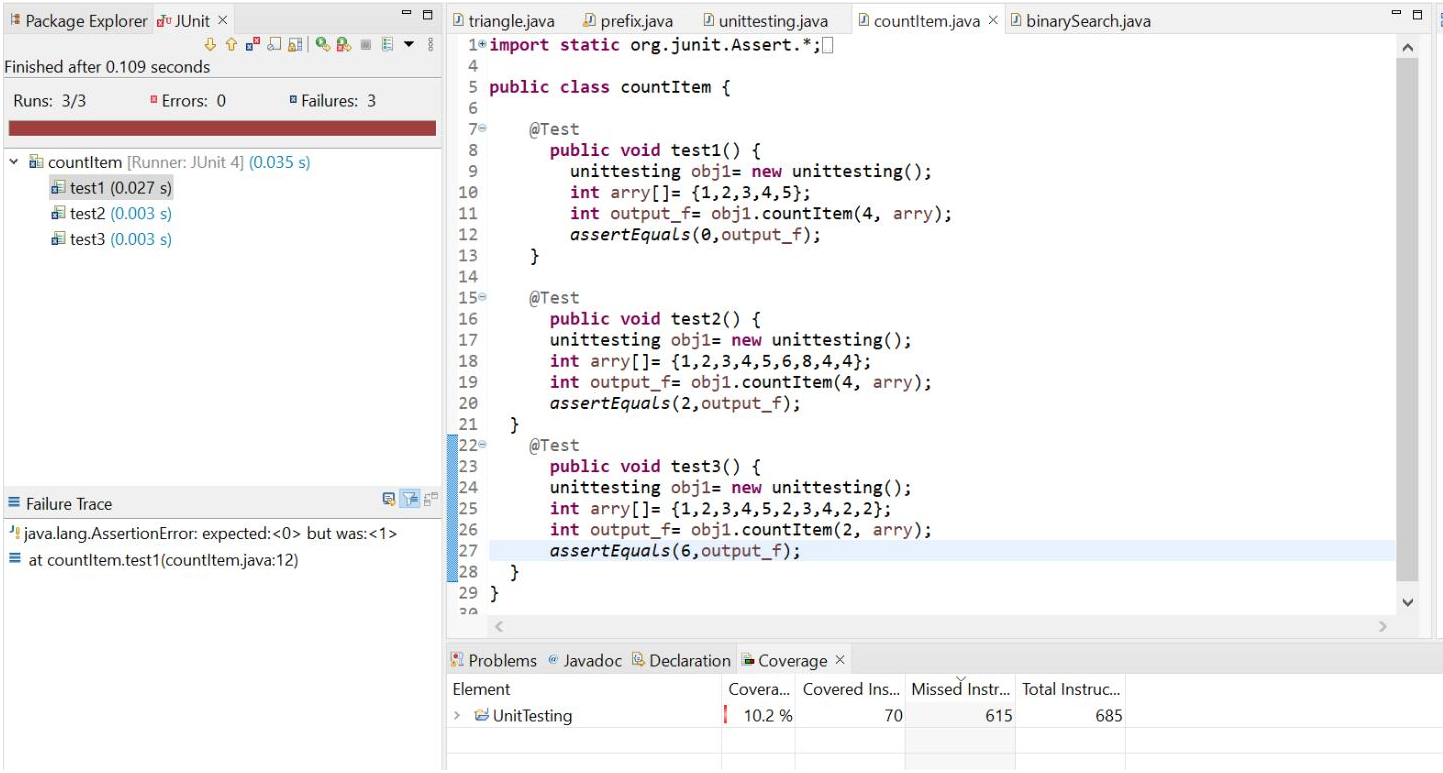
Boundary Value Analysis:

Test with v as a non-existent value and an empty array a[]	-1

Test with v as a non-existent value and a non-empty array a[]	-1
Test with v as an existent value and an array a[] of length 0	-1
Test with v as an existent value and an array a[] of length 1, where v 0 exists	0
Test with v as an existent value and an array a[] of length 1, where v -1 does not exist	-1
Test with v as an existent value and an array a[] of length greater than 0 1, where v exists at the beginning of the array	0
Test with v as an existent value and an array a[] of length greater than the last index where v is found 1, where v exists at the end of the array	the last index where v is found

P2:





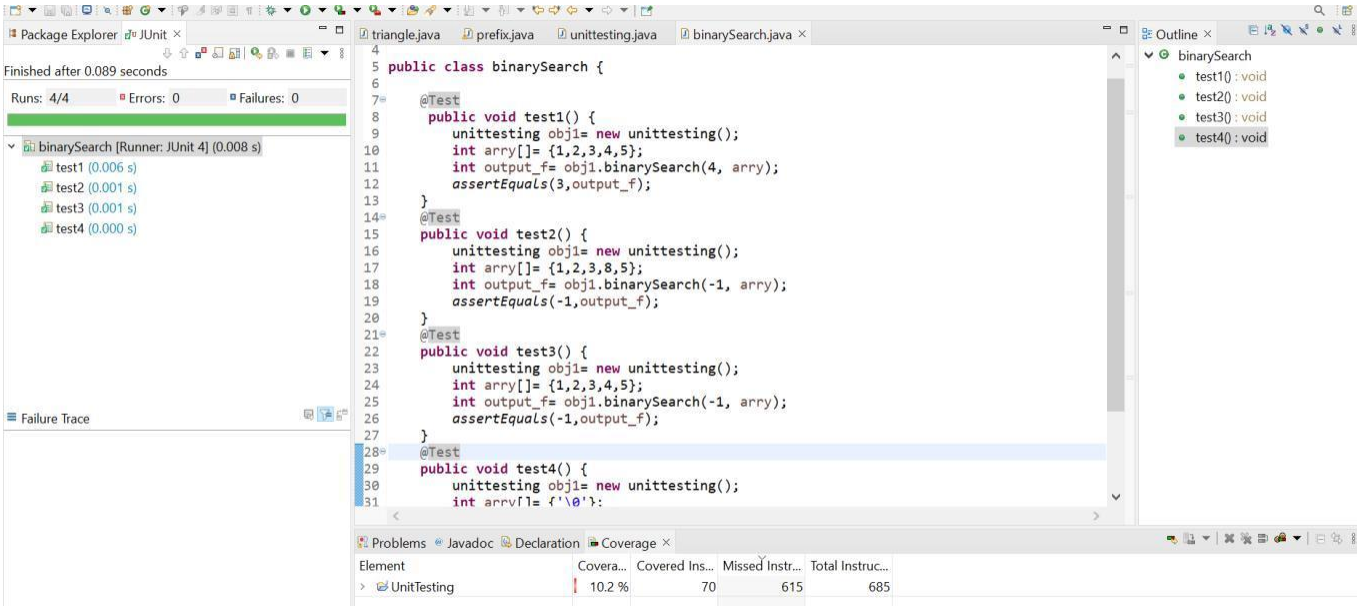
Equivalence Partitioning :

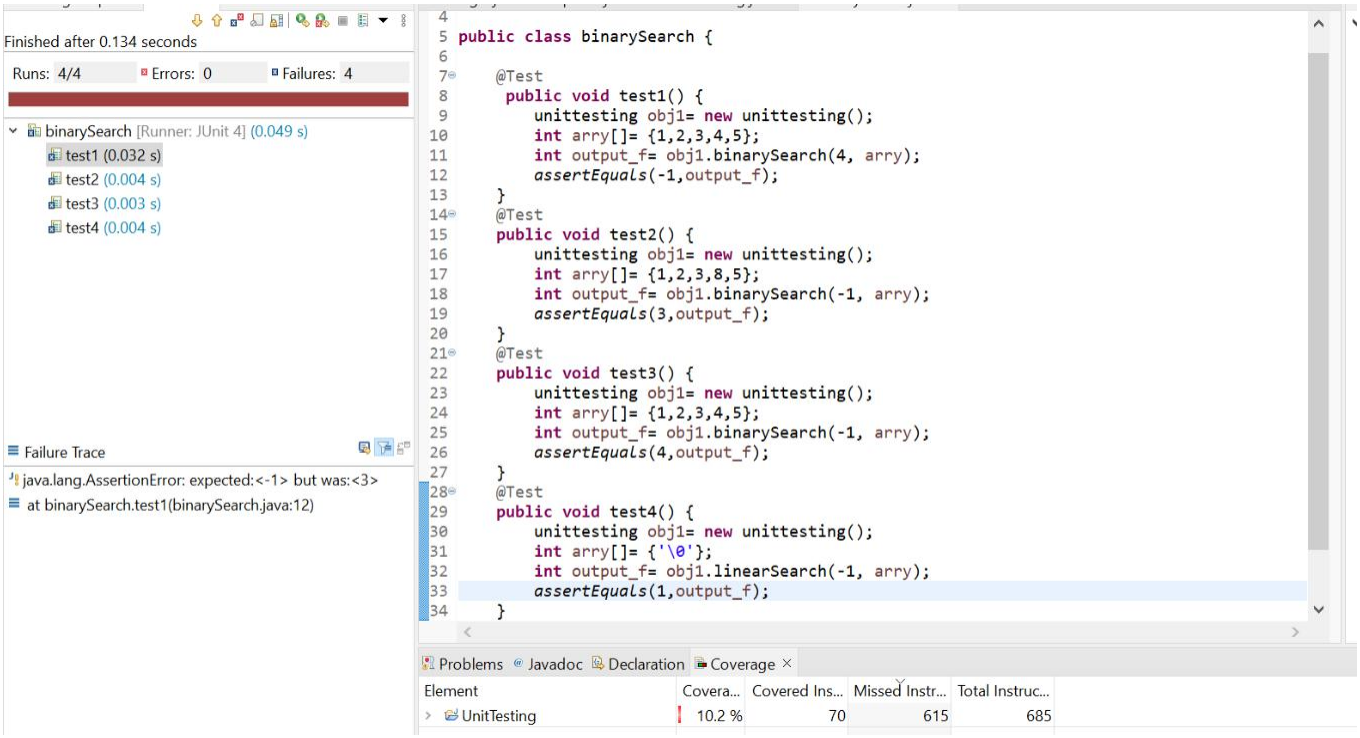
Tester Action and Input Data	Expected Outcome
Test with v as a non-existent value and an empty array a[]	0
Test with v as a non-existent value and a non-empty array a[]	0
Test with v as an existent value and an empty array a[]	0
Test with v as an existent value and a non-empty array a[] the index of v in a[] where v exists	# of occurrence of v in a[]
Test with v as an existent value and a non-empty array a[] -1 where v does not exist	1

Boundary Value Analysis:

Tester Action and Input Data	Expected Outcome
Test with v as a non-existent value and an empty array a[]	0
Test with v as a non-existent value and a non-empty array a[]	0
Test with v as an existent value and an array a[] of length 0	0
Test with v as an existent value and an array a[] of length 1, where v exists	1
Test with v as an existent value and an array a[] of length 1, where v does not exist	0
Test with v as an existent value and an array a[] of length greater than 1, where v exists at the beginning of the array	the number of occurrences of v in a[]
Test with v as an existent value and an array a[] of length greater than 1, where v exists at the end of the array	the number of occurrences of v in a[]
Test with v as an existent value and an array a[] of length greater than 1, where v exists in the middle of the array	the number of occurrences of v in a[]

P3:


Package Explorer
JUnit
Finished after 0.089 seconds
Runs: 4/4
Errors: 0
Failures: 0
binarySearch [Runner: JUnit 4] (0.008 s)
test1 (0.006 s)
test2 (0.001 s)
test3 (0.001 s)
test4 (0.000 s)
Failure Trace
triangle.java
prefix.java
unittesting.java
binarySearch.java
public class binarySearch {
@Test
public void test1() {
unittesting obj1= new unittesting();
int arry[]= {1,2,3,4,5};
int output_f= obj1.binarySearch(4, arry);
assertEquals(3,output_f);
}
@Test
public void test2() {
unittesting obj1= new unittesting();
int arry[]= {1,2,3,8,5};
int output_f= obj1.binarySearch(-1, arry);
assertEquals(-1,output_f);
}
@Test
public void test3() {
unittesting obj1= new unittesting();
int arry[]= {1,2,3,4,5};
int output_f= obj1.binarySearch(-1, arry);
assertEquals(-1,output_f);
}
@Test
public void test4() {
unittesting obj1= new unittesting();
Outline
binarySearch
test1() : void
test2() : void
test3() : void
test4() : void
Problems
Javadoc
Declaration
Coverage
Element
Covera...
Covered Ins...
Missed Instr...
Total Instruc...
UnitTesting
10.2 %
70
615
685


Finished after 0.134 seconds
Runs: 4/4
Errors: 0
Failures: 4
binarySearch [Runner: JUnit 4] (0.049 s)
test1 (0.032 s)
test2 (0.004 s)
test3 (0.003 s)
test4 (0.004 s)
Failure Trace
java.lang.AssertionError: expected:<-1> but was:<3>
at binarySearch.test1(binarySearch.java:12)
public class binarySearch {
@Test
public void test1() {
unittesting obj1= new unittesting();
int arry[]= {1,2,3,4,5};
int output_f= obj1.binarySearch(4, arry);
assertEquals(-1,output_f);
}
@Test
public void test2() {
unittesting obj1= new unittesting();
int arry[]= {1,2,3,8,5};
int output_f= obj1.binarySearch(-1, arry);
assertEquals(3,output_f);
}
@Test
public void test3() {
unittesting obj1= new unittesting();
int arry[]= {1,2,3,4,5};
int output_f= obj1.binarySearch(-1, arry);
assertEquals(4,output_f);
}
@Test
public void test4() {
unittesting obj1= new unittesting();
int arry[]= {'\0'};
int output_f= obj1.linearSearch(-1, arry);
assertEquals(1,output_f);
}
Problems
Javadoc
Declaration
Coverage
Element
Covera...
Covered Ins...
Missed Instr...
Total Instruc...
UnitTesting
10.2 %
70
615
685

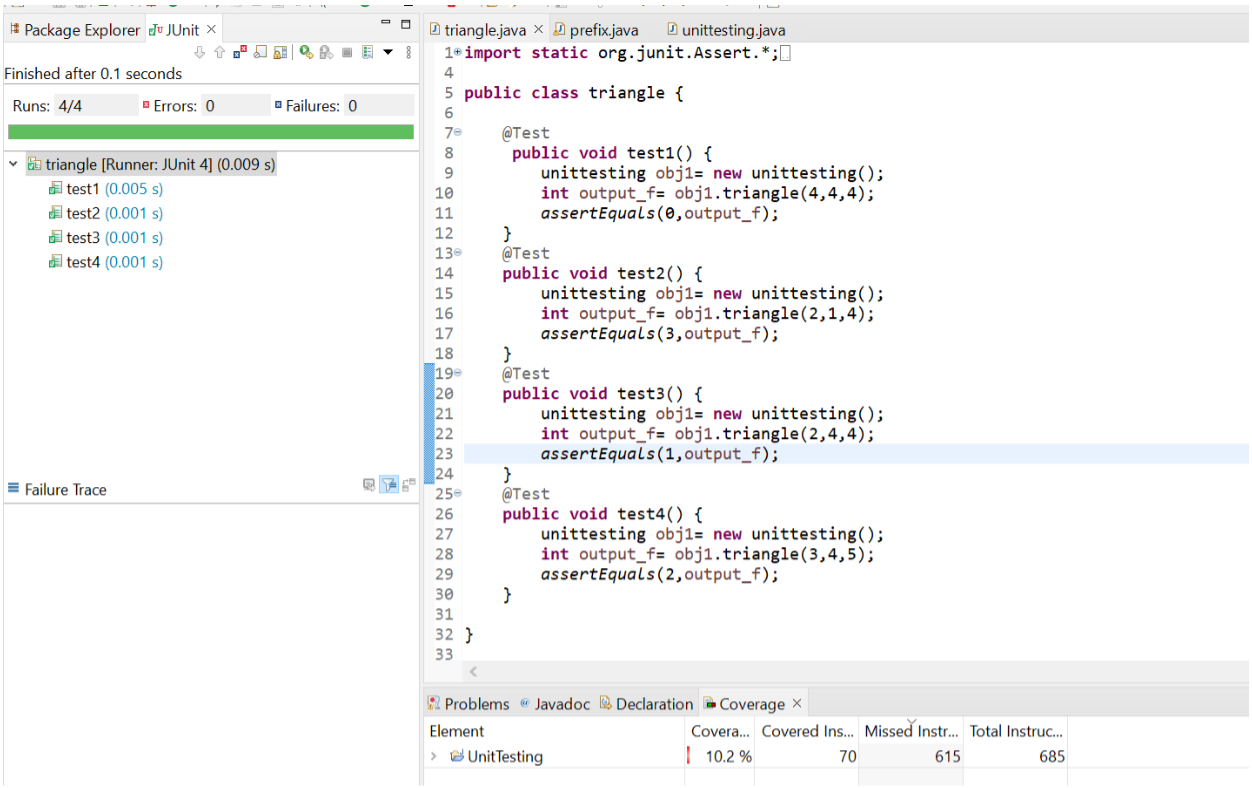
Equivalence Partitioning :

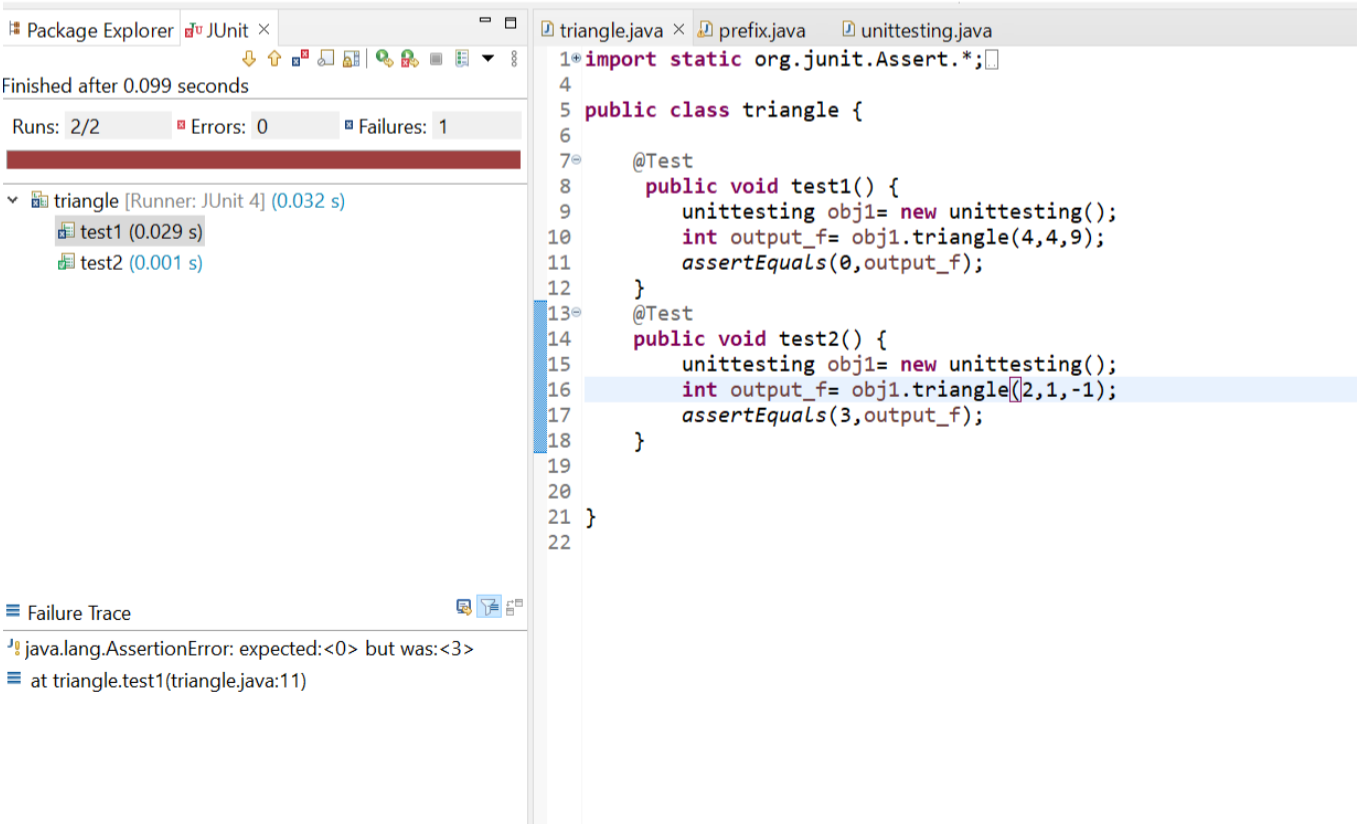
Tester Action and Input Data	Expected Outcome
v=5, a=[1, 3, 5, 7, 9]	2
v=1, a=[1, 3, 5, 7, 9]	0
v=9, a=[1, 3, 5, 7, 9]	4
v=4, a=[1, 3, 5, 7, 9]	-1
v=11, a=[1, 3, 5, 7, 9]	-1

boundary Value Analysis:

Tester Action and Input Data	Expected Outcome
v=1, a=[1]	0
v=9, a=[9]	
v=5, a=[]	-1
v=5, a=[5, 7, 9]	0(smallest in the array)
v=5, a=[1, 3, 5]	2(largest)

P4:


Package Explorer
JUnit
Finished after 0.1 seconds
Runs: 4/4
Errors: 0
Failures: 0
triangle [Runner: JUnit 4] (0.009 s)
test1 (0.005 s)
test2 (0.001 s)
test3 (0.001 s)
test4 (0.001 s)
Failure Trace
triangle.java
prefix.java
unittesting.java
1 import static org.junit.Assert.*;
4
5 public class triangle {
6
7 @Test
8 public void test1() {
9 unittesting obj1= new unittesting();
10 int output_f= obj1.triangle(4,4,4);
11 assertEquals(0,output_f);
12 }
13 @Test
14 public void test2() {
15 unittesting obj1= new unittesting();
16 int output_f= obj1.triangle(2,1,4);
17 assertEquals(3,output_f);
18 }
19 @Test
20 public void test3() {
21 unittesting obj1= new unittesting();
22 int output_f= obj1.triangle(2,4,4);
23 assertEquals(1,output_f);
24 }
25 @Test
26 public void test4() {
27 unittesting obj1= new unittesting();
28 int output_f= obj1.triangle(3,4,5);
29 assertEquals(2,output_f);
30 }
31
32 }
33
Problems
Javadoc
Declaration
Coverage
Element
Covera...
Covered Ins...
Missed Instr...
Total Instruc...
UnitTesting
10.2 %
70
615
685


Package Explorer
JUnit
Finished after 0.099 seconds
Runs: 2/2
Errors: 0
Failures: 1
triangle [Runner: JUnit 4] (0.032 s)
test1 (0.029 s)
test2 (0.001 s)
Failure Trace
java.lang.AssertionError: expected:<0> but was:<3>
at triangle.test1(triangle.java:11)
triangle.java
prefix.java
unittesting.java
1 import static org.junit.Assert.*;
4
5 public class triangle {
6
7 @Test
8 public void test1() {
9 unittesting obj1= new unittesting();
10 int output_f= obj1.triangle(4,4,9);
11 assertEquals(0,output_f);
12 }
13 @Test
14 public void test2() {
15 unittesting obj1= new unittesting();
16 int output_f= obj1.triangle(2,1,-1);
17 assertEquals(3,output_f);
18 }
19
20
21 }
22

Equivalence Partitioning:

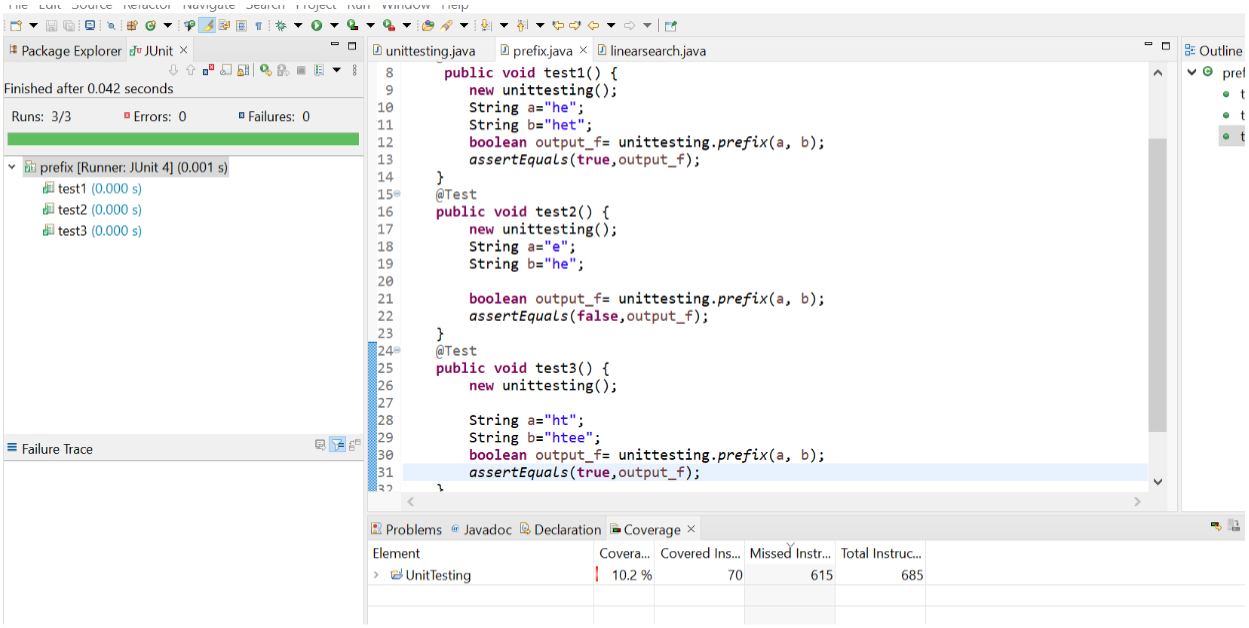
Tester Action and Input Data	Expected Outcome
Valid input: a=3, b=3, c=3	EQUILATERAL
Valid input: a=4, b=4, c=5	ISOSCELES
Valid input: a=5, b=4, c=3	SCALENE
Invalid input: a=0, b=0, c=0	INVALID
Invalid input: a=-1, b=2, c=3	INVALID
Valid input: a=1, b=1, c=1	EQUILATERAL
Valid input: a=2, b=2, c=1	ISOSCELES
Valid input: a=3, b=4, c=5	SCALENE
Invalid input: a=0, b=1, c=1	INVALID
Invalid input: a=1, b=0, c=1	INVALID
Invalid input: a=1, b=1, c=0	INVALID

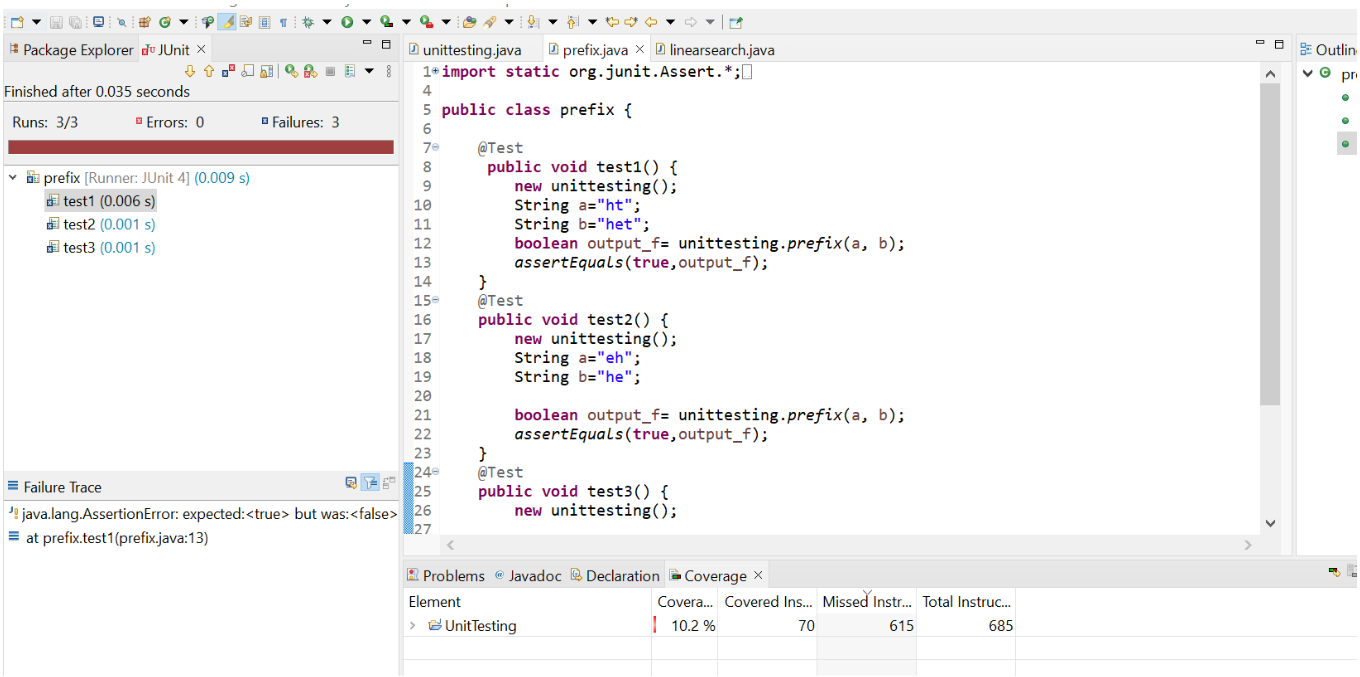
Boundary Value Analysis:

Tester Action and Input Data	Expected Outcome
Invalid inputs: a = 0, b = 0, c = 0	INVALID
Invalid inputs: a + b = c or b + c = a or c + a = b (a=3, b=4, c=8)	INVALID
Equilateral triangles: a = b = c = 1	EQUILATERAL

Equilateral triangles: a = b = c = 100	EQUILATERAL
Isosceles triangles: a = b ≠ c = 10	ISOSCELES
Isosceles triangles: a ≠ b = c = 10	ISOSCELES
Isosceles triangles: a = c ≠ b = 10	ISOSCELES
Scalene triangles: a = b + c - 1	SCALENE
Scalene triangles: b = a + c - 1	SCALENE
Scalene triangles: c = a + b - 1	SCALENE
Maximum values: a, b, c = Integer.MAX_VALUE	INVALID
Minimum values: a, b, c = Integer.MIN_VALUE	INVALID

P5 :





Equivalence Partitioning:

Tester Action and Input Data	Expected Outcome
Valid Inputs: s1= "hello", s2 = "hello world"	true
Valid Inputs: s1= "a", s2 = "abc"	true
Invalid Inputs: s1 = "", s2 = "hello world"	false
Invalid Inputs: s1 = "world", s2 = "hello world"	false

Boundary Value Analysis:

Tester Action and Input Data	Expected Outcome
s1= "", s2 = "abc"	False
s1= "ab", s2 = "abc"	True
s1= "abc", s2 = "ab"	False

s1= "a", s2 = "ab"	True
s1= "hello", s2 = "hellooo"	True
s1= "abc", s2 = "abc"	True
s1= "a", s2 = "b"	False
s1= "a", s2 = "a"	True

P6:

Equivalence Class:

Tester Action and Input Data	Expected Outcome
a = -1, b = 2, c = 3	Invalid input
a = 1, b = 1, c = 1	Equilateral triangle
a = 2, b = 2, c = 3	Isosceles triangle
a = 3, b = 4, c = 5	Scalene right-angled triangle
a = 3, b = 5, c = 4	Scalene right-angled triangle
a = 5, b = 3, c = 4	Scalene right-angled triangle
a = 3, b = 4, c = 6	Not a triangle

Test Case:

Invalid inputs:

a = 0, b = 0, c = 0, a + b = c, b + c = a, c + a = b

Invalid inputs:

a = -1, b = 1, c = 1, a + b = c

Equilateral triangles:

a = b = c = 1, a = b = c = 100

Isosceles triangles:

a = b = 10, c = 5;
a = c = 10, b = 3;
b = c = 10,
a = 6

Scalene triangles:

a = 4, b = 5, c = 6;
a = 10, b = 11, c = 13

Right angled triangle:

a = 3, b = 4, c = 5;
a = 5, b = 12, c = 13

Non-triangle:

a = 1, b = 2, c = 3

Non-positive input:

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

c) Boundary condition A + B > C:

a = Integer.MAX_VALUE, b = Integer.MAX_VALUE, c = 1
a = Double.MAX_VALUE, b = Double.MAX_VALUE, c = Double.MAX_VALUE

d) Boundary condition A = C:

a = Integer.MAX_VALUE,
b = 2,
c = Integer.MAX_VALUE

a = Double.MAX_VALUE,
b = 2.5,
c = Double.MAX_VALUE

e) Boundary condition A = B = C:

a = Integer.MAX_VALUE, b = Integer.MAX_VALUE, c = Integer.MAX_VALUE
a = Double.MAX_VALUE, b = Double.MAX_VALUE, c = Double.MAX_VALUE

f) Boundary condition A^2 + B^2 = C^2:

a = Integer.MAX_VALUE,
b = Integer.MAX_VALUE,
c = Integer.MAX_VALUE

a = Double.MAX_VALUE,
b = Double.MAX_VALUE,
c = Math.sqrt(Math.pow(Double.MAX_VALUE, 2) + Math.pow(Double.MAX_VALUE, 2))

g) Non-triangle:

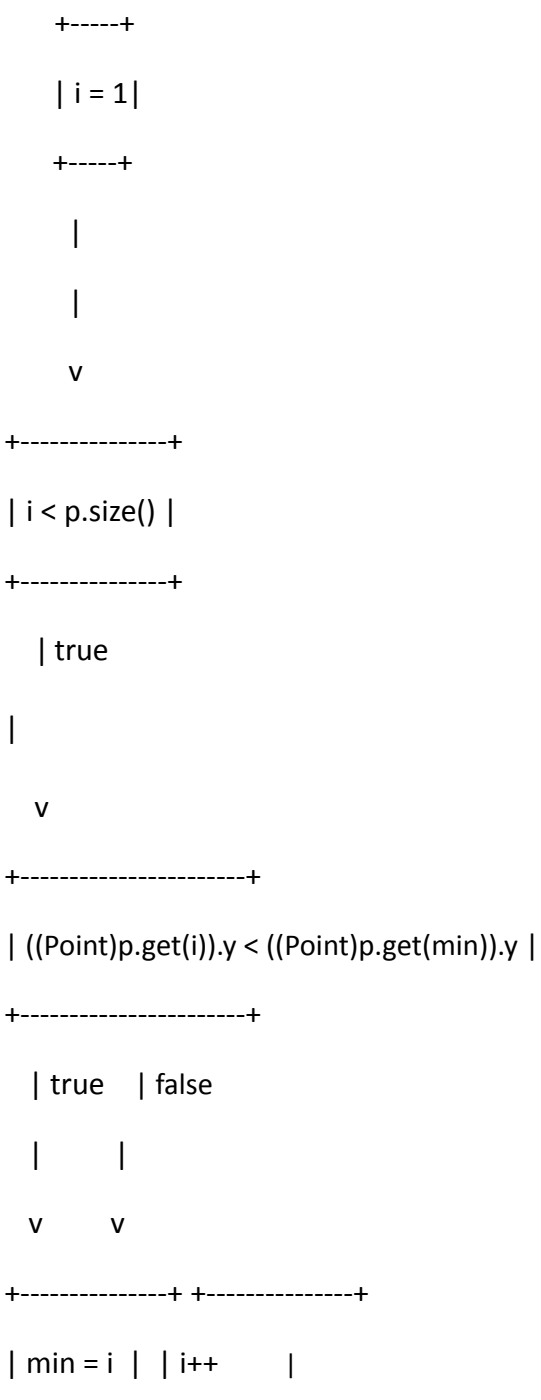
a = 1, b = 2, c = 4 a = 2, b = 4, c = 8

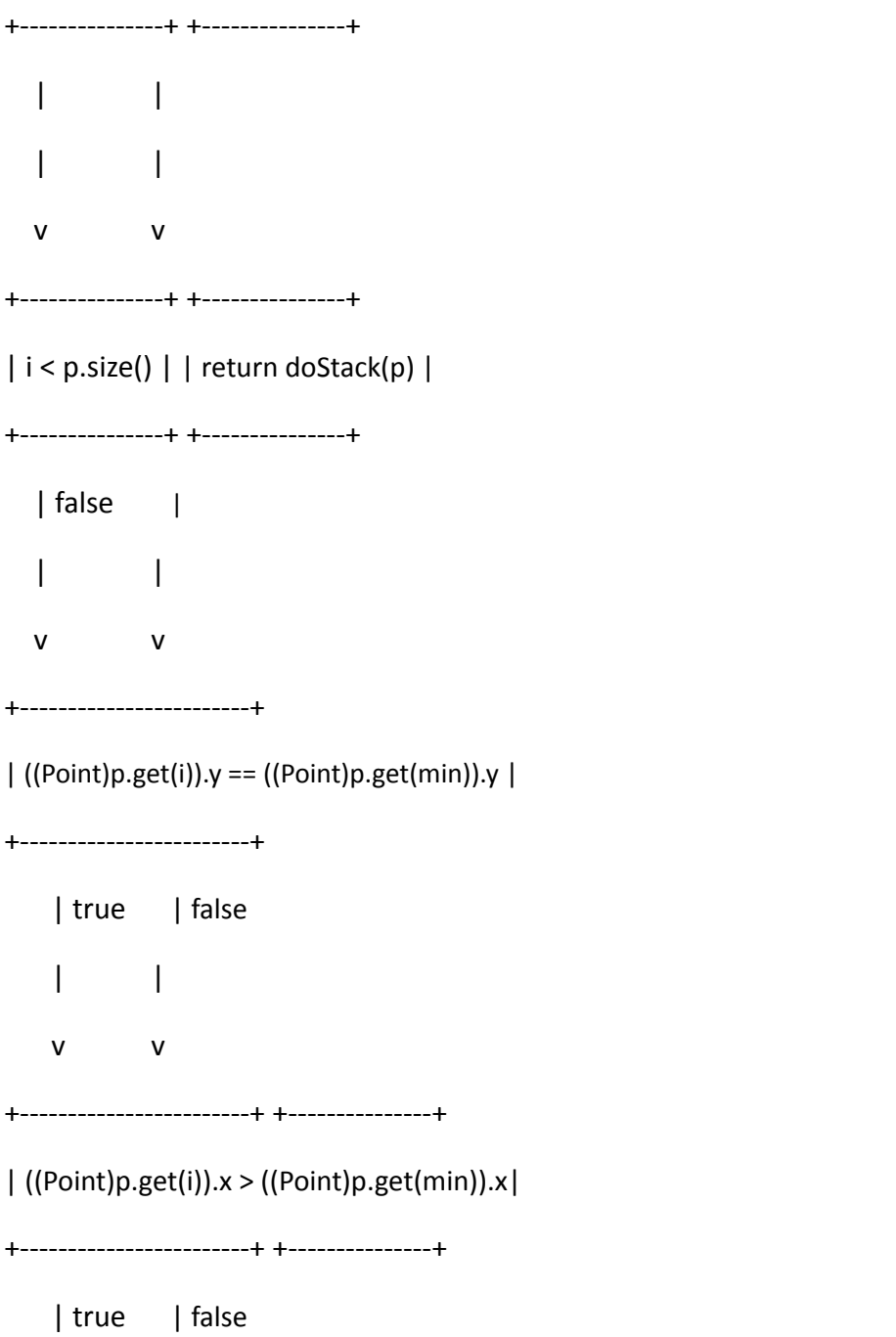
h) Non-positive input:

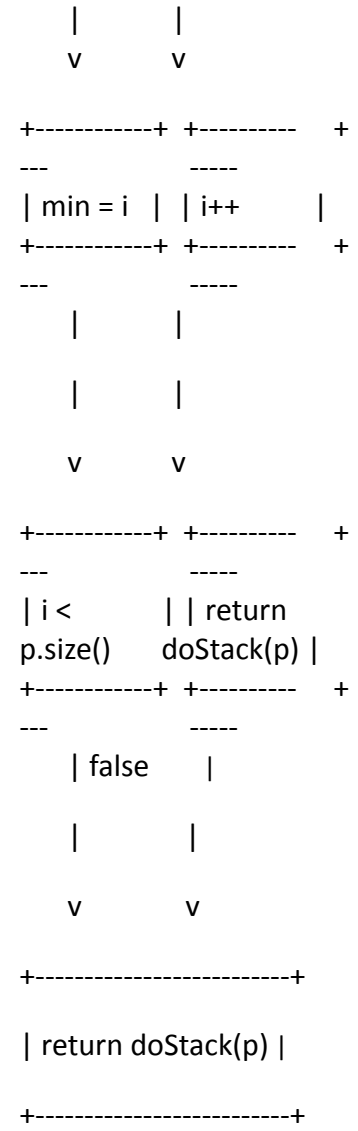
a = -1, b = -2, c = -3 a = 0, b = 1, c = 2

Section B

a) Control Flow graph:




| i < p.size() |
| return doStack(p) |
| false |
| ((Point)p.get(i)).y == ((Point)p.get(min)).y |
| true |
| false |
| ((Point)p.get(i)).x > ((Point)p.get(min)).x|
| true |
| false



b) Test sets for each coverage criterion:

a. Statement Coverage:

Test 1: p = {new Point(0, 0), new Point(1, 1)}
Test 2: p = {new Point(0, 0), new Point(1, 0), new Point(2, 0)}

b. Branch Coverage:

Test 1: p = {new Point(0, 0), new Point(1, 1)}
Test 2: p = {new Point(0, 0), new Point(1, 0), new Point(2, 0)}
Test 3: p = {new Point(0, 0), new Point(1, 0), new Point(1, 1)}

C. Base Coverage:

Test 1: p = {new Point(0, 0), new Point(1, 1)}
Test 2: p = {new Point(0, 0), new Point(1, 0), new Point(2, 0)}
Test 3: p = {new Point(0, 0), new Point(1, 0), new Point(1, 1)}
Test 4: p = {new Point(0, 0), new Point(1, 0), new Point(0, 1)}
Test 5: p = {new Point(0, 0), new Point(0, 1), new Point(1, 1)}