Name : Patni Vikaskumar Prakashbhai

Id: 202001192

Lab - 7

Section A

Based on the input ranges, equivalence classes are given as follows

1. Valid dates: The input triple (day, month, year) that represents a valid date in the Gregorian calendar, such as (14, 11, 2002).
2. Invalid dates: The input triple (day, month, year) that represents an invalid date, such as (30, 2, 2002) or (29, 2, 1900).
3. Out of range dates: The input triple (day, month, year) that are outside the allowed ranges, such as (0, 5, 200) or (15, 15, 2007).

Based on these equivalence classes, we can design the following test cases:

Valid dates:

Input	Expected output
29,07,2007	28,07,2007
1,1,2009	31,12,1999
31,8,1997	30,8,1997

Invalid dates:

Input	Expected output
31,02,2009	Invalid
31,11,2009	Invalid

Out of range dates:

Input	Expected output
0,12,2009	Invalid
15,15,2007	Invalid
31,12,1887	Invalid

Boundary value analysis:

1) Earliest possible date: (1,1,1900)
2) Latest possible date: (31,12,2015)
3) The earliest day of each month: (1, 1, 2000), (1, 2, 2000), (1, 3, 2000),..., (1, 12, 2000)
4) The latest day of each month: (31, 1, 2000), (28, 2, 2000), (31, 3, 2000),..., (31, 12, 2000)
5) Leap year day: (29, 2, 2004)
6) Invalid leap year day: (29, 2, 2001)
7) One day before earliest date: (31, 12, 1899)
8) One day after latest date: (1, 1, 2016)

Boundary test cases:

Input	Expected output
(1,1,1900)	Invalid
(31,12,2015)	(30,12,2015)
(1, 1, 2000)	(31,12,1999)
(31, 1, 2000)	(30,1,2000)
(29,2,2004)	(28,4,2004)
(29,2,2001)	Invalid

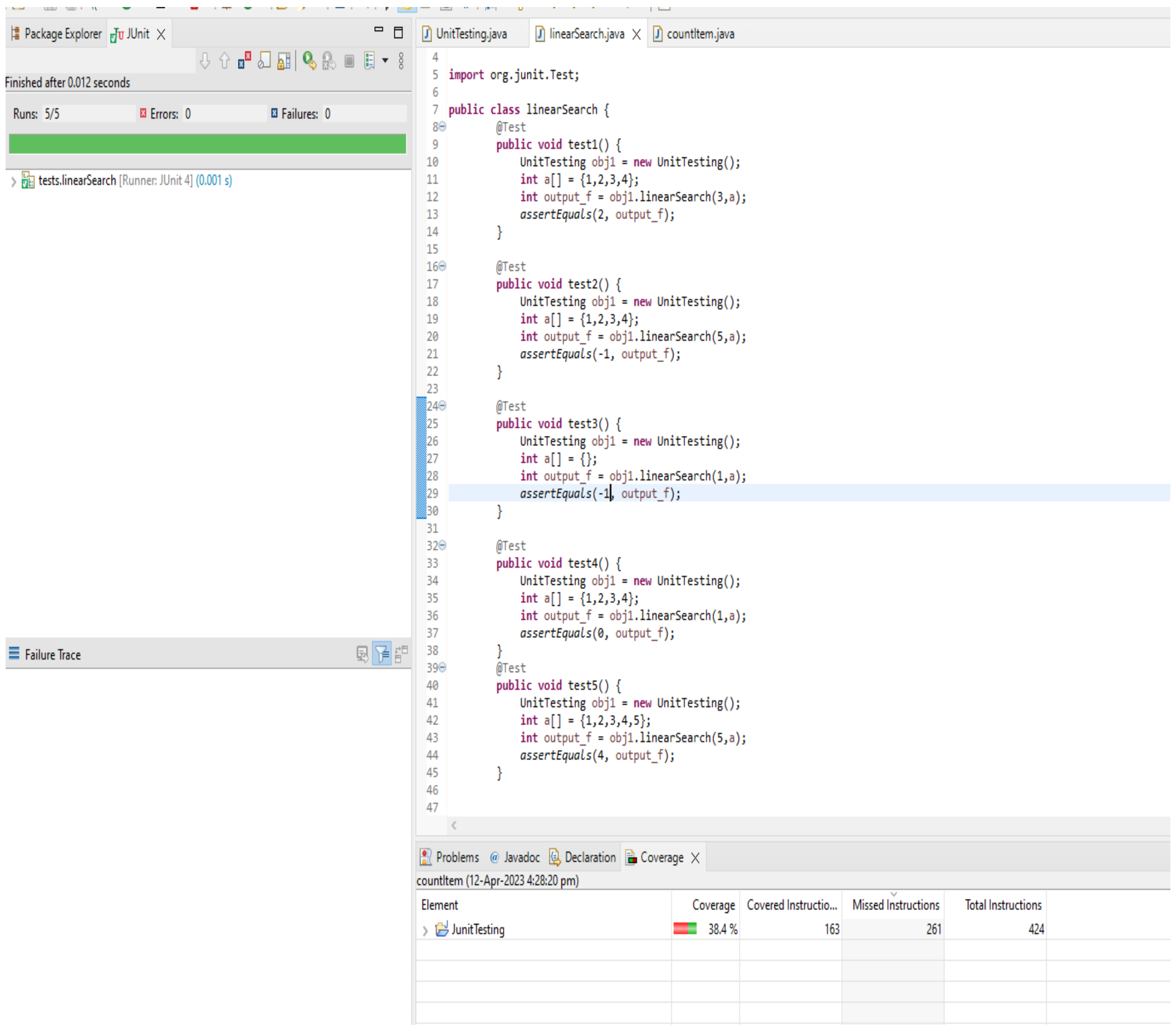
P1:

Equivalence class table

Test Case ID	Array value	target	output	Expected output
1	{1,2,3,4}	3	2	2
2	{1,2,3,4}	5	-1	-1
3	{}	1	-1	-1

Boundary class table

Case id	array	target	output	Expected output
1	{1,2,3,4}	1	0	0
2	{1,2,3,4,5}	5	4	4



Finished after 0.012 seconds

Runs: 5/5 Errors: 0 Failures: 0

tests.linearSearch [Runner: JUnit 4] (0.001 s)

```
import org.junit.Test;

public class linearSearch {
	@Test
	public void test1() {
		UnitTesting obj1 = new UnitTesting();
		int a[] = {1,2,3,4};
		int output_f = obj1.linearSearch(3,a);
		assertEquals(2, output_f);
	}

	@Test
	public void test2() {
		UnitTesting obj1 = new UnitTesting();
		int a[] = {1,2,3,4};
		int output_f = obj1.linearSearch(5,a);
		assertEquals(-1, output_f);
	}

	@Test
	public void test3() {
		UnitTesting obj1 = new UnitTesting();
		int a[] = {};
		int output_f = obj1.linearSearch(1,a);
		assertEquals(-1, output_f);
	}

	@Test
	public void test4() {
		UnitTesting obj1 = new UnitTesting();
		int a[] = {1,2,3,4};
		int output_f = obj1.linearSearch(1,a);
		assertEquals(0, output_f);
	}
	@Test
	public void test5() {
		UnitTesting obj1 = new UnitTesting();
		int a[] = {1,2,3,4,5};
		int output_f = obj1.linearSearch(5,a);
		assertEquals(4, output_f);
	}
```

countItem (12-Apr-2023 4:28:20 pm)

Element	Coverage	Covered Instructio...	Missed Instructions	Total Instructions
JunitTesting	38.4 %	163	261	424

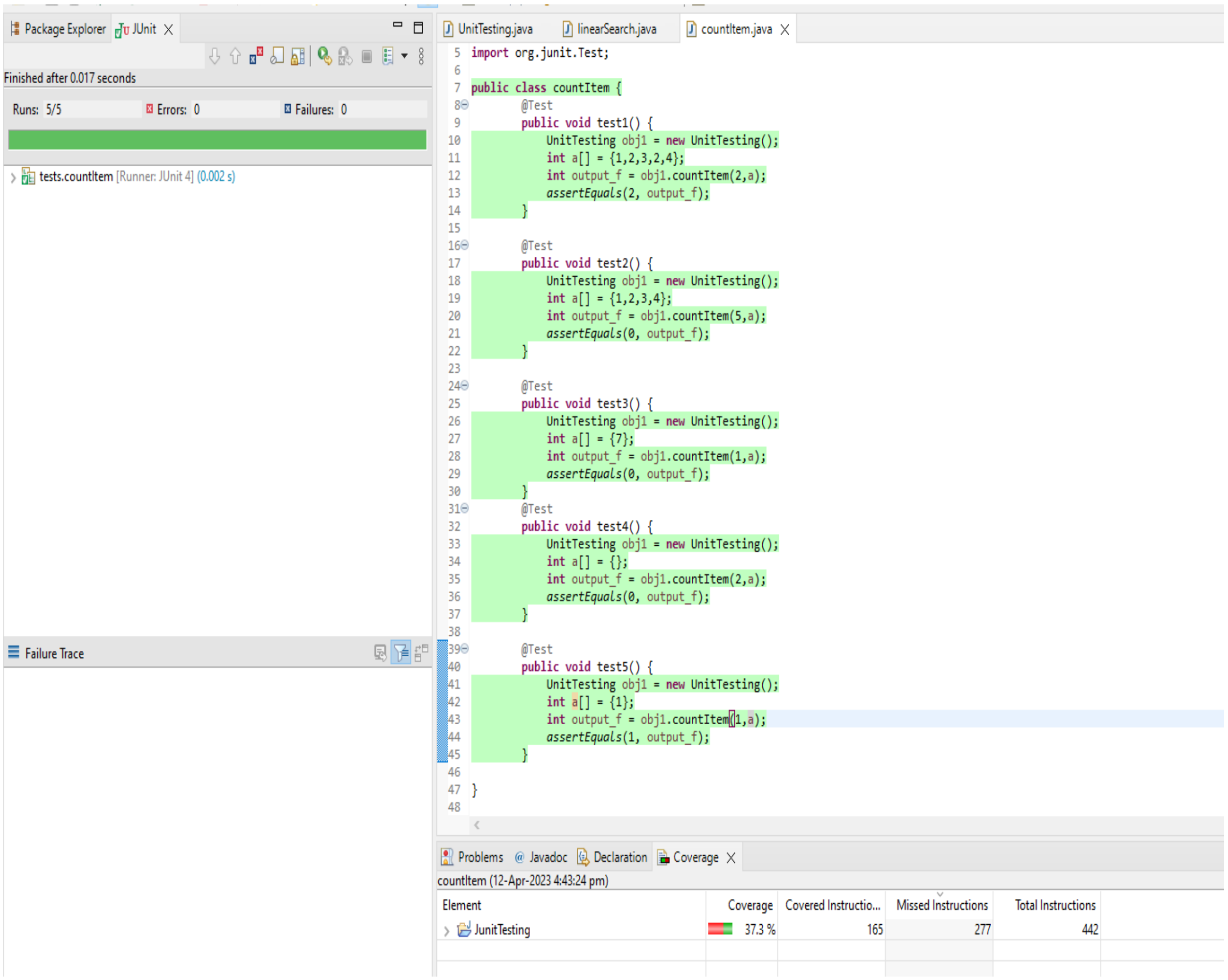
P2:

Equivalence class table

Test Case ID	Array value	target	output	Expected output
1	{1,2,3,2,4}	2	2	2
2	{1,2,3,4}	5	0	0
3	{7}	1	0	0

Boundary class table

Case id	array	target	output	Expected output
1	{}	2	0	0
2	{1}	1	1	1



```java
import org.junit.Test;

public class countItem {
        @Test
        public void test1() {
            UnitTesting obj1 = new UnitTesting();
            int a[] = {1,2,3,2,4};
            int output_f = obj1.countItem(2,a);
            assertEquals(2, output_f);
        }

        @Test
        public void test2() {
            UnitTesting obj1 = new UnitTesting();
            int a[] = {1,2,3,4};
            int output_f = obj1.countItem(5,a);
            assertEquals(0, output_f);
        }

        @Test
        public void test3() {
            UnitTesting obj1 = new UnitTesting();
            int a[] = {7};
            int output_f = obj1.countItem(1,a);
            assertEquals(0, output_f);
        }
        @Test
        public void test4() {
            UnitTesting obj1 = new UnitTesting();
            int a[] = {};
            int output_f = obj1.countItem(2,a);
            assertEquals(0, output_f);
        }

        @Test
        public void test5() {
            UnitTesting obj1 = new UnitTesting();
            int a[] = {1};
            int output_f = obj1.countItem(1,a);
            assertEquals(1, output_f);
        }

}
```

countItem (12-Apr-2023 4:43:24 pm)

Element	Coverage	Covered Instructio...	Missed Instructions	Total Instructions
JunitTesting	37.3 %	165	277	442

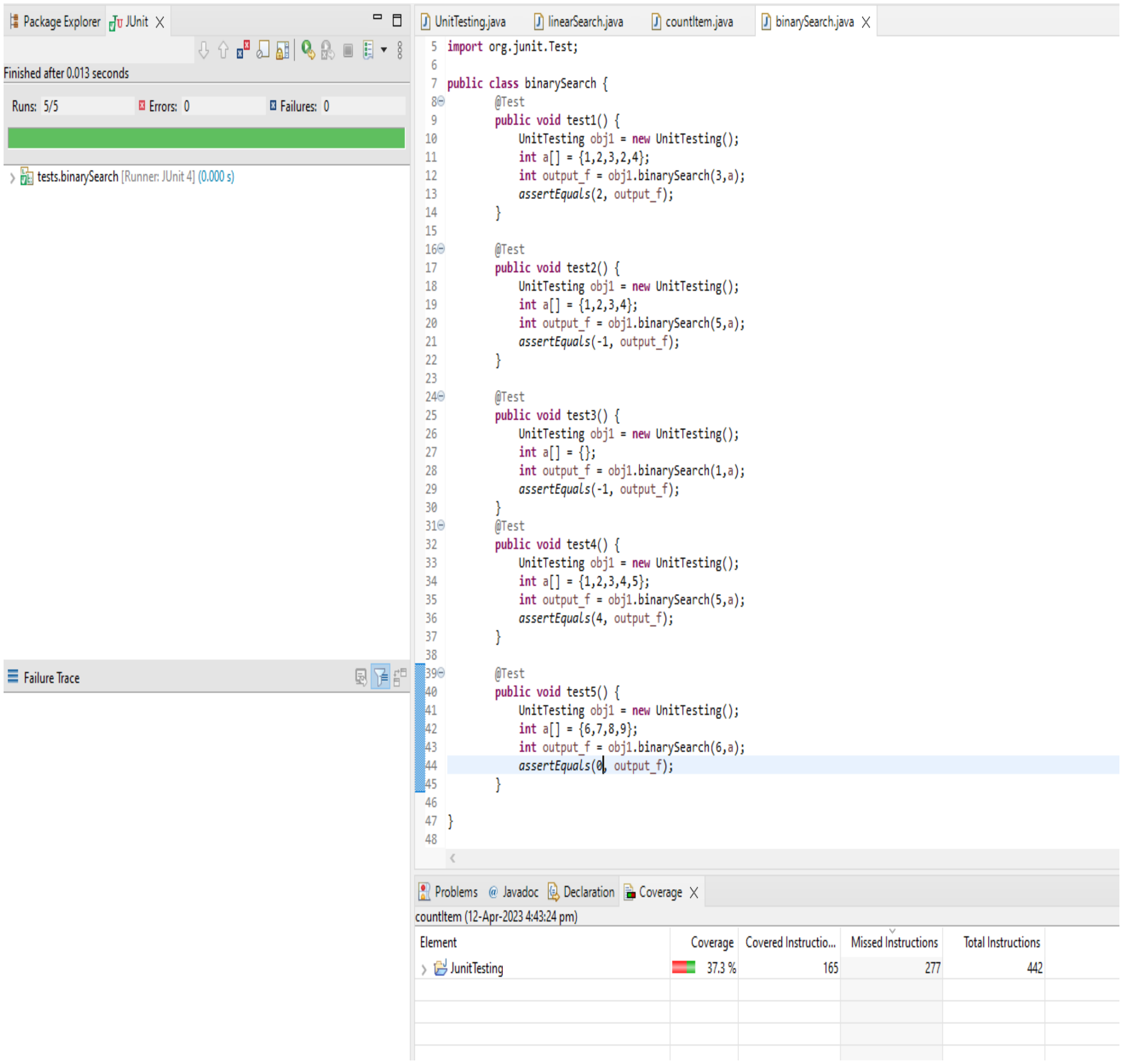
P3:

Equivalence class table

Test Case ID	Array value	target	output	Expected output
1	{1,2,2,3,4}	3	3	3
2	{1,2,3,4}	5	-1	-1
3	{}	1	-1	-1

Boundary class table

Case id	array	target	output	Expected output
1	{1,2,3,4,5}	5	4	4
2	{6,7,8,9}	6	0	0



Finished after 0.013 seconds

Runs: 5/5   Errors: 0   Failures: 0

tests.binarySearch [Runner: JUnit 4] (0.000 s)

```java
import org.junit.Test;

public class binarySearch {
        @Test
        public void test1() {
            UnitTesting obj1 = new UnitTesting();
            int a[] = {1,2,3,2,4};
            int output_f = obj1.binarySearch(3,a);
            assertEquals(2, output_f);
        }

        @Test
        public void test2() {
            UnitTesting obj1 = new UnitTesting();
            int a[] = {1,2,3,4};
            int output_f = obj1.binarySearch(5,a);
            assertEquals(-1, output_f);
        }

        @Test
        public void test3() {
            UnitTesting obj1 = new UnitTesting();
            int a[] = {};
            int output_f = obj1.binarySearch(1,a);
            assertEquals(-1, output_f);
        }
        @Test
        public void test4() {
            UnitTesting obj1 = new UnitTesting();
            int a[] = {1,2,3,4,5};
            int output_f = obj1.binarySearch(5,a);
            assertEquals(4, output_f);
        }

        @Test
        public void test5() {
            UnitTesting obj1 = new UnitTesting();
            int a[] = {6,7,8,9};
            int output_f = obj1.binarySearch(6,a);
            assertEquals(0, output_f);
        }

}
```

countItem (12-Apr-2023 4:43:24 pm)

Element	Coverage	Covered Instructio...	Missed Instructions	Total Instructions
JunitTesting	37.3 %	165	277	442

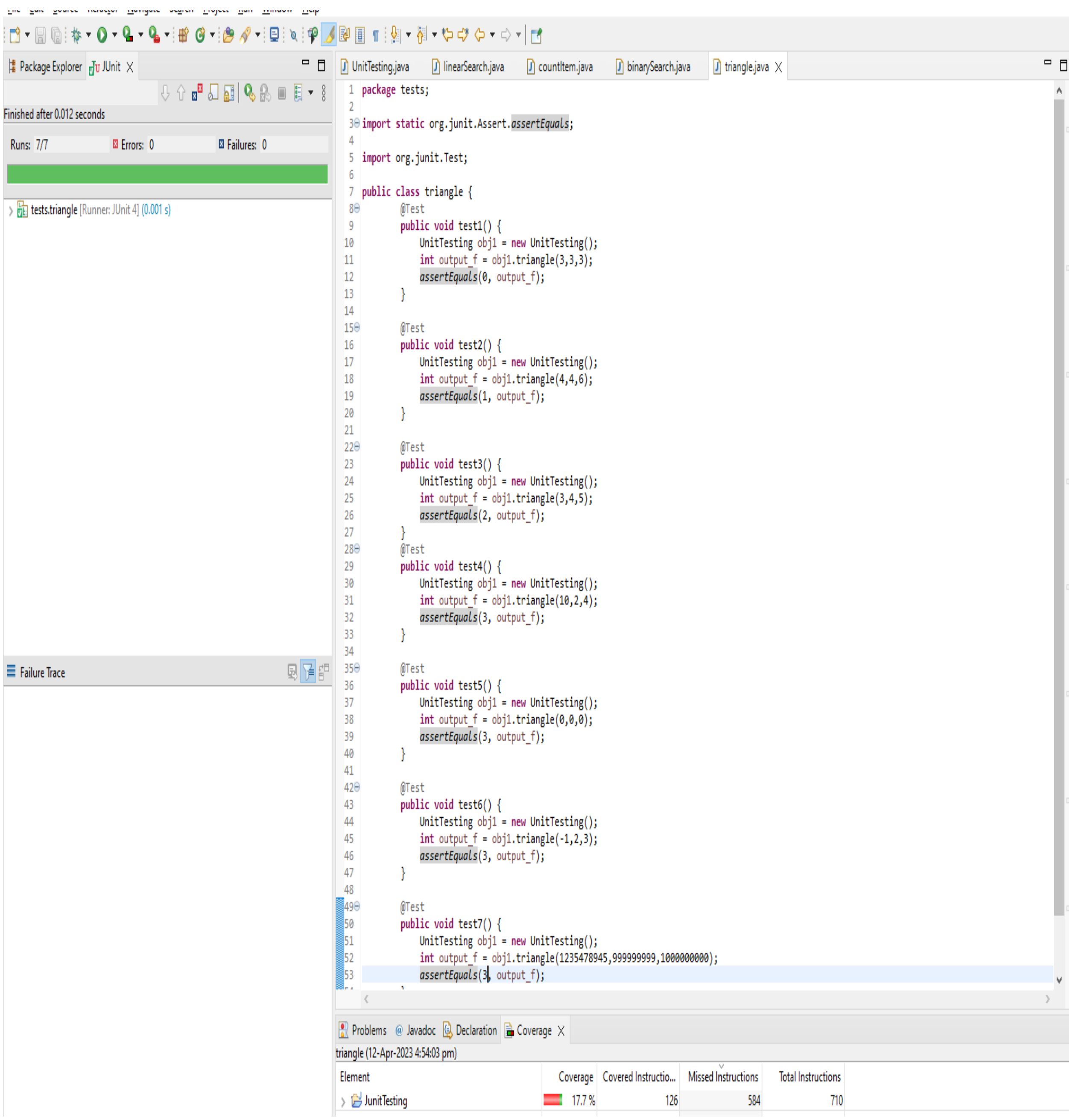
P4

Equivalence class table

Test Case ID	Array value	output	Expected output
1	3,3,3	0	0
2	4,4,6	1	1
3	3,4,5	2	2

Boundary class table

Case id	array	output	Expected output
1	10,2,4	3	3
2	0,0,0	3	3
3	-1,-2,-3	3	3
4	1235478945,999999999,1000000000	3	3



Finished after 0.012 seconds

Runs: 7/7 Errors: 0 Failures: 0

tests.triangle [Runner: JUnit 4] (0.001 s)

```java
package tests;

import static org.junit.Assert.assertEquals;

import org.junit.Test;

public class triangle {
	@Test
	public void test1() {
		UnitTesting obj1 = new UnitTesting();
		int output_f = obj1.triangle(3,3,3);
		assertEquals(0, output_f);
	}

	@Test
	public void test2() {
		UnitTesting obj1 = new UnitTesting();
		int output_f = obj1.triangle(4,4,6);
		assertEquals(1, output_f);
	}

	@Test
	public void test3() {
		UnitTesting obj1 = new UnitTesting();
		int output_f = obj1.triangle(3,4,5);
		assertEquals(2, output_f);
	}
	@Test
	public void test4() {
		UnitTesting obj1 = new UnitTesting();
		int output_f = obj1.triangle(10,2,4);
		assertEquals(3, output_f);
	}

	@Test
	public void test5() {
		UnitTesting obj1 = new UnitTesting();
		int output_f = obj1.triangle(0,0,0);
		assertEquals(3, output_f);
	}

	@Test
	public void test6() {
		UnitTesting obj1 = new UnitTesting();
		int output_f = obj1.triangle(-1,2,3);
		assertEquals(3, output_f);
	}

	@Test
	public void test7() {
		UnitTesting obj1 = new UnitTesting();
		int output_f = obj1.triangle(1235478945,999999999,1000000000);
		assertEquals(3, output_f);
```

triangle (12-Apr-2023 4:54:03 pm)

Element	Coverage	Covered Instructio...	Missed Instructions	Total Instructions
JunitTesting	17.7 %	126	584	710

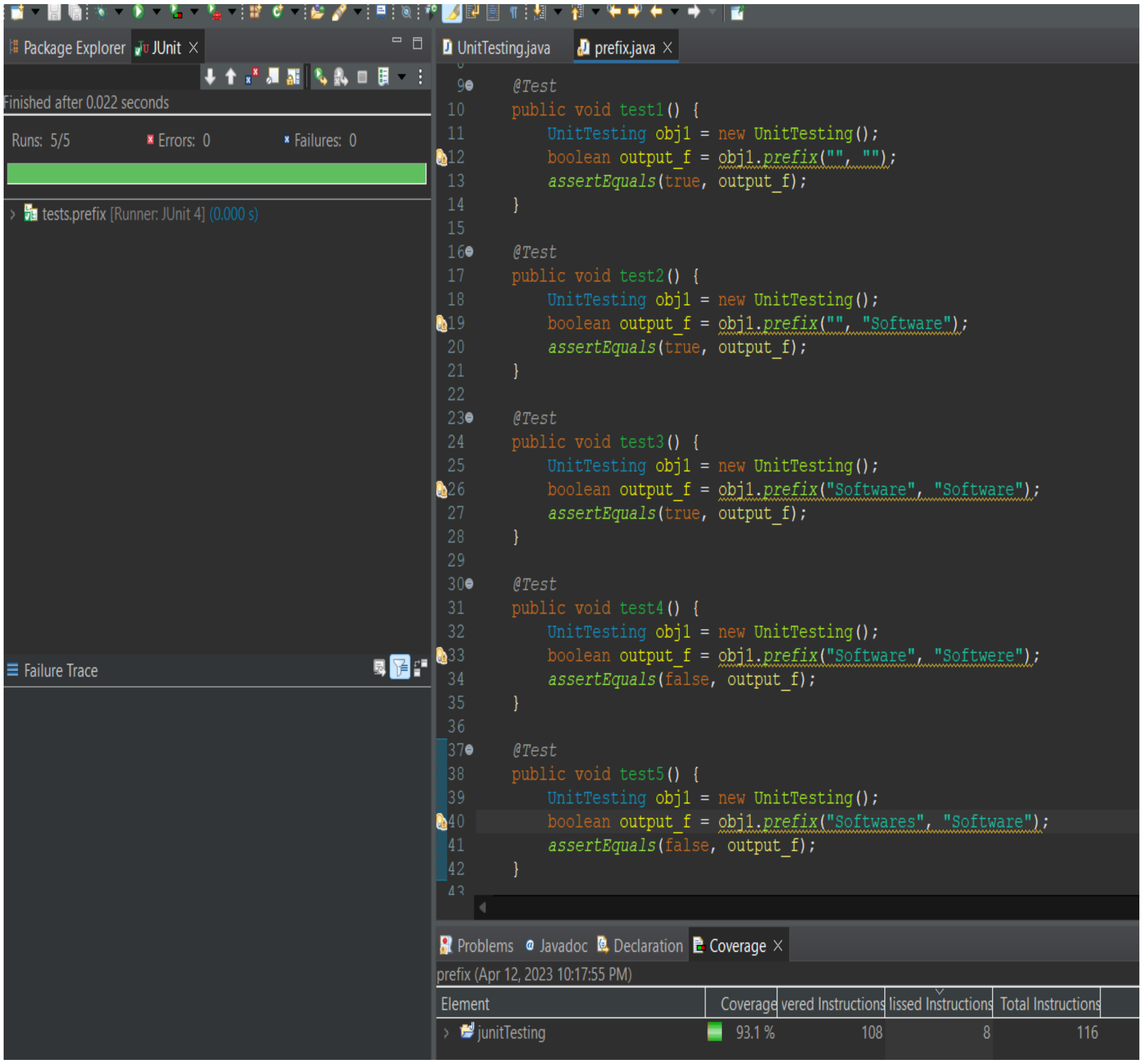
P5:

Equivalence class table

Test case	Strings	output	Expected output
1	“”, “”	true	true
2	“”, “Software”	true	true
3	“Software”, “Software”	true	true
4	“Software”, “Softwere”	false	false
5	“Softwares”, “Software”	false	false

Boundary class tables

Test case	strings	output	Expected output
1	“”, “”	true	true
2	“”, “Software”	true	true
3	“Software”, “Softwere”	False	false
4	“Softwares”, “Software”	false	false



```
Finished after 0.022 seconds

Runs: 5/5        Errors: 0        Failures: 0

> tests.prefix [Runner: JUnit 4] (0.000 s)
```

```java
    @Test
    public void test1() {
        UnitTesting obj1 = new UnitTesting();
        boolean output_f = obj1.prefix("", "");
        assertEquals(true, output_f);
    }

    @Test
    public void test2() {
        UnitTesting obj1 = new UnitTesting();
        boolean output_f = obj1.prefix("", "Software");
        assertEquals(true, output_f);
    }

    @Test
    public void test3() {
        UnitTesting obj1 = new UnitTesting();
        boolean output_f = obj1.prefix("Software", "Software");
        assertEquals(true, output_f);
    }

    @Test
    public void test4() {
        UnitTesting obj1 = new UnitTesting();
        boolean output_f = obj1.prefix("Software", "Softwere");
        assertEquals(false, output_f);
    }

    @Test
    public void test5() {
        UnitTesting obj1 = new UnitTesting();
        boolean output_f = obj1.prefix("Softwares", "Software");
        assertEquals(false, output_f);
    }
```

prefix (Apr 12, 2023 10:17:55 PM)

Element	Coverage	vered Instructions	lissed Instructions	Total Instructions
> junitTesting	🟩 93.1 %	108	8	116

P6:

a) Equivalence classes for the system
EC1: All sides are positive, real numbers.
EC2: One or more sides are negative or zero.
EC3: The sum of the lengths of any two sides is less than or equal to the length of the remaining side (impossible lengths).
EC4: The sum of the lengths of any two sides is greater than the length of the remaining side (possible lengths).

b) Test cases to cover equivalence classes
TC1 (EC1): A=3, B=4, C=5 (right-angled triangle)
TC2 (EC1): A=5, B=5, C=5 (equilateral triangle)
TC3 (EC1): A=5, B=6, C=7 (scalene triangle)
TC4 (EC1): A=5, B=5, C=7 (isosceles triangle)
TC5 (EC2): A=-2, B=4, C=5 (invalid input)
TC6 (EC2): A=0, B=4, C=5 (invalid input)

c) Test cases for boundary condition A+B>C
TC7 (EC4): A=4, B=3, C=6 (sum of A and B > C)

d) Test case for boundary condition A=C
TC8 (EC4): A=5, B=6, C=5 (A equals to C)

e)Test case for the boundary condition A=B=C
TC9 (EC4): A=5, B=5, C=5 (all sides are equal)

f) Test case for the boundary condition A^2 + B^2 = C^2
TC10 (EC4): A=3, B=4, C=5 (right-angled triangle)

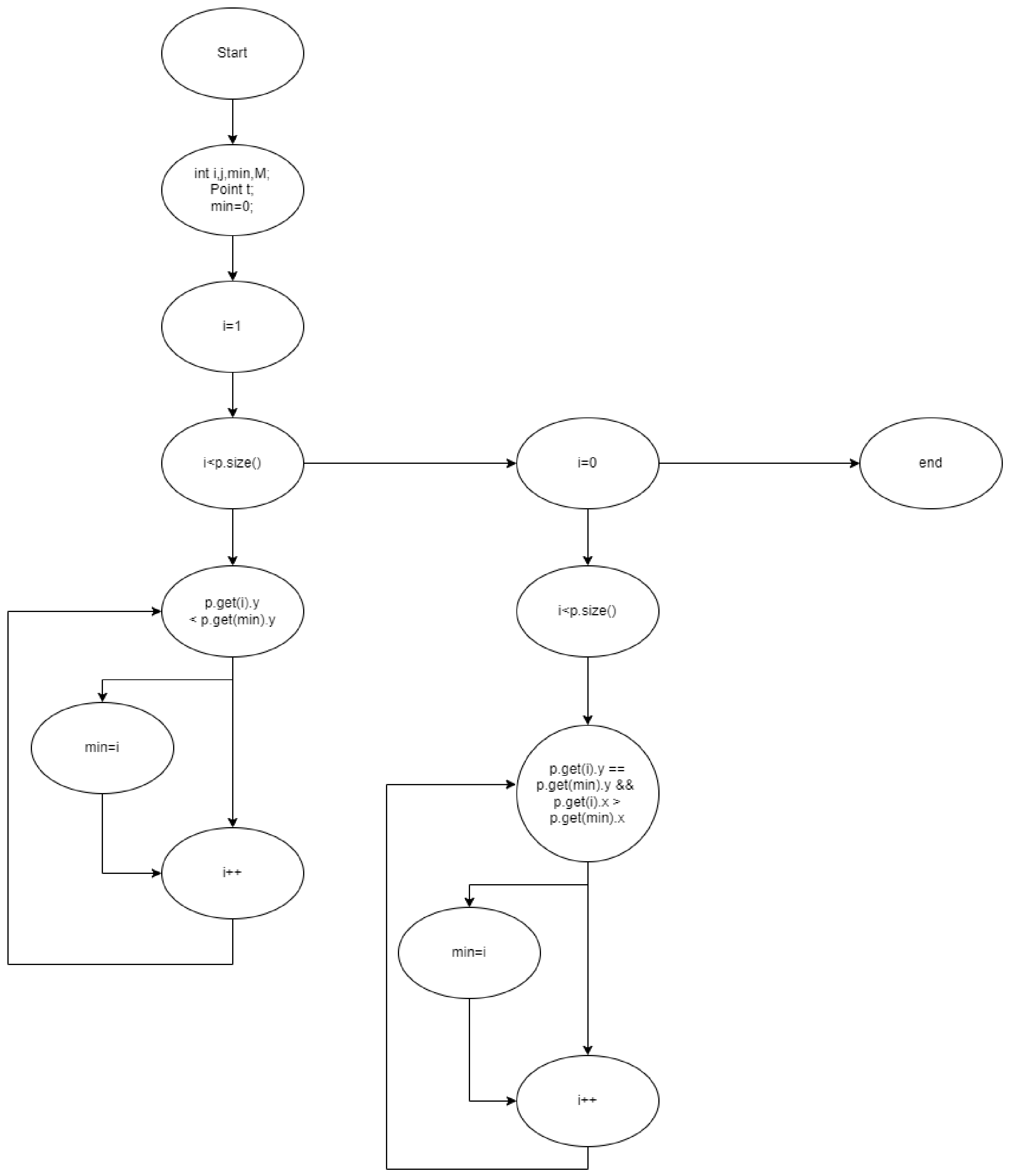
g) Test cases for the boundary condition of non-triangle case:
TC11 (EC3): A=2, B=2, C=4 (sum of A and B is equal to C)

h) For non-positive input, identify test points.
TP1 (EC2): A=0, B=4, C=5 (invalid input)
TP2 (EC2): A=-2, B=4, C=5 (invalid input)

The Test cases TC1 to TC10 covers all identified equivalence classes.

Section B

Control flow graph of doGraham method



Statement coverage test cases: every statement in code is executed at least once

Test 1: p = empty vector
Test 2: p = vector with one point
Test 3: p = vector with two points with the same y component
Test 4: p = vector with two points with different y components
Test 5: p = vector with three or more points with different y components
Test 6: p = vector with three or more points with the same y component

Branch coverage test sets: every branch in code is executed at least once
Test 1: p = empty vector
Test 2: p = vector with one point
Test 3: p = vector with two points with the same y component
Test 4: p = vector with two points with different y components
Test 5: p = vector with three or more points with different y components, and none of them have the same x component
Test 6: p = vector with three or more points with the same y component, and some of them have the same x component
Test 7: p = vector with three or more points with the same y component, and all of them have the same x component

Basic condition coverage test sets: every boolean expression is executed at least once

Test 1: p = empty vector
Test 2: p = vector with one point
Test 3: p = vector with two points with the same y component, and the first point has a smaller x component
Test 4: p = vector with two points with the same y component, and the second point has a smaller x component
Test 5: p = vector with two points with different y components
Test 6: p = vector with three or more points with different y components, and none of them have the same x component
Test 7: p = vector with three or more points with the same y component, and some of them have the same x component
Test 8: p = vector with three or more points with the same y component, and all of them have the same x component.

Examples of such test cases

Test cases :
1) p=[(x=2,y=2),(x=2,y=3),(x=1,y=3),(x=1,y=4)]
2) p=[(x=2,y=3),(x=3,y=4),(x=1,y=2),(x=5,y=6)]
3) p=[(x=1,y=5),(x=2,y=7),(x=3,y=5),(x=4,y=5),(x=5,y=6)] 4) p=[(x=1,y=2)]
5) p=[]

These 5 test cases cover all the tests discussed above.