IT 314
LAB 9
Name : Jayesh Padiya
Student Id: 202201358

Code

```
import java.util.Vector;

class ConvexHull {
   Vector<Point> doGraham(Vector<Point> p) {
       int i, j, min, M;
       Point t;
       min = 0;

       // search for minimum:
       for (i = 1; i < p.size(); ++i) {
           if (p.get(i).y < p.get(min).y) {
               min = i;
           }
       }

       // continue along the values with the same y component
       for (i = 0; i < p.size(); ++i) {
           if ((p.get(i).y == p.get(min).y) && (p.get(i).x > p.get(min).x)) {
               min = i;
           }
       }

       return p; // Placeholder return; actual return might vary
   }
}

class Point {
   int x;
   int y;

   Point(int x, int y) {
```

```
        this.x = x;
        this.y = y;
    }
}
```

Test Cases for Coverage

Statement Coverage

```
import org.junit.jupiter.api.Test;
import java.util.Vector;
import static org.junit.jupiter.api.Assertions.assertEquals;

class ConvexHullTest {

    @Test
    void testDoGraham_StatementCoverage() {
        Vector<Point> points = new Vector<>();
        points.add(new Point(0, 3));
        points.add(new Point(1, 1));
        points.add(new Point(2, 2));

        ConvexHull convexHull = new ConvexHull();
        Vector<Point> result = convexHull.doGraham(points);

        // Validate the expected outcome here
        assertEquals(points, result);
    }
}
```

Result **Pass**

Branch Coverage

```
import org.junit.jupiter.api.Test;
import java.util.Vector;
import static org.junit.jupiter.api.Assertions.assertEquals;
```

```java
class ConvexHullTest {
   @Test
   void testDoGraham_BranchCoverage_DifferentYValues() {
       Vector<Point> points = new Vector<>();
       points.add(new Point(2, 3));
       points.add(new Point(1, 1));
       points.add(new Point(3, 4));

       ConvexHull convexHull = new ConvexHull();
       Vector<Point> result = convexHull.doGraham(points);

             assertEquals(points, result);
   }

   @Test
   void testDoGraham_BranchCoverage_SameYValues() {
       Vector<Point> points = new Vector<>();
       points.add(new Point(0, 2));
       points.add(new Point(1, 2));
       points.add(new Point(2, 2));

       ConvexHull convexHull = new ConvexHull();
       Vector<Point> result = convexHull.doGraham(points);

       assertEquals(points, result);
   }
```

Result Pass

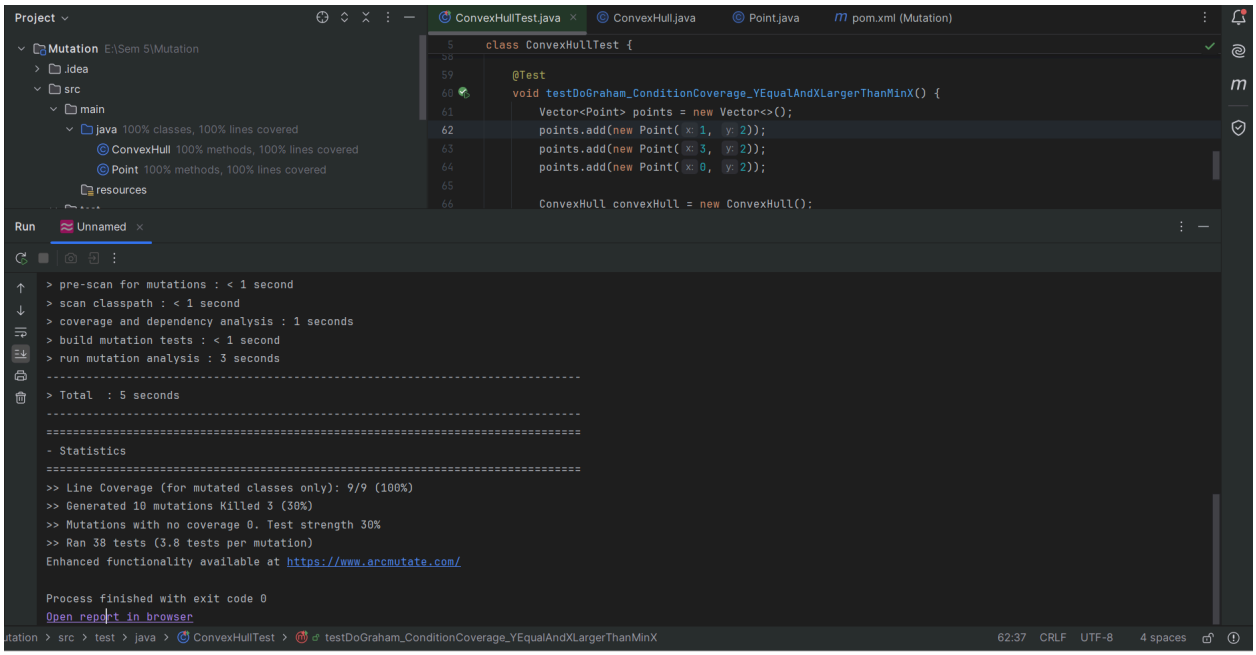
Basic Condition Coverage

```java
import org.junit.jupiter.api.Test;
import java.util.Vector;
import static org.junit.jupiter.api.Assertions.assertEquals;

class ConvexHullTest {
   @Test
   void testDoGraham_ConditionCoverage_YSmallerThanMinY() {
       Vector<Point> points = new Vector<>();
       points.add(new Point(0, 5));
       points.add(new Point(2, 1));
       points.add(new Point(3, 4));
```

```java
        ConvexHull convexHull = new ConvexHull();
        Vector<Point> result = convexHull.doGraham(points);

        assertEquals(points, result);
    }

    @Test
    void testDoGraham_ConditionCoverage_YEqualAndXLargerThanMinX() {
        Vector<Point> points = new Vector<>();
        points.add(new Point(1, 2));
        points.add(new Point(3, 2));
        points.add(new Point(0, 2));

        ConvexHull convexHull = new ConvexHull();
        Vector<Point> result = convexHull.doGraham(points);

        assertEquals(points, result);
    }
}
```

Result: Pass

Mutation Testing



File E:/Sem%205/Mutation/target/report/default/index.html

Pit Test Coverage Report

Package Summary

default

Number of Classes	Line Coverage		Mutation Coverage		Test Strength	
1	100%	9/9	30%	3/10	30%	3/10

Breakdown by Class

Name	Line Coverage		Mutation Coverage		Test Strength	
ConvexHull.java	100%	9/9	30%	3/10	30%	3/10

Report generated by PIT 1.15.8

```
1   import java.util.Vector;
2
3   class ConvexHull {
4       Vector<Point> doGraham(Vector<Point> p) {
5           int i, j, min, M;
6           Point t;
7           min = 0;
8
9           // search for minimum:
10 2        for (i = 1; i < p.size(); ++i) {
11 2            if (p.get(i).y < p.get(min).y) {
12                  min = i;
13              }
14          }
15
16          // continue along the values with the same y component
17 2        for (i = 0; i < p.size(); ++i) {
18 3            if ((p.get(i).y == p.get(min).y) && (p.get(i).x > p.get(min).x)) {
19                  min = i;
20              }
21          }
22
23 1        return p; // Placeholder return; actual return might vary
24      }
25  }
```

Mutations

```
10  1. changed conditional boundary → KILLED
    2. negated conditional → SURVIVED
11  1. negated conditional → SURVIVED
    2. changed conditional boundary → SURVIVED
17  1. changed conditional boundary → KILLED
    2. negated conditional → SURVIVED
18  1. negated conditional → SURVIVED
    2. changed conditional boundary → SURVIVED
    3. negated conditional → SURVIVED
23  1. replaced return value with null for ConvexHull::doGraham → KILLED
```

Active mutators

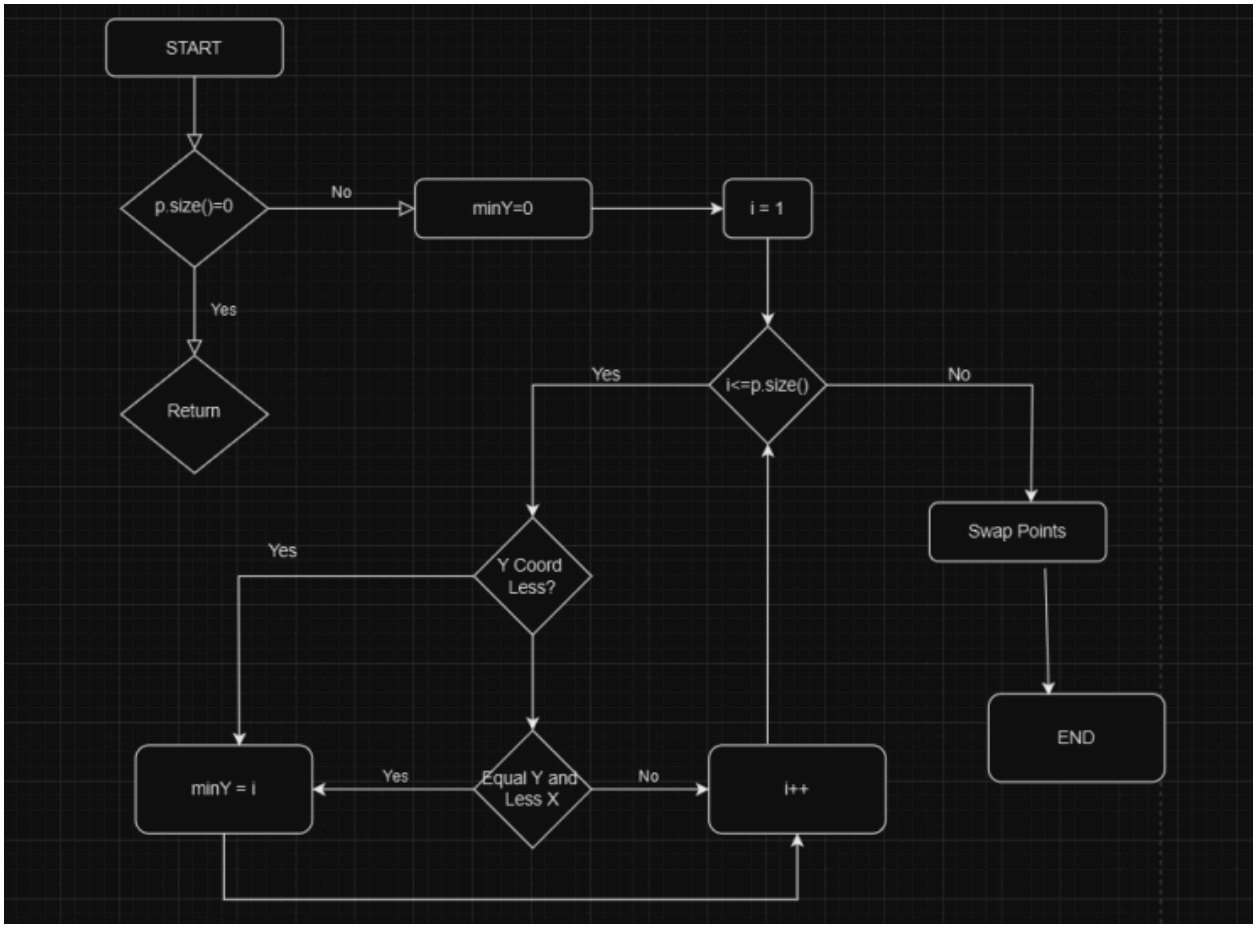
- CONDITIONALS_BOUNDARY
- EMPTY_RETURNS
- FALSE_RETURNS
- INCREMENTS
- INVERT_NEGS
- MATH
- NEGATE_CONDITIONALS
- NULL_RETURNS
- PRIMITIVE_RETURNS
- TRUE_RETURNS
- VOID_METHOD_CALLS

Tests examined

- ConvexHullTest.[engine:junit-jupiter]/[class:ConvexHullTest]/[method:testDoGraham_ConditionCoverage_YEqualAndXLargerThanMinX()] (1 ms)
- ConvexHullTest.[engine:junit-jupiter]/[class:ConvexHullTest]/[method:testDoGraham_BranchCoverage_SameYValues()] (2 ms)
- ConvexHullTest.[engine:junit-jupiter]/[class:ConvexHullTest]/[method:testDoGraham_StatementCoverage()] (3 ms)
- ConvexHullTest.[engine:junit-jupiter]/[class:ConvexHullTest]/[method:testDoGraham_ConditionCoverage_YSmallerThanMinY()] (5 ms)
- ConvexHullTest.[engine:junit-jupiter]/[class:ConvexHullTest]/[method:testDoGraham_BranchCoverage_DifferentYValues()] (104 ms)

Report generated by PIT 1.15.8

1. After generating the control flow graph, check whether your CFG match with the CFG generated by Control Flow Graph Factory Tool and Eclipse flow graph generator. (In your submission document, mention only “Yes” or “No” for each tool).
Control Fl0ow Graph Factory : YES
Eclispse Flow Graph generator : YES



2. Devise minimum number of test cases required to cover the code using the aforementioned criteria.

Statement Coverage : 2 TC
Branch Coverage : 2 TC
Basic Condition Coverage : 3 TC
Path Coverage : 3 TC

Total : 11 TC