Random Walk

1. Conclusion

- First Observation - as per the practical experiment on a random walk, the relation between the n steps and d distance is directly proportional as n increases the value of d also increases
- Second Observation - while running the six values of n, with each of these runnings at least 10 times, it was observed that the value of d doesn't vary with very large values, instead, the values of d vary with a smaller value range.

2. Evidence

- Example - The value set of n used in the practical experiment was of the following range 40, 50, 60, 100, 120, 140, 500, and 600 and d range vary with small value observed in the below table

n	40	50	60	100	120	140	500	600
d	5.83	6.46	6.90	8.92	9.81	10.80	19.37	22.44
	5.52	6.26	6.56	9.10	9.77	10.17	19.28	20.53
	5.78	6.16	6.67	8.75	9.90	10.18	19.55	22.60
	5.57	6.39	6.72	8.94	9.87	10.33	20.48	20.22
	5.55	6.23	7.04	8.28	9,02	10.34	20.10	21.46
	5.40	6.43	7.01	8.53	9.66	10.47	20.25	20.97
	5.74	6.23	6.73	9.00	9.87	10.08	20.81	21.78
	5.57	6.33	6.68	8.90	9.85	10.98	19.54	21.75
	5.52	6.35	7.33	9.27	9.70	10.32	18.98	21.73
	5.64	6.29	6.84	8.63	9.12	10.26	18.86	21.54

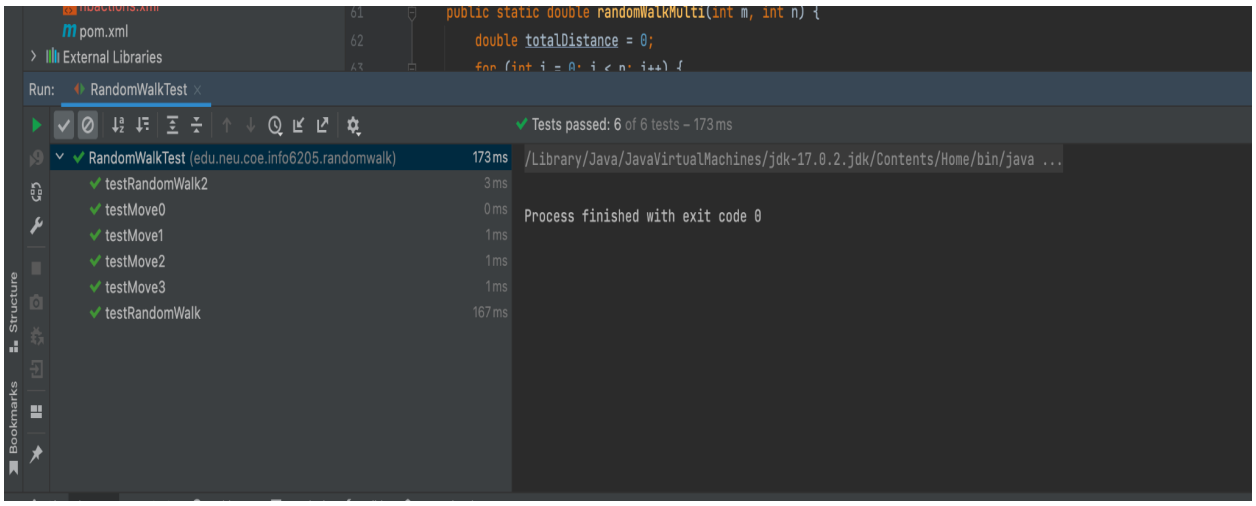
3. Code

```
package edu.neu.coe.info6205.randomwalk;

import java.util.Random;

public class RandomWalk {

  private int x = 0;
  private int y = 0;

  private final Random random = new Random();

  /**
   * Private method to move the current position, that's to say the drunkard moves
   *
   * @param dx the distance he moves in the x direction
   * @param dy the distance he moves in the y direction
   */
  private void move(int dx, int dy) {
     x = x + dx;
     y = y + dy;
  }

  /**
   * Perform a random walk of m steps
   *
   * @param m the number of steps the drunkard takes
   */
  private void randomWalk(int m) {
     for(int i=0; i<m; i++) {
        randomMove();
     }
  }

  /**
   * Private method to generate a random move according to the rules of the situation.
   * That's to say, moves can be (+-1, 0) or (0, +-1).
```

```
   */
  private void randomMove() {
     boolean ns = random.nextBoolean();
     int step = random.nextBoolean() ? 1 : -1;
     move(ns ? step : 0, ns ? 0 : step);
  }

  /**
   * Method to compute the distance from the origin (the lamp-post where the drunkard starts)
to his current position.
   *
   * @return the (Euclidean) distance from the origin to the current position.
   */
  public double distance() {
     double distanceVal = Math.sqrt(Math.pow(x, 2) + Math.pow(y, 2));
     return distanceVal;
  }

  /**
   * Perform multiple random walk experiments, returning the mean distance.
   *
   * @param m the number of steps for each experiment
   * @param n the number of experiments to run
   * @return the mean distance
   */
  public static double randomWalkMulti(int m, int n) {
     double totalDistance = 0;
     for (int i = 0; i < n; i++) {
        RandomWalk walk = new RandomWalk();
        walk.randomWalk(m);
        totalDistance = totalDistance + walk.distance();
     }
     return totalDistance / n;
  }

  public static void main(String[] args) {
     if (args.length == 0)
        throw new RuntimeException("Syntax: RandomWalk steps [experiments]");
     int m = Integer.parseInt(args[0]);
     int n = 30;
```

```
        if (args.length > 1) n = Integer.parseInt(args[1]);
        double meanDistance = randomWalkMulti(m, n);
        System.out.println(m + " steps: " + meanDistance + " over " + n + " experiments");
    }

}
```

4. Unit Test Passing



NEUID - 002922398
Email - rajan.nai@northeastern.edu
Name - Naina Rajan