Random Walk

Conclusion

As per the experiments performed for 9 samples of **n** with a 100 experiments in each experiment cycle, the mean distance **d** remained approximately near for every 10 experiment cycle results. Also it was obvious, with increasing value of n, the distance d increased.

In our experiment we can observe, the distance covered by a drunken man with a fixed **n** number of random steps remains almost closer, irrespective of the random moves made over multiple experiment cycles.

n - number of steps
d - mean of euclidean distance
Number of tests performed - 100 experiments per cycle, 10 cycles for each n steps

Evidence

n steps	50	60	70	80	90	100	150	200	250
Cycle 1	6.51	6.45	7.19	8.25	7.75	9.14	11.71	12.74	14.99
Cycle 2	6.18	6.68	7.42	8.02	8.18	8.96	10.72	11.93	13.64
Cycle 3	6.66	7.16	7.19	8.35	8.61	8.81	10.12	12.60	15.13
Cycle 4	6.05	6.63	6.74	7.65	8.11	8.84	9.94	12.91	14.02
Cycle 5	6.62	7.17	6.77	7.91	8.57	8.04	11.90	12.87	14.30
Cycle 6	5.99	6.78	8.21	8.43	8.19	8.87	10.88	12.43	14.01
Cycle 7	6.30	6.55	7.77	7.50	8.70	9.43	10.55	12.43	13.83
Cycle 8	6.04	6.74	7.48	8.18	8.98	9.08	11.31	11.97	14.10
Cycle 9	6.22	6.79	8.01	8.00	8.56	8.60	10.62	11.36	14.67
Cycle 10	6.56	6.87	7.16	8..23	9.49	8.79	10.69	12.94	15.17

The value of d remains in a closer range for every cycle with the same number of steps.

Code

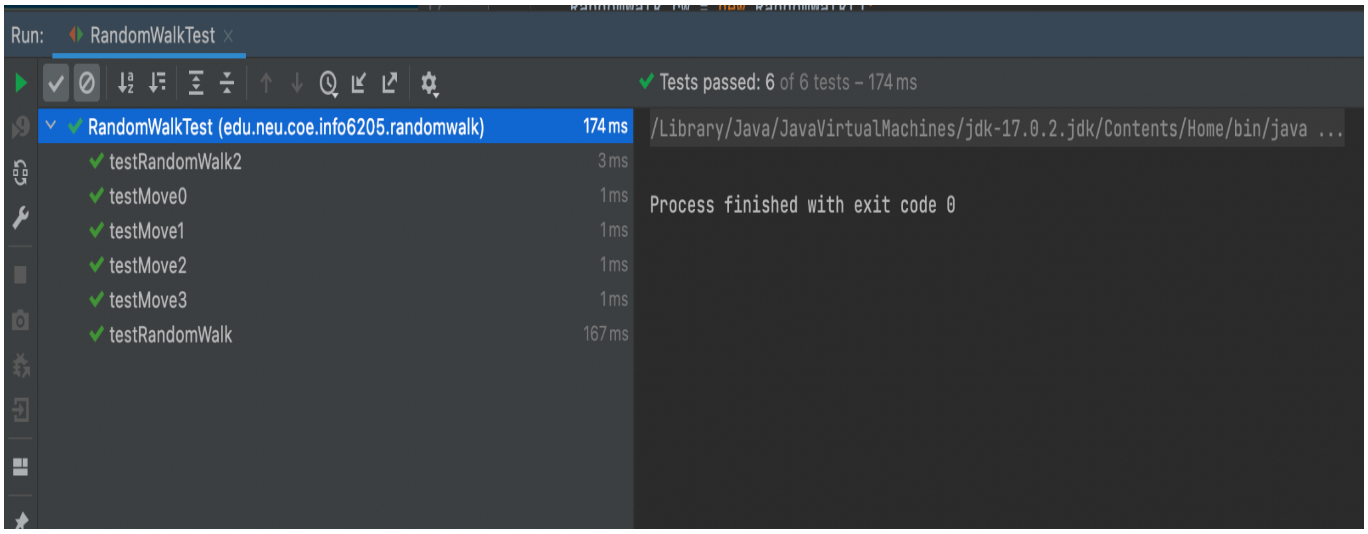
RandomWalk.java

```java
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() {
        // considering origin as post with coordinates (0,0) & (x,y) as final
position after m steps
        double distance = Math.sqrt(Math.pow(x, 2) + Math.pow(y, 2));
        return distance;
    }

    /**
     * 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");
    }

}
```

Screen shots

Unit Tests



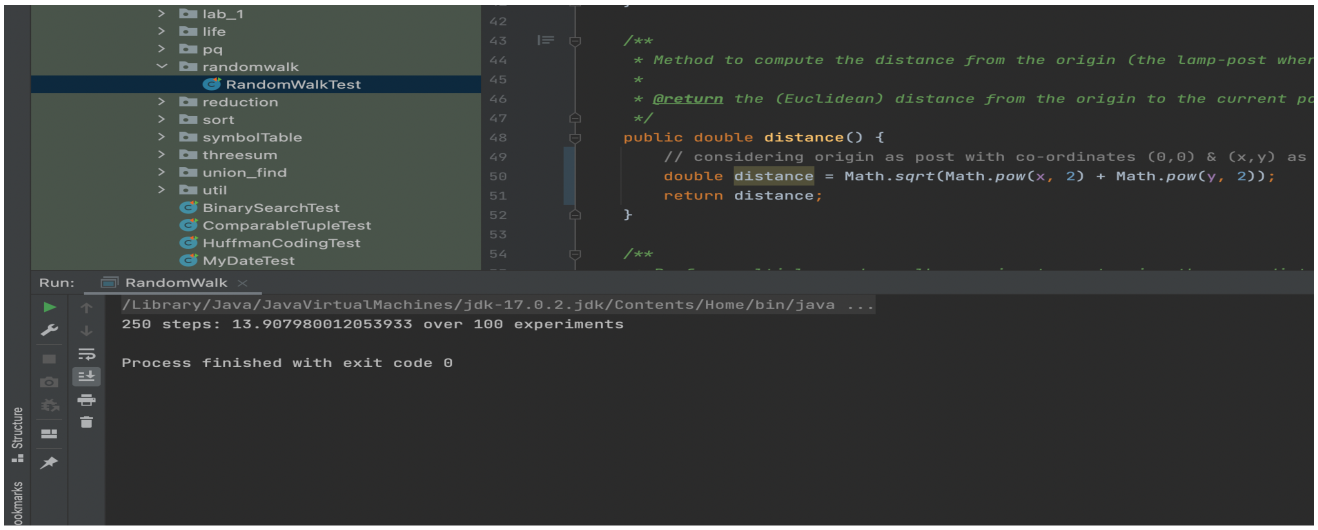
Run: RandomWalkTest

Tests passed: 6 of 6 tests – 174 ms

```
RandomWalkTest (edu.neu.coe.info6205.randomwalk)   174 ms
    testRandomWalk2                                  3 ms
    testMove0                                        1 ms
    testMove1                                        1 ms
    testMove2                                        1 ms
    testMove3                                        1 ms
    testRandomWalk                                 167 ms
```

```
/Library/Java/JavaVirtualMachines/jdk-17.0.2.jdk/Contents/Home/bin/java ...

Process finished with exit code 0
```

Program console



```
/**
 * Method to compute the distance from the origin (the lamp-post wher
 *
 * @return the (Euclidean) distance from the origin to the current po
 */
public double distance() {
    // considering origin as post with co-ordinates (0,0) & (x,y) as
    double distance = Math.sqrt(Math.pow(x, 2) + Math.pow(y, 2));
    return distance;
}

/**
```

Run: RandomWalk

```
/Library/Java/JavaVirtualMachines/jdk-17.0.2.jdk/Contents/Home/bin/java ...
250 steps: 13.907980012053933 over 100 experiments

Process finished with exit code 0
```

Report by,
Thomas John
NEU ID: 002933800