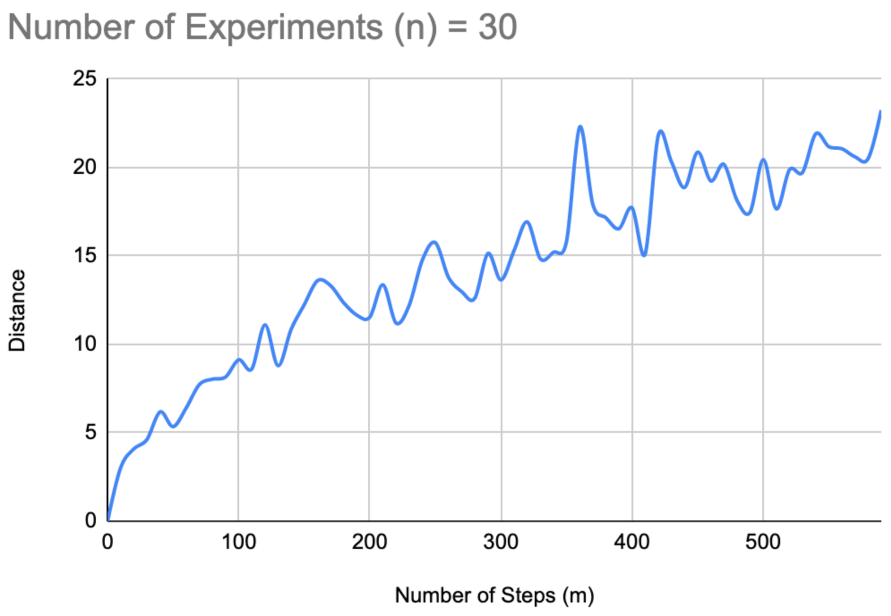
Random Walk

1. **Conclusion:-** The distance d and the number of steps m has the following relationship: $d = \sqrt{m}$
   The relationship between the distance d and the number of steps m in a random walk problem can be derived from a mathematical concept called the central limit theorem. The central limit theorem states that the sum of a large number of random variables will tend to be approximately normally distributed, regardless of the underlying distribution of the variables.
   In the case of a random walk, if we consider the distance traveled in the x-direction and the distance traveled in the y-direction as two independent random variables, the total distance traveled (d) can be represented as the square root of the sum of the squares of the x and y distances (d = sqrt(x^2 + y^2)).
   As the number of steps increases, the x and y distances will become larger, and the total distance will also increase. However, since the x and y distances are independent, the total distance will increase at a slower rate, following the square root function. This is why we use the square root relationship between distance and steps.

2. **Evidence:- (For Number of steps from 0 to 600, data points plotted with a gap of 10 and 30 Experiments carried out on each step)**

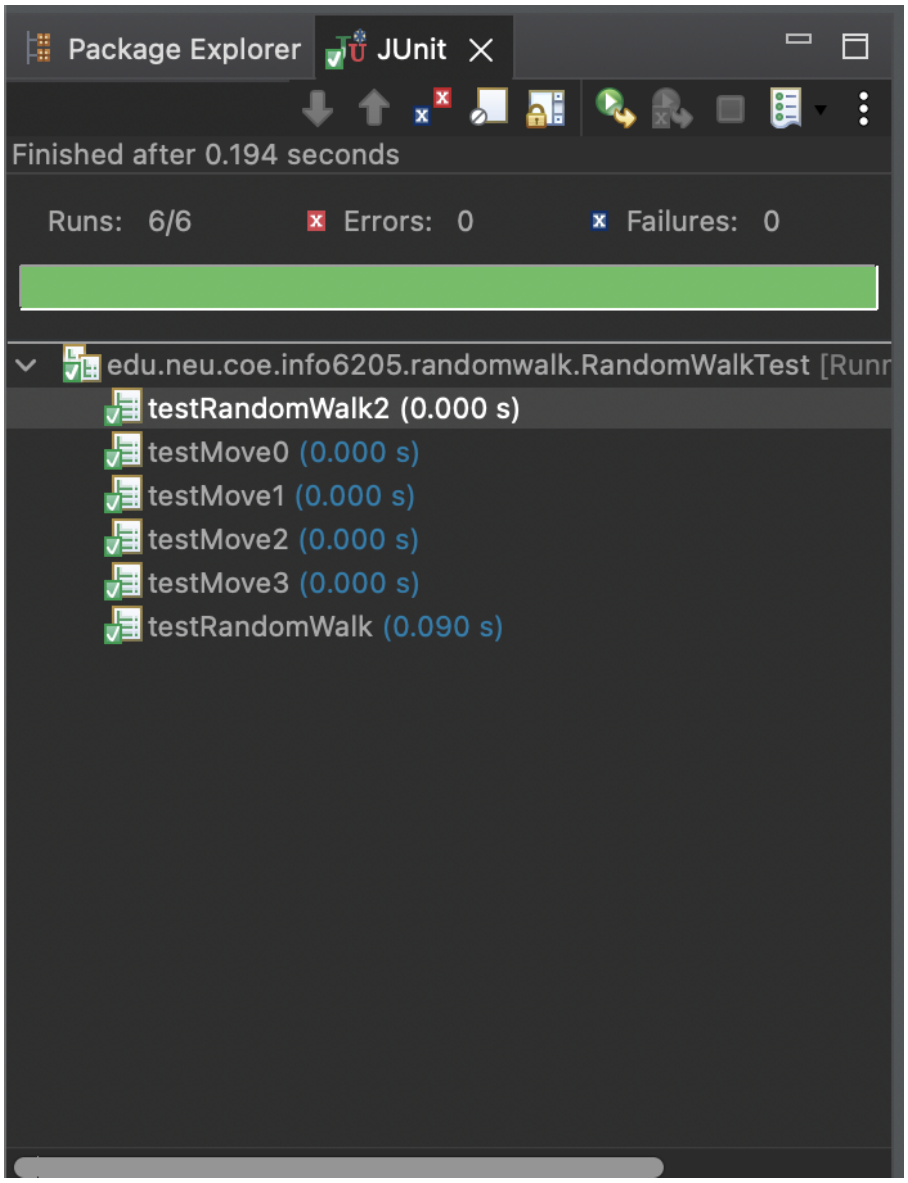


3.Code

```
/*
 * Copyright (c) 2017. Phasmid Software
 */

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 += dx;
      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() {
         return Math.sqrt((x * x) + (y * y));
    }

    /**
     * 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. Screenshot of the unit tests all passing



```
Finished after 0.194 seconds

Runs:  6/6        Errors:  0        Failures:  0

edu.neu.coe.info6205.randomwalk.RandomWalkTest [Runr
    testRandomWalk2 (0.000 s)
    testMove0 (0.000 s)
    testMove1 (0.000 s)
    testMove2 (0.000 s)
    testMove3 (0.000 s)
    testRandomWalk (0.090 s)
```