Name: Prathamesh Mahesh Sahasrabuddhe
NUID: 002117703
Date: 01/29/2022

Programming Structures & Algorithms (INFO 6205)
Assignment 1- Random Walk

AIM:

This assignment will begin to build your algorithmic skills. It will also give you some experience with random number generation in Java.

An important example of a practical experiment is called the "random walk" experiment.

Imagine a drunken man who, starting out leaning against a lamp post in the middle of an open space, takes a series of steps of the same length: 1 meter. The direction of these steps is randomly chosen from North, South, East or West. **After n steps, how far (*d*), generally speaking, is the man from the lamp post?** Note that *d* is the Euclidean distance of the man from the lamp-post.

It turns out that there is a relationship between *d* and *n* which is typically applicable to many different types of stochastic (randomized) experiments. Your task is to implement the code for the experiment and, most importantly, to **deduce the relationship**.

Please clone/pull from the class repository and work on *RandomWalk.java* and *RandomWalkTest.java* each of package *randomwalk* and each under the appropriate source directory. [You may have to remove other java files from the classpath in order to allow the whole project to compile. In IntelliJ/IDEA you can do this for entire packages by right-clicking and choosing "Mark Directory As... Excluded"]. Once you have all the unit tests running, you can do the experiment by running *RandomWalk* as a main program (provide the value of *n* as the first argument).

For this particular assignment, it is **necessary but *not* sufficient** to ensure that the unit tests all run. You must demonstrate via image files, graphs, whatever, what experiments you made in order to come up with the required expression. You will run the experiment for at least six values of *n* and will run each of these at least ten times. That's to say, you will run the program *at least* 60 separate times. Feel free to change the main program so that it will run all your experiments in one shot instead of 60 different runs.

Your submission should include:

1. Your **conclusion** about the relationship between *d* and *n*;
2. Your **evidence** to support that relationship (screen shot and/or graph and/or spreadsheet);
3. Your **code** (*RandomWalk.java* plus anything else that you changed or created);
4. A **screen shot** of the unit tests all passing.

Please note: for this assignment, you do not need to set up github and push your files, as described in the general instructions for submission (Submitting Assignments). Note also that common sense should tell you how d varies with l. Don't spend a lot of time agonizing over this aspect of the assignment. What we are primarily interested in is how d varies with n.

Conclusion:

According to the experiments we have run, we can conclude that the number of steps and the distance are directly proportional to each other. We can see from the results that as the number of steps increases, the distance travelled increases as well. Henceforth, we can say that the distance d (distance travelled by the man from first step till the last step) is directly proportional to n (number of steps taken by the drunk man).

$$d \propto n$$

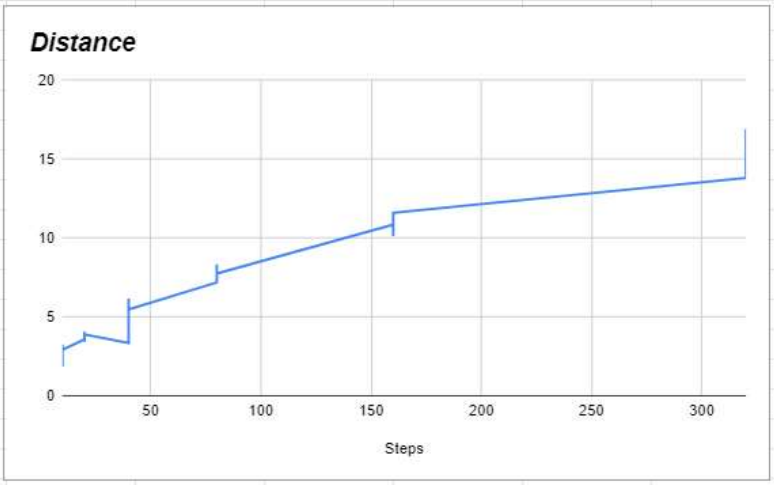
Also, we can see that the distance travelled is approximately equal to the square root of the number of steps taken by the drunk man. Thus, we can also conclude that d (distance travelled by the drunk man) is approximately the square root of n (number of steps taken by the drunk man).

$$d \cong \sqrt{n}$$

Evidence:

A spreadsheet with steps, distance, and number of experiments along with the graph showing distance d being directly proportional to number of steps taken by the drunk man n.

Steps	Distance	Experiments
10	1.850393786	10
10	3.256092305	20
10	2.593009805	30
10	3.188762142	40
10	2.876593079	50
10	2.885586827	60
10	2.72923303	70
10	2.674575189	80
10	2.708311215	90
10	2.90302264	100
20	3.585131288	10
20	3.84124235	20
20	3.535016909	30
20	3.856903939	40
20	3.420255765	50
20	4.061661905	60
20	3.81873968	70
20	4.050474244	80
20	3.775145174	90
20	3.877603049	100
40	3.338632792	10
40	5.239125213	20
40	5.538792091	30
40	5.729932831	40
40	5.767458277	50
40	5.268990588	60
40	5.799503823	70
40	5.745399238	80
40	6.145842058	90



Observations:

Steps	Distance	Experiments
10	1.850393786	10
10	3.256092305	20
10	2.593009805	30
10	3.188762142	40
10	2.876593079	50
10	2.885586827	60
10	2.72923303	70
10	2.674575189	80
10	2.708311215	90
10	2.90302264	100
20	3.585131288	10
20	3.84124235	20
20	3.535016909	30
20	3.856903939	40
20	3.420255765	50
20	4.061661905	60
20	3.81873968	70
20	4.050474244	80
20	3.775145174	90
20	3.877603049	100
40	3.338632792	10
40	5.239125213	20
40	5.538792091	30
40	5.729932831	40
40	5.767458277	50
40	5.268990588	60
40	5.799503823	70
40	5.745399238	80
40	6.145842058	90
40	5.467319176	100

80	7.180992779	10
80	7.538360176	20
80	7.483085064	30
80	7.423188556	40
80	7.71872388	50
80	7.8953585	60
80	8.323754485	70
80	7.99025225	80
80	7.327349487	90
80	7.74355429	100
160	10.85309871	10
160	10.25824681	20
160	10.38783066	30
160	10.56513581	40
160	10.50263658	50
160	11.46628626	60
160	10.12060354	70
160	11.04691706	80
160	11.17431046	90
160	11.59861317	100
320	13.82064629	10
320	14.0914944	20
320	15.93362791	30
320	14.91253246	40
320	16.38854989	50
320	16.18662579	60
320	16.20659924	70
320	16.90140435	80
320	16.15410609	90
320	15.97020815	100

Code:

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

package edu.neu.coe.info6205.randomwalk;

import com.github.sh0nk.matplotlib4j.Plot;
import com.github.sh0nk.matplotlib4j.PythonExecutionException;

import java.io.IOException;
import java.util.Arrays;
import java.util.List;
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) {
        // FIXME do move by replacing the following code
        this.x = this.x + dx;
        this.y = this.y + dy;
    }

    /**
     * Perform a random walk of m steps
     *
     * @param m the number of steps the drunkard takes
     */
    private void randomWalk(int m) {
        // FIXME
        for (int i = 0; i < m; i++) {
            this.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() {
        // FIXME
        // END
        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]");

        for (int i = 0; i < args.length - 1; i++) {
            int count = 1;
            int gap = 0;
            while (count <= 10) {
                int m = Integer.parseInt(args[i]);
                int n = Integer.parseInt(args[i + 1]) + gap;
                double meanDistance = randomWalkMulti(m, n);
                System.out.println(m + " steps: " + meanDistance + " over " +
n + " experiments");
                count++;
                gap = gap + 10;
            }
//          if (args.length > 1) n = Integer.parseInt(args[1]);
            i++;
        }
    }
}
```

Unit Test Cases Passing: