Abhishikth Daniel Merugu (001548340)

**Program Structures & Algorithms**

**Fall 2021**

**Assignment No. 1**

* **Task**

1. Calculated the mean distance (d) of a drunken man walk from the origin point.
2. By conducting multiple experiments, observed the relation between the distance(d) and the number of steps(n).

* **Relationship Conclusion**

1. After multiple experiments we can conclude that the relationship between distance(D) and number of steps is

**D = A \* √N**

Where, D = Euclidean distance

A = coefficient which is ≈1

N = number of steps

* **Evidence to support the conclusion:**

1. **Output**

**Text

Description automatically generated**

1. **Graphical Representation (Attached Excel sheet)**

**Chart, scatter chart

Description automatically generated**

* **Unit tests result:**

Text

Description automatically generated

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) {  
 // TO BE IMPLEMENTED  
 // double c= Math.sqrt((dx\*dx)+(dy\*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) { //5  
 // TO BE IMPLEMENTED  
 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(); //0/1  
 int step = random.nextBoolean() ? 1 : -1; // 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() {  
 // TO BE IMPLEMENTED  
 double c= Math.*sqrt*(Math.*pow*(x,2) + Math.*pow*(y,2)); // Math.sqrt((x\*x)+(y\*y));  
  
 return c;  
 }  
  
 */\*\*  
 \* 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 i=1;  
 for(int j=0;j<6;j++) {  
 int m = 10 \* i; //10 //Integer.parseInt(args[0])  
 i=i\*2;  
 int n = 50;  
 if (args.length > 1) n = Integer.*parseInt*(args[1]);  
 double meanDistance = *randomWalkMulti*(m, n);  
 System.*out*.println(m + " steps: " + meanDistance + " over " + n + " experiments");  
 }  
  
 }  
  
}