**INFO6205: Program Structure and Algorithms**

**Assignment 1 – RandomWalk Algorithm**

**By**

**Ajay Mohandas**

**NEU ID: 001426741**

Contents

[Source Code – RandomWalk.java 1](#_Toc535587032)

[Output Screenshots 4](#_Toc535587033)

[RandomWalk.java 4](#_Toc535587034)

[RandomWalkTest.java 4](#_Toc535587035)

[Analysis 5](#_Toc535587036)

# Source Code – RandomWalk.java

package psa\_randomwalk;

/\*\*

\*

\* @author Ajay Mohandas

\*/

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.io.FileWriter;

import java.io.IOException;

import java.util.Random;

import java.util.Scanner;

public class RandomWalk {

private int x = 0;

private int y = 0;

private static final String COMMA\_DELIMITER = ",";

private static final String NEW\_LINE\_SEPARATOR = "\n";

private static final String FILE\_HEADER = "Steps,Mean\_Distance";

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 n the number of steps the drunkard takes

\*/

public void randomWalk(int n) {

// TO BE IMPLEMENTED ... (n random moves)

// ... END IMPLEMENTATION

for (int i =0; i< n;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() {

//Euclidean formula for calculating distance

double result = 0.0;

result = Math.sqrt(Math.pow(x,2)+ Math.pow(y,2));

return result;

}

/\*\*

\* 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;

}

//args is number of steps to be given by user

public static void main(String[] args) {

/\*if (args.length == 0)

throw new RuntimeException("Syntax: RandomWalk steps [experiments]");

int m = Integer.parseInt(args[0]);\*/

String fileName = "./assignment1.csv";

FileWriter fileWriter = null;

try{

fileWriter = new FileWriter(fileName);

fileWriter.append(FILE\_HEADER.toString());

fileWriter.append(NEW\_LINE\_SEPARATOR);

int m = 50;

int n = 20;

if (args.length > 1) n = Integer.parseInt(args[1]);

for (int i = 0 ; i< 50;i++)

{

double meanDistance = randomWalkMulti(m, n);

System.out.println("Experiment"+(i+1)+ " "+m + " steps: " + meanDistance + " over " + n + " experiments");

fileWriter.append(String.valueOf(m));

fileWriter.append(COMMA\_DELIMITER);

fileWriter.append(String.valueOf(meanDistance));

fileWriter.append(NEW\_LINE\_SEPARATOR);

m+=50;

}

}

catch(Exception e)

{

System.out.println("Error in CSV file");

e.printStackTrace();

}

try {

fileWriter.flush();

fileWriter.close();

}

catch (IOException e) {

System.out.println("Error while flushing/closing fileWriter !!!");

e.printStackTrace();

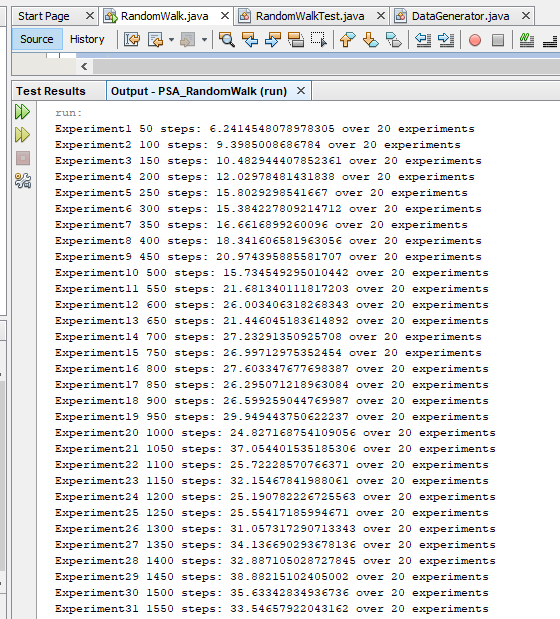
}

}

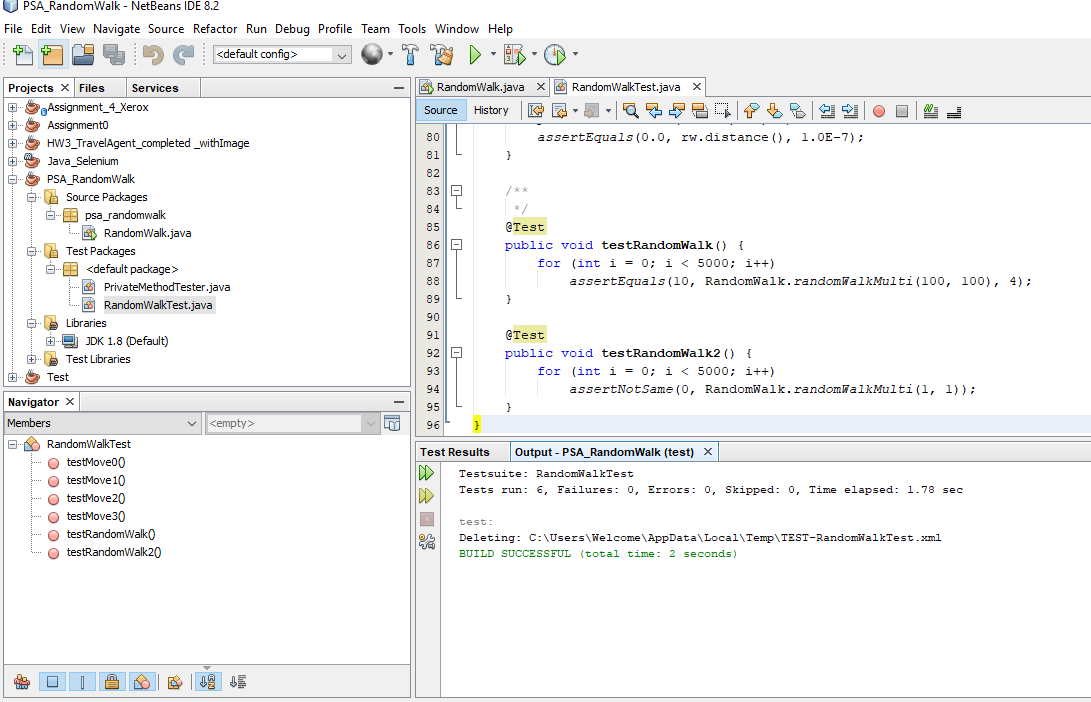
}

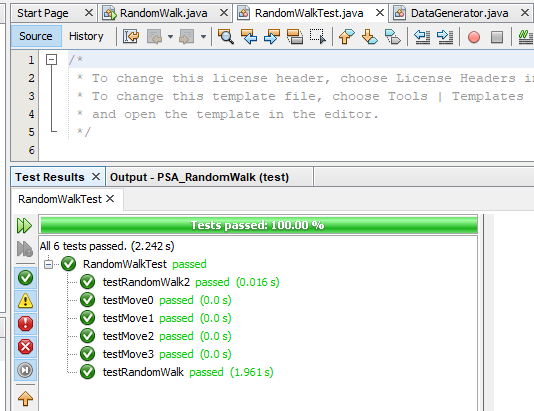
# Output Screenshots

## RandomWalk.java



## RandomWalkTest.java





# Analysis

d🡪 Calculated distance

n🡪 Steps taken by drunkard

l🡪 Length of each steps

From the spreadsheet RandomWalk.xlsx we are getting the formula as

Y = 0.98 \* x0.4851

Which is similar to non-linear equation for nth root curve i.e. y = k \* x1/n

From the graph, Y axis denotes the distance (denoted by d) and X axis denotes the steps taken by the drunkard (denoted by n).

So the relation becomes,

d = 0.98 \* n0.48

Using approximation,

d = 1 \* n0.5

Since length of each step taken by the drunkard is 1, l = 1. Hence the formula become,

d = l \* √n