**PSA**

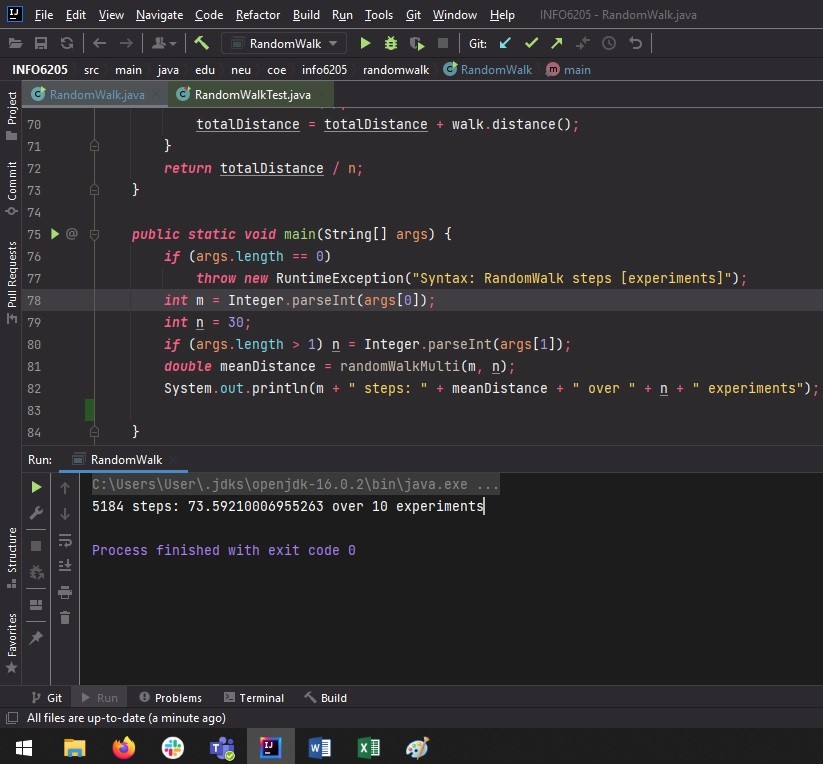
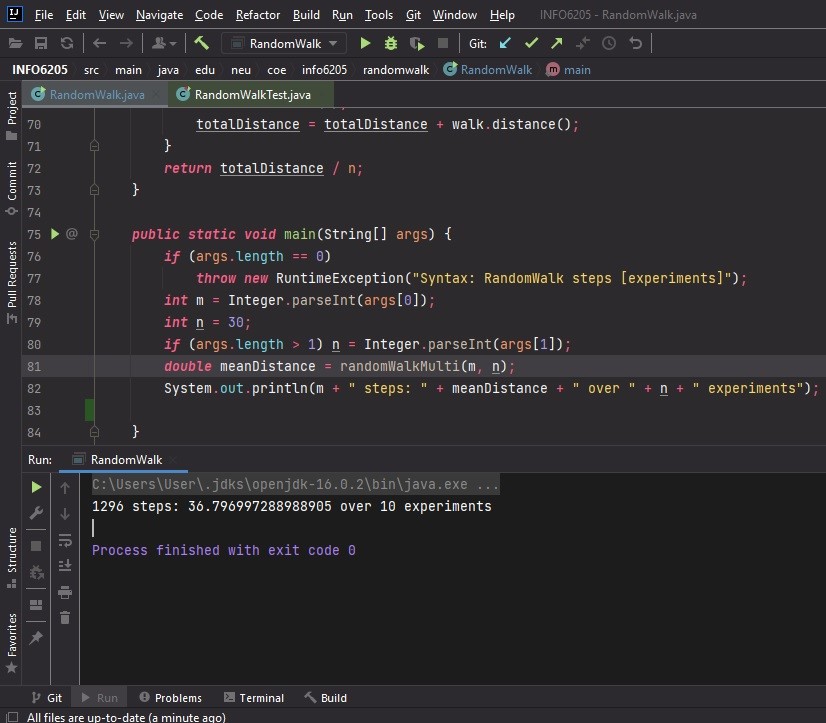
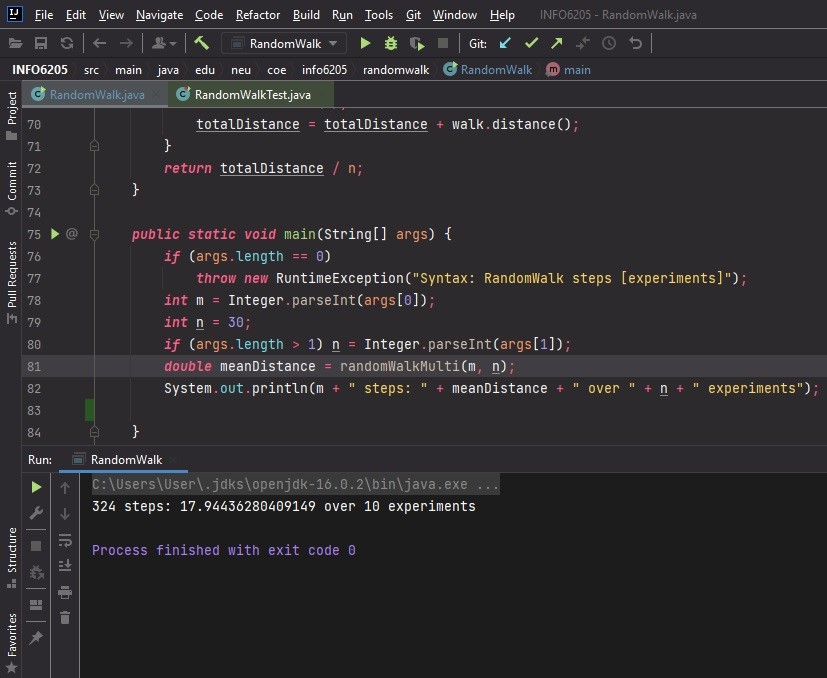
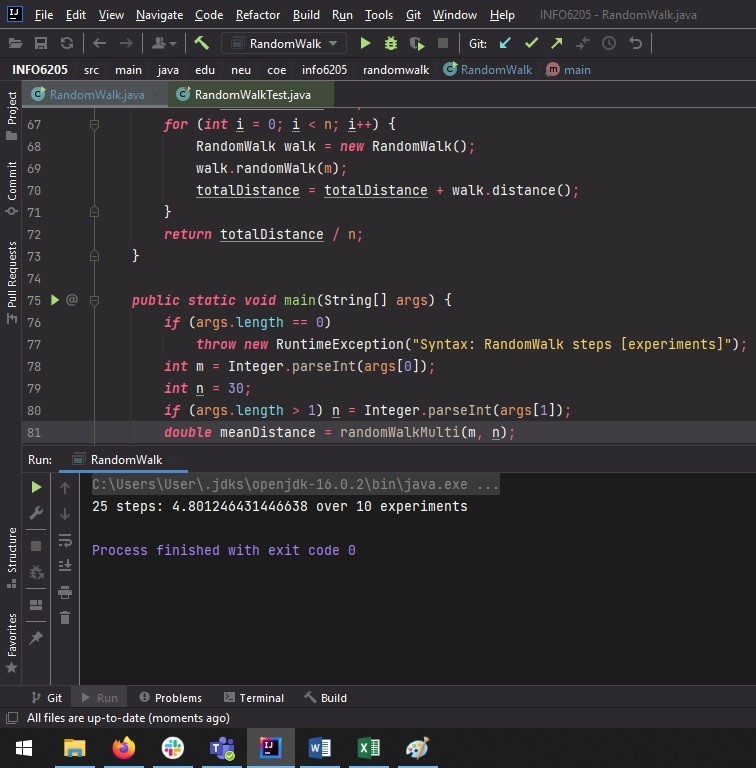
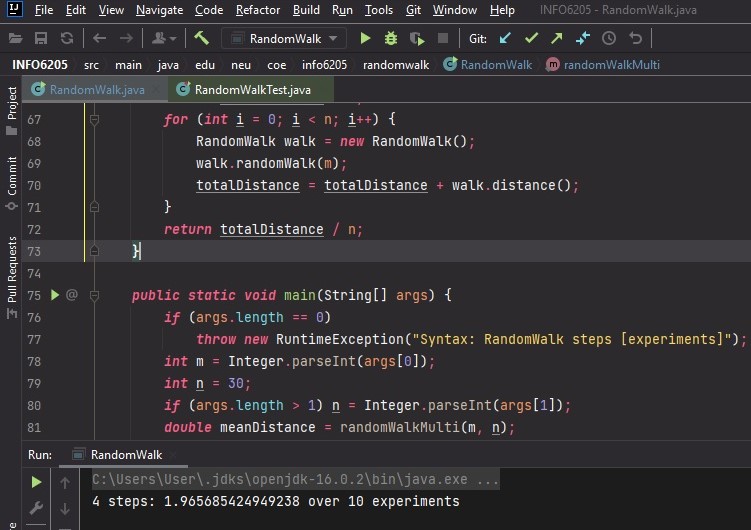
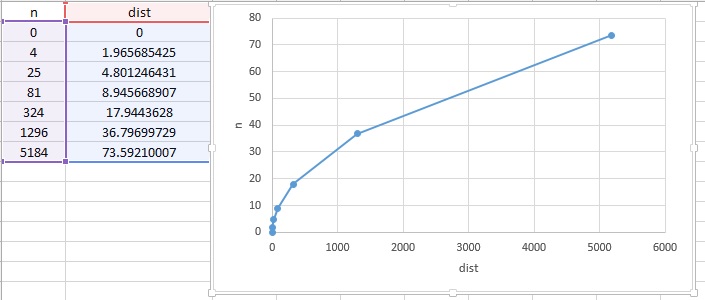
**ASSIGNMENT 1 (RANDOM WALK)**

KEYUR ASHOKBHAI BAROT

SECTION 1

NU ID: 001568664

1. The relationship between d (Euclidean distance) and n (steps) is that **d = √n.**
2. Graph as evidence to support the relationship.



1. **Code for RandomWalk.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() {  
 ***double*** dist = Math.sqrt((y\*y)+(x\*x));  
 ***return*** dist;  
 }  
  
 */\*\*  
 \* 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");  
   
 }  
  
}

1. **Screenshot of all unit tests passing:**

