

Author

[Lan gao](#)

001568670

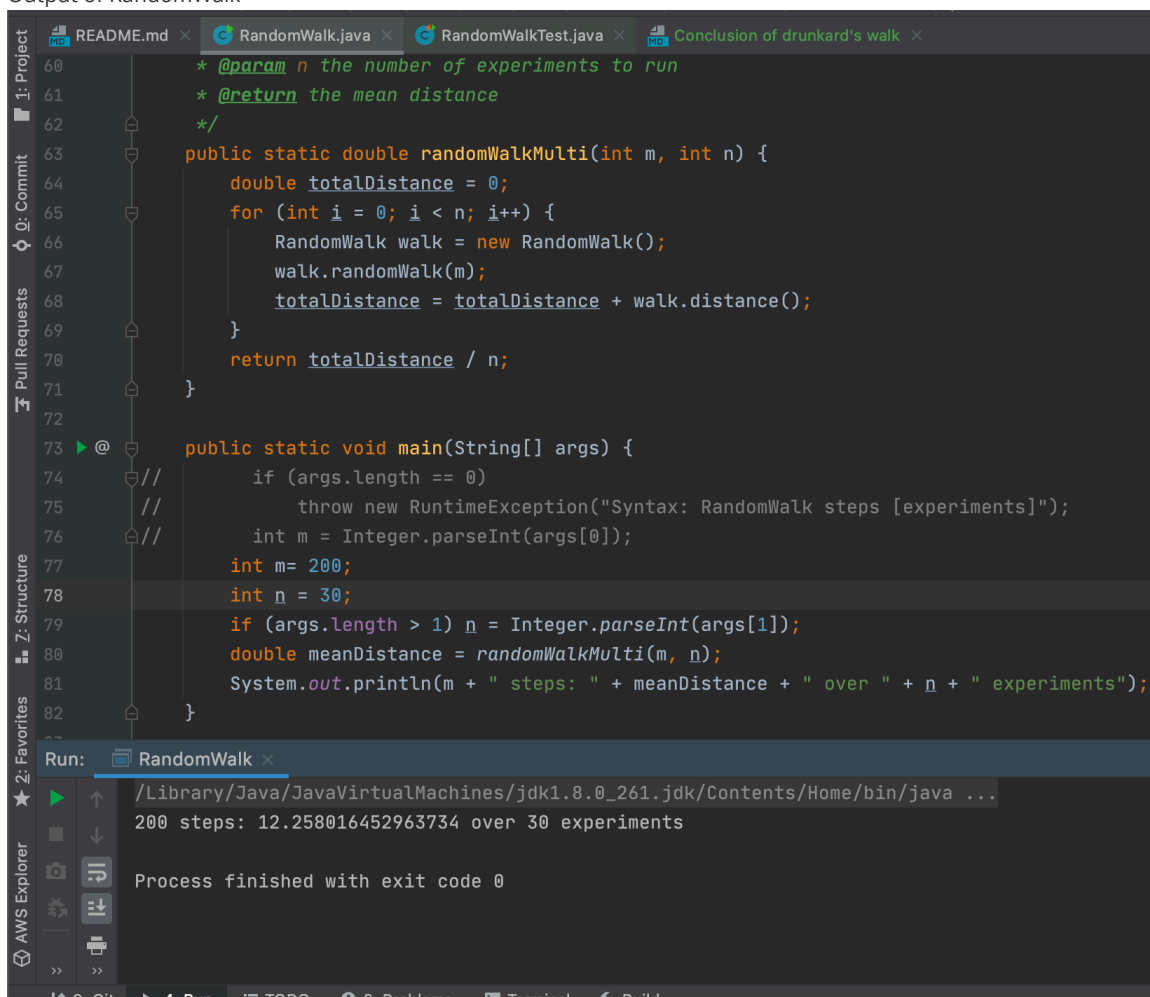
Task

Question:

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.

- Output of RandomWalk



The screenshot shows an IDE with the following components:

- Project Explorer:** Shows files README.md, RandomWalk.java, RandomWalkTest.java, and Conclusion of drunkard's walk.
- Code Editor:** Displays the RandomWalk.java file with the following code:

```
60  * @param n the number of experiments to run
61  * @return the mean distance
62  */
63  public static double randomWalkMulti(int m, int n) {
64      double totalDistance = 0;
65      for (int i = 0; i < n; i++) {
66          RandomWalk walk = new RandomWalk();
67          walk.randomWalk(m);
68          totalDistance = totalDistance + walk.distance();
69      }
70      return totalDistance / n;
71  }
72
73  @
74  public static void main(String[] args) {
75      if (args.length == 0)
76          throw new RuntimeException("Syntax: RandomWalk steps [experiments]");
77      int m = Integer.parseInt(args[0]);
78      int m= 200;
79      int n = 30;
80      if (args.length > 1) n = Integer.parseInt(args[1]);
81      double meanDistance = randomWalkMulti(m, n);
82      System.out.println(m + " steps: " + meanDistance + " over " + n + " experiments");
83  }
```
- Run Console:** Shows the execution of RandomWalk with the following output:

```
/Library/Java/JavaVirtualMachines/jdk1.8.0_261.jdk/Contents/Home/bin/java ...
200 steps: 12.258016452963734 over 30 experiments
Process finished with exit code 0
```

- Tests Results

```

Run: RandomWalkTest
Tests passed: 7 of 7 tests - 2 s 297 ms
RandomWalkTest (edu.neu.2 s 297 ms)
  testRandomWalk2 10 ms
  testMove0 2 ms
  testMove1 1 ms
  testMove2 1 ms
  testMove3 1 ms
  testRandomWalk3 2 s 81 ms
  testRandomWalk 201 ms

steps:139 Expected value: 11.7898 Actual value:10.4838 Difference:1.3868
steps:108 Expected value: 10.3923 Actual value:9.2247 Difference:1.1676
steps:110 Expected value: 10.4881 Actual value:9.1772 Difference:1.3108
steps:148 Expected value: 12.1655 Actual value:10.7447 Difference:1.4208
steps:18 Expected value: 4.2426 Actual value:3.7688 Difference:0.4818
steps:196 Expected value: 14.0000 Actual value:12.4277 Difference:1.5723
steps:173 Expected value: 13.1529 Actual value:11.6633 Difference:1.4897
steps:59 Expected value: 7.6811 Actual value:6.7240 Difference:0.9572
steps:70 Expected value: 8.3666 Actual value:7.4044 Difference:0.9622
steps:57 Expected value: 7.5498 Actual value:6.7501 Difference:0.7997
steps:94 Expected value: 9.6954 Actual value:8.5966 Difference:1.0988
steps:55 Expected value: 7.4162 Actual value:6.5728 Difference:0.8434
steps:62 Expected value: 7.8740 Actual value:7.8093 Difference:0.7847
steps:85 Expected value: 9.2195 Actual value:8.1468 Difference:1.0728
steps:199 Expected value: 14.1067 Actual value:12.4000 Difference:1.7067
steps:19 Expected value: 4.3589 Actual value:3.8467 Difference:0.5122
steps:44 Expected value: 6.6332 Actual value:5.8474 Difference:0.7858
steps:151 Expected value: 12.2882 Actual value:10.8895 Difference:1.3987
steps:101 Expected value: 10.0499 Actual value:8.9544 Difference:1.0955
steps:80 Expected value: 8.9443 Actual value:8.0420 Difference:0.9022
steps:74 Expected value: 8.6023 Actual value:7.6875 Difference:0.9148
steps:4 Expected value: 2.0000 Actual value:1.7549 Difference:0.2451
steps:66 Expected value: 8.1240 Actual value:7.1659 Difference:0.9581
steps:143 Expected value: 11.9583 Actual value:10.5671 Difference:1.3912
steps:168 Expected value: 12.9615 Actual value:11.4771 Difference:1.4844
steps:77 Expected value: 8.7750 Actual value:7.8473 Difference:0.9276

Process finished with exit code 0

```

- My own test

```

@Test
public void testRandomWalk3(){
    Random random = new Random();
    for(int i = 0; i < 100; i++){
        int steps = random.nextInt( bound: 200);
        double expected = Math.sqrt(steps);
        double average = RandomWalk.randomWalkMulti(steps, n: 10000);
        System.out.printf("steps:%d Expected value: %.4f Actual value: %.4f Difference: %.4f\n", steps, expected, average,
            assertEquals(expected, average, delta: 3);
    }
}

```

Conclusion

n : number of steps

d : the distance between the man and the lamp post.

Delta is the difference between the expected value and the actual value. The more tests, the more likely Delta will be to zero

$$\sqrt{n} = d \pm \Delta$$

Prove

According to the given topic, we can only get the expected value of the distance, that is, to find the following expected value

$$E_n(X^2 + Y^2) = \sum (x^2 + y^2)P(X = x, Y = y)$$

According to the same possibility of the four directions, it can be concluded that

$$P(X = x + 1, Y = y) = P(X = x + 1, Y = y | X = x, Y = y)P(X = x, Y = y) = \frac{1}{4}P(X = x, Y = y)$$

Therefore, for $N = n + 1$:

$$E_{n+1}(X^2 + Y^2) = \frac{1}{4} \sum [(x + 1)^2 + y^2] + [x^2 + (y + 1)^2] + [(x - 1)^2 + y^2] + [x^2 + (y - 1)^2]P(X = x, Y = y)$$

After simplified, we got :

$$E_{n+1}(X^2 + Y^2) = \sum (x^2 + y^2 + 1)P(X = x, Y = y) = E_n(X^2 + Y^2) + \sum P(X = x, Y = y)$$

Absolutely

$$\sum P(X = x, Y = y) = 1$$

So, we get

$$E_n(X^2 + Y^2) = n$$

That is to say, the number of steps is the square of the expected Euclidean distance

Provement of the test results

- When $n=10000$, $\delta=2$ could pass the test successfully;

The screenshot shows a Java code editor with a test method `testRandomWalk3()`. The code generates random steps and compares the expected value (sqrt(steps)) with the actual value from `RandomWalk.randomWalkMulti()`. The `assertEquals` call has `delta: 2` highlighted. Below the code, the 'Run' console shows 'Tests passed: 7 of 7 tests - 2 s 239 ms'. A detailed log shows multiple test cases with expected and actual values and differences, all within the tolerance.

```
@Test
public void testRandomWalk3(){
    Random random = new Random();
    for(int i = 0; i < 100; i++){
        int steps = random.nextInt( bound: 200);
        double expected =Math.sqrt(steps);
        double average = RandomWalk.randomWalkMulti(steps, n: 10000);
        System.out.printf("steps:%d Expected value: %.4f Actual value: %.4f Difference: %.4f\n", steps, expected, average, expected-average);
        assertEquals(expected, average, delta: 2);
    }
}
```

Run: RandomWalkTest x
Tests passed: 7 of 7 tests - 2 s 239 ms

Test Case	Expected value	Actual value	Difference
steps:16	4.0000	3.5112	0.4888
steps:66	8.1240	7.2452	0.8788
steps:9	3.0000	2.6806	0.3194
steps:162	12.7279	11.3539	1.3740
steps:179	13.3791	12.0319	1.3472
steps:63	7.9373	7.0344	0.9029
steps:47	6.8557	6.0715	0.7842
steps:117	10.8167	9.5842	1.2324

Process finished with exit code 0

- When $n=100$, $\delta=2$ failed to pass the test;

The screenshot shows the same test code but with `n: 100` highlighted. The 'Run' console shows 'Tests failed: 1, passed: 6 of 7 tests - 241 ms'. A detailed log shows the failure for 'steps:13' with an expected value of 3.7840 and an actual value of 11.6246, which is outside the tolerance. The stack trace indicates the failure in the `assertEquals` method.

```
public void testRandomWalk3(){
    Random random = new Random();
    for(int i = 0; i < 100; i++){
        int steps = random.nextInt( bound: 200);
        double expected =Math.sqrt(steps);
        double average = RandomWalk.randomWalkMulti(steps, n: 100);
        System.out.printf("steps:%d Expected value: %.4f Actual value: %.4f Difference: %.4f\n", steps, expected, average, expected-average);
        assertEquals(expected, average, delta: 2);
    }
}
```

Run: RandomWalkTest x
Tests failed: 1, passed: 6 of 7 tests - 241 ms

Test Case	Expected value	Actual value	Difference
steps:13	3.7840	11.6246	7.8406

Process finished with exit code 255

That is to say, as the number of test getting bigger, delta will tend to 0.