Name: \_\_\_\_Eyad Alsahori\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |
| --- |
| **ASSIGNMENT GOALS:**   1. **Become familiar with characteristics of the Drunkard Walker problem.** 2. **Gain more experience with Java Collection Classes.** 3. **Use both starter Classes and enum to complete required functionality.** |

NOTE package up programming assignment Classes under: **edu.cuny.csi.csc330.lab5**

1. Read Pragmatic Programmer Textbook – start of Chapter 5 pp 129 ~ 134 (middle of) (Bend or Break Chapter)
2. Read Textbook Chapter 7 - “403 - 481”. Once again, much of this material will be a review of what you already know, but it’s a worthy reinforcement **(20 pts):**

**Complete the following exercises starting on page 482:**

5) when the program runs

6) length

9) N/2

**Complete the following Short Answer exercises on bottom of page 485:**

3)

|  |  |  |
| --- | --- | --- |
| (a) | (b) | ( c) |
| 2 | 14 | 8 |

7)

Inorder for it to find the correct value that is stored in the last element, it needs read all 10,000 elements.

**(2) The Walking Drunkard (or Random Walk) Problem – focus on Encapsulation and putting Java Collections to work (80 pts)**:

Imagine an infinite grid of streets where locations/intersections are represented as integer pairs (x, y) – or more precisely, (avenue, street) pairs. For example, you might be on the intersection of 8th Ave and 52nd Street. Now consider someone that has had 1 drink too many – that person starts at a given intersection and then stumbles to the next intersection, but in a completely random, poorly-thought-out direction. So the “drunkard” might start at 7th Ave and 90th street - then randomly **picks one of eight directions** as a next step. Notice that “diagonal” steps represent a 2 linear-block move while the others represent a single linear block move:

1. Forward / North – 7th Ave and 91st Street
2. Right / East – 6th Ave and 91st Street.
3. Backward / South – 6th Ave and 90th Street.
4. Left / West – 7th Ave and 90th Street
5. Forward-Right / Northeast – 6th Ave and 91st Street
6. Backward-Right / Southeast – 5th Ave and 90th Street
7. Backward-Left / Southwest – 6th Ave and 89th Street
8. Forward-Left / Northwest – 7th Ave and 90th Street

See NYC grid below as an example, but our grid can be larger in scope – and as you may experience, negative values for Streets / Avenues are allowed.

Lab 5 is all about modeling random movement and keeping key statistics on a block-by-block trip covered by drunken (or **random)** behavior ;-)

Very often as a professional software developer, you’ll be asked to modify existing or **partially completed code**, rather than start an application from scratch. That’s exactly what you’re being asked to do here.

You’re being provided with four starter “entities”:

1. **Direction enum** – defines Direction constants, and methods related to using Direction values.
2. **Intersection Class** – simply models the street corner – or the (avenue, street) coordinates.
3. **DrunkWalker Class** – models a drunkard “navigating” intersection to intersection.
4. **DrunkWalkTester Class** – is a “test harness” for the DrunkWalker Class.

Your job will be to submit all 4 Classes/enums with updates that make them far more functional than they were when you found them.

**Here’s what needs to be done:**

1. **Direction enum** – already encompasses 4 of the 8 (plus NONE) Direction-set values required. You need to:
   1. Add the 4 additional Values
   2. Make appropriate changes to the getNextRandom() method to consider the 4 new Direction values.
2. **Intersection**:
   1. Use Eclipse code generating features (in Editor pane ‘right-click’->source ) to generate:
      1. a constructor that accepts avenue and street values (and of course assigns them to the Class’ data members).
      2. getters/setters for the two private data members (avenue, street).
      3. hashCode() and equals() methods
      4. toString() method.
   2. Populate the main() method to perform the following:
      1. Create 2 instances of Intersection that represent two **different** grid/street corners.
      2. Display the value of both instances using toString() – either explicitly or implicitly.
      3. Test the equality of the two instances you created using the equals() method – display the results of the if/equals test – proving that equals is working properly.
3. **DrunkWalker** – models the drunkard “navigating” intersection to intersection. Your update tasks for this Class are follows:
   1. Implement the hollowed out

DrunkWalker(int avenue, int street) constructor.

* 1. Complete the implementation of the following methods. In all cases, let the comments in each method be your guide:
     1. public void step().
     2. public String getLocation()
     3. public void fastForward(int steps)
     4. public int howFar()
     5. private void takeAStep()
     6. public void displayWalkDetails()
  2. add the appropriate Collection data members to support the methods above …
  3. Implement a toString() method using the Eclipse->Source feature (do this last – when you have the full structure of the Class worked out. You don’t need to include all data member’s state in toString() – you decide what you want to see displayed while developing and testing the application.

1. **DrunkWalkTester** – is a “test harness” for the DrunkWalker Class.

You will leave the existing code in the runTest() method alone, but you’ll add to it as dictated by the comments that appear at the end of that method:

/\*\*

\* Expand the test above to include the following ...

\* **Create a 2nd instances of DrunkWalker - Harvey**

**\* Have then race each - which instance (billy or harvey)**

**\* manages to walk a greater distance with 200 steps?**

**\***

**\* Also invoke the displayWalkDetails() on both instances.**

\*/



**Files you need to submit**:

1. This Word Document – with part 1 questions provided above and your program output pasted below.
2. The 4 completed source files for the following Classes/enum:
   * 1. Intersection
     2. DrunkWalker
     3. DrunkWalkTester
     4. Direction

**Sample Output**

|  |
| --- |
| **Intersection**:  Corner 1: Intersection [avenue=6, street=23]  Corner 2: Intersection [avenue=7, street=41]  Corners are not equal.  [Intersection [avenue=1, street=3], Intersection [avenue=3, street=4], Intersection [avenue=6, street=23], Intersection [avenue=7, street=34], Intersection [avenue=7, street=41]]  **DrunkWalker**:  billy's location after 1 steps: Current location: DrunkWalker [avenue=7, street=23]  billy's location after 2 steps: Current location: DrunkWalker [avenue=7, street=22]  billy's location after 3 steps: Current location: DrunkWalker [avenue=6, street=21]  Current location after fastForward(): Current location: DrunkWalker [avenue=6, street=21]  That's 2 blocks from start.  Start Time: Tue Oct 18 15:07:44 EDT 2016  End time - After FF: Tue Oct 18 15:07:45 EDT 2016  Starting Location: Intersection [avenue=10, street=42]  Current/Ending Location: Intersection [avenue=11, street=61]  Distance (blocks): 20  Number of steps taken: 100  Number of unique intersections visited: 71  Visited Intersection [avenue=8, street=42] 3 times!  Visited Intersection [avenue=8, street=63] 2 times!  Visited Intersection [avenue=9, street=41] 2 times!  Visited Intersection [avenue=9, street=42] 3 times!  Visited Intersection [avenue=9, street=43] 2 times!  Visited Intersection [avenue=9, street=63] 3 times!  Visited Intersection [avenue=10, street=41] 2 times!  Visited Intersection [avenue=10, street=42] 3 times!  Visited Intersection [avenue=10, street=43] 2 times!  Visited Intersection [avenue=10, street=44] 2 times!  Visited Intersection [avenue=10, street=49] 4 times!  Visited Intersection [avenue=10, street=54] 2 times!  Visited Intersection [avenue=10, street=55] 3 times!  Visited Intersection [avenue=10, street=62] 2 times!  Visited Intersection [avenue=10, street=64] 2 times!  Visited Intersection [avenue=11, street=49] 2 times!  Visited Intersection [avenue=11, street=50] 5 times!  Visited Intersection [avenue=11, street=62] 2 times!  Visited Intersection [avenue=12, street=47] 2 times!  **DrunkWalkTester:**  Enter the starting avenue value: 5  Enter the starting street value: 23  Billy's Current location: DrunkWalker [avenue=-2, street=28]  That's 12 blocks from start.  Starting Location: Intersection [avenue=5, street=23]  Current/Ending Location: Intersection [avenue=-2, street=28]  Distance (blocks): 12  Number of steps taken: 50  Number of unique intersections visited: 32  Visited Intersection [avenue=-3, street=26] 3 times!  Visited Intersection [avenue=-2, street=27] 2 times!  Visited Intersection [avenue=-1, street=25] 2 times!  Visited Intersection [avenue=-1, street=27] 3 times!  Visited Intersection [avenue=0, street=26] 2 times!  Visited Intersection [avenue=0, street=27] 3 times!  Visited Intersection [avenue=0, street=28] 2 times!  Visited Intersection [avenue=1, street=26] 2 times!  Visited Intersection [avenue=1, street=27] 2 times!  Visited Intersection [avenue=1, street=29] 2 times!  Visited Intersection [avenue=3, street=26] 3 times!  Visited Intersection [avenue=4, street=25] 2 times!  Visited Intersection [avenue=5, street=24] 2 times!  Visited Intersection [avenue=5, street=25] 2 times! |

**Paste Your Output Below**

|  |
| --- |
| **Intersection Class**:  <here>  Corner1[Intersection ave= 6, street=23]  Corner2[Intersection ave= 7, street=41]  1170  1219  These Corners are not equal.  **DrunkWalker Class**:  <here>  Current location after fastForward(): [DrunkWalker] Current Intersection: [Intersection ave= 7, street=25] Starting Position: [Intersection ave= 6, street=23]  That's 3 blocks from start.  Starting Location: [Intersection ave= 6, street=23]  Current/Ending Location: [Intersection ave= 7, street=28]  Number of steps taken: 6  Unique Intersection Count: 6  Intersections Visited More Than Once: 1  visited intersection: [[Intersection ave= 7, street=27]]  visited intersection: [[Intersection ave= 7, street=26]]  visited intersection: [[Intersection ave= 7, street=28]]  visited intersection: [[Intersection ave= 7, street=25], [Intersection ave= 7, street=25]]  visited intersection: [[Intersection ave= 7, street=24]]  visited intersection: [[Intersection ave= 6, street=25]]  visited intersection: [[Intersection ave= 6, street=24]]  Unique Visits:  [[Intersection ave= 7, street=27], [Intersection ave= 7, street=26], [Intersection ave= 7, street=28], [Intersection ave= 7, street=24], [Intersection ave= 6, street=25], [Intersection ave= 6, street=24]]  Non-Unique Visits:  [[Intersection ave= 7, street=25]]  **DrunkWalkTester Class:**  <here>  Enter the starting avenue value: 3  Enter the starting street value: 20  Billy's [DrunkWalker] Current Intersection: [Intersection ave= -2, street=1025] Starting Position: [Intersection ave= 3, street=20]  That's 1010 blocks from start.  Starting Location: [Intersection ave= 3, street=20]  Current/Ending Location: [Intersection ave= -2, street=1025]  Number of steps taken: 1010  Unique Intersection Count: 1113  Intersections Visited More Than Once: 394  Race Information:-  harvey:  [DrunkWalker] Current Intersection: [Intersection ave= 3, street=20] Starting Position: [Intersection ave= 3, street=20]  billy:  [DrunkWalker] Current Intersection: [Intersection ave= -2, street=1025] Starting Position: [Intersection ave= 3, street=20]  Starting Location: [Intersection ave= 3, street=20]  Current/Ending Location: [Intersection ave= -2, street=1025]  Number of steps taken: 1010  Unique Intersection Count: 1113  Intersections Visited More Than Once: 394  Hence: THE WINNER IS Billy WITH 1010 steps!!! |