COMP 1501 - ASSIGNMENT 4 IF, WHILE, STRINGS AND FILE I/O

Given: Tuesday Nov 10th
Due Date: Friday Nov 20th (before 11:59 pm)

Points for early submission

- 0 points = November 20th after midnight. Assignments handed in after midnight will be awarded zero points but will be accepted up **until 6am the following morning without any penalties**.
- 1 point = November 19th before midnight.
- 2 points = November 18th before midnight.
- 3 points = November 17th before midnight.
- 4 points = November 16th before midnight.

Case Scenario:

You have been hired to design and write a program to assist the organizers of an orienteering competition. The program will help to compute final statistics about the results of individual teams that were part of the competition.

Each participating team of explorers will be given a list of compass directions and distances and their task is to follow the directions to arrive at the precise final destination.

So what's the challenge?

Each successive instruction is handed out at the location reached by following preceding instruction. The organizers would like some assistance tracking the progress of each team and determining each team's final location. The program will run at the end of the competition and print out a summary report.

Input:

Input of the information about the orienteering will be done through file input. When the program starts, it will ask the user to input a file name, for example, `data1.txt`.

Below is a basic breakdown of the structure of the file. The file could contain information about a single team or for multiple teams that are part of the orienteering competition.

- For information about an individual team, the first line will be the team name (which will always be only one word).
- Each following line contains one instruction for a move in the order that each team must travel. A move consists of the direction (N for north, S for south, E for east, W for west, then NE, SW, etc. for the other four directions) followed by a distance.
- There will always be a sentinel record to mark the end of moves for a team. This line will contain XX (you need only check for the sentinel direction).
- There may be no moves for a team but there will always be a sentinel record marking the end of information for that team.
- After the sentinel record, the records for the next team of orienteers begin.

The starting location for all teams will be at (x,y) = (0,0).

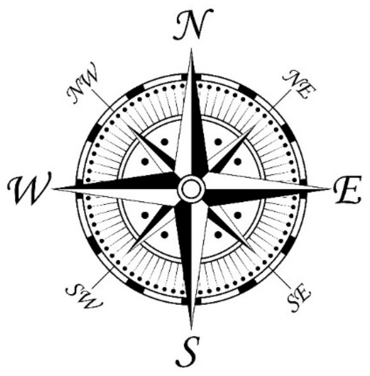
For example, a possible data file might be `data1.txt` (shown below)

```
Aardvarks
N 10
E 4.5
NE .98
W 3
XX
Beetles
XX
Chimeras
W 14.7
XX
```

data1.txt

3 teams
- Aardvarks
  - 4 moves: North 10, East 4.5, NE 0.98, West 3
- Beetles
  - 0 moves
- Chimeras
  - 1 move: West 14.7

Meaning of data1.txt



Once the program finishes processing information on one file, it will ask the user if they would like to continue processing another file. If the user chooses “yes”, the program will ask the user to input another file name and will process the above steps again. If they choose “no” the program ends.

Output:

There must be an output section for each team.

This will normally be a statement containing the team name, the final location with respect to the starting position, the number of moves made by the team and the distance (as the crow flies, i.e. in a straight line) of the team from its starting position. Starting position is always (0,0). This should be neatly formatted and readable.

For example, the report for Team Aardvarks shown in the above input file should look something like the following:

```
Team Aardvarks
   Final location:  10.7 km North and 2.2 km East from origin
   Number of moves:  4
   Distance covered as the crow flies:  10.92 km
   Total distance:  18.48 km
```

After the progress for all teams has been calculated and reported, a summary should be printed giving the name of the team which covered the furthest actual distance (not as the crow flies) and the number of individual moves made by the team.

For example:

```
Team Aardvarks covered the largest distance.
   Total distance:  18.48 km
   Total number of moves:  4
```

Note: Decimal output should be formatted to 2 decimal places (use the `DecimalFormat` class or the `String.format` method)

Explanation of how these values were obtained:

- Team Aardvarks starting location: (X,Y) = (0,0)
- Movements to the North or East result in a positive change to the current location
- Movements to the South or West result in a negative change to the current location
- Positive or negative change relates to an update of the x or y location.
- For diagonal moves, see the note below the example.

Step	Starting location (X,Y)	Move	Result Location (After the move) (X,Y)	Total Distance traveled (After the move)
0 (start)	(0, 0)			0
1	(0, 0)	North 10	(0, 10)	10
2	(0, 10)	East 4.5	(4.5, 10)	14.5
3	(4.5, 10)	North East 0.98 (*)	(5.2, 10.7)	15.48
4	(5.2, 10.7)	West 3	(2.2, 10.7)	18.48

End location: (2.2, 10.7)

(*) Note: If a team moves diagonally (northeast, northwest, southeast, or southwest), then you must modify the movement along the cardinal directions appropriately. The change will be the distance divided by the square root of 2 in each appropriate direction. So, for North East 0.98, the distance is 0.9, the change for each X, and y location is then:

$$change = \frac{distance}{\sqrt{2}}$$

$$change = \frac{0.98}{\sqrt{2}}$$

$$change = \frac{0.98}{\sqrt{2}} = 0.7$$

For square roots, you will need to use the `Math.sqrt` method.

To determine the "as the crow flies" distance you just need to compute the distance between the start point and the end point. So for our example, the start point is (xS,yS)=(0,0) and the end point is (xE,yE) = (2.2, 10.7)

The formula for distance between two points is shown below

$$crow\ distance = \sqrt{(xS - xE)^2 + (yS - yE)^2}$$

In our case, (xS,yS) will always be zero so we can shorten the formula to

$$crow\ distance = \sqrt{(xE)^2 + (yE)^2}$$

So for our example, we get

$$crow\ distance = \sqrt{(2.2)^2 + (10.7)^2}$$

$$crow\ distance = \sqrt{4.84 + 114.49}$$

$$crow\ distance = \sqrt{4.84 + 114.49}$$

$$crow\ distance = 10.92$$

Solution Strategy:

Your program must be decomposed into several methods.

Submission Instructions

Submit a BlueJ project folder (with all its contents) to the "submit" folder, which appears under the "I:" drive each time you log in to a university PC. This folder must include a ".java" file containing an executable class with a main method. If you are submitting from a non-university computer (e.g. a home PC), you can access the submit folder via secure.mtroyal.ca, into which you can upload a compressed (e.g. ".zip") version of your ENTIRE folder.

Your submitted folder name must follow the format illustrated below:

```
     <LastName>_<FirstName>_Asg4
```

The text file `data1.txt` (with data) used in the making and testing of this assignment MUST also be included in this folder.

Documentation Template:

Replace all highlighted text with actual values.

File header

```
/**
 *  <include description of the class here>
 * @author <your name>
 * @version 1.0
 * Last Modified: <date> - <action> <who made the change>
 */
```