CSE 271 Project 0 – CSE 174 Review
Spring 2022
Assigned: 1/31/2022
Due: 2/5/2022

Introduction:

In this project, we are going to review fundamental CSE 174 topics while further improving our abilities with the Eclipse IDE. In this project, you will be implementing methods, comments, performing File IO via files and the console, reviewing the Miami University Programming Style Guidelines, and working with ArrayList and Point objects. Specifically, your assignment is to develop a program that reads a set of points from a given text file. Each point has an x-coordinate and y-coordinate. The program will store this data as an ArrayList of Point objects, and then allow the user to show those points, remove points, and to find points at a given distance from the origin. A sample run/output of this program appears at the end of this assignment.

Provided Files:

1. **points.txt**, and **points_many.txt**: These files contain data for several points. Each point is defined by a pair of values in the order x-coordinate and a y-coordinate, separated by one or more white spaces (blank lines, tabs, or newline characters). You may use this file to help you test your code. The data is intentionally formatted differently to help you practice with Java's Scanner methods.
2. **PointPlotter.java**: This file contains the main() method and the menu() method that are used for testing. You may temporarily modify this file but you will not be submitting it.
3. **PointProcessor.java**: This file contains basic starter code for the methods you will be ***documenting and implementing*** as part of this project. This will be the only file you will be submitting.

Directions:

1. Create a project in your Eclipse IDE called **Project0** and add the provided started code and file to the correct positions in your project in the Eclipse Package Explorer. That is, the .java files need to go into the "**src**" folder while the **points.txt** and **points_many.txt** should be placed in the "**Project0**" folder.
2. Implement the methods in the **PointProcessor.java** class file.
3. Add appropriate comments to each method as well as to any line of code you believe requires additional explanation.
4. Make sure you follow the Miami University Programming Style Guidelines. Briefly, no line of code should be over 80 characters in length and no method should exceed 25 lines (not including comments or empty lines).
5. Use the provided **PointPlotter.java** class with the main() method to test your program.

Descriptions for the methods in PointProcessor.java:

1. **public static ArrayList<Point> readPointsFromFile(String fileName)**: This method must read a list of points from a given text file into an ArrayList and return the list. Recall that each point is represented by a pair of integer values corresponding to its x-coordinate and y-coordinate. Each coordinate is assumed to be separated by one or more white spaces (blank lines, tabs, or newline characters). Points are returned in the same order as they appeared in the given text file.

2. **public static int cabDistance(Point pt)**: This method, which can be completed in one line of code, returns the "taxicab" or "Manhattan" distance of a point from the origin (0, 0). This distance is defined as the absolute value of the x-coordinate plus the absolute value of the y-coordinate.
3. **public static void showPoint(Point pt)**: This method prints the point's coordinates in parentheses, separated b a comma, followed by that point's cab distance from the origin. Usa a tab ("\t") character to separate the point from its distance. A sample output is as follows (note that visually, the tab changes number of spaces and certain values may produce a different looking output which is okay):

   (-26, 2)	28

4. **public static void showAllPoints(ArrayList<Point> ptList)**: This method uses the showPoint() method to print all the points in the ptList parameter. This is done one line at a time with the index of the point to the left of it. If ptList is empty, then this method just prints "Empty list". A sample output is as follows (note that visually, the tab changes number of spaces and certain values may produce a different looking output which is okay):

   [0] (-26, 2)	28
   [1] (-11, -20)	31

5. **public static ArrayList<Point> findAll(ArrayList<Point> ptList, int dist)**: This method returns a new list of points that contains all points that are exactly at a given cab distance. The returned list of points are in the same order as they were in the given ptList parameter. If no points match, then this method returns an empty list (not a null list).

Test Run:

Enter points text file name: points.txt
Loaded 10 points from points.txt

What to do next:
1. Show all points
2. Find point with given distance from (0,0)
3. Quit
Enter choice: 1
[0] (-26, 2)	28
[1] (-11, -20)	31
[2] (42, 48)	90
[3] (25, -60)	85
[4] (25, -65)	90
[5] (67, -32)	99
[6] (67, -35)	102
[7] (-35, 73)	108
[8] (-53, 69)	122
[9] (59, -100)	159

What to do next:
1. Show all points
2. Find point with given distance from (0, 0)
3. Quit
Enter choice: 2

```
Find a point with what distance from origin? 90
[0] (42, 48)       90
[1] (25, -65)      90

What to do next:
1. Show all points
2. Find point with given distance from (0, 0)
3. Quit
Enter choice: 2
Find a point with what distance from origin? 100
Empty list

What to do next:
1. Show all points
2. Find point with given distance from (0, 0)
3. Quit
Enter choice: 3
```

Submission Instructions:

After you have completed the project, locate your source code (**PointProcessor.java**) in your workspace and submit it to the corresponding Project 0 assignment's CODE plugin.

Rubric:

Task	Grade
Correctly implemented the readPointsFromFile() method	20
Correctly implemented the cabDistance() method	15
Correctly implemented the showPoint() method	15
Correctly implemented the showAllPoints() method	20
Correctly implemented the findAll() method	20
Program contained correct JavaDoc documentation and individual comments where needed for the **PointProcessor.java** class and followed the Miami University coding guidelines	10
Total	**100**