COMP 2150 Assignment 1 – Summer 2022
Basic OOP concepts

This assignment is due by 11:59pm on Friday May 30th.

Submission format: On the course page on UMLearn, go to Assessments ➔ Assignments and upload your submissions to the Assignment 1 folder.

- Please follow both the "Programming Standards" and "Assignment Guidelines" for all work you submit.
- Ensure that you submit a signed Blanket Honesty Declaration before the assignment due date.

Question 1

You have been contracted by a rental-car management company to create a system for managing car rentals. The rental-car management company owns many different car rental depots with different rental cars available at each site and wants to efficiently assign rental cars to the "renters".

Implement a Java system for managing car rentals across several rental depots. Input to the program will be a data file that describes the rental depots, and a list of the rentals that renter want to reserve, as well as the location of the renter for each reservation. Output will be neatly-formatted charts for each month showing the assignment of all rental-cars ordered by rental depot, and a list of the parkers showing their parking space reservation.

Each rental-car is reserved month-by-month, but renters will reserve a set of months over the year. For example, a renter can request months 2 – 6, which means they will reserve a car from February to June (inclusive). Other renters could be assigned that car in month 1 or anytime during months 7 – 12.

To best serve their renters, the rental-car management company will assign rental-cars to renters based on proximity to their location. That is, renters indicate where they wish to pick up their rental-car, and the management company will assign them the closest available car. For our system, each rental depots and renter destination is described using an integer 2D coordinate system (x,y), and "closest" is defined as the minimal Euclidean distance; that is:

$$distance = \sqrt{(X_2 - X_1)^2 + (Y_2 - Y_1)^2}$$

calculated using real numbers. If two rental depots have exactly equal distance, choose the one with the lowest ID (the first in the input file) first.

Input File:

The first line in the input file will contain an integer describing the number of rental depots n that will be managed by the management company. This is followed by the description of n rental depots.

Each rental depot has various types of rental-cars in a variety of colors (For renter's preference) and a finite number of rental-cars available in each color for each model. Because of the differing preferences of the communities each rental depot services, the number of rental-cars available in each color can be different. In the input file, a rental depot is given as an integer number of available colors on the first line of input, followed by one line for each color. This line will have a string stating the color option followed by m integers corresponding to the number of rental-cars available in that color for each of the m car models, separated by commas. There is one more line at the end containing the location of the rental depot. For example:

```
3
red,2
black,8
blue,16
-153,100
```

This describes a rental depot with 3 vehicle colors options. The rental depot is located at (-153,100) in the coordinate system. This depot has a total of 26 rental-cars (2+8+16). In the input file, this would be followed immediately by the description of the next rental depot, followed by the next, to n rental depots.

Rental-cars at a depot are identified by a four character code. The first character is the depot ID, which is an upper case letter: A for the first depot, B for the depot, and so on. There will be at most 26 depots. The remaining three characters are digits indicating the car ID of given rental-car: 000 for the first designated car, 001 for the second and so on. There will never be more than 1000 total rental-cars.

Immediately following the rental depot descriptions is a list of the renters that wish want to reserve vehicles, one renter per line. Each renter has the following values, separated by commas:

- The renter's last name: any characters except for a comma.
- The renter's chosen color option.
- The renter's pick-up location coordinates: integer x, integer y.
- The starting and ending months of the renter's reservations: two integers in the range 0 – 11 with the second number greater than or equal to the first.

These values continue until a blank line occurs in the input file. For example:

```
Ogden,red,-177,103,3,8
Lee,black,-120,95,5,7
Lukas,blue,-153,100,1,12
Gupta,blue,-148,88,6,6
```

For example, Lee is renting a black car with a location of (-120,95), between months 5 and 7 (May to July).

After a blank line, there may be one or more event days listed. Event days are described on a line with these, comma-separated values:

- The name of the event: any characters except for a comma
- The location of the event, using coordinates: integer x, integer y.
- The month of the event: one integer in the range 0 – 11.
- The event rental color option.
- The number of rental-cars required by the event co-ordinators: an integer.

These values may continue until the end of the input file. For example:

```
Box-spring Dan's Grand Opening Event,100,75,8,white,8
```

You can assume that all events would be on different days.

Part A: Assigning Cars

The renter's want cars in specific colors. Rental cars are reserved in the order they are given in the input file. They are reserved using a greed algorithm; that is:

- Find the rental depot closest to the renter's pickup location, using minimum Euclidean distance.
- Determine if there is a rental-car available in the desired color at that depot:
    - The rental car must be the desired color requested by the renter.
    - The rental car must be available for ALL months the renter requested.
- If a matching rental-car was found, assign the rental car to the renter for the given timeframe.
- If **no** matching rental-car is found, proceed to the next nearest rental depot.
- If no rental-car car be found, then all rentals of that color are taken. In this case, print the name of the renter and stop assigning rental-cars to renters. (Note that this may leave unrented cars)

Part B: Event Days

Event day parking will first assign available cars for the event day rental from the nearest available depot, but once all available rentals are exhausted additional vehicles may need to be rented from other nearby depots. Event day rental-cars will be unavailable to standard renters during the month the event takes place. The rental-cars, however can be used for other event reservations during the same month.

The algorithm for assigning rental-cars on event days are as follows:

- Start with the number of rental-cars required, e
- Find the rental depot closest to the event, using minimum Euclidean distance.
- Assign all available vehicles to that event, starting with vehicles which are already assigned to another event during the same month before assigning any other rental-car which matches the color requirements of the event.
- If e is not zero when all rental-cars are assigned to the event, then assign the remaining rental-cars from the next nearest depot.
- If there are insufficient rental-cars available for the event, print the name of the event and stop the event day assignment (Note again that may leave unrented cars)

Part C: Output

The car rental reservation data is used in two ways.

First, traffic enforcement needs to know who is renting any vehicle at any given time to identify suspects if the rental-car is used unlawfully. This will require the output of a timeline that outlines the renter's name for a one year timeframe (12 months).

Car's without any reservations do not need to display a blank schedule and can instead be skipped. Here is what that output should look like for the first level of the input described above:

```
- - - A001 - - -
Ogden    3-8

- - - A003 - - -
Lee      5-7

- - - A011 - - -
Lukas    1-12

- - - A012 - - -
Gupta    6-6
```

For months that the rental-car is reserved for events, the name on the rental-car assignment schedule should simply say “Event” for any month which they are assigned to events. The program must output this schedule for every rental-car in each rental depot with active booking once the end of the input file is reached or when no more reservations can be filled.

Notes

Remember that, according to the programming standards, you should follow the rules of abstraction and object-oriented programming as well as possible. Use classes and collections of classes where appropriate. You can use any standard Java container classes or data types you like.

Your program must read the data from an input file. If the program is run with a command-line argument, use that as the input file name. If it is not, read input from the default file name “a1q1.txt” instead.

There are no data format errors in the input files. Your program should still be able to deal with challenging situations like edge cases.

In your README file, describe to the marker the classes and data structures that you used to implement your solution. Indicate the relationships between classes and containers that hold them.