Jan Aldous Torres

Viagogo Coding Challenge

## To run:

Use terminal to go into the src directory. Enter:

javac Test.java

java Test

The output should be similar to the previous

## Assumptions made:

* There are a max of 999 Events as Events.id only has 3 digits
* The user can search nearestEvents() from a coordinate not in the grid (i.e. -11,11 in a -10 to 10 by -10 to 10 grid)
* The grid has equal number of negative and positive coordinates for both its length and width. (i.e. cannot have -10 to 11 for x axis or y axis, must be -10 to 10)

## How might you change your program if you needed to support multiple events at the same location?

Firstly, we need to change Grid.events to be Grid.coordinates. In the current class Grid, the events are stored in a List. To support multiple events at the same location, a List of Coordinate objects should instead be used. Coordinates will contain a coordinate and list of events in that coordinate like below:

class Coordinate

Point coordinate

List<Event> events {tickets sorted by cheapest price} // contains the events in that coordinate

Secondly, whenever an event is made, Grid.coordinates should be updated as follows:

1: check if its point is the same as an existing Coordinate in Grid.coordinates

2: If it is add event to the coordinate’s events list attribute, if not create a new Coordinate object and add the event to its events list attribute and add the new coordinate to Grid.coordinates

Thirdly, the 5 nearest events should be calculated like this:

Let List<Event> e be an empty list of Events

1: create list x of CoordinateDistances from Grid.coordinates

2: sort list x according to distance and then cheapest price // implement same as existing code

3: n <- 0

4: for i <- 0; i < CoordinateDistances.size(); i++

5: List<Events> events <- x.get(i)

6: for j <- 0; j < events.size(); j++

7: e.add(events.get(j))

8: n++;

9: if (n > 5) then break;

10: return e

This is similar to the current implementation, except CoordinateDistance is used instead of EventDistance and the program selects the 5 nearest by iterating through the first coordinates’s events to obtain the 5 events.

We change EventDistance to CoordinateDistance with attributes:

int distance;

Coordinate coordinate; // instead of just Event event

## How would you change your program if you were working with a much larger world size?

The following may be implemented:

-Increase the size of the Grid to the correct length and width

-Implement quicksort for sorting in findNearestEvents() to implement the fastest sorting algorithm

Assuming no events will be updated or created, we can have a cache of previously searched coordinates through a List of CoordinateSearch:

class CoordinateSearch

Point c

List<Event> events {size = 5}

int numOfSearches

The list can have a set length and whenever we need to add and the list is full, the CoordinateSearch with the lowest numOfSearches is replaced by the new CoordinateSearch.

When findNearestEvents() is called, the program first searches the list of previous searches. If it has not been searched before, the method calculates the nearest 5 events using the current implementation. If it has been searched, the method returns the CoordinateSearch with the same coordinate events.