Assignment #8 – Graph

Objectives:

The main objectives of this assignment is:

- Understand Graph ADT
- Implement Graph ADT using Adjacency Map structure
- Implement efficiently using location aware design pattern.

The Assignment:

Implement AdjacencyGraph.java

Your implementation needs to support both directed and undirected graph. You need to implement the vertex's adjacency outgoing and incoming information using maps. Please note that the text book has many lines of code using the hash map.

Important Notes:

a) Be sure to include Time Complexity annotation and TCJ comments for each method. Use **n** to represent the number of vertices and **m** to represent the number of edges in the graph. The cost can be expressed with more than one variables, expressions, functions, like **O(n+m), O(m log n), O(deg(v)),** etc.

b) For each Vertex, you need to include the location aware information, which is an reference of the node

c) For each Edge, you need to include the location aware information, which is an reference of the node

d) For each Vertex, you need to store outgoing and incoming Adjacency Map.

Getting Started:

1. Download and import the code starter zip file. : For Eclipse:

2. You will need to implement the nested class myVertex and myEdge. You should feel free to copy over ANY data structures to this project that you have written during the semester.

   If you didn't do well with the related previous assignments, please meet with the TA/instructor to get approval and instructions to import library cs2321util at least 3 days before the project due.

Submission:

First, look at the following checklist:

1. Have you tested every method?
2. Do you ever `import` from `java.util`. If so, be sure you only import allowed components (like `Iterator`, Exceptions, etc.). Unless the assignment specifically mentions it is permissible, you should never include any of Java's native data structures.
3. Do all your methods have a `@TimeComplexity` indicator?
4. Do you provide adequate `TCJ` comments to prove your time complexity?
5. Is the indentation easily readable? You can have Eclipse correct indentation by highlighting all code and select "Source → Correct Indentation".
6. Are comments well organized and concise?
7. Does the program meet all required interfaces?
8. Again, Have you tested every method in some way?

If you have reviewed your code and all the criteria on the checklist are acceptable, follow the submission procedure.

Grading Criteria:

1. Correctly implementing the graph type: 50 points. It is important that you implement graph efficiently. Please note the following important requirement.
    a) If you don't use Adjacency Map, you will get 50% of what you got from the autograder test cases.
    b) If you don't have location aware reference pointer in Vertex object, you will lose 5 points
    c) If you don't have location aware reference pointer in Edge object, you will lose 5 points

2. Correct annotation of time complexity: 10 points
3. Good TCJ: 5 points
4. Clear concise code with good commenting: 5 points

Q&A:

- Do I get same points no matter which one (Edge List, Adjacency List, Adjacency Map, Adjacency Matrix) I implement?
    - No. You will get 50% if you didn't implement Adjacency Map.