Lab 16

CSE 274

In this lab we implement Graph data structure using adjacency lists.

I. The Graph class

The `Graph` class is as follows:

```
class Graph
{
private LinkedList[] AdjacencyList;
private Vertex[] VertexArray;
int size;
int maxSize;
}
```

The `VertexArray` is an array that stores vertexes of the graph. The `AdjacencyList` is an array of Linked Lists. The ith entry of `AdjacencyList` stores the vertexes that are neighbor with the specific vertex stored at ith entry of `VertexArray`. The variable `maxSize` is the size of `VertexArray` and `AdjacencyList` arrays. The `size` is the number of vertexes in the graph. The constructor of the `Graph` class initializes the variables `size` and `maxSize`, and allocates memory to `AdjacencyList` and `VertexArray` arrays.

II. The addVertex method

The `addVertex` method of the `Graph` class can be used to add vertexes to the graph:

```
public void addVertex(String label)
{
Vertex newVertex = new Vertex(label);
AdjacencyList[size] = new LinkedList();
VertexArray[size] = newVertex;
size = size + 1;
}
```

III. the addEdge method

The `Graph` class has the following method to add edges to the graph:

```
public void addEdge(String uLabel, String vLabel)
```

Develop the `addEdge` method using the following logic:

```
Vertex vVertex = null

for (int i = 0; i < size; i++)
      if the label of VertexArray[i] is vLabel
            vVertex = VertexArray[i]

for (int i = 0; i < size; i++)
      if the label of VertexArray[i] is uLabel
            if AdjacencyList[i] does not have a Vertex with vLabel
                  AdjacencyList[i].insertFirst(vVertex)
                  return
```

Use the following lines of code to test the developed method:

```
Graph myGraph = new Graph(100);

myGraph.addVertex("A");
myGraph.addVertex("B");
myGraph.addVertex("C");
myGraph.addVertex("D");
myGraph.addVertex("E");
myGraph.addVertex("F");
myGraph.addVertex("G");
myGraph.addVertex("H");
myGraph.addVertex("I");
myGraph.addVertex("J");
myGraph.addVertex("K");

myGraph.addEdge("A", "B");
myGraph.addEdge("B", "A");
myGraph.addEdge("B", "E");
myGraph.addEdge("E", "B");
myGraph.addEdge("A", "C");
myGraph.addEdge("C", "A");
myGraph.addEdge("B", "D");
myGraph.addEdge("D", "B");
myGraph.addEdge("D", "C");
myGraph.addEdge("C", "D");
myGraph.addEdge("C", "E");
myGraph.addEdge("E", "C");
myGraph.addEdge("E", "F");
myGraph.addEdge("F", "E");
myGraph.addEdge("D", "G");
myGraph.addEdge("G", "D");
myGraph.addEdge("E", "G");
myGraph.addEdge("G", "E");
myGraph.addEdge("G", "H");
myGraph.addEdge("H", "G");
myGraph.addEdge("C", "I");
myGraph.addEdge("I", "C");
myGraph.addEdge("A", "J");
myGraph.addEdge("J", "A");

myGraph.display();
```

The expected output is printed below:

```
Adjacency List:

A: J C B
B: D E A
C: I E D A
D: G C B
```

```
E: G F C B
F: E
G: H E D
H: G
I: C
J: A
K:
```

IV. The depthFirstTraversal

The `depthFirstTraversal` method receives a label as an argument, and performs the depth-first traversal of the graph starting at the vertex with the given label:

```
public void depthFirstTraversal(String label)
```

Develop the `depthFirstTraversal` method using the following logic:

```
System.out.println("Depth First Traversal from Vertex: " + label)

Vertex[] VisititedVertexes = new Vertex[size]
int VisitedCounter = 0

Vertex CurrentVertex = null

for (int i = 0; i < size; i++)
      if (VertexArray[i].label.equals(label))
            CurrentVertex = VertexArray[i]

if (CurrentVertex == null)
      return

Visit the CurrentVertex
VisititedVertexes[VisitedCounter] = CurrentVertex
VisitedCounter++

StackV myStack = new StackV(size)
push CurrentVertex on myStack

while myStack is not empty

      CurrentVertex = myStack.pop()
      Link currentLink = null
      for (int i = 0; i < size; i++)
            if VertexArray[i] and CurrentVertex are same references
                  currentLink = AdjacencyList[i].first

      while currentLink is not null
            NeighborVertex is the Data inside currentLink
            if NeighborVertex is already visited
                  currentLink = currentLink.next
                  continue
```

```
            Visit NeighborVertex
            VisititedVertexes[VisitedCounter] = NeighborVertex
            VisitedCounter++

            push CurrentVertex on myStack
            push NeighborVertex on myStack
            break
```

Use the following lines of code to test the developed method:

```
Graph myGraph = new Graph(100);
myGraph.addVertex("A");
myGraph.addVertex("B");
myGraph.addVertex("C");
myGraph.addVertex("D");
myGraph.addVertex("E");
myGraph.addVertex("F");
myGraph.addVertex("G");
myGraph.addVertex("H");
myGraph.addVertex("I");
myGraph.addVertex("J");
myGraph.addVertex("K");
myGraph.addEdge("A", "B");
myGraph.addEdge("B", "A");
myGraph.addEdge("B", "E");
myGraph.addEdge("E", "B");
myGraph.addEdge("A", "C");
myGraph.addEdge("C", "A");
myGraph.addEdge("B", "D");
myGraph.addEdge("D", "B");
myGraph.addEdge("D", "C");
myGraph.addEdge("C", "D");
myGraph.addEdge("C", "E");
myGraph.addEdge("E", "C");
myGraph.addEdge("E", "F");
myGraph.addEdge("F", "E");
myGraph.addEdge("D", "G");
myGraph.addEdge("G", "D");
myGraph.addEdge("E", "G");
myGraph.addEdge("G", "E");
myGraph.addEdge("G", "H");
myGraph.addEdge("H", "G");
myGraph.addEdge("C", "I");
myGraph.addEdge("I", "C");
myGraph.addEdge("A", "J");
myGraph.addEdge("J", "A");
myGraph.depthFirstTraversal("A");
myGraph.depthFirstTraversal("K");
```

The expected output is printed below:

```
Depth First Traversal from Vertex: A
A
J
C
I
E
G
H
D
B
F
Depth First Traversal from Vertex: K
K
```

V. Submitting the Assignment

When submitting your response to this assignment, keep the above lines of code in the body of the `main` method.