

Graph:

The Graph class represents the full graph structure.

This is a set of vertices (Node) and connections between them (edges with weights).

The graph is implemented through a contiguous list **adjacencyList** : a map where the key is the vertex id, and the value is the neighbours of this vertex and the distance to them.

► Key methods:

1. **addEdge(from, to, weight)**: adds an edge between the vertices from and to with the specified weight. If the reverse edge already exists, it is used.
2. **getDistance(from, to)**: returns the weight of an edge between two vertices.
3. **getNeighbors(node)**: returns all neighbors of the specified vertex.
4. **getNodes()**: returns the set of all vertices of the graph.

The screenshot shows an IDE interface with a project structure on the left and code editors on the right. The project structure includes a .idea folder, a src folder containing main/java/com/university/routing/algorithms, main/java/com/university/routing/Map, and main/java/com/university/routing/models. The models folder contains three files: Graph.java, Node.java, and Main.java. The Graph.java file is open in the editor, showing its implementation. The Node.java file is also circled in red at the bottom of the project tree. The code in Graph.java is as follows:

```
package com.university.routing.models;

public class Graph { //Class for representing a graph
    private Map<String, Map<String, Integer>> adjacencyList;

    public Graph() { adjacencyList = new HashMap<>(); }

    public Map<String, Map<String, Integer>> getAdjacencyList() { return adjacencyList; }

    //Adds an edge from the "from" vertex to the "to" vertex with the specified "weight"
    public void addEdge(String from, String to, int weight) {
        adjacencyList.putIfAbsent(from, new HashMap<>());
        adjacencyList.get(from).put(to, weight);

        // Убедитесь, что для вершины "from" существует пустой список смежных вершин
        adjacencyList.putIfAbsent(from, new HashMap<>());
    }
}
```