

In the project, the Node and Graph classes are closely related, as they are used together to represent the graph structure and work with vertices, edges, and their weights.

Node:

Node Class represents a single vertex of the graph, which is used to find a path (for example, in the A* algorithm).

► Each Node has:

1. **id**: a unique vertex identifier (coordinates).
2. **gCost**: The cost of the path from the starting point to this vertex.
3. **hCost**: A heuristic estimating the distance from this vertex to the endpoint.
4. **fCost**: The sum of gCost + hCost, used to evaluate priorities in the A* algorithm.

it is calculated dynamically via the getFCost() method. fCost is not stored in the Node object, but is calculated each time as the sum of gCost + hCost.

► Key methods:

1. **getFocus()**: returns the overall priority of the vertex.
2. **compareTo()**: Compares two vertices by their fCost for ordering.

The screenshot shows an IDE interface with the Node.java file open. The code defines a class Node that implements Comparable<Node>. It contains fields for id (String), gCost (int), and hCost (int). The constructor sets these values. The getFCost() method returns the sum of gCost and hCost. The Node class is highlighted with a red circle. The Project view on the left shows the Bot-AI_Side [route-optimizer] project structure, including src/main/java/com.university.routing.algorithms, Map, and models packages, with Node being the active file.

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

public class Node implements Comparable<Node> {
    private String id;
    private int gCost; // Cost from start to this node
    private int hCost; // Heuristic (estimated cost to goal)

    public Node(String id, int gCost, int hCost) {
        this.id = id;
        this.gCost = gCost;
        this.hCost = hCost;
    }

    public int getFCost() { return gCost + hCost; }
}
```