**Classes :**  
  
**NodeNew Class:**

* The **Node** class represents a node in the graph.
* Properties:
  + **id**: An integer representing the unique identifier of the node.
  + **hasMarker**: A boolean indicating whether the node has received a marker message.
  + **successors**: A list of nodes representing the successors of the current node.
  + **state**: A list of integers representing the recorded state of the node.
  + **channelOccupied**: A list of booleans indicating the occupancy status of outgoing channels.
  + **messageLog**: A list of strings storing the messages sent and received by the node.
  + **mtx**: An object used for synchronization purposes.
* Methods:
  + Node(int nodeId, int numChannels): Constructor to initialize the node with an ID and the number of outgoing channels.
  + receiveMarker(int markerValue): Method called when a node receives a marker message. It records the state, logs the message, and propagates the marker to successors.
  + propagateMarker(int markerValue): Method to propagate the marker message to successors.
  + markChannelAsOccupied(int successorId): Method to mark the outgoing channel to a successor as occupied and log the message.
  + markChannelAsEmpty(int successorId): Method to mark the outgoing channel to a successor as empty and log the message.
  + getState(): Method to simulate the state recording process. It creates a separate thread, sleeps for 2 seconds, and records the state.

**GraphNew Class:**

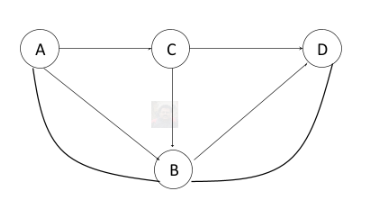
The Graph class represents the graph that contains nodes and their connections.

* Properties:
  + nodes: A list of nodes in the graph.
* Methods:
  + Graph(): Constructor to initialize the graph with an empty list of nodes.
  + addNode(Node node): Method to add a node to the graph.
  + addEdge(Node fromNode, Node toNode): Method to add an edge between two nodes in the graph.
  + startAlgorithm(Node initiator, int markerValue): Method to initiate the Chandy-Lamport State Recording Algorithm by sending a marker message from the initiator node.
  + checkTermination(): Method to check if all nodes have recorded their states and display the recorded states and message logs.

**InputReader class**

The **InputReader** class is responsible for reading the input graph from a file. It provides a static method **readGraph** that takes the filename as a parameter and returns the input graph as a 2D array.

Graph Used





**Explaining InputCL.txt**

4 7 // 4 – No. of Nodes , 7 – No. of Edges

0 1 // 0->1 edge between 0 and 1

0 2 // 0->2

1 3 // 1-> 3

1 0 // 1->0

2 1 // 2->1

2 3 //2->3

3 1 //3->1

0 23 // 0 – initiator node , 23- MarkerValue

**Output :**

Initiating Chandy-Lamport State Recording Algorithm.

Node 0 received marker with value 23.

Propagating marker with value 23 from Node 0 to Node 1

Channel 0 -> 1 marked as OCCUPIED.

Node 1 received marker with value 23.

Propagating marker with value 23 from Node 1 to Node 3

Channel 1 -> 3 marked as OCCUPIED.

Node 3 received marker with value 23.

Channel 1 -> 3 marked as EMPTY.

Channel 0 -> 1 marked as EMPTY.

Propagating marker with value 23 from Node 0 to Node 2

Channel 0 -> 2 marked as OCCUPIED.

Node 2 received marker with value 23.

Channel 0 -> 2 marked as EMPTY.

State Recording Completed

State stacks recorded:

State recorded by Node 0: 1 2 0 0 0 0 0

Message Log for Node 0:

Received marker: 23

Sent message 'Hi' to Node 1

Sent message 'Hi' to Node 2

State recorded by Node 1: 0 0 3 0 0 0 0

Message Log for Node 1:

Received marker: 23

Sent message 'Hi' to Node 3

State recorded by Node 2: 0 0 0 0 1 3 0

Message Log for Node 2:

Received marker: 23

State recorded by Node 3: 0 0 0 0 0 0 1

Message Log for Node 3:

Received marker: 23

State Recording Algorithm Terminated.

\*\*NEED MORE MODIFICATION \*\*