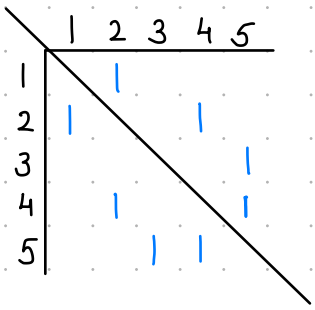
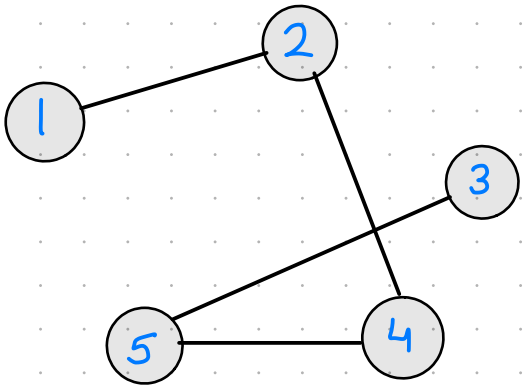
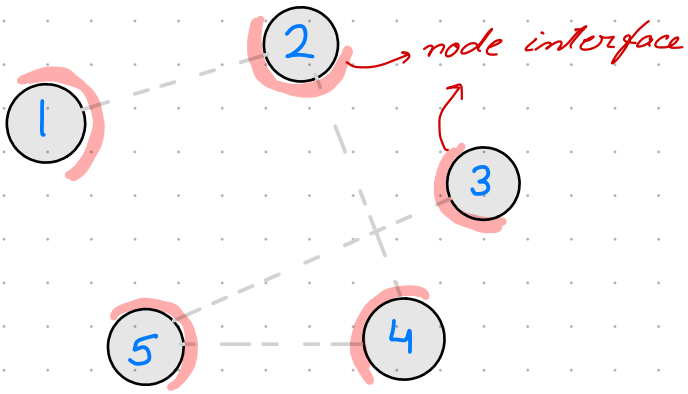
Overlay

Step 1 : Input





Q: Matrix pre-defined (fixed), or to be constructed first time the program is run (user prompt) ?



→ Every node has its own interface

→ during initialization :

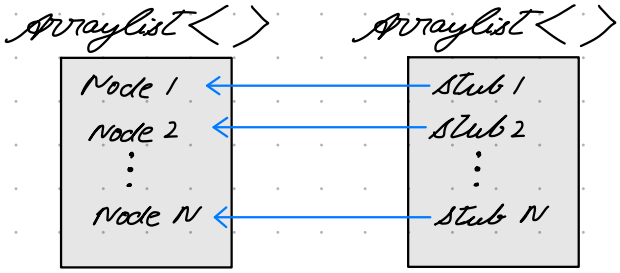
↳ associate every node with its interface

How?

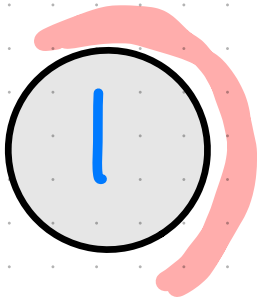
↳ Create a stub for every node instance and associate them.

how to manage?

↳ Make a list of stubs (ArrayList?) and keep adding stubs to it.



What IS a node ?



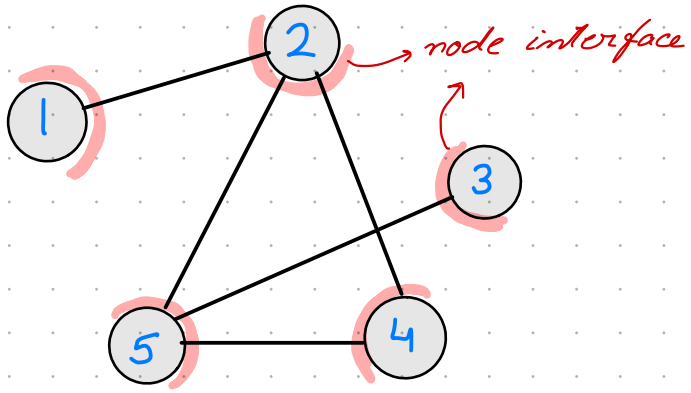
→ Each node is a seperate running entity.

→ Some attributes :

↳ node ID

↳ All (connected) neighbours' IDs

How ??



How to 'connect' nodes

1 → 2
2 → 1,4,5
3 → 5
4 → 2,5

basically store the IDs of all connected neighbours ??
(Not sure rn.)

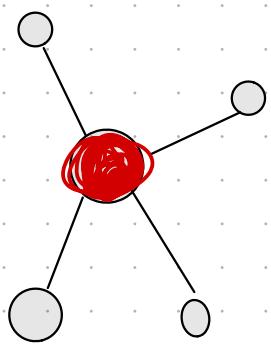
INITIALIZE:

```
Node node = new Node(0);   // ID or x
NodeInterface nitf = (NodeInterface) UnicastRemoteObject.exportObject... ... (node, 0)
//add the node and nitf to their respective arraylists.
Registry registry = LocateRegistry.getRegistry();
registry.bind("NodexService", nitf);
```

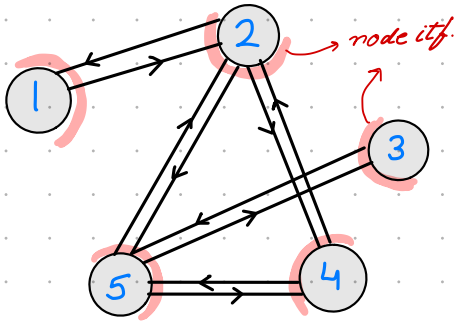
Can we get rid of this part ??

CONNECT:

Iterate through ALL IDs in input matrix:
  if ID in input matrix matches: (==1)
    ⋮

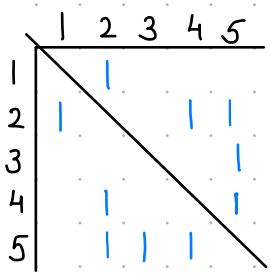


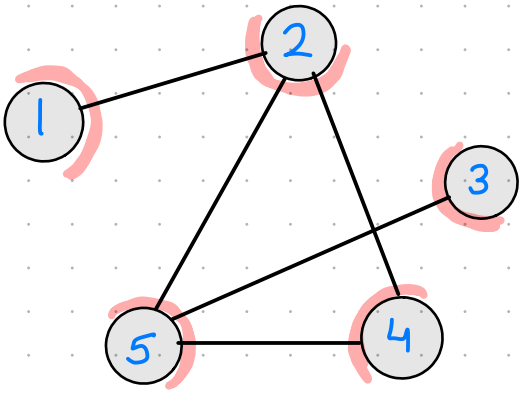
```
Node node = new Node(0);   // ID or x
NodeInterface nitf = (NodeInterface) UnicastRemoteObject.exportObject...
//add the node and nitf to their respective arraylists.  ... (node, 0)
Registry registry = LocateRegistry.getRegistry();
registry.bind("NodexService", nitf);
```



Iterate through ALL IDs in input matrix:
  if ID in input matrix matches: (==1) y

```
Registry registry = LocateRegistry.getRegistry("localhost");
NodeInterface nitf = (NodeInterface) registry.lookup("NodeyService");
Node node = new Node(y, nitf);
```





readFile():

- read the matrix file
- Identify :
  - ↳ No. of nodes (line 1)
  - ↳ IDs of all nodes (line 2)
  - ↳ Neighbourhood connection (rest all lines)

inputMatrix.txt

C: > Users > archi > IdeaProjects > OverlayProject > inputMatrix.txt

```
1  5
2
3  01234
4
5  01000
6  10011
7  00001
8  01001
9  01110
```

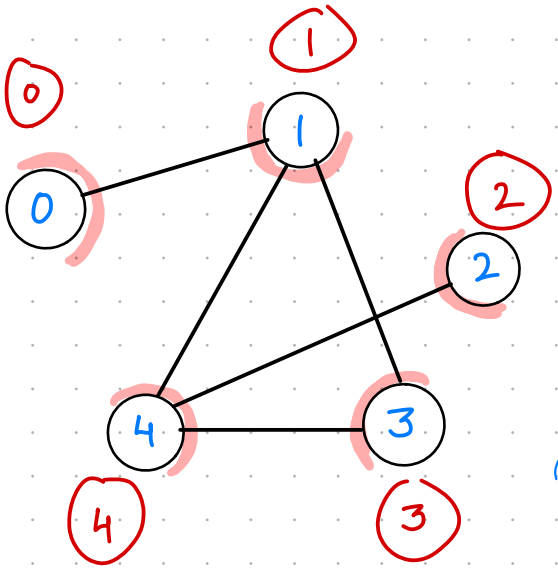
(ignore space lines)

- Create a hashmap :

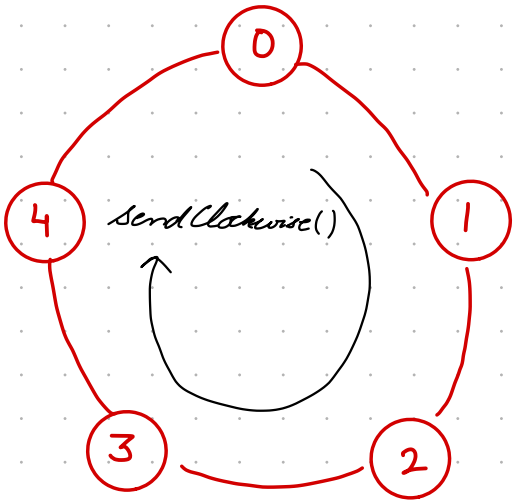
nodeNeighbourMap

key → Node ID

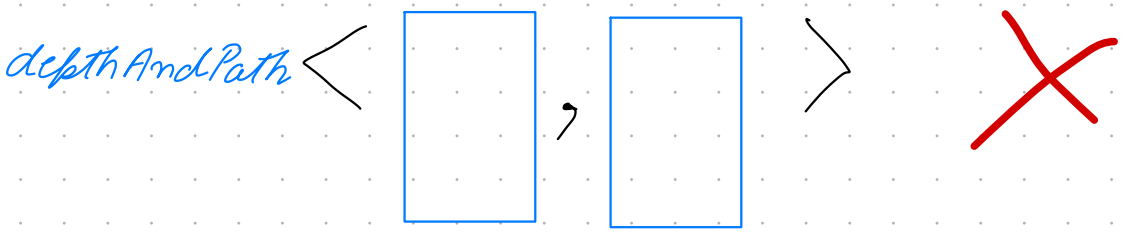
value → All neighbours' IDs

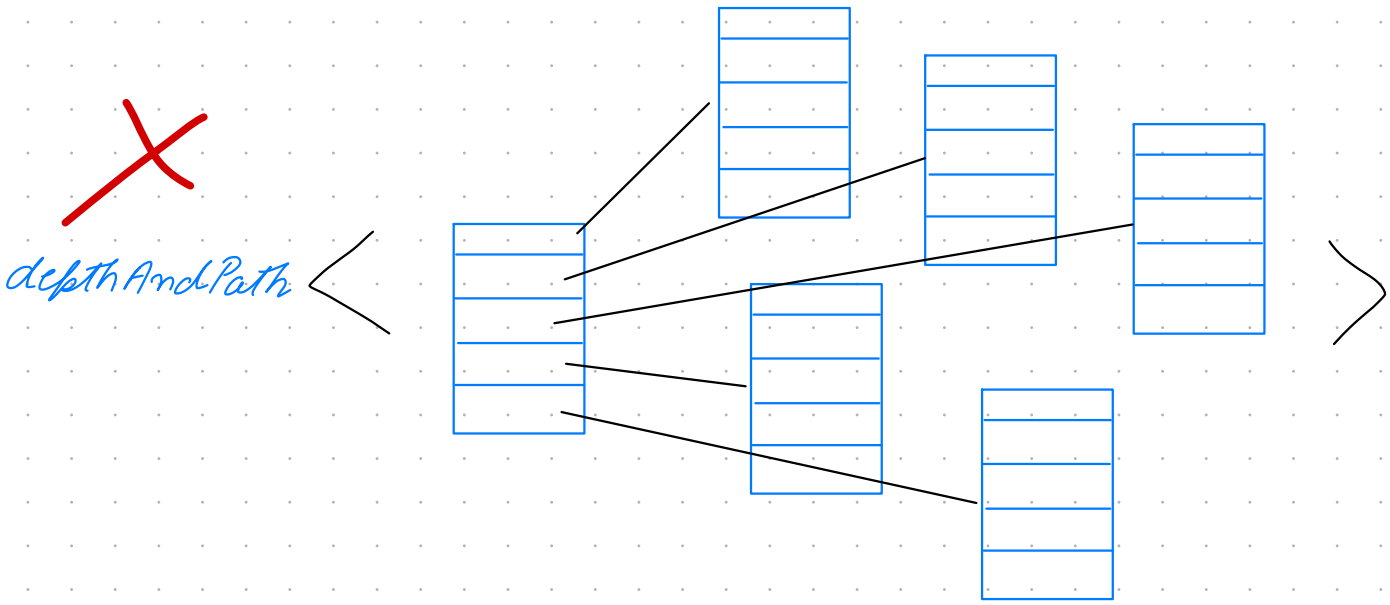


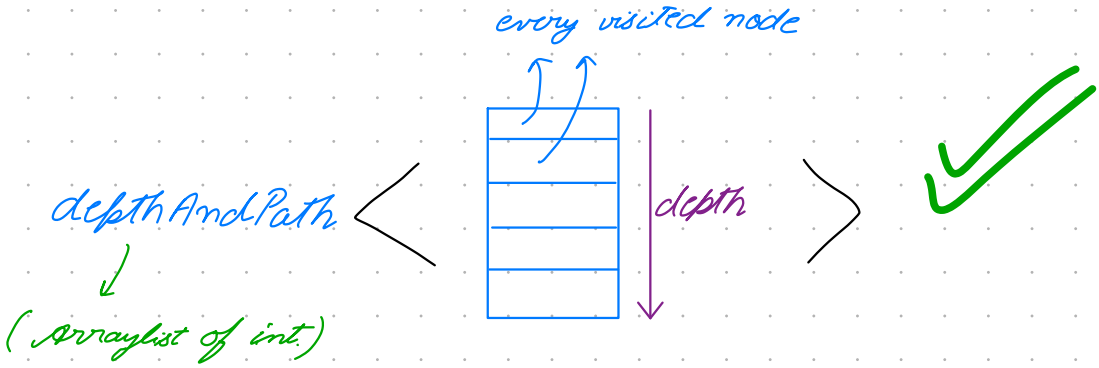
keeping the numbering (mapping) as same.



FUNC: computeRouting(...) (in PhysicalNode class)







computeRouting (int destinationPhyNode, ArrayList<Integer> Path)

- path.add (current node)
- if node reached:
  - return path
- else:
  - iterate thru ALL neighbours
    - computeRouting (int destinationPhyNode, ArrayList<Integer> Path)