**TOKEN BASED ALGORITHM FOR RING TOPOLOGY**

**Class And Methods :**

1. The Node class represents a node in a ring network. It has:
   * An ID
   * A boolean indicating if it has the token
   * A queue of request IDs
   * A reference to the Ring it belongs to
2. The Node constructor initializes the node's fields.
3. The getId(), hasToken(), setToken(boolean), and getRequestQueue() methods are getters and setters for the node's fields.
4. The printLocalQueue() method prints the node's request queue.
5. The requestToken() method either processes requests if it has the token, passes the token to the next node, or requests the token from the next node. I don't fully understand the logic here regarding when a node passes vs requests the token.
6. The passToken(int) method passes the token to the node with the given ID by setting hasToken to false for itself and true for the other node. It then processes any requests in the new token holder's queue or has it request the token from the next node. I don't fully understand under what conditions it will have the new token holder request the token vs process requests.
7. The Ring class represents the ring network. It has:
   * An array of Nodes
   * The ID of the node currently holding the token (phold)
   * A random number generator
8. The Ring constructor initializes the ring by creating its nodes and choosing a random node to hold the initial token.
9. The requestToken(int) method has the node with the given ID request the token.
10. The getNodes() and getNumNodes() methods are getters for the ring's fields.
11. The getPhold() method returns the ID of the node currently holding the token.
12. The Main class creates a Ring with (Lets say X ) nodes and has each node request the token. It then prints each node's request queue.

**Example:** Lets consider there are 5 nodes and firstly Node0 raised a request , then Node2, Node3 and Node1 , one after another ,has raised request .

Initially our program randomly chooses a node that has the Token. In this case Node1.

**Output :**

1 has the token #\_#

Node 0 is requesting token from node 1

Node 0 passed the token to 1

Node 2 is requesting token from node 3

Node 2 added itself into its neighbor's queue

Node 3 is requesting token from node 4

Node 3 added itself into its neighbor's queue

Node 1 entered CS.

Node 1 passed token to node 2

2 has the token #\_#

Node 2 entered CS.

Node 2 passed token to node 3

3 has the token #\_#

Node 3 entered CS.

Node 3 `s queue has node 2

Node 3 passed token to node 2

2 has the token #\_#

Node 2 entered CS.

Node 2 passed token to node 3

3 has the token #\_#

Node 3 entered CS.

Node 3 passed token to node 4

4 has the token #\_#

Node 4 entered CS.

Node 4 `s queue has node 3

Node 4 passed token to node 3

3 has the token #\_#

Node 3 entered CS.

Node 3 passed token to node 4

4 has the token #\_#

Node 4 entered CS.

Node 4 passed token to node 0

0 has the token #\_#

Node 0 entered CS.

Node 0 passed token to node 1

1 has the token #\_#

Token returned to Phold (node 1).

Node 0 local queue: []

Node 1 local queue: []

Node 2 local queue: []

Node 3 local queue: []

Node 4 local queue: []   
  
  
  
  
  
At the end Empty queues shows that the local queues of each node is empty means no request is pending in the network.

**Output 2:**

4 has the token #\_#

Node 0 is requesting token from node 1

Node 0 added itself into its neighbor's queue

Node 2 is requesting token from node 3

Node 2 added itself into its neighbor's queue

Node 3 is requesting token from node 4

Node 3 passed the token to 4

Node 1 is requesting token from node 2

Node 1 added itself into its neighbor's queue

Node 4 entered CS.

Node 4 passed token to node 0

0 has the token #\_#

Node 0 entered CS.

Node 0 passed token to node 1

1 has the token #\_#

Node 1 entered CS.

Node 1 `s queue has node 0

Node 1 passed token to node 0

0 has the token #\_#

Node 0 entered CS.

Node 0 passed token to node 1

1 has the token #\_#

Node 1 entered CS.

Node 1 passed token to node 2

2 has the token #\_#

Node 2 entered CS.

Node 2 `s queue has node 1

Node 2 passed token to node 1

1 has the token #\_#

Node 1 entered CS.

Node 1 passed token to node 2

2 has the token #\_#

Node 2 entered CS.

Node 2 passed token to node 3

3 has the token #\_#

Node 3 entered CS.

Node 3 `s queue has node 2

Node 3 passed token to node 2

2 has the token #\_#

Node 2 entered CS.

Node 2 passed token to node 3

3 has the token #\_#

Node 3 entered CS.

Node 3 passed token to node 4

4 has the token #\_#

Token returned to Phold (node 4).

Node 0 local queue: []

Node 1 local queue: []

Node 2 local queue: []

Node 3 local queue: []

Node 4 local queue: []