Name of Student: Pushkar Sane			
Roll Number: 45		**Lab Assignment Number: 5**	
Title of Lab Assignment: Implementation of mutual exclusion using the token ring algorithm.			
DOP: 07-10-2024		**DOS: 14-10-2024**	
CO Mapped:	**PO Mapped:**		**Signature:**

Practical No. 5

Aim: Implementation of mutual exclusion using the token ring algorithm.

Theory:

The Token Ring algorithm is a method used in distributed systems to achieve mutual exclusion, allowing multiple nodes (processes) to safely access shared resources without conflict. The algorithm is based on the concept of a token, a special control message that is passed around the nodes in a network topology arranged in a logical ring.

Key Concepts:

1. Token: A unique token circulates among the nodes in the ring. Only the node holding the token can enter its critical section (the part of the code that accesses shared resources).
2. Ring Topology: The nodes are logically arranged in a circular manner, where each node is connected to two neighbors. The token is passed sequentially from one node to its immediate neighbor.
3. Critical Section: This is the part of the code where shared resources are accessed. Only one node can be in its critical section at any time, which prevents race conditions and ensures data integrity.

Algorithm Steps:

1. Initialization: At the start, one node is assigned the token. This node can enter its critical section when it needs to access shared resources.
2. Requesting Access: If a node wants to enter its critical section, it waits until it receives the token. If it does not have the token, it continues to wait.
3. Entering Critical Section: When a node receives the token, it enters its critical section, performs its operations, and then prepares to release the token.
4. Releasing the Token: After completing its work in the critical section, the node releases the token to the next node in the ring. This enables the next node to access the critical section.
5. Passing the Token: The token continues to circulate around the ring, ensuring that each node gets a chance to access the critical section in a fair manner.

Advantages:

1. Fairness: Each node gets an opportunity to access the critical section.
2. Deadlock-Free: The algorithm prevents deadlock as there’s always a token available for some node.
3. Simplicity: The logic of token passing is straightforward and easy to implement.

Disadvantages:

1. Token Loss: If a node fails and the token is lost, the system can be deadlocked. Recovery mechanisms are needed.
2. Latency: The time to wait for the token can lead to increased latency, especially in large networks.
3. Single Point of Failure: The system depends on the continuous circulation of the token, making it vulnerable to failures.

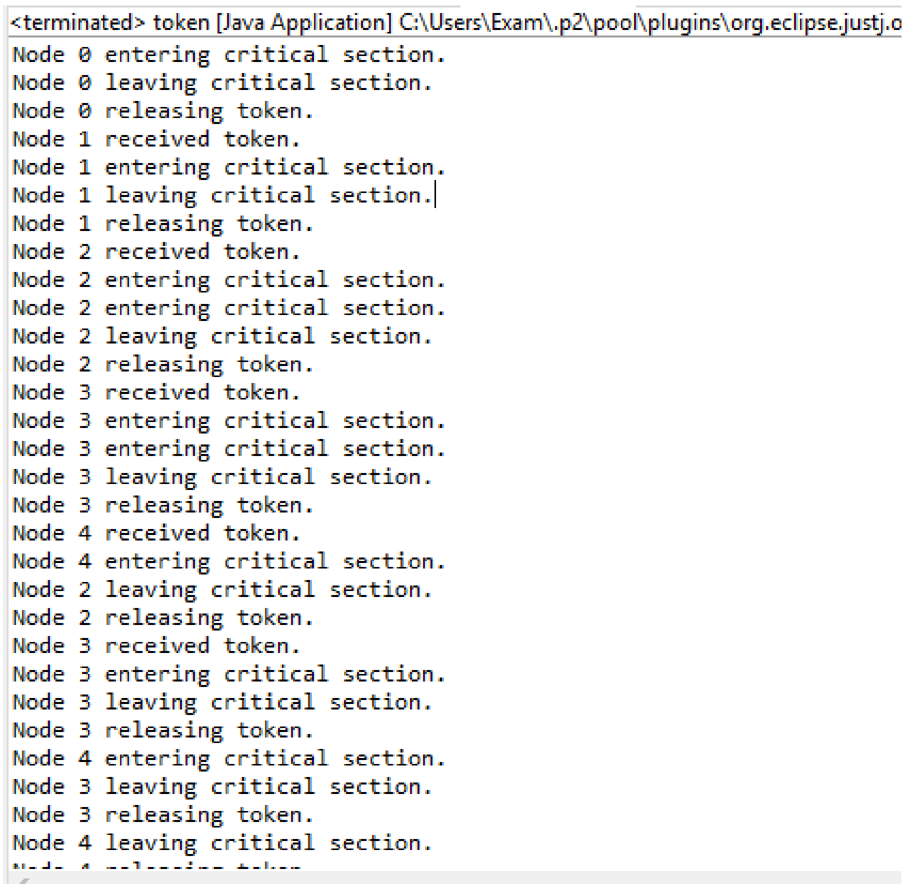
Applications:

The Token Ring algorithm is widely used in networking protocols and distributed systems, such as LANs (Local Area Networks), where multiple devices need to share the same communication channel while ensuring data integrity and avoiding conflicts.

Code:

```
import java.util.Random;
class TokenRingNode extends Thread {
    private final int id;
    private final TokenRing ring;
    private boolean hasToken;

    public TokenRingNode(int id, TokenRing ring) {
        this.id = id;
        this.ring = ring;
        this.hasToken = false;
    }

    @Override
    public void run() {
        while (true) {
            if (hasToken) {
                enterCriticalSection();
                releaseToken();
                break; // Exit after using the token
            }

            try {
                Thread.sleep(new Random().nextInt(1000));
            } catch (InterruptedException e) {
                Thread.currentThread().interrupt();
            }
        }
    }

    public void enterCriticalSection() {
        System.out.println("Node " + id + " entering critical section.");
        try {
            Thread.sleep(new Random().nextInt(1000) + 500);
        } catch (InterruptedException e) {
            Thread.currentThread().interrupt();
        }
        System.out.println("Node " + id + " leaving critical section.");
    }

    public void releaseToken() {
        hasToken = false;
        System.out.println("Node " + id + " releasing token.");
        int nextNodeId = (id + 1) % ring.getSize();
        ring.getNode(nextNodeId).receiveToken();
    }

    public void receiveToken() {
        if (!hasToken) {
            hasToken = true;
            System.out.println("Node " + id + " received token.");
            run(); // Start requesting access
        }
    }

    public void requestToken() {
        if (hasToken) {
            return;
        }
        try {
            Thread.sleep(new Random().nextInt(1000)); // Simulate delay
        } catch (InterruptedException e) {
            Thread.currentThread().interrupt();
        }
        run();
    }

    public void setHasToken(boolean hasToken) {
        this.hasToken = hasToken;
    }
}

class TokenRing {
    private final TokenRingNode[] nodes;
    public TokenRing(int numNodes) {
        nodes = new TokenRingNode[numNodes];
        for (int i = 0; i < numNodes; i++) {
            nodes[i] = new TokenRingNode(i, this);
        }
        nodes[0].setHasToken(true);
    }

    public void start() {
        for (TokenRingNode node : nodes) {
            node.start();
        }
    }
    public int getSize() {
        return nodes.length;
    }

    public TokenRingNode getNode(int index) {
        return nodes[index];
    }
}

public class token {
    public static void main(String[] args) {
        int numNodes = 5;
        TokenRing ring = new TokenRing(numNodes);
        ring.start();
    }
}
```

Output:



```
<terminated> token [Java Application] C:\Users\Exam\.p2\pool\plugins\org.eclipse.justj.o
Node 0 entering critical section.
Node 0 leaving critical section.
Node 0 releasing token.
Node 1 received token.
Node 1 entering critical section.
Node 1 leaving critical section.
Node 1 releasing token.
Node 2 received token.
Node 2 entering critical section.
Node 2 entering critical section.
Node 2 leaving critical section.
Node 2 releasing token.
Node 3 received token.
Node 3 entering critical section.
Node 3 entering critical section.
Node 3 leaving critical section.
Node 3 releasing token.
Node 4 received token.
Node 4 entering critical section.
Node 2 leaving critical section.
Node 2 releasing token.
Node 3 received token.
Node 3 entering critical section.
Node 3 leaving critical section.
Node 3 releasing token.
Node 4 entering critical section.
Node 3 leaving critical section.
Node 3 releasing token.
Node 4 leaving critical section.
```

Conclusion:

The Token Ring algorithm is a fundamental method for achieving mutual exclusion in distributed systems. Its reliance on a unique token to control access to critical sections makes it an efficient and effective solution for managing shared resources in a synchronized manner.