Experiment 07 – Design an program to illustrate token based algorithm

Learning Objective: Learn and implement a token-based algorithm in Java for process synchronization.

Aim: To design a program that illustrates the token-based algorithm for achieving mutual exclusion in distributed systems.

Tools: Java Development Kit (JDK), IDE (e.g., IntelliJ, Eclipse), Terminal.

Theory:

Principles of Token-Based Algorithm:

A token-based algorithm is a method used in distributed systems for mutual exclusion. It ensures that only one process can access a critical section at a time, preventing race conditions.

The Protocol:

1. **Initialization:**
    - A token is created and assigned to a process.
    - Other processes wait for the token before proceeding.
2. **Token Passing:**
    - The process holding the token executes its critical section.
    - Once done, it passes the token to another process.
3. **Completion:**
    - The process completes execution and releases the token.
    - The cycle continues until all processes have completed their tasks.

Security Considerations:

- Ensure token integrity to prevent duplication.
- Avoid token loss to prevent system deadlock.
- Implement recovery mechanisms for token failures.

Properties of Token-Based Algorithm:

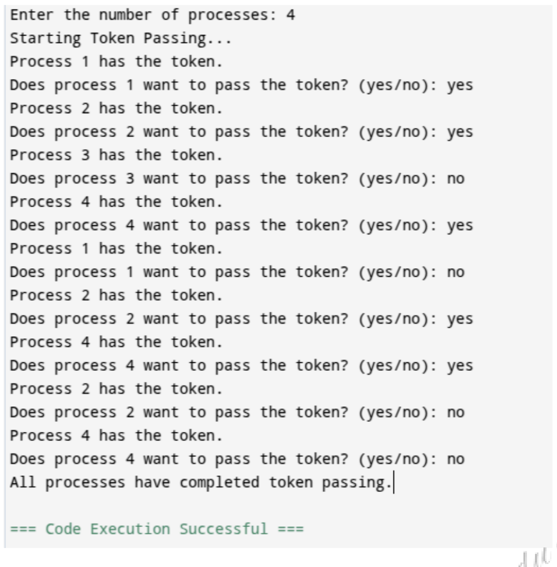
1. **Mutual Exclusion:** Ensures only one process executes in the critical section.
2. **Fairness:** All processes get a fair chance to access the critical section.
3. **Efficiency:** Reduces unnecessary wait times.
4. **Deadlock Prevention:** Proper token handling prevents indefinite blocking.
5. **Scalability:** Suitable for large distributed systems.

Implementation:

Java Code :-

```
import java.util.LinkedList;
import java.util.Queue;
import java.util.Scanner;

class TokenBasedAlgorithm {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the number of processes: ");
        int numProcesses = scanner.nextInt();

        Queue<Integer> processQueue = new LinkedList<>();
        for (int i = 1; i <= numProcesses; i++) {
            processQueue.add(i);
        }

        System.out.println("Starting Token Passing...");
        while (!processQueue.isEmpty()) {
            int process = processQueue.poll();
            System.out.println("Process " + process + " has the token.");

            System.out.print("Does process " + process + " want to pass the token?
  (yes/no): ");
            String response = scanner.next();

            if (response.equalsIgnoreCase("yes")) {
                processQueue.add(process);
            }
        }

        System.out.println("All processes have completed token passing.");
        scanner.close();
    }
}
```

Output:



```
Enter the number of processes: 4
Starting Token Passing...
Process 1 has the token.
Does process 1 want to pass the token? (yes/no): yes
Process 2 has the token.
Does process 2 want to pass the token? (yes/no): yes
Process 3 has the token.
Does process 3 want to pass the token? (yes/no): no
Process 4 has the token.
Does process 4 want to pass the token? (yes/no): yes
Process 1 has the token.
Does process 1 want to pass the token? (yes/no): no
Process 2 has the token.
Does process 2 want to pass the token? (yes/no): yes
Process 4 has the token.
Does process 4 want to pass the token? (yes/no): yes
Process 2 has the token.
Does process 2 want to pass the token? (yes/no): no
Process 4 has the token.
Does process 4 want to pass the token? (yes/no): no
All processes have completed token passing.

=== Code Execution Successful ===
```

-

Learning Outcomes: The student should have the ability to:

LO1.1 Explain the fundamental principles of token-based synchronization.
LO1.2 Identify and describe the mechanisms that ensure mutual exclusion.
LO1.3 Implement a basic token-based synchronization algorithm in Java.
LO1.4 Analyze potential risks and ethical considerations.
.

Course Outcomes: Students will be able to design and implement a token-based algorithm that ensures mutual exclusion in distributed systems.

Conclusion:

Viva Questions:

1. What is the significance of a token-based algorithm in process synchronization?
2. How does token passing ensure mutual exclusion?
3. What happens if the token is lost in a distributed system?
4. How can token duplication be prevented?

For Faculty Use:

Correction Parameter s	Formative Assessment [40%]	Timely completion of Practical [40%]	Attendance / Learning Attitude [20%]	
Marks Obtained				