**CODE:**

import java.util.concurrent.Semaphore;

public class ds {

static int numProcesses = 5;

static Semaphore[] semaphores = new Semaphore[numProcesses];

static boolean[] isHoldingToken = new boolean[numProcesses];

static int nextProcess = 0;

public static void main(String[] args) throws InterruptedException {

// Initialize semaphores

for (int i = 0; i < numProcesses; i++) {

semaphores[i] = new Semaphore(0);

isHoldingToken[i] = false;

}

isHoldingToken[0] = true; // The first process initially holds the token

// Start the processes

Thread[] threads = new Thread[numProcesses];

for (int i = 0; i < numProcesses; i++) {

int id = i;

threads[i] = new Thread(() -> {

while (true) {

try {

semaphores[id].acquire(); // Wait for permission to enter critical section

criticalSection(id);

releaseToken(id);

} catch (InterruptedException e) {

e.printStackTrace();

}

}

});

threads[i].start();

}

// Start the token passing loop

while (true) {

Thread.sleep(1000); // Wait for some time

int currentProcess = nextProcess;

nextProcess = (nextProcess + 1) % numProcesses; // Pass the token to the next process

if (isHoldingToken[currentProcess]) {

semaphores[nextProcess].release(); // Signal the next process to enter critical section

}

}

}

static void criticalSection(int id) {

System.out.println("Process " + id + " entered critical section");

try {

Thread.sleep(1000); // Simulate some work

} catch (InterruptedException e) {

e.printStackTrace();

}

System.out.println("Process " + id + " exited critical section");

}

static void releaseToken(int id) {

isHoldingToken[id] = false;

isHoldingToken[nextProcess] = true;

}

}

**OUTPUT:**

