PRACTICAL NO. 5

Mutual Exclusion

LOB5: Understand the mechanism of mutual exclusion using token ring algorithm.
LO5: Implement program based on mutual exclusion using token ring algorithm.

Mutual Exclusion:

A mutual exclusion (mutex) is a program object that prevents simultaneous access to a shared resource. This concept is used in concurrent programming with a critical section, a piece of code in which processes or threads access a shared resource. Only one thread owns the mutex at a time, thus a mutex with a unique name is created when a program starts. When a thread holds a resource, it has to lock the mutex from other threads to prevent concurrent access of the resource. Upon releasing the resource, the thread unlocks the mutex.

Mutex comes into the picture when two threads work on the same data at the same time. It acts as a lock and is the most basic synchronization tool. When a thread tries to acquire a mutex, it gains the mutex if it is available, otherwise the thread is set to sleep condition. Mutual exclusion reduces latency and busy-waits using queuing and context switches. Mutex can be enforced at both the hardware and software levels.

Disabling interrupts for the smallest number of instructions is the best way to enforce mutex at the kernel level and prevent the corruption of shared data structures. If multiple processors share the same memory, a flag is set to enable and disable the resource acquisition based on availability. The busy-wait mechanism enforces mutex in the software areas.

Token Ring Algorithm for Mutual Exclusion:

For this algorithm, we assume that there is a group of processes with no inherent ordering of processes, but that some ordering can be imposed on the group. For example, we can identify each process by its machine address and process ID to obtain an ordering. Using this imposed ordering, a logical ring is constructed in software. Each process is assigned a position in the ring and each process must know who is next to it in the ring (Figure).

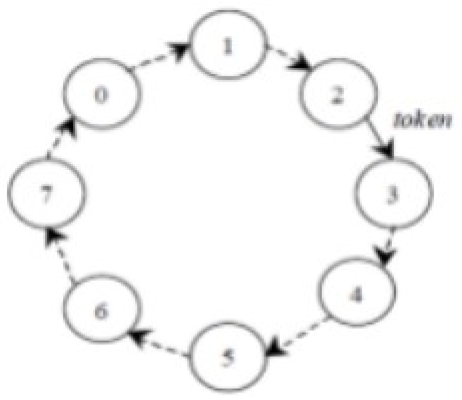


Figure: Token Ring example

1. The ring is initialized by giving a token to process 0. The token circulates around the ring (process n passes it to (n+1) mod ring_size.
2. When a process acquires the token, it checks to see if it is attempting to enter the critical section. If so, it enters and does its work. On exit, it passes the token to its neighbor.
3. If a process isn't interested in entering a critical section, it simply passes the token along.

Only one process has the token at a time and it must have the token to work on a critical section, so mutual exclusion is guaranteed. Order is also well-defined, so starvation cannot occur.

The biggest Drawback of this algorithm is that if a token is lost, it will have to be generated. Determining that a token is lost can be difficult.

Exercise:

a) Write a java program to implement mutual exclusion using Token ring algorithm.

Server.java:

```
import java.net.DatagramPacket;
import java.net.DatagramSocket;

class TokenServer {
        public static void main(String[] args) throws Exception {
                // TODO Auto-generated method stub
                while (true) {
                        Server sr = new Server();
                        sr.recPort(8000);
                        sr.recData();
                }

        }
}

public class Server {
	boolean hasToken = false;
	boolean sendData = false;
	int recport;

	void recPort(int recport) {
		// TODO Auto-generated method stub
		this.recport = recport;

	}

	public void recData() throws Exception {
		// TODO Auto-generated method stub
		byte bu[] = new byte[256];
		DatagramSocket ds;
		DatagramPacket dp;

		String str;
		ds = new DatagramSocket(recport);
		dp = new DatagramPacket(bu, bu.length);

		ds.receive(dp);
		ds.close();

		str = new String(dp.getData(), 0, dp.getLength());

		System.out.println("This message is " + str);
	}

}
```

TokenClient1.java:

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;

public class TokenClient1 {
	public static void main(String[] args) throws Exception {
		// TODO Auto-generated method stub
		InetAddress lclhost;
		BufferedReader br;
		String str = "";
		TokenClient12 tkcl, tkser;

		boolean hasToken;
		boolean setSendData;

		while (true) {
			lclhost = InetAddress.getLocalHost();
			tkcl = new TokenClient12(lclhost);
			tkser = new TokenClient12(lclhost);

			tkcl.setSendPort(9004);
			tkcl.setRecPort(8002);
			lclhost = InetAddress.getLocalHost();
			tkser.setSendPort(9000);
			if (tkcl.hasToken == true) {
				System.out.println("Do You want to enter the
Data -> YES/NO");
				br = new BufferedReader(new
InputStreamReader(System.in));
				str = br.readLine();
				if (str.equalsIgnoreCase("yes")) {
					System.out.println("Ready to Send Data");
					tkser.setSendData = true;
					tkser.sendData();
					tkser.setSendData = false;
				} else if (str.equalsIgnoreCase("no")) {
					System.out.println("I am in Else");
					tkcl.hasToken = false;
					tkcl.sendData();
					tkcl.recData();
					System.out.println("i am leaving Else");
				}
			} else {
				System.out.println("ENTERING RECIVING MODE");
				tkcl.recData();
				hasToken = true;
			}
```

```
                }

        }

}

class TokenClient12 {
        InetAddress lclhost;
        int sendport, recport;
        boolean hasToken = true;
        boolean setSendData = false;
        TokenClient12 tkcl, tkser;

        TokenClient12(InetAddress lclhost){
                this.lclhost = lclhost;
        }
        void setSendPort(int sendport) {
                this.sendport = sendport;
        }
        void setRecPort(int recport) {
                this.recport = recport;
        }

        void sendData() throws Exception{
                BufferedReader br;
                String str = "TOKEN";
                DatagramSocket ds;
                DatagramPacket dp;
                if(setSendData == true) {
                        System.out.println("Sending");
                        System.out.println("Enter the Data");
                        br = new BufferedReader(new
InputStreamReader(System.in));
                        str = "Client one..."+br.readLine();
                        System.out.println("Now Sending");
                }
                ds = new DatagramSocket(sendport);
                dp = new DatagramPacket(str.getBytes(), str.length(),
lclhost,sendport-1000);
                ds.send(dp);
                ds.close();
                setSendData = false;
                hasToken = false;
        }

        void recData() throws Exception{
                String msgStr;
                byte buffer[] = new byte[256];
                DatagramSocket ds;
                DatagramPacket dp;
                ds = new DatagramSocket(recport);
                dp = new DatagramPacket(buffer, buffer.length);
                ds.receive(dp);
                ds.close();
                msgStr = new String(dp.getData(),0, dp.getLength());
                System.out.println("The Data is "+ msgStr);
                if(msgStr.equals("TOKEN")) {
                        hasToken = true;
                }
        }
}
```

TokenClient2.java:

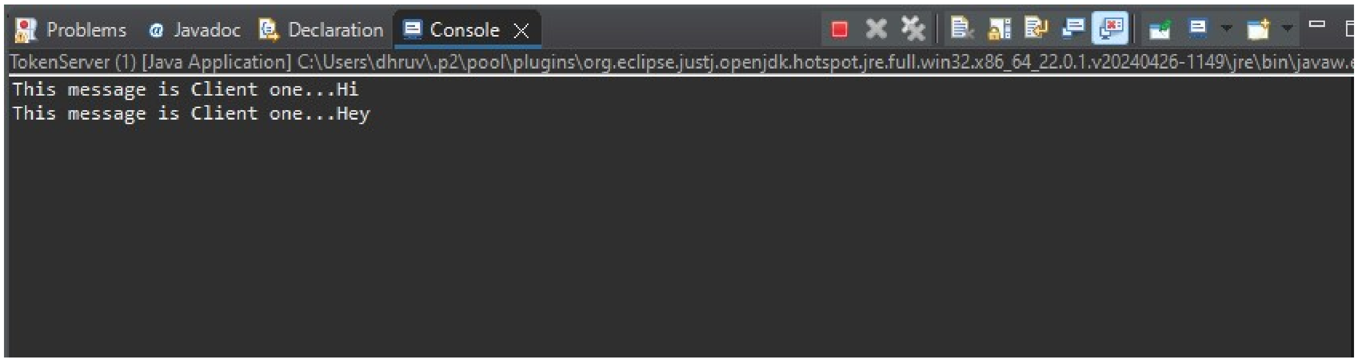
```
import java.io.*;
import java.net.*;
public class TokenClient2 {
  static boolean setSendData ; static boolean hasToken ;
public static void main(String arg[]) throws Exception
{
  InetAddress lclhost; BufferedReader br;
  String str1;
  TokenClient21 tkcl;
  TokenClient21 ser;
  while(true)
  {
     lclhost=InetAddress.getLocalHost();
     tkcl = new TokenClient21(lclhost);
     tkcl.setRecPort(8004);
     tkcl.setSendPort(9002);
     lclhost=InetAddress.getLocalHost();
     ser = new TokenClient21(lclhost);
     ser.setSendPort(9000);
     System.out.println("entering if");
     if(hasToken == true)
```

```
        {
           System.out.println("Do you want to enter the Data –> YES/NO");
           br=new BufferedReader(new InputStreamReader(System.in));
           str1=br.readLine();
           if(str1.equalsIgnoreCase("yes"))
           {
              System.out.println("ignorecase");
              ser.setSendData = true;
              ser.sendData();
           }
           else if(str1.equalsIgnoreCase("no"))
           {
              tkcl.sendData();
              tkcl.hasToken=false; tkcl.sendData(); tkcl.recData();
              hasToken=false;
           }
        }
        else
        {
           System.out.println("entering recieving mode");
           tkcl.recData();
           hasToken=true;
        }
     }
  }
  }
  class TokenClient21
  {
     InetAddress lclhost;
     int sendport,recport;
     boolean setSendData = false;
     boolean hasToken = false;
     TokenClient21 tkcl;
     TokenClient21 ser;
     TokenClient21(InetAddress lclhost)
     {
        this.lclhost = lclhost;
     }
     void setSendPort(int sendport)
     {
        this.sendport = sendport;
     }
```

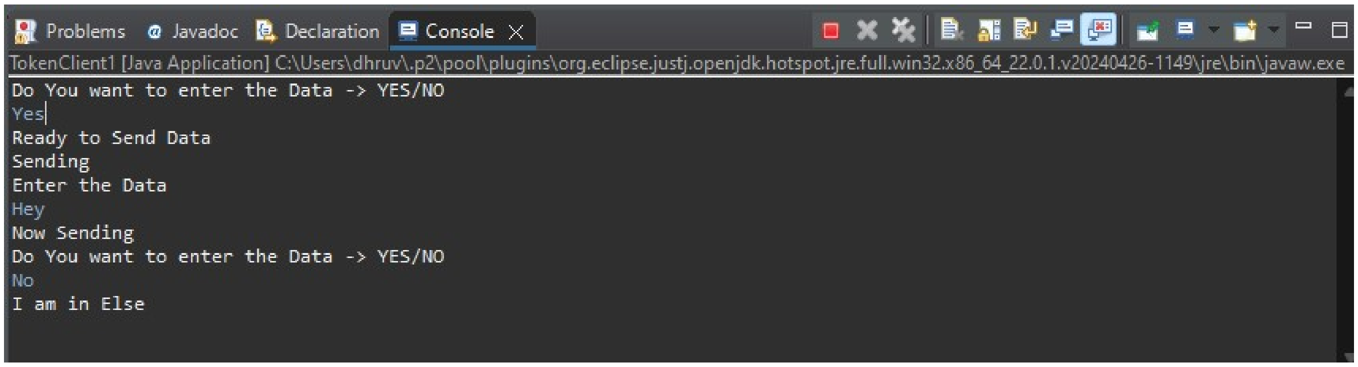
```
void setRecPort(int recport)
{
   this.recport = recport;
}
void sendData() throws Exception
{
   System.out.println("case");
   BufferedReader br;
   String str="Token";
   DatagramSocket ds;
   DatagramPacket dp;
   if(setSendData == true)
   {
      System.out.println("Enter the Data");
      br=new BufferedReader(new InputStreamReader(System.in));
      str ="ClientTwo….." + br.readLine();
   }
   ds = new DatagramSocket(sendport);
   dp = new DatagramPacket(str.getBytes(),str.length(),lclhost,sendport-1000);
   ds.send(dp);
   ds.close();
   System.out.println("Data Sent");
   setSendData = false; hasToken = false;
}
void recData()throws Exception
{
   String msgstr;
   byte buffer[] = new byte[256];
   DatagramSocket ds;
   DatagramPacket dp;
   ds = new DatagramSocket(recport);
   ds = new DatagramSocket(4000);
   dp = new DatagramPacket(buffer,buffer.length);
   ds.receive(dp);
   ds.close();
   msgstr = new String(dp.getData(),0,dp.getLength());
   System.out.println("The data is "+msgstr);
   if(msgstr.equals("Token"))
   {
      hasToken = true;
   }
}
```

}

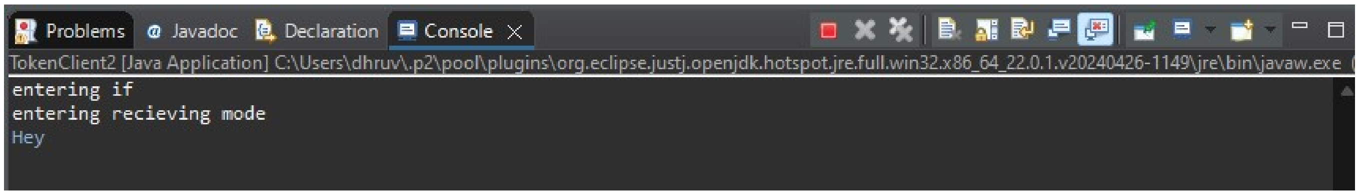
Output:



```
TokenServer (1) [Java Application] C:\Users\dhruv\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_22.0.1.v20240426-1149\jre\bin\javaw.e
This message is Client one...Hi
This message is Client one...Hey
```



```
TokenClient1 [Java Application] C:\Users\dhruv\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_22.0.1.v20240426-1149\jre\bin\javaw.exe
Do You want to enter the Data -> YES/NO
Yes
Ready to Send Data
Sending
Enter the Data
Hey
Now Sending
Do You want to enter the Data -> YES/NO
No
I am in Else
```



```
TokenClient2 [Java Application] C:\Users\dhruv\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_22.0.1.v20240426-1149\jre\bin\javaw.exe
entering if
entering recieving mode
Hey
```