Title: Implementation of mutual exclusion algorithm

1) Token Ring

Code:

TokenServer.java

```
package lab_04;

import java.net.*;

public class TokenServer {
	public static void main(String agrs[]) throws Exception {
		int port = 8000;
		System.out.println("Server Started on " + port);
		while (true) {
			Server sr = new Server();
			sr.recPort(port);
			sr.recData();
		}
	}
}

class Server {

	boolean hasToken = false;
	boolean sendData = false;
	int recport;

	void recPort(int recport) {
		this.recport = recport;
	}

	void recData() throws Exception {
		byte buff[] = new byte[256];
		DatagramSocket ds;
		DatagramPacket dp;
		String str;

		ds = new DatagramSocket(recport);
```

```
                dp = new DatagramPacket(buff, buff.length);
                ds.receive(dp);
                ds.close();

                str = new String(dp.getData(), 0, dp.getLength());
                System.out.println("The message is " + str);
        }
}
```

TokenClient.java

```
package lab_04;

import java.io.*;
import java.net.*;

public class TokenClient {
        public static void main(String arg[]) throws Exception {
                InetAddress localhost;
                BufferedReader br;
                String str = "";
                TokenClientInside tokenClient, tokenServer;

                while (true) {
                        localhost = InetAddress.getLocalHost();
                        tokenClient = new TokenClientInside(localhost);
                        tokenServer = new TokenClientInside(localhost);
                        tokenClient.setSendPort(9004);
                        tokenClient.setRecPort(8002);
                        localhost = InetAddress.getLocalHost();
                        tokenServer.setSendPort(9000);

                        if (tokenClient.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");
                                        tokenServer.setSendData = true;
                                        tokenServer.sendData();
                                        tokenServer.setSendData = false;
                                } else if (str.equalsIgnoreCase("no")) {
                                        System.out.println("Token Waiting...");
                                        tokenClient.sendData();
                                        tokenClient.recData();
                                }
                        } else {
                                System.out.println("ENTERING RECEIVING
MODE...");
                                tokenClient.recData();
                        }
                }
        }

}

class TokenClientInside {
        InetAddress localhost;
        int sendPort, recPort;
        boolean hasToken = true;
        boolean setSendData = false;

        TokenClientInside(InetAddress localhost) {
                this.localhost = localhost;
        }

        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("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(), localhost,
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

```
package lab_04;

import java.io.*;
import java.net.*;

public class TokenClient2 {
	static boolean setSendData;
	static boolean hasToken;

	public static void main(String arg[]) throws Exception {
		InetAddress localhost;
		BufferedReader br;
		String str1;
		TokenClientInside2 tokenClient;
		TokenClientInside2 Server;
		while (true) {
			localhost = InetAddress.getLocalHost();
			tokenClient = new TokenClientInside2(localhost);
			tokenClient.setRecPort(8004);
			tokenClient.setSendPort(9002);
			localhost = InetAddress.getLocalHost();
			Server = new TokenClientInside2(localhost);
			Server.setSendPort(9000);
			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("Ready to send");
					Server.setSendData = true;
					Server.sendData();
				} else if (str1.equalsIgnoreCase("no")) {
```

```
                                        System.out.println("Token Waiting...");
                                        tokenClient.sendData();
                                        hasToken = false;
                                }
                        } else {
                                System.out.println("ENTERING RECIEVING
MODE...");
                                tokenClient.recData();
                                hasToken = true;
                        }
                }
        }
}

class TokenClientInside2 {
        InetAddress localhost;
        int sendPort, recPort;
        boolean setSendData = false;
        boolean hasToken = false;

        TokenClientInside2(InetAddress localhost) {
                this.localhost = localhost;
        }

        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("Enter the Data");
                        br = new BufferedReader(new
InputStreamReader(System.in));
                        str = "Client Two: " + br.readLine();
                        System.out.println("Now sending...");
                }
                ds = new DatagramSocket(sendPort);
                dp = new DatagramPacket(str.getBytes(), str.length(), localhost,
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);
                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;
                }
        }

}
```

Screenshots:

Token Server:

```
TokenServer [Java Application] C:\Program Files\Java\jdk-15\bin\javaw.exe  (27-Oct-2020, 11:21:52 am)
Server Started on 8000
The message is Client One: Helo
The message is Client Two: Hi!
The message is Client One: How are you???
The message is Client Two: Doing good! What about you?
```

TokenClient1:

```
TokenClient [Java Application] C:\Program Files\Java\jdk-15\bin\javaw.exe  (27-Oct-2020, 11:21:55 am)
Do you want to enter the Data -> Yes/No?
Yes
Ready to send
Enter the Data:
Helo
Now sending...
Do you want to enter the Data -> Yes/No?
No
Token Waiting...
The data is Token
Do you want to enter the Data -> Yes/No?
Yes
Ready to send
Enter the Data:
How are you???
Now sending...
Do you want to enter the Data -> Yes/No?
No
Token Waiting...
```

TokenClient2:

```
TokenClient2 [Java Application] C:\Program Files\Java\jdk-15\bin\javaw.exe  (27-Oct-2020, 11:21:59 am)
The data is Token
Do you want to enter the Data -> YES/NO
Yes
Ready to send
Enter the Data
Hi!
Now sending...
Data sent
Do you want to enter the Data -> YES/NO
No
Token Waiting...
Data sent
ENTERING RECIEVING MODE...
The data is Token
Do you want to enter the Data -> YES/NO
Yes
Ready to send
Enter the Data
Doing good! What about you?
Now sending...
Data sent
Do you want to enter the Data -> YES/NO
```