SIDDHI S.KOTRE
ROLL NO: 50

Module 5 - Mutual Exclusion

Implementation of Mutual Exclusion using Token Ring Technique

Aim: 1.Write a program to demonstrate token ring technique.

Hint:

1.Create a server program which accepts data from clients.

2.Create 2 client programs who send a token to the next client when it enters a busy state.

Theory:

The Token Ring technique is a popular method for implementing mutual exclusion in distributed systems. It operates by passing a token (a special message) between nodes (in this case, clients), and only the node holding the token is allowed to enter the critical section (busy state). After completing its critical section, the node passes the token to the next node in the ring.

Code:

TokenRingServer.java

```
package practical;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
public class TokenRingServer {

	public static DatagramSocket datagramSocket;
	 public static DatagramPacket datagramPacket;

	public static void main(String[] args) throws Exception {
				datagramSocket = new DatagramSocket(1000);
				System.out.println("Siddhi Santosh Kotre: 50\n");
				System.out.println("Token Server is running on port 1000 and waiting for
messages...");
		while (true) {
		   byte buffer[] = new byte[1024];
		   datagramSocket.receive(datagramPacket = new DatagramPacket(buffer, buffer.length));
		   String receivedMessage = new String(datagramPacket.getData(), 0,
datagramPacket.getLength());
		   System.out.println("Server message received: " + receivedMessage);
		}
		    }
	}
```

TokenRingClient1.java

```
package practical;
	import java.io.BufferedReader;
	import java.io.InputStreamReader;
	import java.net.DatagramPacket;
	import java.net.DatagramSocket;
	import java.net.InetAddress;
	public class TokenRingClient1 {

		public static DatagramSocket datagramSocket;
	    public static DatagramPacket datagramPacket;
	    public static BufferedReader bufferedReader;

		public static void main(String[] args) throws Exception {
			boolean hasToken = true;
	        datagramSocket = new DatagramSocket(100);
	        System.out.println("Siddhi Santosh Kotre: 50\n");
	        System.out.println("Client 1 started and listening on port 100.");
	        while (true) {
	            if (hasToken) {
	                System.out.println("\n Client 1 currently holds the token.");
	                System.out.println("Do you want to write data to the server? (yes/no)");
	                bufferedReader = new BufferedReader(new InputStreamReader(System.in));
	                String response = bufferedReader.readLine();
	                if (response.equalsIgnoreCase("yes")) {
	                    System.out.println("Client 1 is ready to write data.");
	                    System.out.println("Please enter the data to send to the server:");
	                    String data = "Client 1 -> " + bufferedReader.readLine();
	                    byte buffer[] = new byte[1024];
	                    buffer = data.getBytes();
	                    datagramSocket.send(new DatagramPacket(buffer, buffer.length,
InetAddress.getLocalHost(), 1000));
	                    System.out.println("Client 1 sent data to the server: " + data);
	                } else if (response.equalsIgnoreCase("no")) {
	                    System.out.println("Client 1 is in busy state. Passing the token to Client 2...");
	                    String tokenMessage = "token";
	                    byte buffer[] = new byte[1024];
	                    buffer = tokenMessage.getBytes();
	                    datagramSocket.send(new DatagramPacket(buffer, buffer.length,
InetAddress.getLocalHost(), 200));
	                    hasToken = false;
	                }
	            } else {
	                System.out.println("\nClient 1 is waiting for the token...");
	                byte buffer[] = new byte[1024];
	                datagramSocket.receive(datagramPacket = new DatagramPacket(buffer,
buffer.length));
```

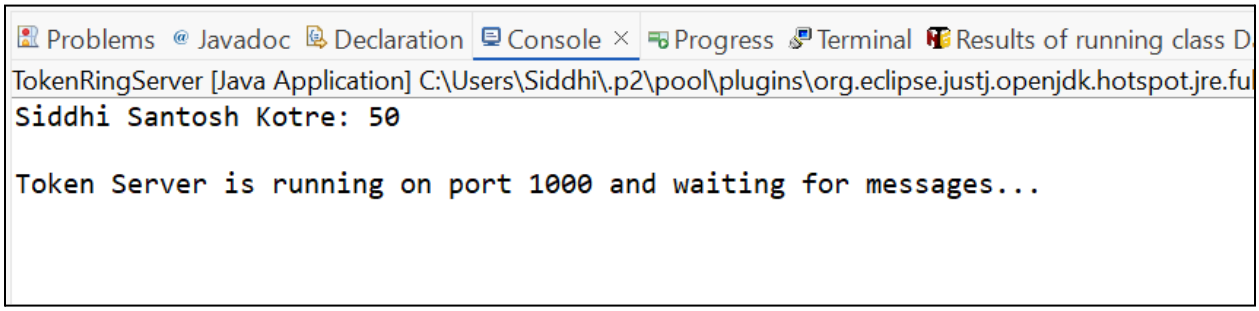
```
                String clientMessage = new String(datagramPacket.getData(), 0,
datagramPacket.getLength());
                System.out.println("Client 1 received message: " + clientMessage);
                if (clientMessage.equals("token")) {
                    hasToken = true;
                    System.out.println("Client 1 has received the token and is leaving the busy state.");
                }
            }
        }
            }
    }
```

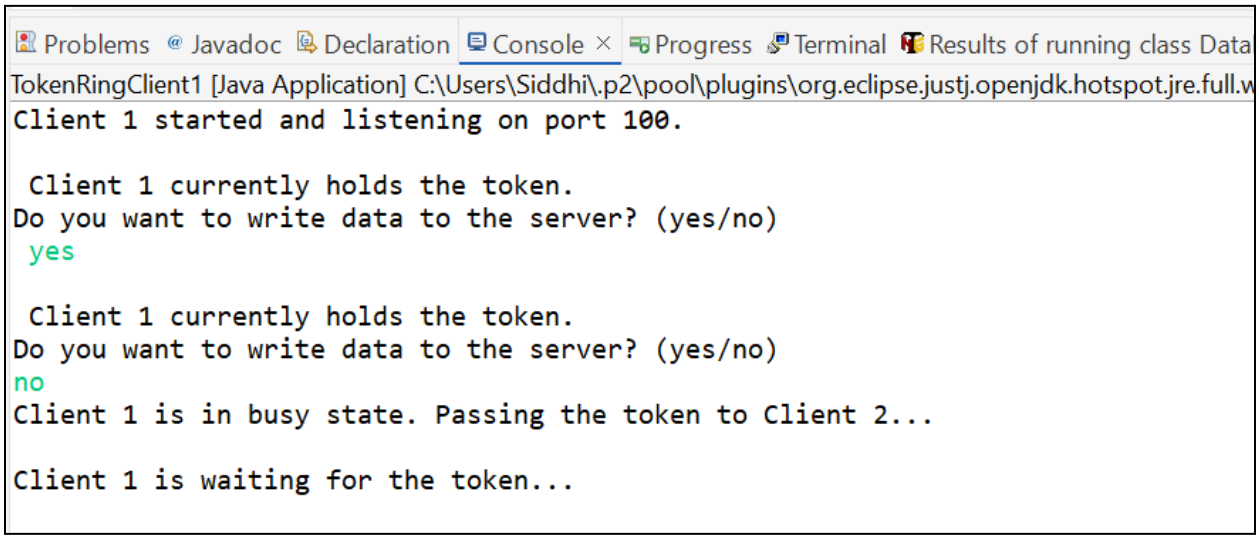
TokenRingClient2.java

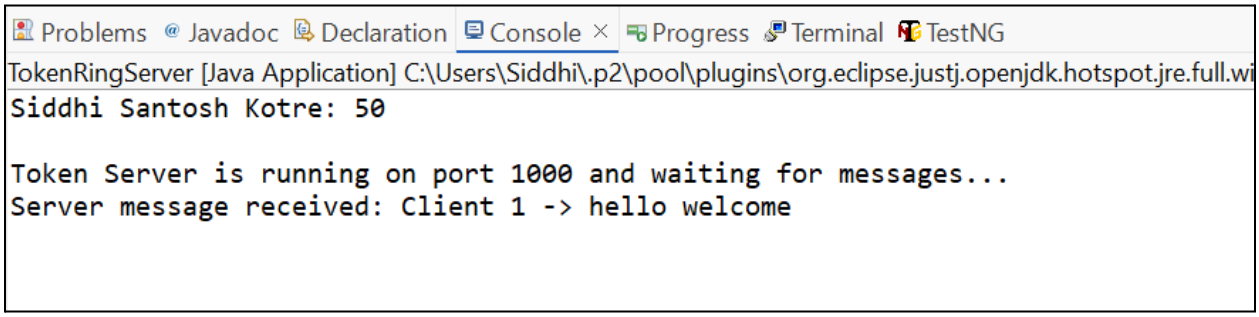
```
package practical;
    import java.io.BufferedReader;
    import java.io.InputStreamReader;
    import java.net.DatagramPacket;
    import java.net.DatagramSocket;
    import java.net.InetAddress;
    public class TokenRingClient2 {

        public static DatagramSocket datagramSocket;
      public static DatagramPacket datagramPacket;
      public static BufferedReader bufferedReader;

        public static void main(String[] args) throws Exception {
            boolean hasToken = false;
        datagramSocket = new DatagramSocket(200);
        System.out.println("Siddhi Santosh Kotre: 50\n");
        System.out.println("Client 2 started and listening on port 200.");
        while (true) {
          if (hasToken) {
           System.out.println("\nClient 2 currently holds the token.");
            System.out.println("Do you want to write data to the server? (yes/no)");
            bufferedReader = new BufferedReader(new InputStreamReader(System.in));
            String response = bufferedReader.readLine();
            if (response.equalsIgnoreCase("yes")) {
                System.out.println("Client 2 is ready to write data.");
            System.out.println("Please enter the data to send to the server:");
            String data = "Client 2 -> " + bufferedReader.readLine();
            byte buffer[] = new byte[1024];
            buffer = data.getBytes();
            datagramSocket.send(new DatagramPacket(buffer, buffer.length,
InetAddress.getLocalHost(), 1000));
            System.out.println("Client 2 sent data to the server: " + data);
          } else if (response.equalsIgnoreCase("no")) {
```

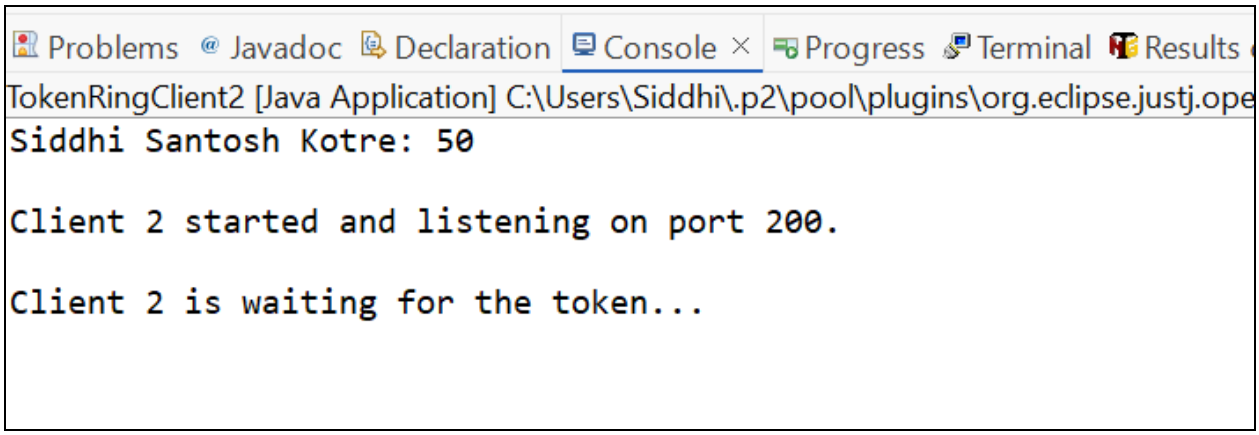
```
                        System.out.println("Client 2 is in busy state. Passing the token back to Client
1...");
                    String tokenMessage = "token";
                    byte buffer[] = new byte[1024];
                    buffer = tokenMessage.getBytes();
                    datagramSocket.send(new DatagramPacket(buffer, buffer.length,
InetAddress.getLocalHost(), 100));
                    System.out.println("Token passed back to Client 1.");
                    hasToken = false;
                }
            } else {
                try {
                        System.out.println("\nClient 2 is waiting for the token...");
                    byte buffer[] = new byte[1024];
                    datagramSocket.receive(datagramPacket = new DatagramPacket(buffer,
buffer.length));
                    String clientMessage = new String(datagramPacket.getData(), 0,
datagramPacket.getLength());
                    System.out.println("Client 2 received message: " + clientMessage);
                    if (clientMessage.equals("token")) {
                        hasToken = true;
                        System.out.println("Client 2 has received the token and is leaving the busy
state.");
                    }
                } catch (Exception e) {
                    // Handle exception if needed
                }
            }
        }
    }
```

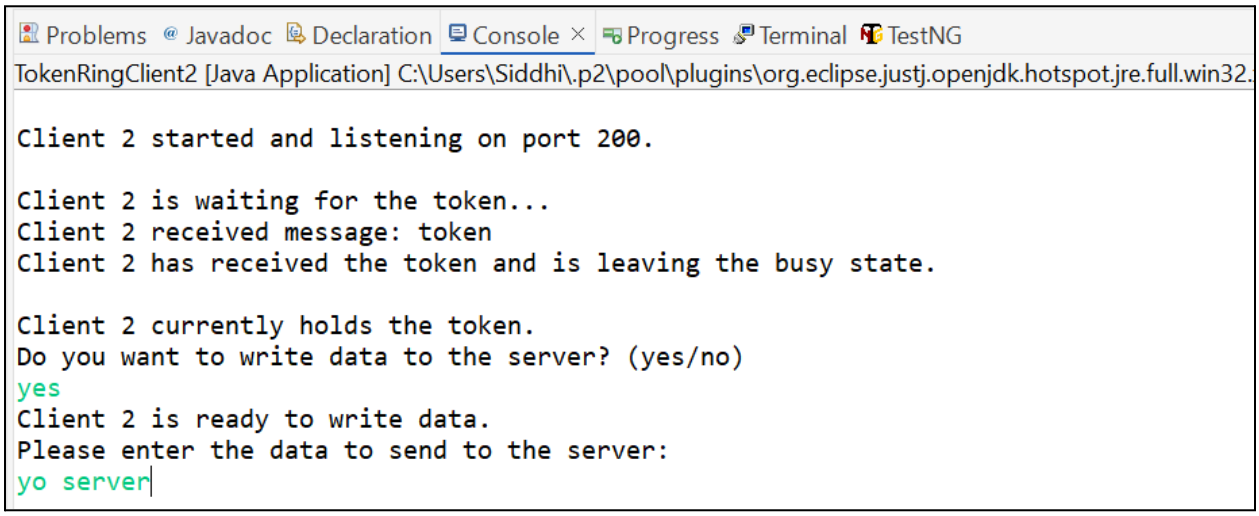
Output:

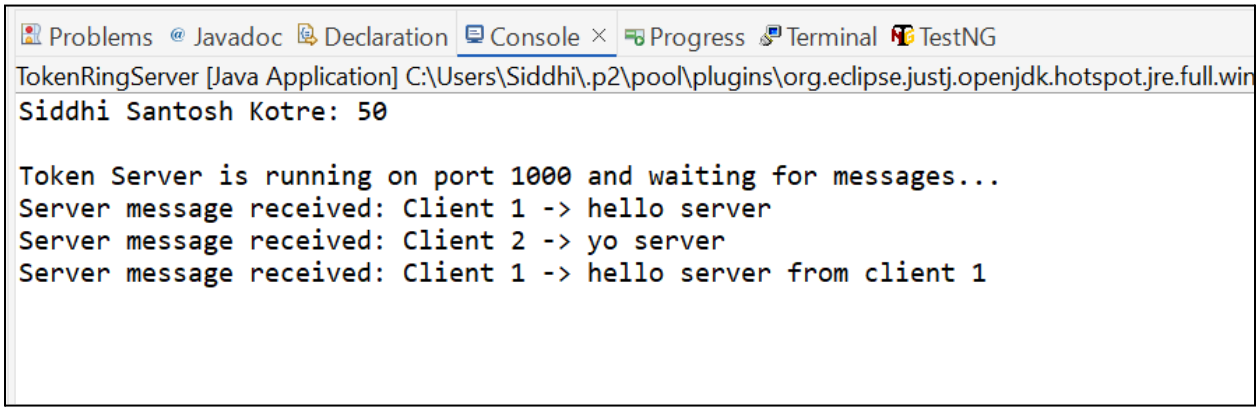

Problems Javadoc Declaration Console Progress Terminal Results of running class D
TokenRingServer [Java Application] C:\Users\Siddhi\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.ful
Siddhi Santosh Kotre: 50
Token Server is running on port 1000 and waiting for messages...


Problems Javadoc Declaration Console Progress Terminal Results of running class Data
TokenRingClient1 [Java Application] C:\Users\Siddhi\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.w
Client 1 started and listening on port 100.
Client 1 currently holds the token.
Do you want to write data to the server? (yes/no)
yes
Client 1 currently holds the token.
Do you want to write data to the server? (yes/no)
no
Client 1 is in busy state. Passing the token to Client 2...
Client 1 is waiting for the token...


Problems Javadoc Declaration Console Progress Terminal TestNG
TokenRingServer [Java Application] C:\Users\Siddhi\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.wi
Siddhi Santosh Kotre: 50
Token Server is running on port 1000 and waiting for messages...
Server message received: Client 1 -> hello welcome


Problems Javadoc Declaration Console Progress Terminal Results
TokenRingClient2 [Java Application] C:\Users\Siddhi\.p2\pool\plugins\org.eclipse.justj.ope
Siddhi Santosh Kotre: 50
Client 2 started and listening on port 200.
Client 2 is waiting for the token...


Problems Javadoc Declaration Console Progress Terminal TestNG
TokenRingClient2 [Java Application] C:\Users\Siddhi\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.
Client 2 started and listening on port 200.
Client 2 is waiting for the token...
Client 2 received message: token
Client 2 has received the token and is leaving the busy state.
Client 2 currently holds the token.
Do you want to write data to the server? (yes/no)
yes
Client 2 is ready to write data.
Please enter the data to send to the server:
yo server


Problems Javadoc Declaration Console Progress Terminal TestNG
TokenRingServer [Java Application] C:\Users\Siddhi\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win
Siddhi Santosh Kotre: 50
Token Server is running on port 1000 and waiting for messages...
Server message received: Client 1 -> hello server
Server message received: Client 2 -> yo server
Server message received: Client 1 -> hello server from client 1