Assignment 1

الاسم : شمس الدين عبدالرحمن علي علي الجوهري

Section : 2

ID: 1700684

i)

Role of Mobile Agents in this case:

We have 3 nodes:

· Surveillance and sensor node

· Server node

· Driver (Computer) node

Surveillance and sensor node gathers information on the area from the set of sensors, surveillance cameras, electronic traffic signs.

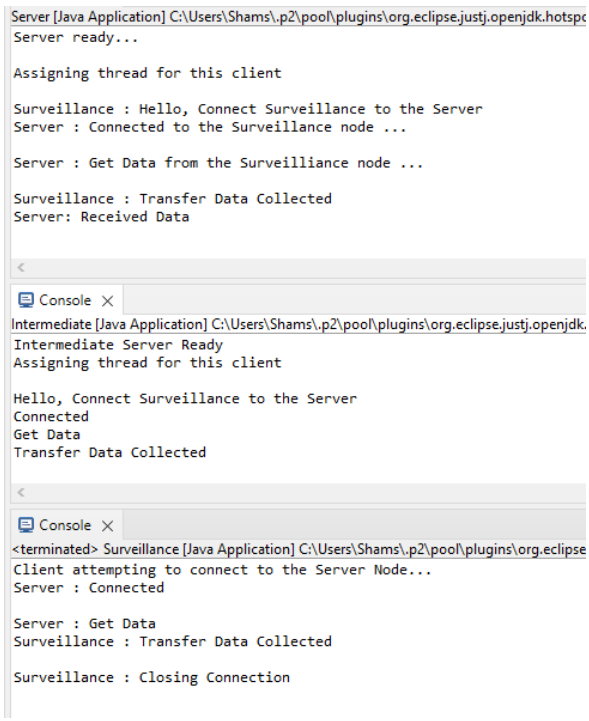
Server node receives the information gathered from the surveillance and sensor nodes and starts processing them to find the best route for the driver.

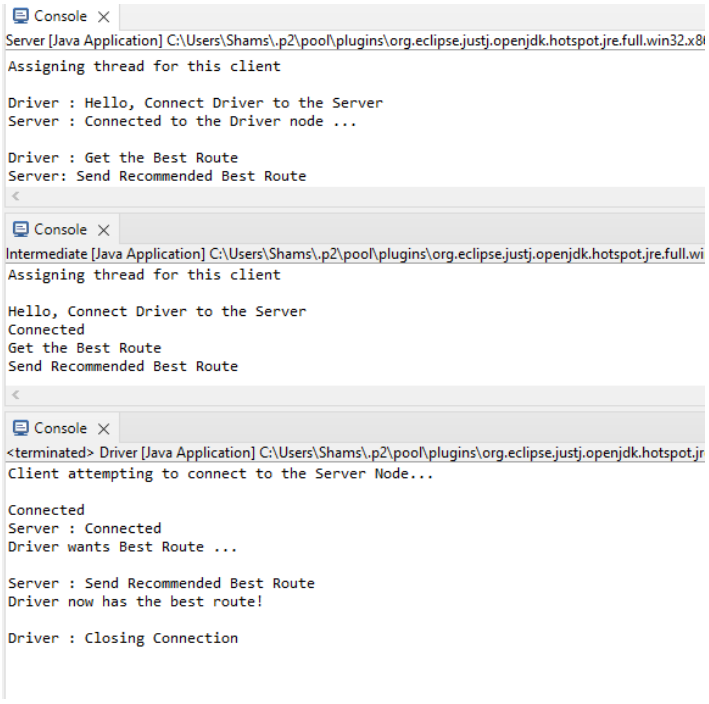
Driver node receives the recommendation of the best route.

Mobile agent can be used as an intermediate between server and two clients which reads from the clients and writes the client message to the server and also reads from the server and sends the server message to the clients.

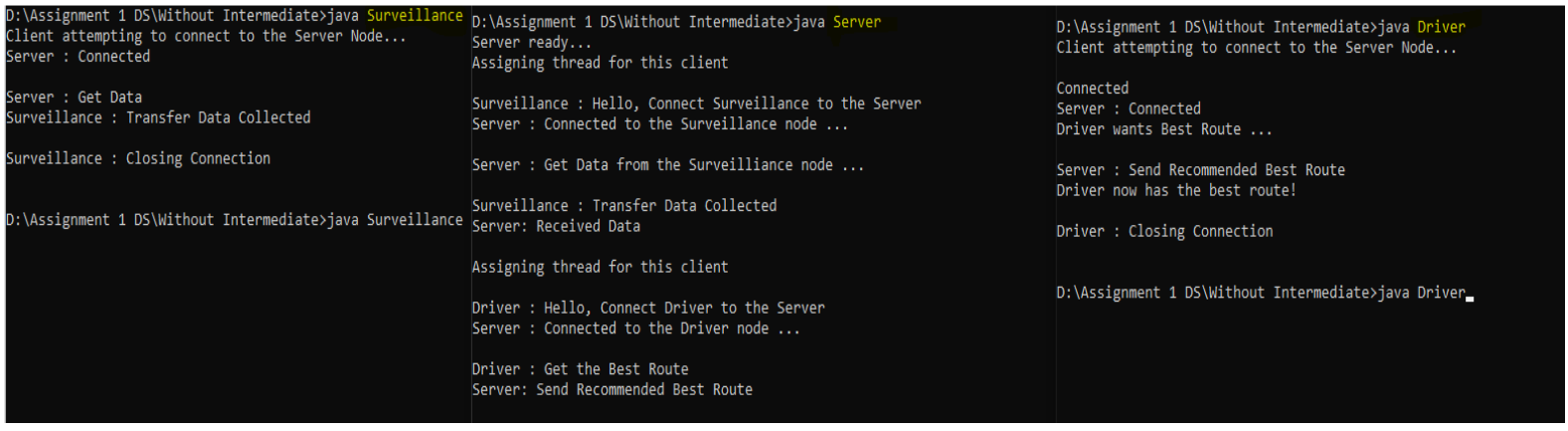
ii)

With intermediate: (Compiled and ran code on Eclipse IDE)





Without intermediate: (Compiled and ran code on the command prompt)

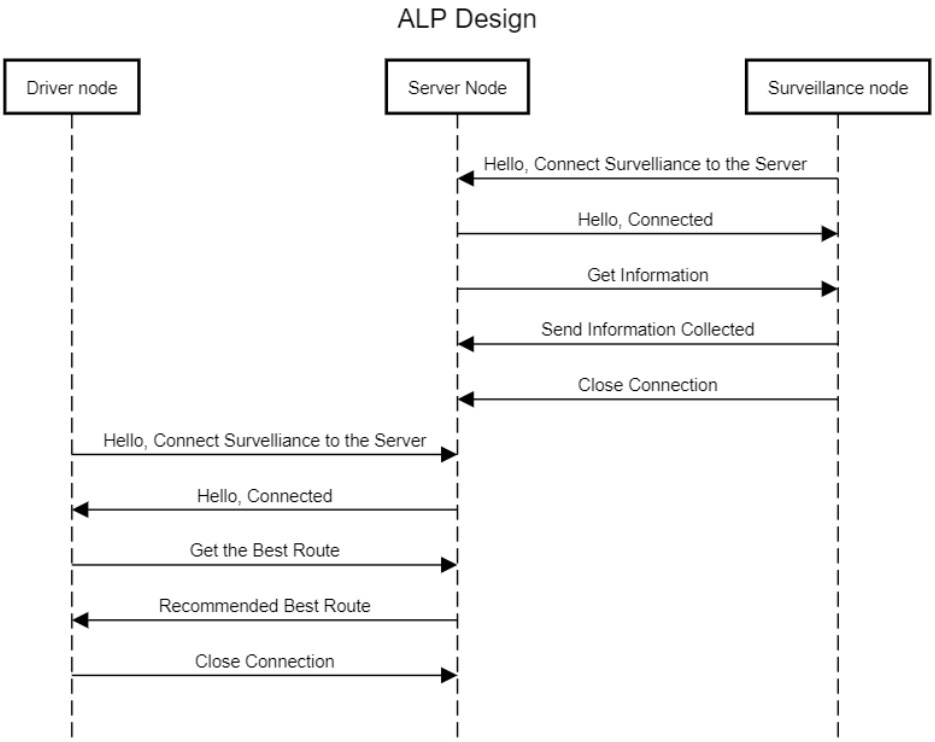


Intermediate Code:

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

public class Intermediate {

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

        try {
            ServerSocket IntermediateServer = new ServerSocket(5858);
            Socket ClientSocket = new Socket("LocalHost", 2222);
            System.out.println("Intermediate Server Ready");

        while(true)
        {
            /* Wait and accept a connection a connection */
            Socket Client = IntermediateServer.accept();
            DataOutputStream DataOut = new DataOutputStream(Client.getOutputStream());
            DataInputStream DataIn = new DataInputStream(Client.getInputStream());
            System.out.println("Assigning thread for this client \n");
            IntermediateHandler thread = new IntermediateHandler(Client, ClientSocket, DataIn, DataOut);
            thread.start();
        }

        }catch (IOException e) {    e.printStackTrace();    }

    }

}


class IntermediateHandler extends Thread{

    DataInputStream  ClientDataIn, ServerDataIn;
    DataOutputStream ClientDataOut, ServerDataOut;
    private Socket client;
    private Socket server;

    public IntermediateHandler(Socket server, Socket client, DataInputStream DataIn, DataOutputStream DataOut) {
        this.server = server;
        this.client = client;
        this.ClientDataIn = DataIn;
        this.ClientDataOut= DataOut;
    }


    @Override
    public synchronized void run(){
        String ServerMessage = null;
        String ClientMessage = null;
        try{
         ServerDataIn = new DataInputStream(client.getInputStream());
         ServerDataOut = new DataOutputStream(client.getOutputStream());
        }
         catch(Exception e){}
        while(true)
        {
            try {
                try{
                    sleep(1500);
                    }
                    catch(Exception e){}

                if ( ClientDataIn.available() > 0)
                    ClientMessage = ClientDataIn.readUTF();
                else
                    ClientMessage = null;


                if (ClientMessage != null)
                {
                    ServerDataOut.writeUTF(ClientMessage);
                    System.out.println(ClientMessage);
                }


                if ( ServerDataIn.available() > 0)
                    ServerMessage = ServerDataIn.readUTF();
                else
                    ServerMessage = null;


                if (ServerMessage != null)
                {
                    ClientDataOut.writeUTF(ServerMessage);
                    System.out.println( ServerMessage );
                }

            } catch (IOException e) {
                e.printStackTrace();
                break;
            }
        }
        try
        {
            this.ClientDataIn.close();
            this.ClientDataOut.close();
        }
        catch(EOFException e){
            e.printStackTrace();
        }
        catch(IOException e){
            e.printStackTrace();
        }
    }


}
```

Rest of the files can be found at the end.

iii)



title ALP Design
participant Driver node
participant Server Node
participant Surveillance node

Surveillance node ->Server Node: Hello, Connect Surveliance to the Server
Surveillance node <-Server Node: Hello, Connected
Surveillance node <-Server Node: Get Information
Surveillance node ->Server Node: Send Information Collected
Surveillance node ->Server Node: Close Connection
Driver node ->Server Node: Hello, Connect Surveliance to the Server
Driver node <- Server Node: Hello, Connected
Driver node ->Server Node: Get the Best Route
Driver node <-Server Node: Recommended Best Route
Driver node -> Server Node: Close Connection

iv) Implementation Code

Server.java:

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

public class Server {

	public static void main(String[] args) throws IOException {
		try {
			ServerSocket MyServer = new ServerSocket(5858);
			System.out.println("Server ready...");

			while (true) {
				/* Wait and accept a connection a connection */
				Socket Client = MyServer.accept();
				DataOutputStream DataOut = new DataOutputStream(Client.getOutputStream());
				DataInputStream DataIn = new DataInputStream(Client.getInputStream());
				System.out.println("Assigning thread for this client \n");
				ClientHandler thread = new ClientHandler(Client, DataIn, DataOut);
				thread.start();
			}

		} catch (IOException e) {
			e.printStackTrace();
		}

	}

}

class ClientHandler extends Thread {

	final DataInputStream DataIn;
	final DataOutputStream DataOut;
	final Socket client;

	public ClientHandler(Socket Client, DataInputStream DataIn, DataOutputStream DataOut) {
		this.client = Client;
		this.DataIn = DataIn;
		this.DataOut = DataOut;
	}

	@Override
	public synchronized void run() {

		String Message;

		while (true) {
			try {

				Message = new String(DataIn.readUTF());

				/* Surveillance Node Connection */
				if (Message.equals("Hello, Connect Surveillance to the Server")) {
					System.out.println("Surveillance : " + Message);
					DataOut.writeUTF("Connected");
					System.out.println("Server : Connected to the Surveillance node ... \n");

					// ASK SURVEILLANCE NODE FOR DATA TO PROCESS
					DataOut.writeUTF("Get Data");
				System.out.println("Server : Get Data from the Surveilliance node ... \n");
				}

				/* Driver Node Connection */
				else if (Message.equals("Hello, Connect Driver to the Server")) {
					System.out.println("Driver : " + Message);
					DataOut.writeUTF("Connected");
					System.out.println("Server : Connected to the Driver node ... \n");
				}

				/* Surveillance Node Data Transfer to Server */
				else if (Message.equals("Transfer Data Collected")) {
					System.out.println("Surveillance : " + Message);
					System.out.println("Server: Received Data \n");
```

```
                                        }

                                        /* Reccemonding the best route */
                                        else if (Message.equals("Get the Best Route")) {
                                                /* Sending Reccemondation to the driver node */
                                                System.out.println("Driver : " + Message);
                                                System.out.println("Server: Send Recommended Best Route \n");
                                                DataOut.writeUTF("Send Recommended Best Route");
                                        }

                                } catch (EOFException e) {
                                        break;
                                } catch (IOException e) {
                                        e.printStackTrace();
                                }
                        }

                        try {
                                DataOut.close();
                                DataIn.close();
                                client.close();
                        } catch (EOFException e) {
                                e.printStackTrace();
                        } catch (IOException e) {
                                e.printStackTrace();
                        }

                }

}
```

Client : Surveillance.java

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

public class Surveillance {

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

                Socket SurveillanceNode = new Socket("LocalHost", 5858);
                DataInputStream Surveillance_In = new DataInputStream(SurveillanceNode.getInputStream());
                DataOutputStream Surveillance_Out = new DataOutputStream(SurveillanceNode.getOutputStream());

                System.out.println("Client attempting to connect to the Server Node... ");
                Surveillance_Out.writeUTF("Hello, Connect Surveillance to the Server");

                String Message = new String(Surveillance_In.readUTF());
                if (Message.equals("Connected")) {
                        System.out.println("Server : " + Message + "\n");
                }

                Message = new String(Surveillance_In.readUTF());
                if (Message.equals("Get Data")) {
                        System.out.println("Server : " + Message);
                        Surveillance_Out.writeUTF("Transfer Data Collected");
                        System.out.println("Surveillance : Transfer Data Collected \n");
                }

                System.out.println("Surveillance : Closing Connection \n");
                Surveillance_In.close();
                Surveillance_Out.close();
                SurveillanceNode.close();
        }

}
```

Client : Driver.java

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

public class Driver {

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

		Socket DriverNode = new Socket("LocalHost", 5858);

		DataInputStream Driver_In = new DataInputStream(DriverNode.getInputStream());
		DataOutputStream Driver_Out = new DataOutputStream(DriverNode.getOutputStream());

		System.out.println("Client attempting to connect to the Server Node... \n");

		Driver_Out.writeUTF("Hello, Connect Driver to the Server");

		String Message = new String(Driver_In.readUTF());
		System.out.println(Message);

		if (Message.equals("Connected")) {
			System.out.println("Server : " + Message);
			System.out.println("Driver wants Best Route ... \n");
			Driver_Out.writeUTF("Get the Best Route");
		}

		Message = new String(Driver_In.readUTF());

		if (Message.equals("Send Recommended Best Route")) {
			System.out.println("Server : " + Message);
			System.out.println("Driver now has the best route! \n");
		}

		System.out.println("Driver : Closing Connection \n");
		Driver_In.close();
		Driver_Out.close();
		DriverNode.close();

	}

}
```