Applied Distributed Systems COIT 13229

Assessment item 1—Java UDP Client Server Application

Due date:	**Week 5 T1-18 –** 11.45PM AEST, Friday 6th April 2018	**ASSESSMENT**
Weighting:	20%	
Length:	N/A	**1**

Objectives

This assessment item is designed to test your understanding in Java UDP networking, file reading and writing using client/server application development.

Assessment task

Your task for this assignment is, to design and implement UDP (User Datagram Protocol) API Socket in a client/server model. In java Datagram Socket uses connectionless protocol with basic send and receive methods. This assignment is built on the learning outcomes from the supplied code for UDPClient and UDPServer from week2 lecture and tutorial materials.

The UDPClient can send offline text messages to the UDP server and because this protocol is not reliable the server should acknowledge the sender that it has received the message. Multiple clients should be able to send messages to the server and no threading is required on the server. The server waits constantly receiving and acknowledging the clients for the messages it receives. When a message is received by the server the IP address of the client socket is used to determine the client's location. However multiple clients can also connect from same location when all the clients and the server are running locally on the same machine. So we distinguish each of them by a location which the user needs to enter before the interaction can start with the server. The client should only be able to transmit messages after entering a valid location. A valid name can be any alphanumeric but cannot be empty. The UDPClient should be able to continuously chat to the UDPServer. The server should store all the messages received in to an ArrayList and a timer should be set up to write contents of the ArrayList to a file.

The timer Schedule can be implemented in java as follows:

eg) class WriteToFile extends TimeTask and overrides run() method to do the file writing.

```
Timer tm=new Timer();

 tm.schedule(new WriteToFile(),long delay, long period);
```

java inbuilt Collections.sort() can be used to sort the ArrayList before writing to the file. Since the contents of the input and output file is in text format, Java IO classes Scanner and PrintWriter can be used for simplicity.

Case Study

The Hills School is a leading private educational institution providing primary school education for children across three major cities (Melbourne, Sydney and Brisbane) in Australia. Currently each school maintains a manual log book which each parent/guardian need to sign in and sign off when they drop and pick up their children from school. The school would like to have a centralised system to keep track of this process electronically. When a child is enrolled the school registers a mobile number and a pin which the parent need to use to sign in and sign off. A Kiosk will be set up to run the UDPClient. When the kiosk is powered up in the morning, the school location need to be entered. However in reality this is not required because when the server receives a message from the kiosk the client socket address is used to determine the location. For simplicity and testing since multiple clients can run locally, we are required to enter the location when the client starts and this cannot be empty. The server should persist the client sign in and sign off details to a log file and also display this information on the server side screen.

Part 1 Design: Java UDP Networking, and writing to a text file

A simple GUI (Graphical User Interface) java application needs to be developed for UDPClient and UDPServer.

A JOptionPane can be used to get the location at the start of the Client application and this cannot be empty. A better way is to use a dropdown box for location as in sample screens provided, this eliminated data entry errors.

The message to be transmitted consists of mobile number, pin and reason (Sign In/Sign Out). The message fields need to be separated by a colon ":" before dispatching to the server, this is not entered by the user. No information need to be stored on the client side and all validations need to be done on the server side.

Client sends to UDP Server a message consisting of:

Location: Mobile Number: Pin: Reason: Time

UDPServer stamps this message with the IP address from the received client and adds this to an ArrayList :

Location: Mobile Number: Pin: Reason: Time: IP address of the client socket

The server should initially load from an existing text file with a list of phone numbers and pin's which the admin staff always keep up to date. You can implement this in two parallel String arrays containing mobile numbers and pin's or use two separate Array Lists. This text file with phone numbers and pin numbers will be provided.

Also Java provides String. Split (":") to break up the incoming sentence in to an array of Strings

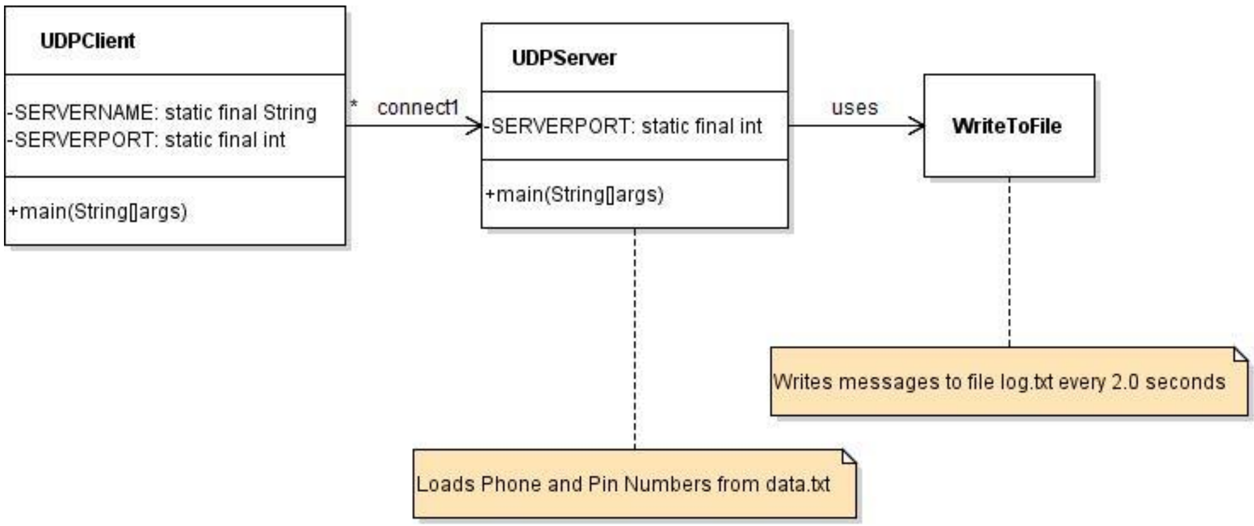
On the client side after the message is sent a Status with single ("*") and when acknowledged by the server should be shown as ("**") along with any error message if the phone number and pin's do not match up with the server's list. Possible states include ("Success", "Invalid number", "Invalid Pin"). The server first validates phone number before proceeding to validate the pin number.

Every 2 seconds the server should sort the list on the school location and over writes the entire list to a file

User should have an option to clear the Array List on the server.

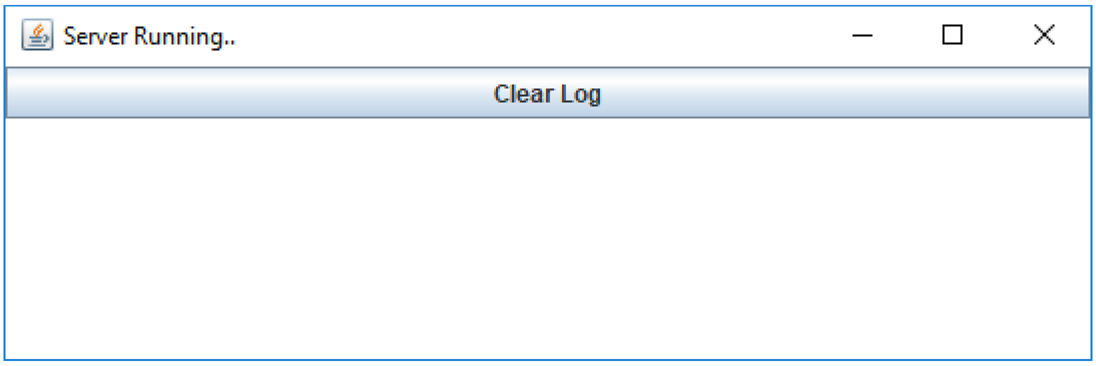
Class Diagram

SERVERNAME="localhost"   SERVERPORT=8888

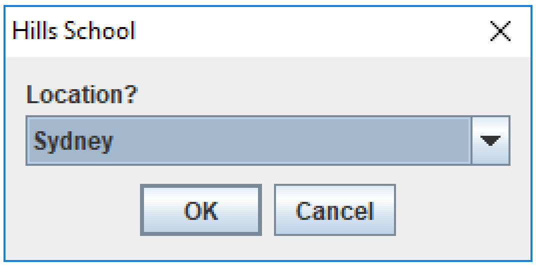

UDPClient
-SERVERNAME: static final String
-SERVERPORT: static final int
+main(String[]args)
* connect1
UDPServer
-SERVERPORT: static final int
+main(String[]args)
uses
WriteToFile
Writes messages to file log.txt every 2.0 seconds
Loads Phone and Pin Numbers from data.txt

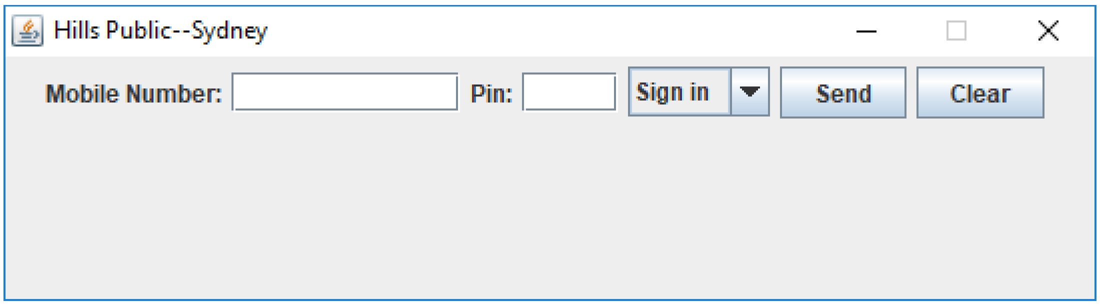
User Interface Screens

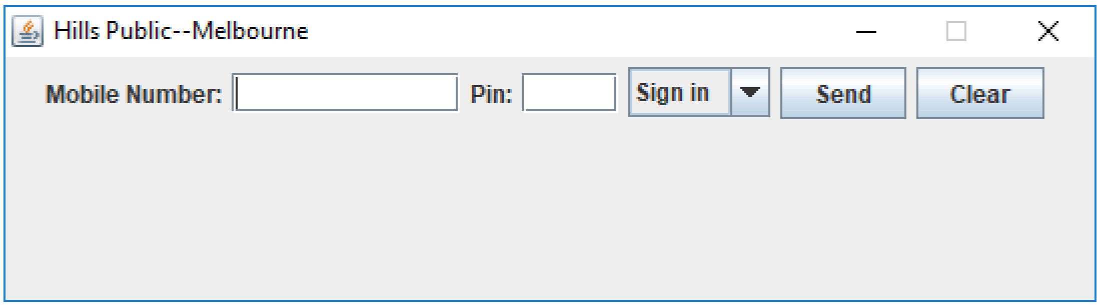
Starting Server

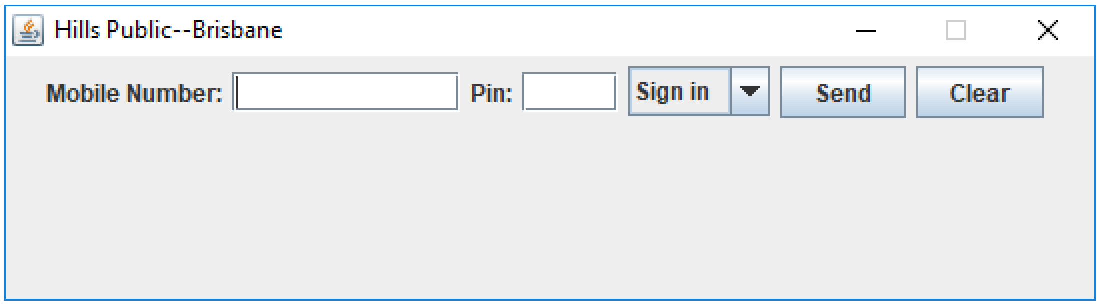


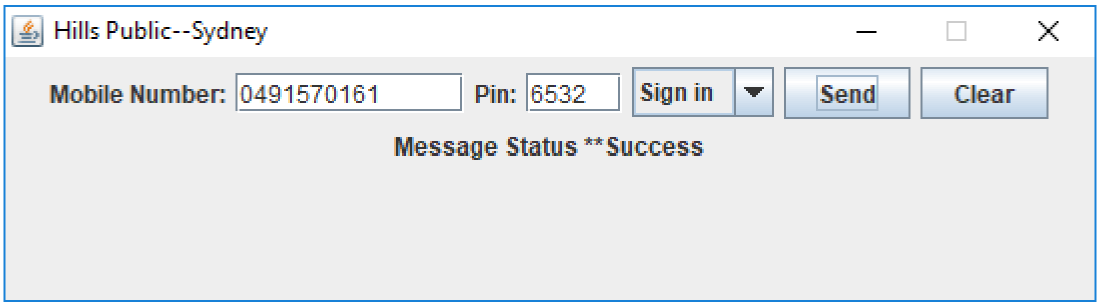
Starting Clients'





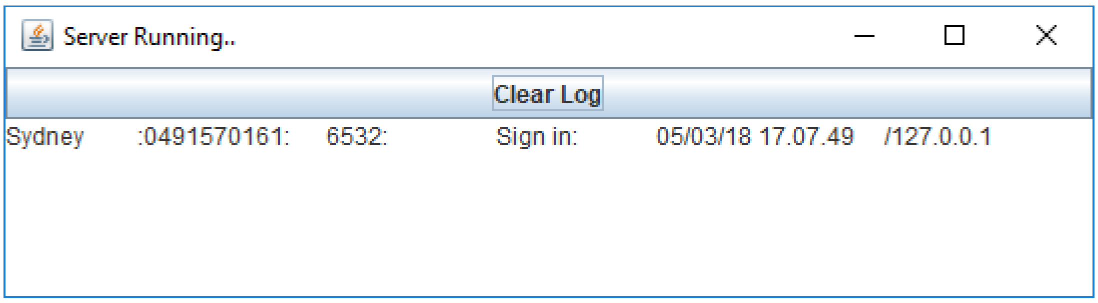






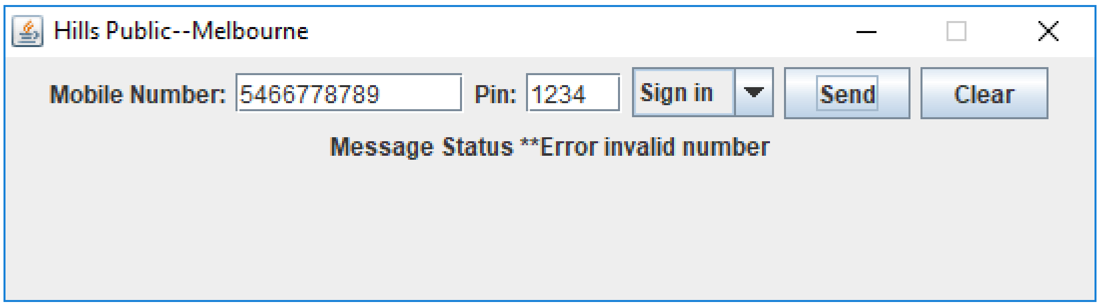
Mobile Number: 0491570161 Pin: 6532

Message Status **Success



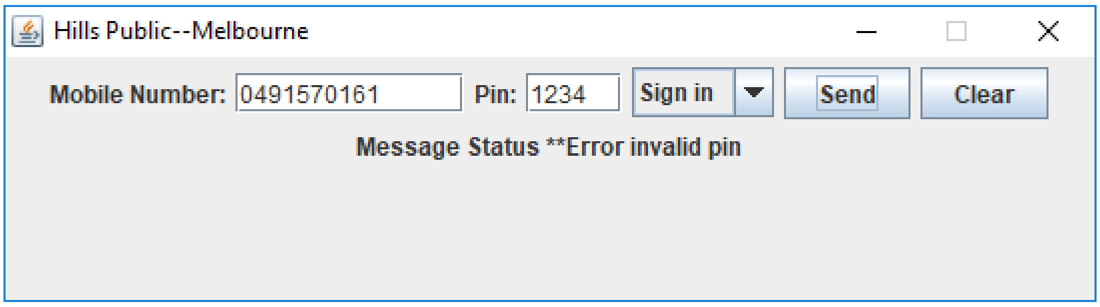
```
Sydney     :0491570161:    6532:          Sign in:        05/03/18 17.07.49    /127.0.0.1
```

Error Screens



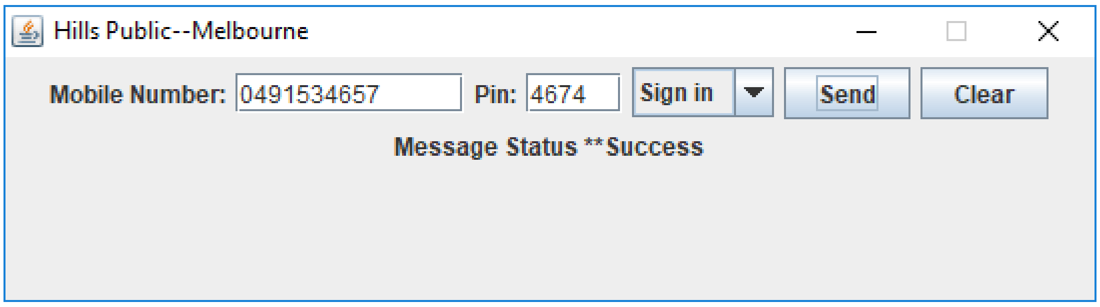
Mobile Number: 5466778789 Pin: 1234

Message Status **Error invalid number



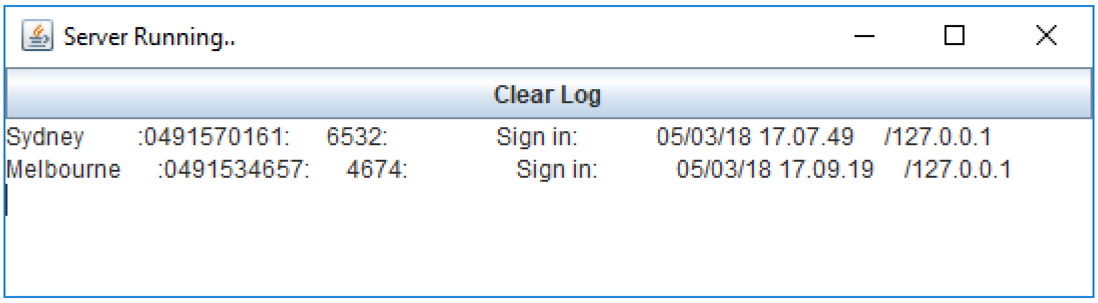
Mobile Number: 0491570161 Pin: 1234

Message Status **Error invalid pin

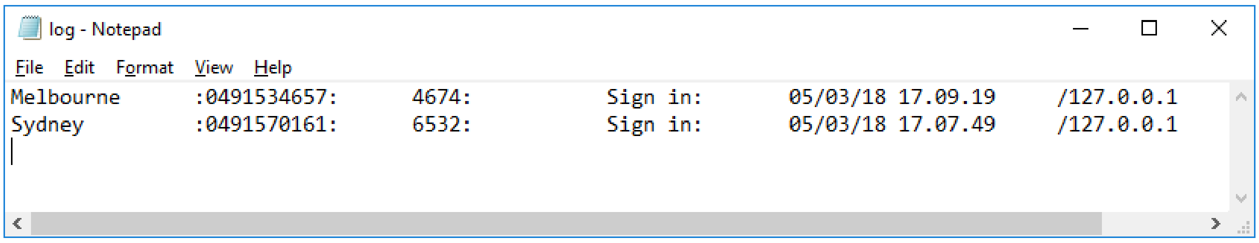


Mobile Number: 0491534657 Pin: 4674

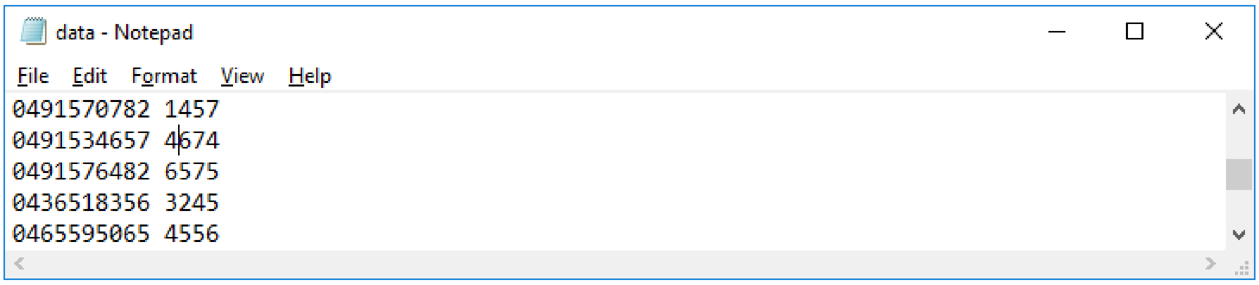
Message Status **Success



```
Sydney      :0491570161:    6532:          Sign in:        05/03/18 17.07.49    /127.0.0.1
Melbourne      :0491534657:    4674:          Sign in:        05/03/18 17.09.19    /127.0.0.1
```



```
Melbourne       :0491534657:      4674:          Sign in:       05/03/18 17.09.19      /127.0.0.1
Sydney          :0491570161:      6532:          Sign in:       05/03/18 17.07.49      /127.0.0.1
```



```
0491570782 1457
0491534657 4674
0491576482 6575
0436518356 3245
0465595065 4556
```

Note: since all the applications are running on the same machine locally, the local host IP address (127.0.0.1) is same for every client.

Part 2: Program use and test instruction

After the implementing the framework, prepare an end user instruction manual about how to use your software along with a brief description where this technology can be used and what happens to the messages if the server is dead or not running.

Submission

You need to provide the following files in your submission.

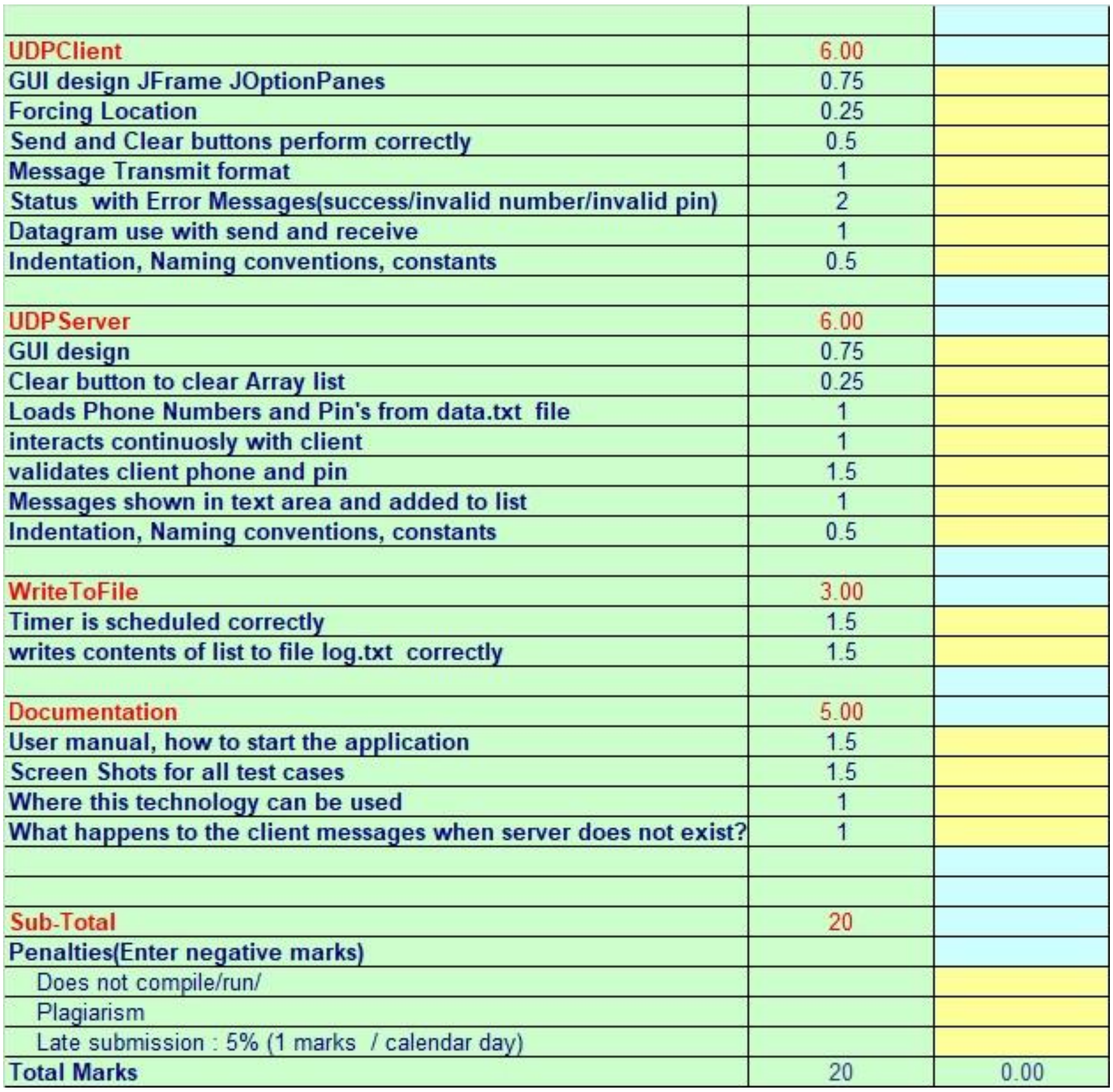
1. All java files mentioned in class diagram. In-line comments on the data structures and control statements used in the programs are required. These source code files must be able to be compiled by the standard JDK (Java Development Kit).

   (http://www.oracle.com/technetwork/java/index.html).

2. A Microsoft Word document to address the issues as specified in the Part 2 above.

All the required files must be compressed into a zip file for submission. You must submit your assignment via the unit web site. **Any hardcopy or email submission will not be accepted. After the marked assignments are returned, any late submissions will not be accepted.**

Marking Guide



UDPClient	6.00	
GUI design JFrame JOptionPanes	0.75	
Forcing Location	0.25	
Send and Clear buttons perform correctly	0.5	
Message Transmit format	1	
Status with Error Messages(success/invalid number/invalid pin)	2	
Datagram use with send and receive	1	
Indentation, Naming conventions, constants	0.5	
UDPServer	6.00	
GUI design	0.75	
Clear button to clear Array list	0.25	
Loads Phone Numbers and Pin's from data.txt file	1	
interacts continuosly with client	1	
validates client phone and pin	1.5	
Messages shown in text area and added to list	1	
Indentation, Naming conventions, constants	0.5	
WriteToFile	3.00	
Timer is scheduled correctly	1.5	
writes contents of list to file log.txt correctly	1.5	
Documentation	5.00	
User manual, how to start the application	1.5	
Screen Shots for all test cases	1.5	
Where this technology can be used	1	
What happens to the client messages when server does not exist?	1	
Sub-Total	20	
Penalties(Enter negative marks)		
Does not compile/run/		
Plagiarism		
Late submission : 5% (1 marks / calendar day)		
Total Marks	20	0.00