1 Objectives

(i.) Learn the fundamentals of network programming.

(ii.) Understand the end-to-end communication.

2 Tools

(i.) Java

(ii.) Socket Programming

3 Procedure

3.1 Server-side

i. Defining the socket to listen to a specific port.

```java
ServerSocket        SerSock         = new ServerSocket(1234);
```

ii. Specifying the port to listen for clients. Whenever a client is connected, a message is shown.

```java
Socket              Sock            = SerSock.accept();
System.out.println("Client␣Connected.␣Enter␣'Exit/exit'␣to␣
    disconnect.");
```

iii. Recieving the first message from client.

```java
  ClientInput = ReadBuffer.readLine();
```

iv. This is the termination condition. Whenever the client sends message to exit, the loop breaks and stops recieving messages.

```java
  while(!ClientInput.equalsIgnoreCase("Exit"))
```

v. Prints the message sent from client to the server's console and waits for the server to send a response.

```java
System.out.println("Client:␣" + ClientInput);
System.out.print("Server:␣");
```

vi. Takes an input from the server and sends it to the client through the **PrintWriter**. If the server responds with termination message, the loop breaks and stops sending/recieving messages.

```java
SendToClient = UserInput.nextLine();
WriteToClient.println(SendToClient);
if(SendToClient.equalsIgnoreCase("Exit"))
  break;
```

vii. Read reply from the client, continue the loop if termination message is not recieved.

```
ClientInput = ReadBuffer.readLine();
```

viii. The connection has been terminated. This part of code releases the resources occupied *i.e. the port*.

```
RecieveFromClient.close();
WriteToClient.close();
Sock.close();
```

3.2 Client-side

i. Specifying the client socket to search for server in a defined port. Once a server is found and connected to it, a confirmation message is shown to the user console.

```
Socket        CliSock       = new Socket("localhost", 1234);
System.out.println("Connected␣to␣Server.␣Enter␣'Exit/exit'␣to␣
    disconnect.");
```

ii. Taking an user input to make the first attempt to chat. The message is sent to the server through the **PrintWriter**.

```
System.out.print("Client:␣");
SendToServer = UserInput.nextLine();
WriteToServer.println(SendToServer);
```

iii. This checks for termination message. If not recieved, the loop will continue to execute.

```
while(!SendToServer.equalsIgnoreCase("Exit"))
```

iv. This reads message from the server and prints out on the client's console. If termination message is recieved, the loop will break and connection will be terminated. Otherwise, the console will prompt the client for another message and send it to the server.

```
ServerInput = ReadBuffer.readLine();
System.out.println("Server:␣" + ServerInput);
if(ServerInput.equalsIgnoreCase("Exit"))
  break;
```

v. The connection has been terminated. This part of code releases the resources occupied *i.e. the port*.

```
RecieveFromServer.close();
WriteToServer.close();
CliSock.close();
```

4 Appendix

4.1 Server code

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


public class server{
  public static void main(String[] args){
    try{
            ServerSocket SerSock = new ServerSocket(1234);
            Socket Sock = SerSock.accept();
            System.out.println("Client␣Connected.␣Enter␣'Exit/exit'␣to␣
                disconnect.");

            InputStreamReader RecieveFromClient = new InputStreamReader
                (Sock.getInputStream());
            BufferedReader ReadBuffer = new BufferedReader(
                RecieveFromClient);
            String ClientInput;
            String SendToClient;
            Scanner UserInput = new Scanner(System.in);
      PrintWriter WriteToClient = new PrintWriter(Sock.getOutputStream
          (), true);


            ClientInput = ReadBuffer.readLine();
            while(!ClientInput.equalsIgnoreCase("Exit")){
              System.out.println("Client:␣" + ClientInput);

              System.out.print("Server:␣");
              SendToClient = UserInput.nextLine();
              WriteToClient.println(SendToClient);
              if(SendToClient.equalsIgnoreCase("Exit"))
                break;
              ClientInput = ReadBuffer.readLine();
            }
            System.out.println("Connection␣Terminated");

            RecieveFromClient.close();
            WriteToClient.close();
            Sock.close();
        }
        catch(IOException e){
            System.out.println(e.toString());
        }
  }
}
```

4.2 Client Code

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

```


public class client{
  public static void main(String[] args){
    try{
      Socket CliSock = new Socket("localhost", 1234);
      System.out.println("Connected␣to␣Server.␣Enter␣'Exit/exit'␣to␣disconnect.");

      InputStreamReader RecieveFromServer = new InputStreamReader(CliSock.getInputStream());
      BufferedReader ReadBuffer = new BufferedReader(RecieveFromServer);
      String ServerInput;
      String SendToServer;
      Scanner UserInput = new Scanner(System.in);
      PrintWriter WriteToServer = new PrintWriter(CliSock.getOutputStream(), true);

      System.out.print("Client:␣");
      SendToServer = UserInput.nextLine();
      WriteToServer.println(SendToServer);
      while(!SendToServer.equalsIgnoreCase("Exit")){
        ServerInput = ReadBuffer.readLine();
        System.out.println("Server:␣" + ServerInput);
        if(ServerInput.equalsIgnoreCase("Exit"))
          break;

        System.out.print("Client:␣");
        SendToServer = UserInput.nextLine();
        WriteToServer.println(SendToServer);
      }
      System.out.println("Connection␣Terminated");

      RecieveFromServer.close();
      WriteToServer.close();
      CliSock.close();
    }
    catch(IOException e){
      System.out.println(e.toString());
    }
  }
}
```

5 Conclusion

As it was observed, the server-client connection was established succesfully. The two way communication went on untill any one side terminated the connection. The goal was to understand the end-to-end communication which has been achieved.