Network Programming

oleh : Yuliana Setiowati

Praktikum 1:

Buatlah program dibawah ini. Program di bawah ini untuk mengetahui nama komputer lokal.

```
import java.net.*;
public class getName{
       public static void main(String args[]) throws Exception{
          InetAddress host = null ;
          host = InetAddress.getLocalHost();
          System.out.println("Nama komputer Anda :" + host.getHostName());
       }
}
```

Praktikum 2:

Buatlah program dibawah ini dengan nama IPtoName.java, program ini bertujuan untuk mendapatkan nama komputer dari Alamat IP.

```
import java.net.*;

public class IPtoName{
   public static void main(String args[]){
      if (args.length == 0){
         System.out.println("Pemakaian : java IPtoName <IP Address>");
         System.exit(0);
      }
      String host = args[0] ;
      InetAddress address = null ;
      try{
        address = InetAddress.getByName(host);
      }catch(UnknownHostException e){
        System.out.println("invalid IP");
        System.exit(0);
      }
      System.out.println(address.getHostName());
   }
}
```

Jalankan dengan argumen IP komputer lokal dan komputer lain.

```
$ javac IPtoName.java
$ java IPtoName <IP Address Anda>
$ java IPtoName <IP Address teman Anda>
$ java IPtoName <IP Address sembarang>
```

Praktikum 3:

Buatlah program di bawah ini, masukkan misal www.detik.com maka akan ditampilkan Alamat IP dari www.detik.com. Masukkan host name : java.sun.com, berapakah Alamat IPnya?

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

public class IPFinder{
   public static void main(String args[]) throws IOException{
      String host;
      BufferedReader input = new BufferedReader(new InputStreamReader(System.in));
      System.out.println("Enter host name : ");
      host = input.readLine();
      try{
         InetAddress address = InetAddress.getByName(host);
         System.out.println("IP Address : " + address);
      }catch(UnknownHostException e){
         System.out.println("Could not find " + host);
      }
   }
}
```

Socket TCP

Java menyediakan obyek Socket dan ServerSocket untuk komunikasi soket TCP. ServerSocket digunakan pada sisi aplikasi server, sedangkan Socket digunakan baik pada sisi aplikasi server maupun client. Berikut adalah langkah-langkah membuat komunikasi socket di server:

1. Buatlah sebuah objek ServerSocket. Konstruktor ServerSocket memerlukan port number (1024-65535) sebagai argumen. Sebagai contoh :

   ```
   ServerSocket servSock = new ServerSocket(1234);
   ```

   Server akan menunggu koneksi dari client pada port 1234

2. Server dalam kondisi menunggu (listen). Operasi ini pada intinya menunggu permintaan koneksi dari sisi client.
   ```
   Socket link = servSock.accept();
   ```

3. Buat input dan output stream. Stream ini digunakan untuk berkomunikasi dengan client. Objek InputStreamReader digunakan untuk menerima respon dari client. Sedangkan PrintWriter untuk mengirimkan data ke client.

   ```
   BufferedReader in = new BufferedReader(new
   InputStreamReader(link.getInputStream()));
   PrintWriter out = new PrintWriter(link.getOutputStream(),true);
   ```

4. Saling berkirim dan menerima pesan. Gunakan method readLine() untuk menerima data dan method println() untuk mengirim data.

   ```
   out.println("Message " + numMessages + ":" + message);
   message = in.readLine();
   ```

5. Menutup socket
   ```
   Link.close();
   ```

Langkah-langkah komunikasi socket pada client:

1. Bangun koneksi ke server. Buatlah sebuah objek Socket, yang mempunyai konstruktor dengan dua argumen:
   - IP address server
   - Port Number

   Port number untuk server dan client haruslah sama.

   ```
   Socket link = new Socket(InetAddress.getLocalHost(),1234);
   ```

2. Buat input dan output stream. Stream ini digunakan untuk berkomunikasi dengan client. Objek InputStreamReader digunakan untuk menerima respon dari client. Sedangkan PrintWriter untuk mengirimkan data ke client.

   ```
   BufferedReader in = new BufferedReader(new
   InputStreamReader(link.getInputStream()));
   PrintWriter out = new PrintWriter(link.getOutputStream(),true);
   ```

3. Saling berkirim dan menerima pesan. Gunakan method readLine() untuk menerima data dan method println() untuk mengirim data.

   ```
   out.println("Message " + numMessages + ":" + message);
   message = in.readLine();
   ```

4. Menutup socket

   ```
   Link.close();
   ```

Praktikum 4:

Buatlah program Server dan Client. Program server hanya bisa terkoneksi dengan 1 client.

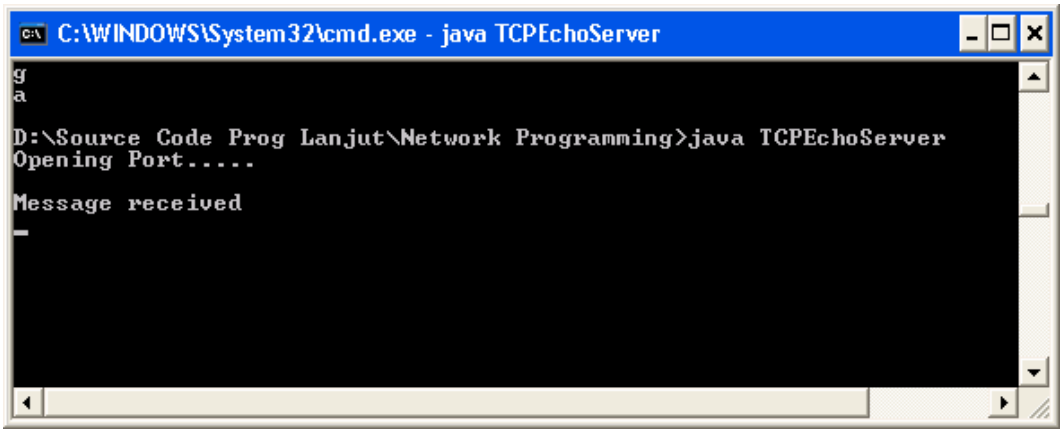
```
import java.io.*;
import java.net.*;

public class TCPEchoServer{
   private static ServerSocket  servSock;
   private static final int PORT = 1234 ;
   public static void main(String args[]){
      System.out.println("Opening Port.....\n");
      try{
         servSock = new ServerSocket(PORT);
      }catch(IOException e){
         System.out.println("Unable to attach to port");
         System.exit(1);
      }

      do{
         run();
      }while(true);
   }

   private static void run(){
      Socket link = null ;
      try{
```

```
          link = servSock.accept();
          BufferedReader in = new BufferedReader(new
InputStreamReader(link.getInputStream()));
          PrintWriter out = new PrintWriter(link.getOutputStream(),true);

          int numMessages=0;
          String message=in.readLine();

          while(!message.equals("close")){
            System.out.println("Message received");
            numMessages++;
            out.println("Message " + numMessages + ":" + message);
            message = in.readLine();
          }

          out.println(numMessages + " message received.");
       }catch(IOException e){
          e.printStackTrace();
       }
       finally{
          try{
             System.out.println("*********Closing Connection****");
             link.close();
          }
          catch(IOException e){
             System.out.println("Unable to disconnect");
             System.exit(1);
          }
       }
   }
}


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

public class TCPEchoClient{
   private static String strHost ;
   private static InetAddress host ;
   private static final int PORT = 1234;
   public static void main(String args[]){
      try{
         host = InetAddress.getLocalHost();
         //strHost = args[0] ;
         //host = InetAddress.getByName(strHost);
      }catch(UnknownHostException e){
         System.out.println("Host ID Not Found");
         System.exit(1);
      }
      run();

   }

   private static void run(){
      Socket link = null ;
      try{
        link = new Socket(host, PORT);

        BufferedReader in = new BufferedReader(new
InputStreamReader(link.getInputStream()));
        PrintWriter out = new PrintWriter(link.getOutputStream(),true);
```

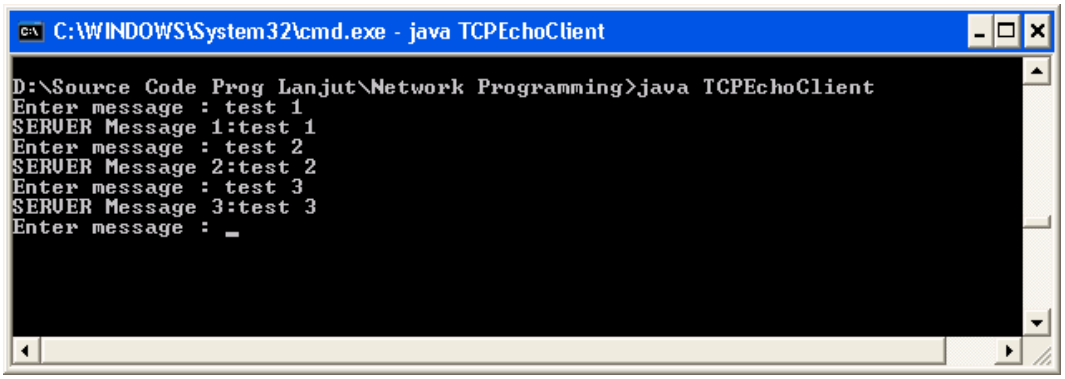
```
        BufferedReader userEntry = new BufferedReader(new
InputStreamReader(System.in));

        String message, response ;

        do{
           System.out.print("Enter message : ");
           message = userEntry.readLine();
           out.println(message);
           response = in.readLine();
           System.out.println("SERVER " + response);
        }while(!message.equals("close"));
      }catch(IOException e){
        e.printStackTrace();
      }
      finally{
        try{
           System.out.println("closing connection");
           link.close();
        }
        catch(IOException e){
           System.out.println("Unable to disconnect!");
           System.exit(1);
        }
      }
   }
}
```

Output program :

Jika kita jalankan TCPEchoServer pertama kali maka akan tampak seperti berikut:

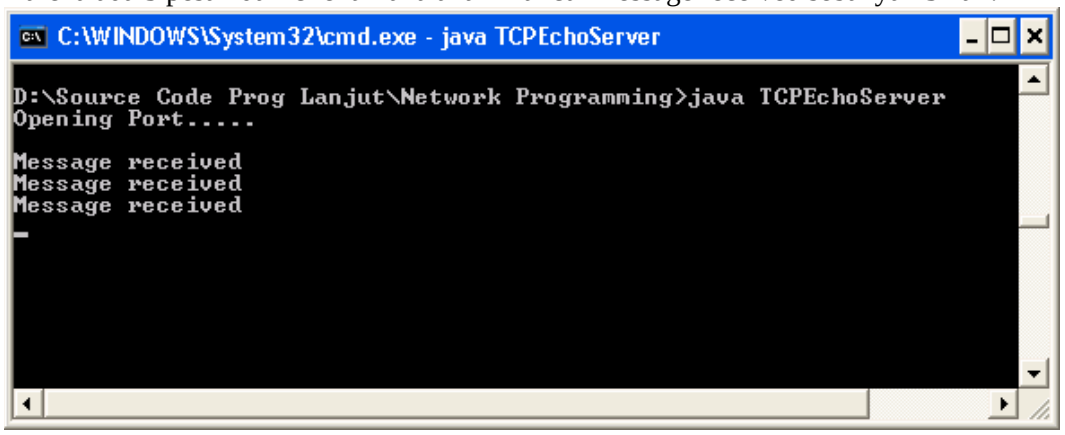


Client
Masukkan pada client string seperti berikut:



Server

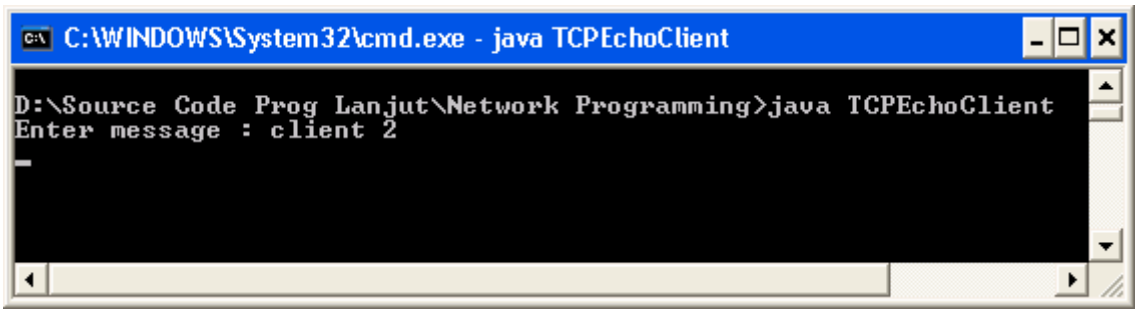
Karena ada 3 pesan dari client maka akan muncul message received sebanyak 3 kali.

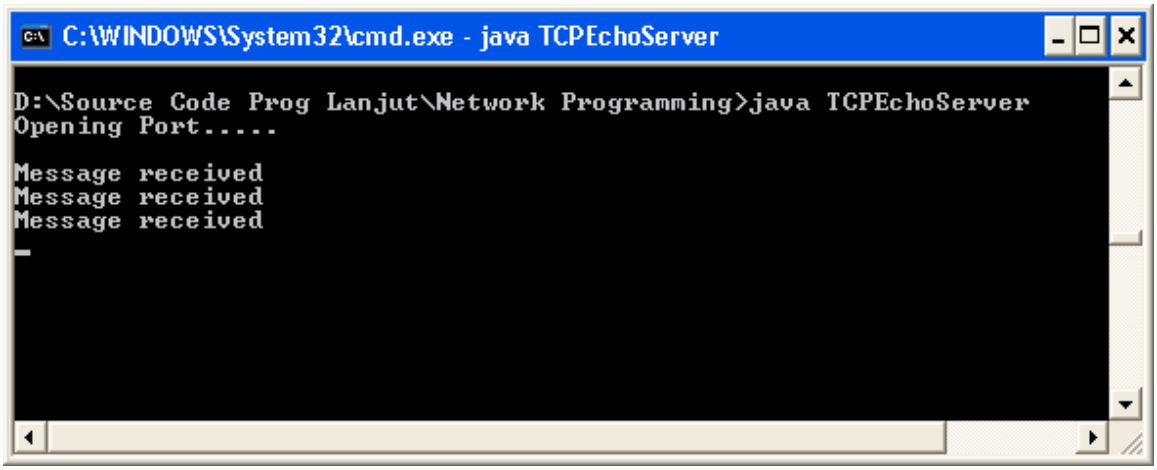


Tambahkan sebuah client lagi dengan cara menjalankan

```
java TCPEchoClient
```

Apa yang terjadi ? ternyata program tidak berjalan lagi. Untuk menghentikan tekan ctrl C. Begitu juga dengan server, server tidak menerima message dari client. Program yang sudah kita kerjakan ini hanya mampu menangani 1 client saja, belum mampu menangani multi client.





Praktikum 5 :
Program di bawah ini server dapat menangani lebih dari 1 client.

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

public class MultiEchoServer{
   private static ServerSocket  servSock;
   private static final int PORT = 1234 ;
   public static void main(String args[]) throws IOException{
      System.out.println("Opening Port.....\n");
      try{
         servSock = new ServerSocket(PORT);
      }catch(IOException e){
         System.out.println("Unable to attach to port");
         System.exit(1);
      }

      do{
         Socket client = servSock.accept();
         ClientHandler handler = new ClientHandler(client);
         handler.start();
      }while(true);
   }
}

class ClientHandler extends Thread{
   private Socket client ;
   private BufferedReader in ;
   private PrintWriter out ;

   public ClientHandler(Socket socket){
      client = socket ;

      try{
         in = new BufferedReader(new
InputStreamReader(client.getInputStream()));
         out = new PrintWriter(client.getOutputStream(),true);
      }catch(IOException e){
         e.printStackTrace();
      }
   }
```

```
   public void run(){
      try{
         String received ;
         do{
            received = in.readLine();
            System.out.println(received);
            out.println("ECHO : " + received);
         }while(!received.equals("QUIT"));
      }catch(IOException e){
         e.printStackTrace();
      }
      finally{
         try{
            if (client != null){
               System.out.println("Closing down connection");
               client.close();
            }
         }catch(IOException e){
            e.printStackTrace();
         }
      }
   }
}

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

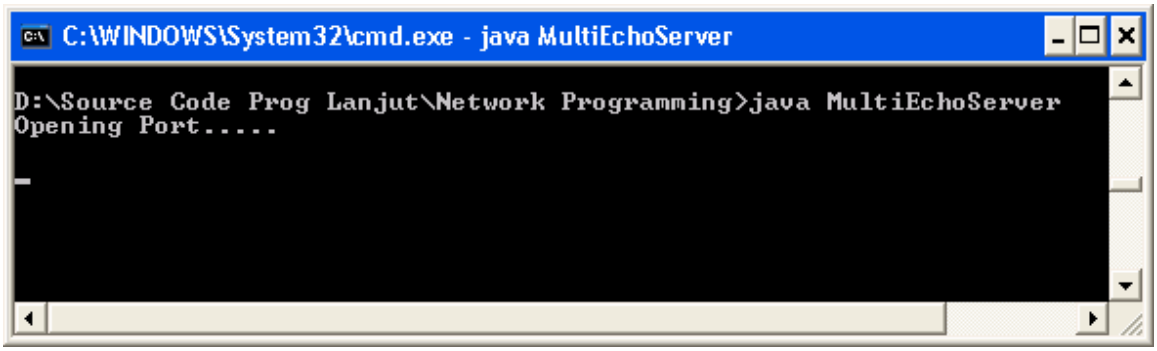
`public class MultiEchoClient{`

```
   private static InetAddress host ;
   private static final int PORT = 1234;
   private static Socket link ;
   private static BufferedReader in ;
   private static PrintWriter out ;
   private static BufferedReader keyboard ;

   public static void main(String args[]){
      try{
         host = InetAddress.getLocalHost();
         link = new Socket(host,PORT);

         in = new BufferedReader(new InputStreamReader(link.getInputStream()));
         out = new PrintWriter(link.getOutputStream(),true);

         keyboard = new BufferedReader(new InputStreamReader(System.in));

         String message, response;
         do{
             System.out.print("Enter message(QUIT to exit)");
             message = keyboard.readLine();
             out.println(message);
             response = in.readLine();
             System.out.println(response);
         }while(!message.equals("QUIT"));

      }catch(UnknownHostException e){
          System.out.println("Host ID not found!");
      }
      catch(IOException e){
         e.printStackTrace();
      }
      finally{
         try{
```

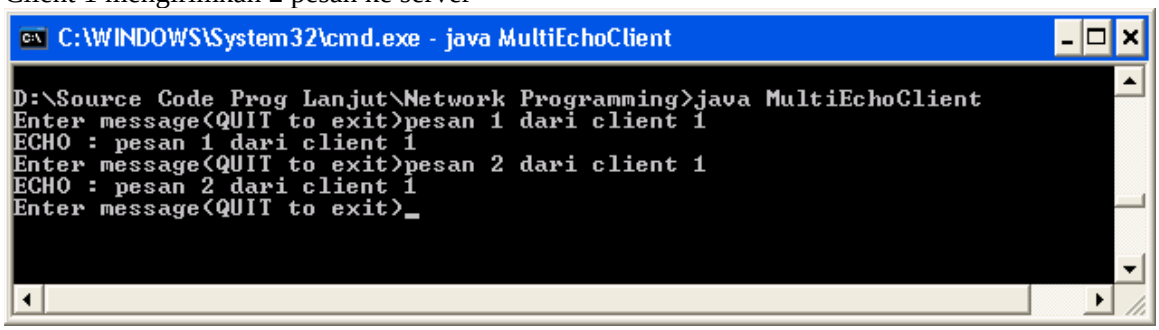
```
            if (link != null){
                System.out.println("Closing down connection");
                link.close();
            }
        }
        catch(IOException e){
            e.printStackTrace();
        }
    }
  }
}
```

Output program :

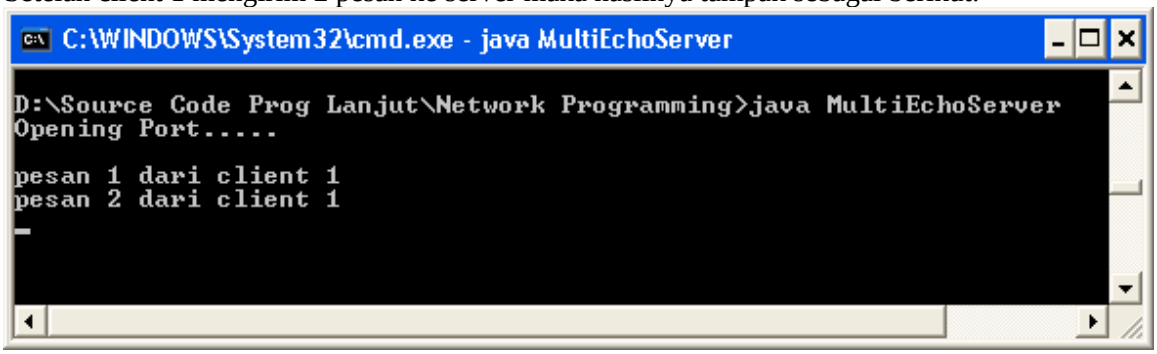
Server pertama kali dijalankan.



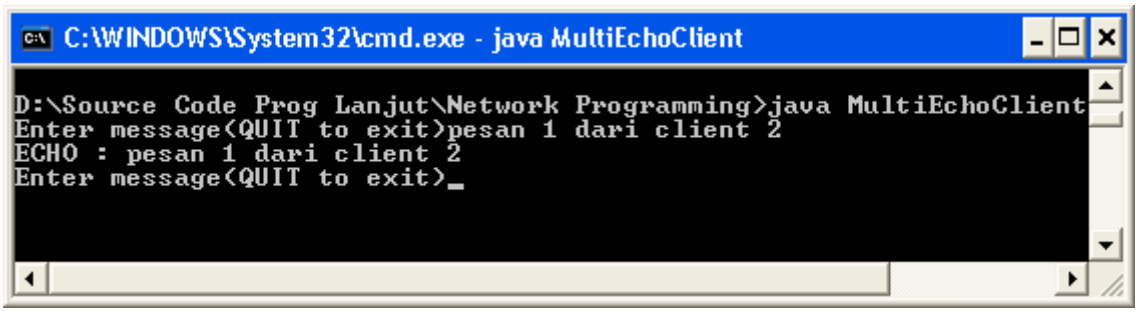
Client 1 mengirimkan 2 pesan ke server



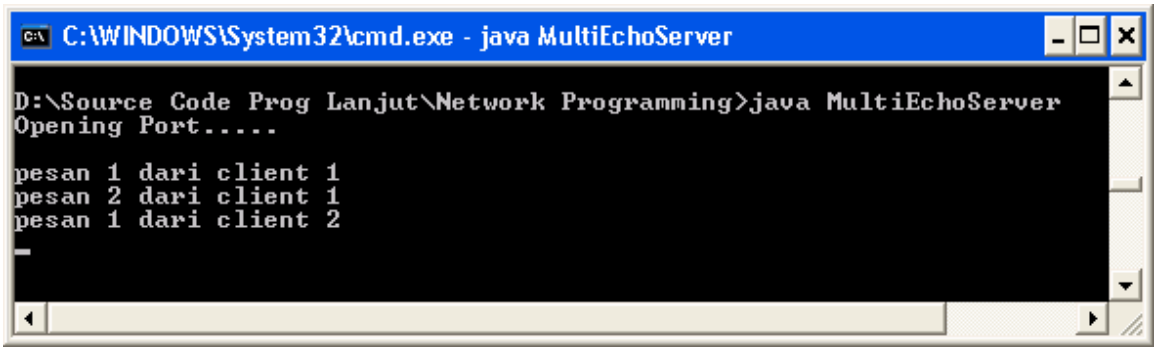
Setelah client 1 mengirim 2 pesan ke server maka hasilnya tampak sebagai berikut:



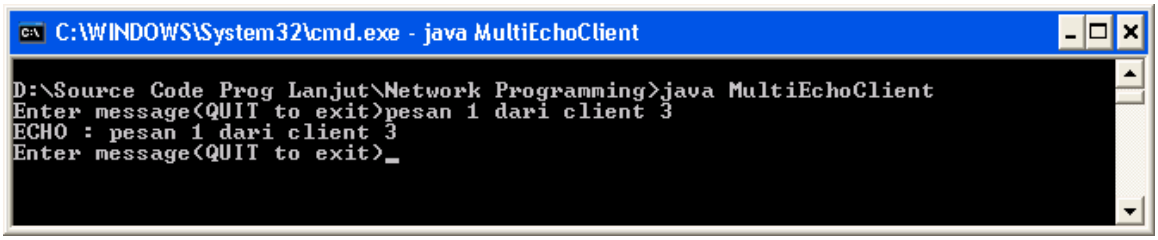
Client 2 mengirimkan 1 pesan



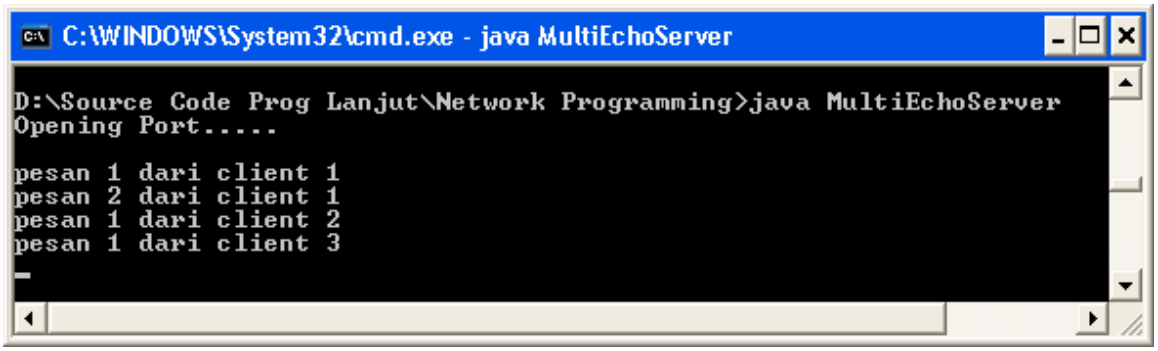
Setelah client 2 mengirim 1 pesan ke server maka hasilnya tampak sebagai berikut:



Client 3 mengirimkan 1 pesan



Setelah client 3 mengirim 1 pesan ke server maka hasilnya tampak sebagai berikut:



Socket Datagram (UDP)

Untuk protokol UDP, perbedaannya adalah socket di sisi server sama dengan socket di sisi client dan tidak ada operasi listen pada sisi server. Kemudian saat paket data dikirimkan alamat socket penerima harus disertakan sebagai argumen.

Langkah-langkah untuk membuat server:

1. Buatlah sebuah objek DatagramSocket. Konstruktor DatagramSocket mempunyai satu argument yaitu no port.

```
DatagramSocket dgramSocket = new DatagramSocket(1234);
```

2. Buat buffer untuk datagram yang masuk.

```
byte[] buffer = new byte[256];
```

3. Buatlah sebuah objek DatagramPacket untuk datagram yang masuk. Konstruktor dari objek ini memerlukan dua argumen yaitu array byte dan besar dari array ini

```
DatagramPacket inPacket = new DatagramPacket(buffer,buffer.length);
```

4. Menerima datagram yang masuk

```
dgramSocket.receive(inPacket);
```

5. Menerima alamat pengirim dan port yang masuk

```
InetAddress clientAddress = inPacket.getAddress();
int clientPort = inPacket.getPort();
```

6. Mengambil data dari buffer. Untuk memudahkan penanganan, maka data di ambil sebagai string. Konstruktor String yang digunakan mempunyai 3 argumen yaitu
   - Array dengan tipe byte
   - Posisi awal array yang akan diambil(posisi 0)
   - Jumlah byte yang akan diambil (sama dengan besar buffer)

```
String   messageIn = new
String(inPacket.getData(),0,inPacket.getLength());
```

7. Buatlah datagram respon. Buatlah sebuah objek DatagramPacket, dengan konstruktor yang memerlukan 3 argument.
   - Array dengan tipe byte yang berisi message response
   - Besar response
   - Alamat client
   - No port

```
DatagramPacket outPacket = new DatagramPacket(messageOut.getBytes(),
messageOut.length(), clientAddress, clientPort);
```

8. Kirim datagram response

```
dgramSocket.send(outPacket);
```

9. Close DatagramSocket.

```
dgramSocket.close();
```

Langkah-langkah untuk membuat client:

1. Buatlah sebuah objek DatagramSocket. Pembuatan objek DatagramSocket hampir sama seperti pada program server tapi untuk client tidak menggunakan no port

```
DatagramSocket dgramSocket = new DatagramSocket();
```

2. Buatlah datagram yang akan keluar. Lihat langkah 7 pada server.

```
DatagramPacket outPacket = new DatagramPacket(messageOut.getBytes(),
messageOut.length(), clientAddress, clientPort);
```

3. Kirim message datagram.
```
dgramSocket.send(outPacket);
```

4. Buatlah sebuah buffer untuk datagram yang masuk.
```
byte[] buffer = new byte[256];
```

5. Buatlah sebuah objek DatagramPacket untuk datagram yang masuk. Konstruktor dari objek ini memerlukan dua argumen yaitu array byte dan besar dari array ini
```
DatagramPacket inPacket = new DatagramPacket(buffer,buffer.length);
```

6. Menerima datagram yang masuk
```
dgramSocket.receive(inPacket);
```

7. Mengambil data dari buffer. Untuk memudahkan penanganan, maka data di ambil sebagai string. Konstruktor String yang digunakan mempunyai 3 argumen yaitu
   - Array dengan tipe byte
   - Posisi awal array yang akan diambil(posisi 0)
   - Jumlah byte yang akan diambil (sama dengan besar buffer)

```
String   messageIn = new
String(inPacket.getData(),0,inPacket.getLength());
```

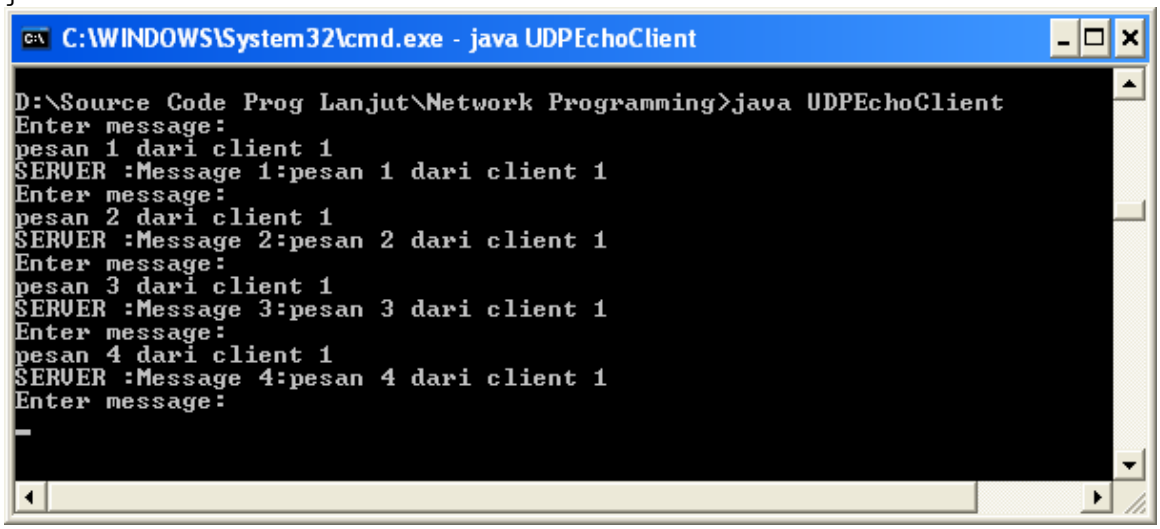
8. Close DatagramSocket.
```
dgramSocket.close();
```

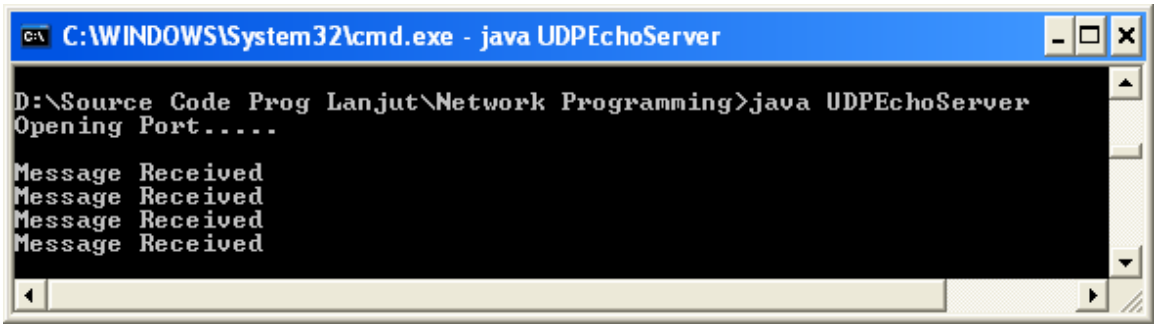
Praktikum 6 :

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

public class UDPEchoServer{
   private static final int PORT = 1234 ;
   private static DatagramSocket dgramSocket ;
   private static DatagramPacket inPacket, outPacket ;
   private static byte[] buffer  ;

   public static void main(String args[]){
      System.out.println("Opening Port.....\n");
      try{
         dgramSocket = new DatagramSocket(PORT);
      }catch(SocketException e){
         System.out.println("Unable to attachto port !");
         System.exit(1);
      }
      run();
   }

   private static void run(){
       try{
```

```
            String messageIn, messageOut;
            int numMessages=0;
            do{
               buffer = new byte[256] ;
               inPacket = new DatagramPacket(buffer,buffer.length);
               dgramSocket.receive(inPacket);
               InetAddress clientAddress = inPacket.getAddress();
               int clientPort = inPacket.getPort();

               messageIn = new
String(inPacket.getData(),0,inPacket.getLength());
               System.out.println("Message Received");

               numMessages++;
               messageOut = ("Message " + numMessages + ":" + messageIn);
               outPacket = new DatagramPacket(messageOut.getBytes(),
messageOut.length(), clientAddress, clientPort);
               dgramSocket.send(outPacket);

            }while(true);


        }
        catch(IOException e){
            e.printStackTrace();
        }
        finally
        {
            System.out.println("\nClosing Connection...");
            dgramSocket.close();
        }

    }

}

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

public class UDPEchoClient{
    private static InetAddress host ;
    private static final int PORT = 1234 ;
    private static DatagramSocket dgramSocket ;
    private static DatagramPacket inPacket, outPacket ;
    private static byte[] buffer  ;

    public static void main(String args[]){
        try{
            host = InetAddress.getLocalHost();
        }
        catch(UnknownHostException e){
            System.out.println("Host ID Not Found");
            System.exit(1);

        }
        run();
    }

    private static void run(){
        try{
           dgramSocket = new DatagramSocket();
           BufferedReader userEntry = new BufferedReader(new
InputStreamReader(System.in));
```

```
            String message="", response="";
            do{
                System.out.println("Enter message:");
                message = userEntry.readLine();
                if (!message.equals("CLOSE")){
                    outPacket = new
DatagramPacket(message.getBytes(),message.length(),host,PORT);
                    dgramSocket.send(outPacket);
                    buffer = new byte[256];
                    inPacket = new DatagramPacket(buffer, buffer.length);
                    dgramSocket.receive(inPacket);
                    response = new
String(inPacket.getData(),0,inPacket.getLength());
                    System.out.println("SERVER :" + response);
                }
            } while(!message.equals("CLOSE"));
        }
        catch(IOException e){
             e.printStackTrace();
        }finally{
            System.out.println("Closing Connection");
            dgramSocket.close();
        }
    }
}
```





MODUL PRAKTIKUM

Praktikum 1:

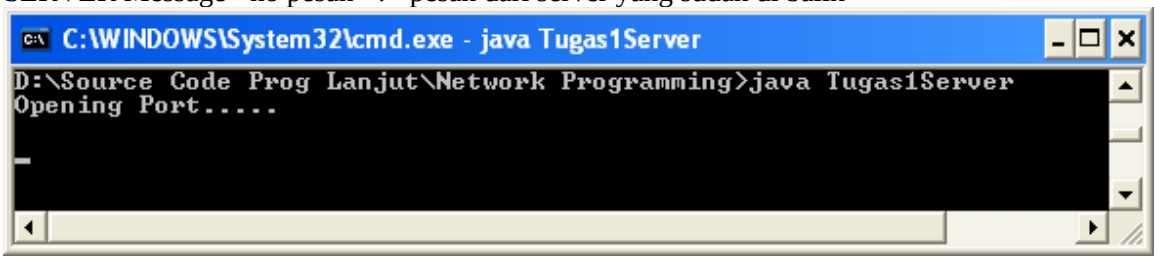
Buatlah program server-client dengan menggunakan TCP Socket. Pertama kali aktifkan server. Misal pada client ketik ”salam” selanjutnya string ini akan dikirim ke server, pada server string akan dibalik sehingga menjadi malas. String yang sudah dibalik tadi selanjutnya dikirim kembali ke client.

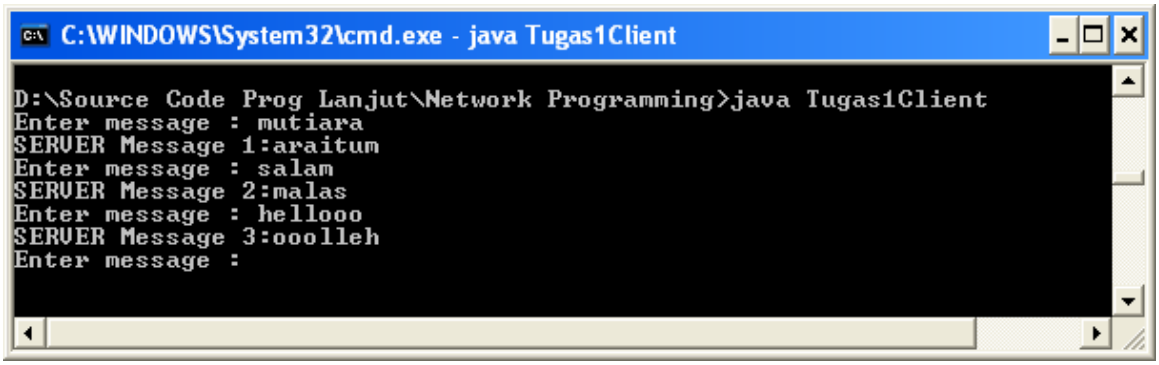
Sedangkan pada server tampilkan pesan

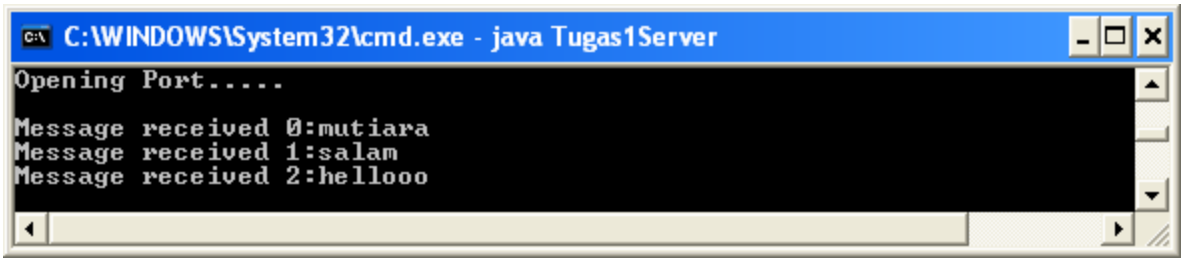
Message received <no pesan> : <pesan yang dikirim>

Pada client akan menampilkan pesan :

SERVER Message <no pesan> : <pesan dari server yang sudah di balik>







Praktikum 2

Kembangkan dari program TCPEchoServer dan TCPEchoCLient, buatlah program Client-Server menggunakan Socket, terdapat 1 server dan 1 client. Skenario program yang akan kalian kerjakan adalah sebagai berikut:

- Server hanya mengenali dua user yaitu ”Andi” dan ”Budi”
- Setiap user mempunyai message box pada server yang dapat menerima maksimum 10 pesan.

- Setiap user dapat mengirim atau membaca pesan.
- Setiap ada pesan yang masuk maka jumlah pesan akan ditambah(Tapi jika message box sudah penuh maka pesan akan diabaikan). Jika pesan sudah dibaca maka jumlah pesan akan diset 0.
- Pada saat user mengirim pesan maka ketikkan :

```
Andi send
Enter message : <pesan>
```

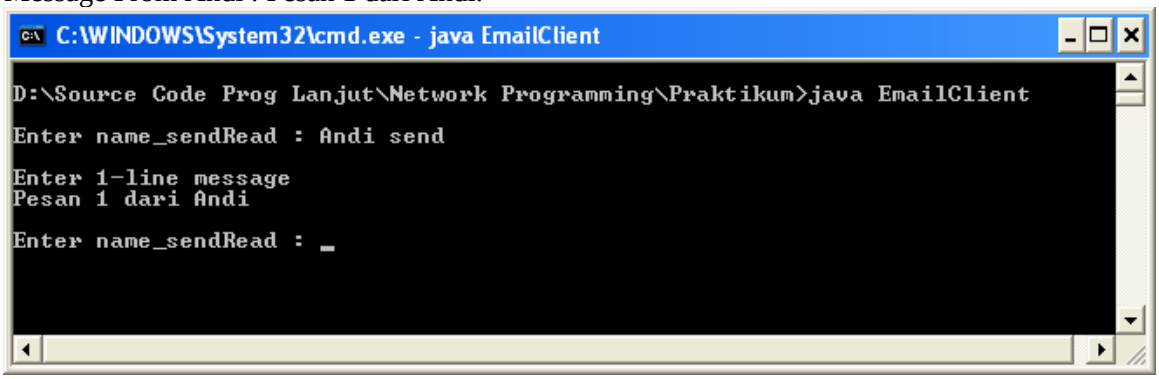
- Pesan yang dikirim oleh Andi akan masuk ke message box Budi atau sebaliknya, pesan yang dikirim oleh Budi akan masuk ke message box Andi.
- Pada saat user membaca pesan maka user mengetikkan :

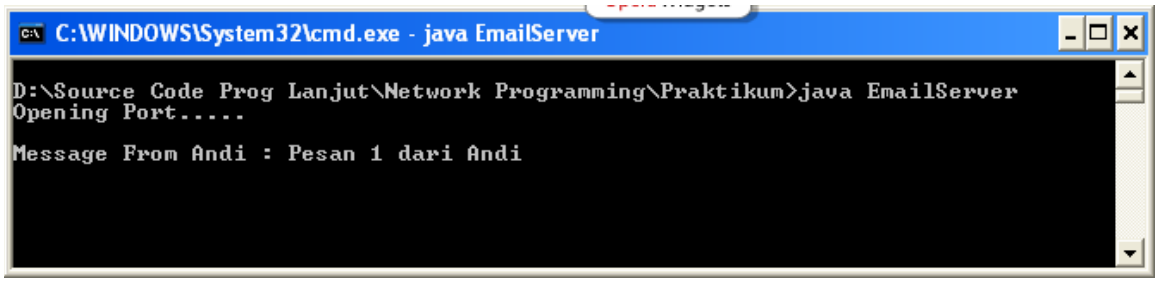
```
Andi read
```

  Maka akan tampak pesan-pesan yang ada pada message box.
- Jika telah selesai ketik close

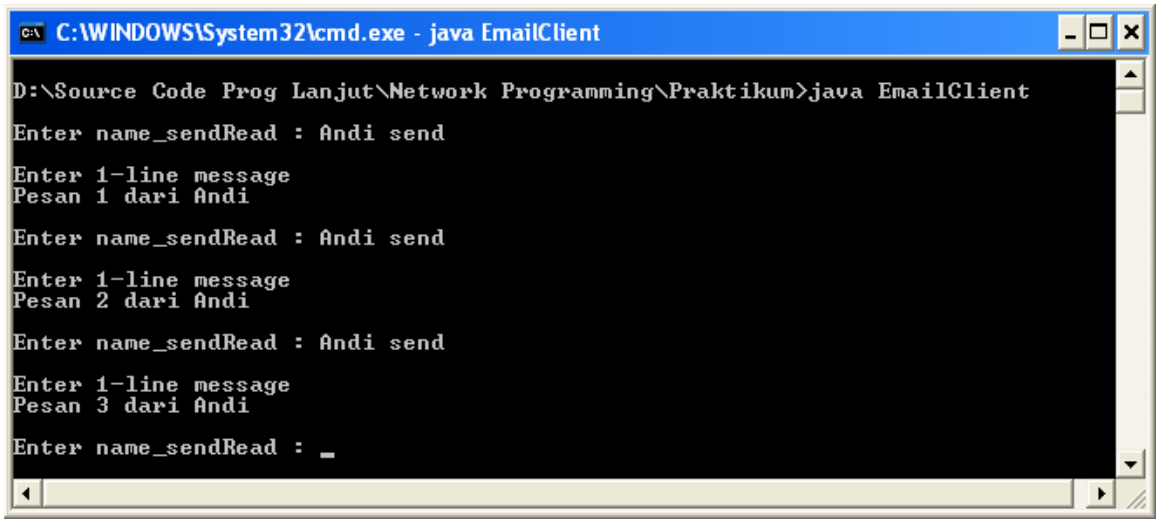
Output program :

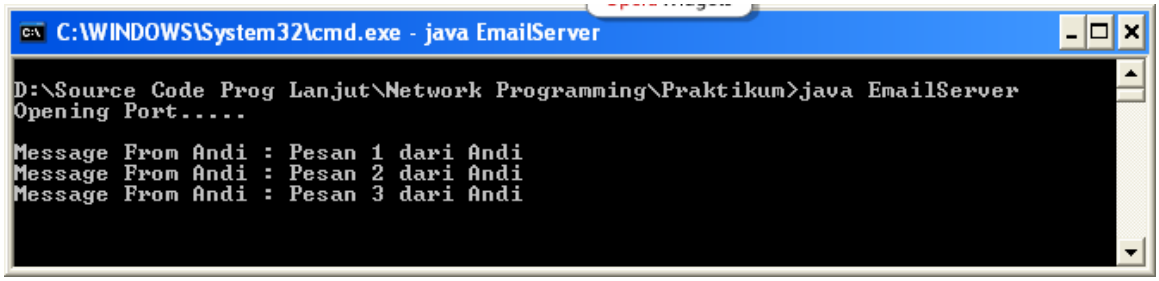
Pertama-tama pada client ketikkan Andi send, kemudian masukkan pesan yang akan dikirimkan ke Andi. Jika sudah, maka pada server, tampilkan pesan :
Message From Andi : Pesan 1 dari Andi.



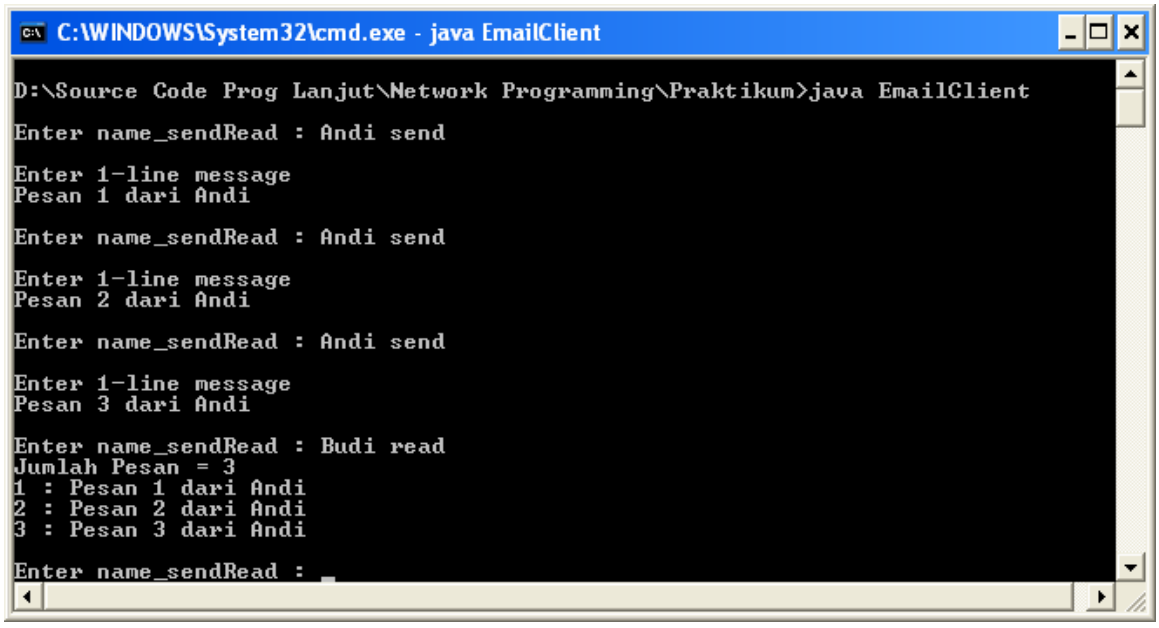


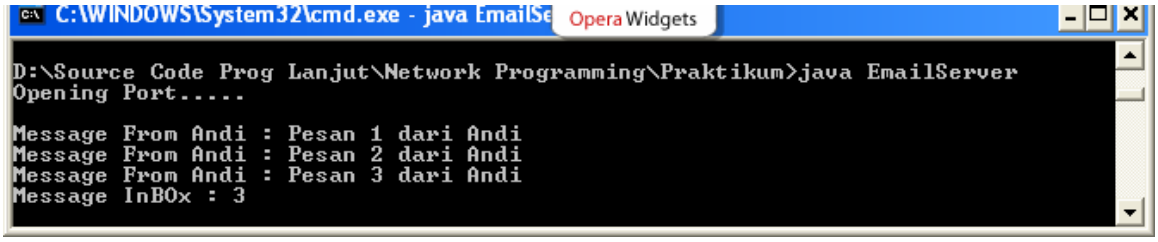
Sebagai user Andi lakukan pengiriman pesan sebanyak 3 kali dengan cara yang sama. Hasil akan tampak seperti di bawah ini:



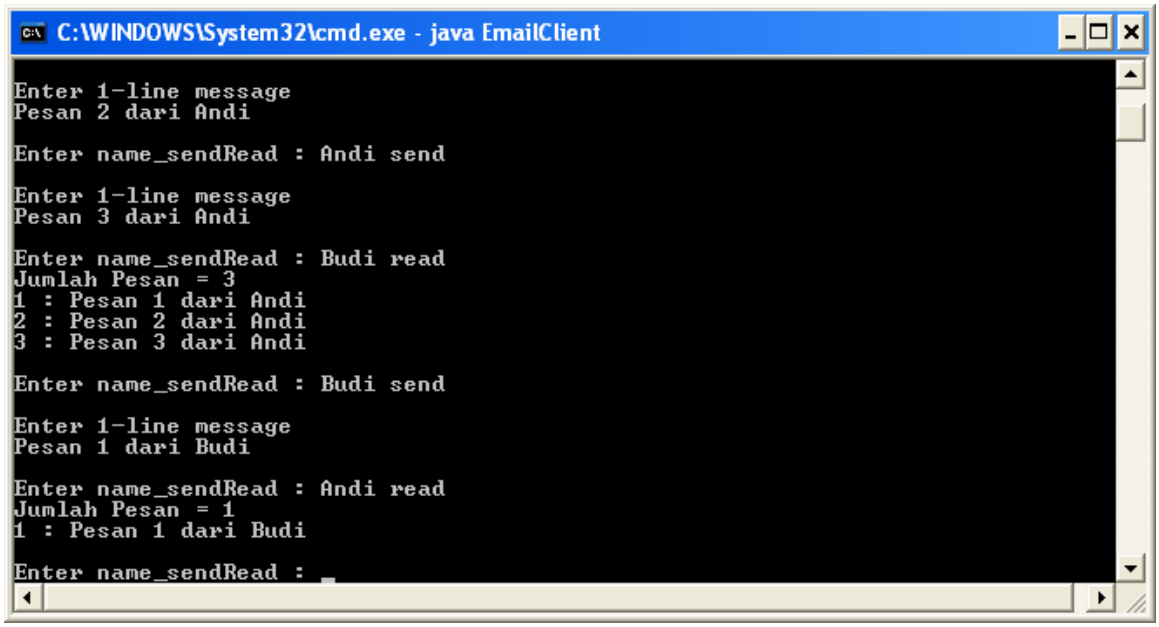


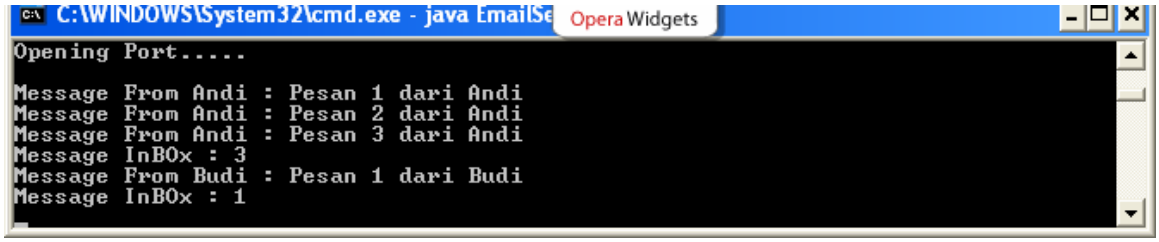
Sebagai user Budi lakukan perintah read untuk membaca message box, karena Andi telah mengirim pesan sebanyak 3 kali, maka message box Budi terdapat 3 pesan. Output seperti gambar di bawah ini:





Selanjutnya user Budi melakukan perintah send maka ketikkan pesan. Lakukanlah sebanyak 1 kali, selanjutnya user Andi melakukan read, maka pada message Box Andi terdapat 1 pesan.





Program

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


public class EmailServer{
   private static ServerSocket  servSock;
   private static final int PORT = 50000 ;
   private static final int MAX = 10 ;
   private static String[] mailbox1 = new String[MAX] ;
   private static String[] mailbox2 = new String[MAX] ;
   private static int messageInBox1 = 0 ;
   private static int messageInBox2 = 0 ;
```

```
   public static void main(String args[]){
      System.out.println("Opening Port.....\n");
      try{
         servSock = new ServerSocket(PORT);
      }catch(IOException e){
         System.out.println("Unable to attach to port");
         System.exit(1);
      }

      do{
         run();
      }while(true);
   }

public static void doSend(String mailbox[], int messageInBox, BufferedReader
in) throws IOException{
         //menerima pesan dari client yang akan disimpan di mailbox
        //tampilkan pesan tersebut di Server

        if (messageInBox == MAX)
           System.out.println("INBOX FULL");
        else
           mailbox[messageInBox] = message ;

     }

public static void doRead(String mailbox[], int messageInBox, PrintWriter out)
throws IOException{
       //Tampilkan di server berapa message yang ada di mailbox
       //Jumlah message yang ada di mailbox kirim ke client
       //lakukan perulangan sebanyak jumlah message, selanjutnya tiap message
        //kirim ke client

 }
   private static void run(){
      Socket link = null ;
      try{
          link = servSock.accept();
          BufferedReader in = new BufferedReader(new
InputStreamReader(link.getInputStream()));
          PrintWriter out = new PrintWriter(link.getOutputStream(),true);
          String message="", name="", sendRead="" ;

          do{

          //menerima pesan dari client berupa name dan sendRead (Andi read)
          //lakukan pemecahan String dg menggunakan StringTokenizer simpan
          //name:Andi sendRead = read
            if (name.equals("Andi")) {
               if (sendRead.equals("send")){
                   System.out.print("Message From Andi : ");
                   doSend(mailbox2, messageInBox2, in);
                    if (messageInBox2<MAX)
                        messageInBox2++;
               }
               else if (sendRead.equals("read")){
                     doRead(mailbox1, messageInBox1, out);
                     messageInBox1 = 0 ;
               }
            }
            else if (name.equals("Budi"))
            {
```

```
                if (sendRead.equals("send")){
                    System.out.print("Message From Budi : ");
                    doSend(mailbox1, messageInBox1, in);
                     if (messageInBox1<MAX)
                         messageInBox1++;
                }
                else if (sendRead.equals("read")){
                       doRead(mailbox2, messageInBox2, out);
                       messageInBox2 = 0 ;
                }
             }
            }
         }while(!message.equals("close"));


      }catch(IOException e){
         e.printStackTrace();
      }
      finally{
         try{
            System.out.println("*********Closing Connection****");
            link.close();
         }
         catch(IOException e){
            System.out.println("Unable to disconnect");
            System.exit(1);
         }
      }
   }
}
```

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

public class EmailClient{
   private static String strHost ;
   private static InetAddress host;
   private static final int PORT = 50000;
   private static BufferedReader userEntry ;
   private static BufferedReader in ;
   private static PrintWriter out;

   public static void main(String args[]){
      userEntry = new BufferedReader(new InputStreamReader(System.in));
      try{
         host = InetAddress.getLocalHost();
         //strHost = args[0] ;
         //host = InetAddress.getByName(strHost);
      }catch(UnknownHostException e){
         System.out.println("Host ID Not Found");
         System.exit(1);
      }
      run();

   }
   private static void doSend() throws IOException{
      //masukkan message
      //kirim ke server
   }

   private static void doRead() throws IOException{
```

```
      //menerima dari server jumlah pesan yang ada di mailbox dalam bentuk
string
      //ubah menjadi int
      //lakukan sebanyak jumlah pesan, untuk menerima dari pesan dari server

   }

   private static void run(){
      Socket link = null ;
      try{
        link = new Socket(host, PORT);

        in = new BufferedReader(new InputStreamReader(link.getInputStream()));
        out = new PrintWriter(link.getOutputStream(),true);
        String message;
        do{
           String name="", sendRead="" ;
           do{
           //masukkan name_sendRead misal Andi send
           //kirim ke server

           if (!message.equals("close")){
           //lakukan pemecahan name=”Andi” sendRead = “send”
           }
           }while(name.equals("Andi") && name.equals("Budi"));

           if (!message.equals("close")){
            if (name.equals("Andi")) {
               if (sendRead.equals("send")){
                   doSend();
               }
               else if (sendRead.equals("read")){
                   doRead();
               }
            }
            else if (name.equals("Budi"))
            {
               if (sendRead.equals("send")){
                   doSend();
               }
               else if (sendRead.equals("read")){
                    doRead();
               }
            }
           }
        }while(!message.equals("close"));
      }catch(IOException e){
        e.printStackTrace();
      }
      finally{
        try{
           System.out.println("closing connection");
           link.close();
        }
        catch(IOException e){
           System.out.println("Unable to disconnect!");
           System.exit(1);
        }
      }
   }
}
```

Praktikum 3

Pada praktikum 3, soal sama seperti praktikum 2, tetapi server dapat menangani lebih dari 1 client (Server dapat menangani multi client). Program dikembangkan dari MultiEchoServer.java dan MultiEchoClient.java. Skenario program adalah sebagai berikut:

- Server dapat mengenali sembarang user. Setiap user baru yang masuk akan ditambahkan dalam sebuah object Vector dengan nama user.

Andi	Budi	Candra	Dian	Edi

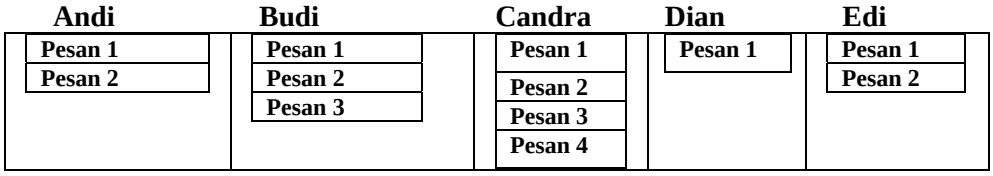
```
Vector user = new Vector();
```

- Setiap user mempunyai message box pada server. Untuk menampung semua mailbox user buatlah sebuah objek dari class Vector dengan nama mailBoxUser.

```
Vector mailBoxUser = new Vector();
```

Karena setiap user mempunyai mailbox sendiri, maka buatlah mailbox untuk setiap user yang dibuat dari objek vector. Kemudian tambahkan pada mailBoxUser. No urut mailbox sama dengan no urut user pada objek user.

```
Vector mailBoxAndi = new Vector();
mailBoxUser.add(mailBoxAndi);
```



Dari contoh diatas mailbox Andi mempunyai 2 message, mailbox Budi mempunyai 3 message, mailbox Candra mempunyai 3 message, mailbox Dian mempunyai 1 message sedangkan mailbox Edi mempunyai 2 message.

Untuk memasukkan pesan ke mailbox Andi adalah sebagai berikut:

```
Vector mailBoxAndi = new Vector();
mailBoxAndi.add("pesan 1");
mailBoxAndi.add("pesan 2");
mailBoxUser.add(mailBoxAndi);
```

- Setiap user dapat mengirim atau membaca pesan. Untuk mengirim ketik perintah "send", sedangkan untuk membaca ketik perintah "read".
- Setiap ada pesan yang masuk maka jumlah pesan akan ditambah. Jika pesan sudah dibaca maka jumlah pesan akan diset 0.
- Pertama kali pada saat program

  Program di client masukkan nama user

```
Enter name : Andi
```

  Program di server akan menampilkan

```
Andi connect.........
```

- Pada saat user mengirim pesan maka ketikkan :

```
Enter send or read : send
Send to : Budi
Enter 1-line message : pesan 1
```

Andi akan mengirim pesan ke Budi, pesan ini akan masuk ke mailbox Budi. Tambahkan pada pesan dengan nama user yang mengirim pesan tersebut. Sehingga pesan tersebut akan menjadi:

```
pesan 1 from Andi
```

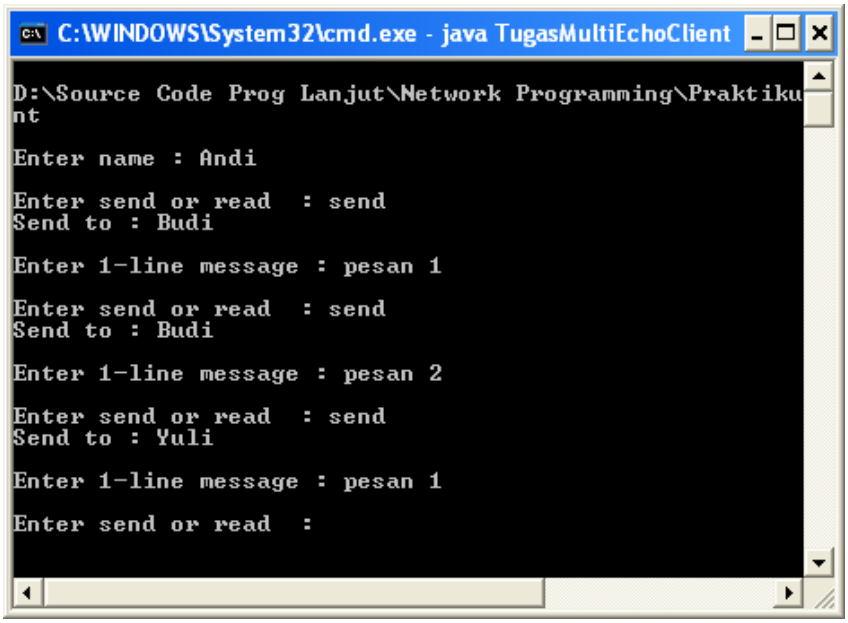
- Pada saat user membaca pesan maka user mengetikkan :

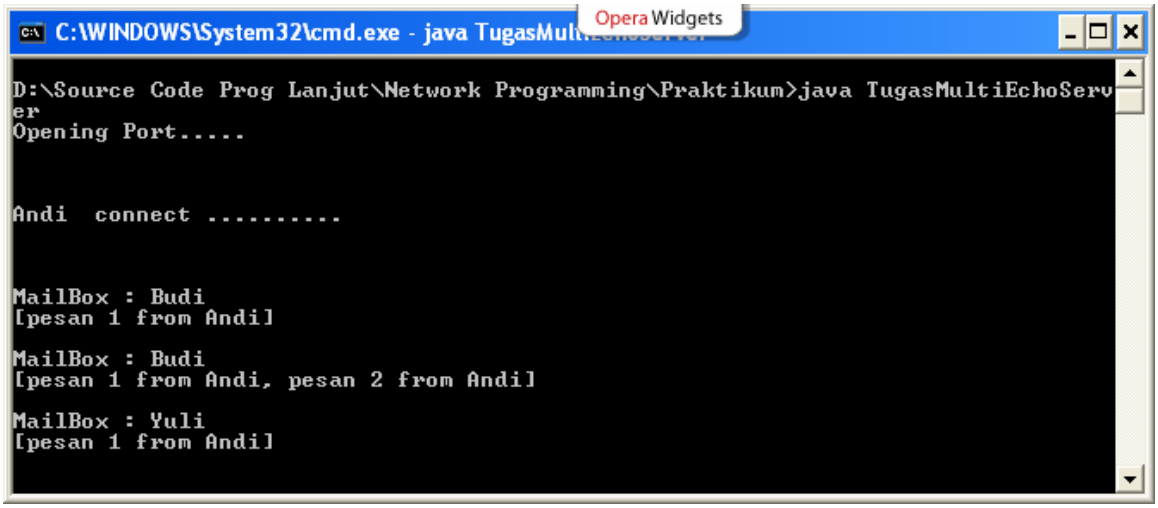
```
Enter send or read : read
Jumlah Pesan : 2
1. Pesan 1 from Budi
2. Pesan 1 form Ita
```

  Maka akan tampak pesan-pesan yang ada pada message box.
- Lakukan dengan cara yang sama dengan user yang berbeda.
- Jika telah selesai ketik close

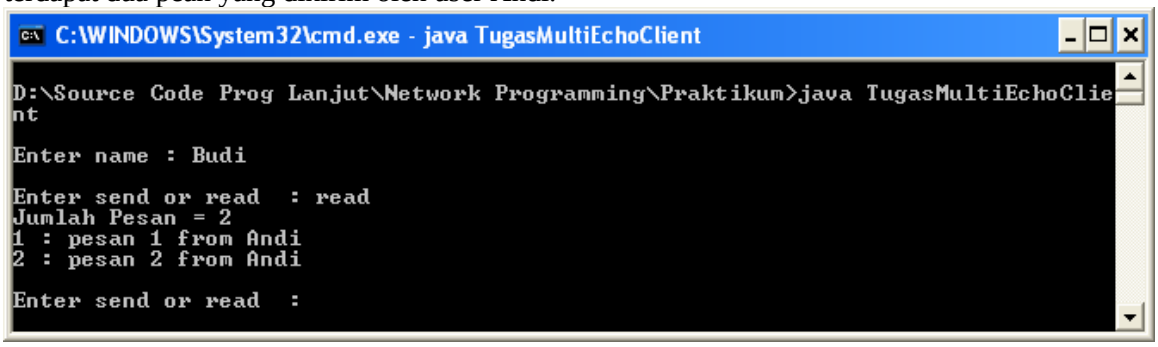
Output program:

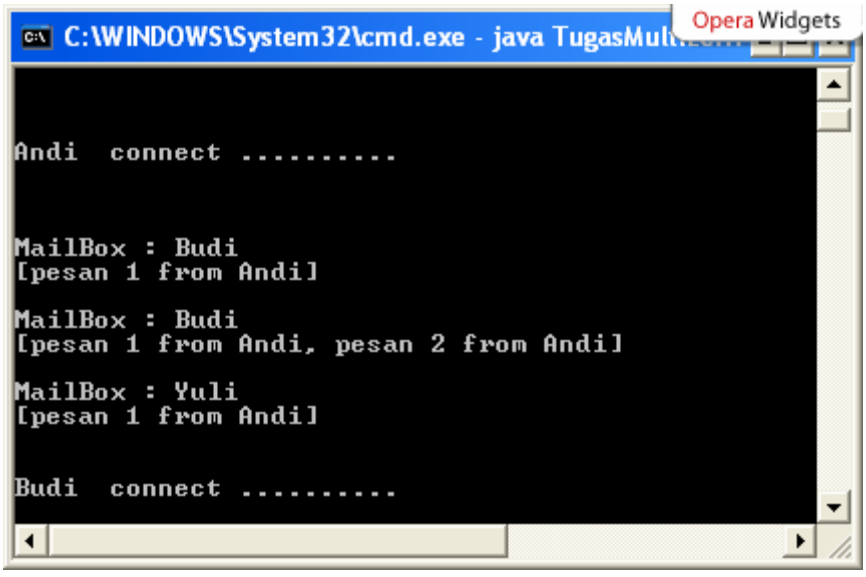
Dibawah ini user Andi terhubung ke Server. Selanjutnya pilih read/send. Pada output program ini user Andi melakukan send message ke Budi sebanyak 2 kali dan send message ke Yuli sebanyak 1 kali.



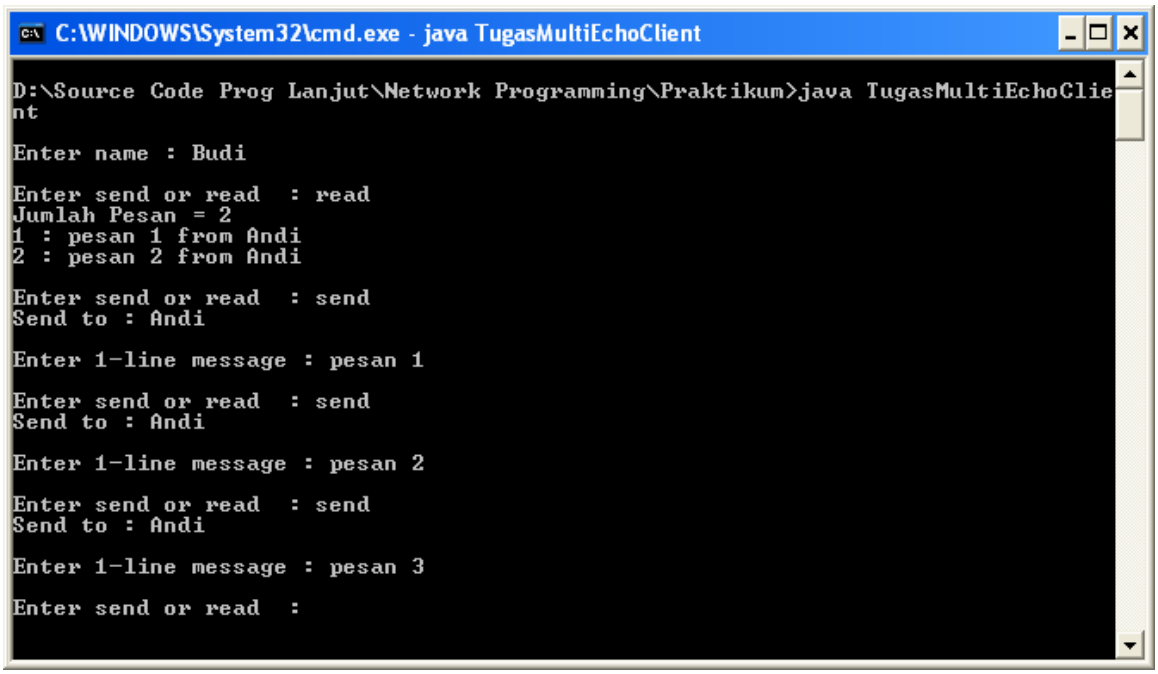


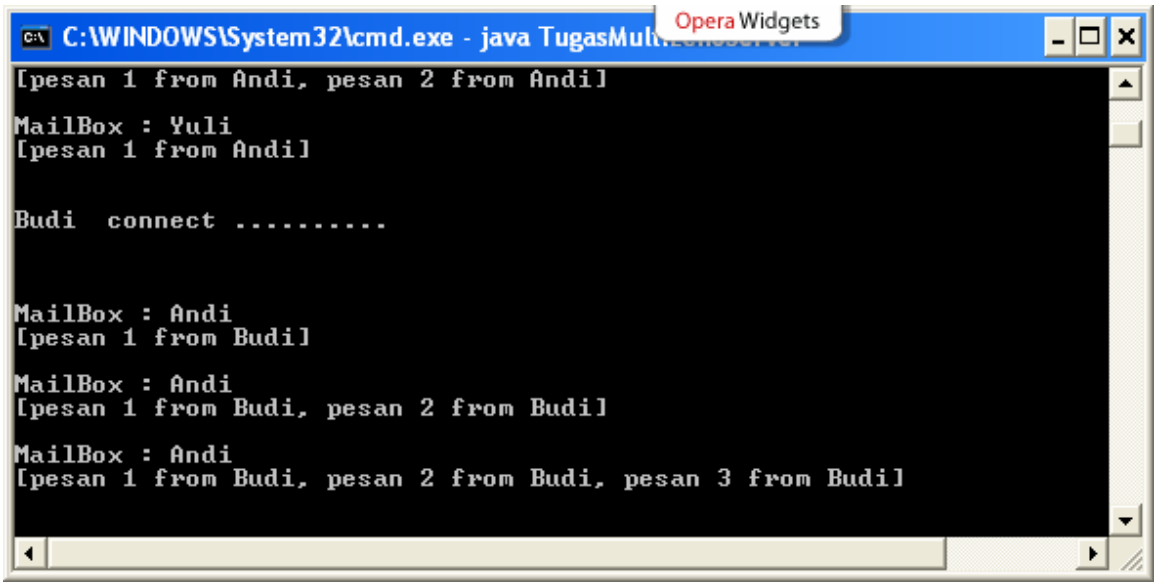
Selanjutnya user Budi terhubung ke Server, user Budi melakukan read, pada mailBox Budi terdapat dua pean yang dikirim oleh user Andi.



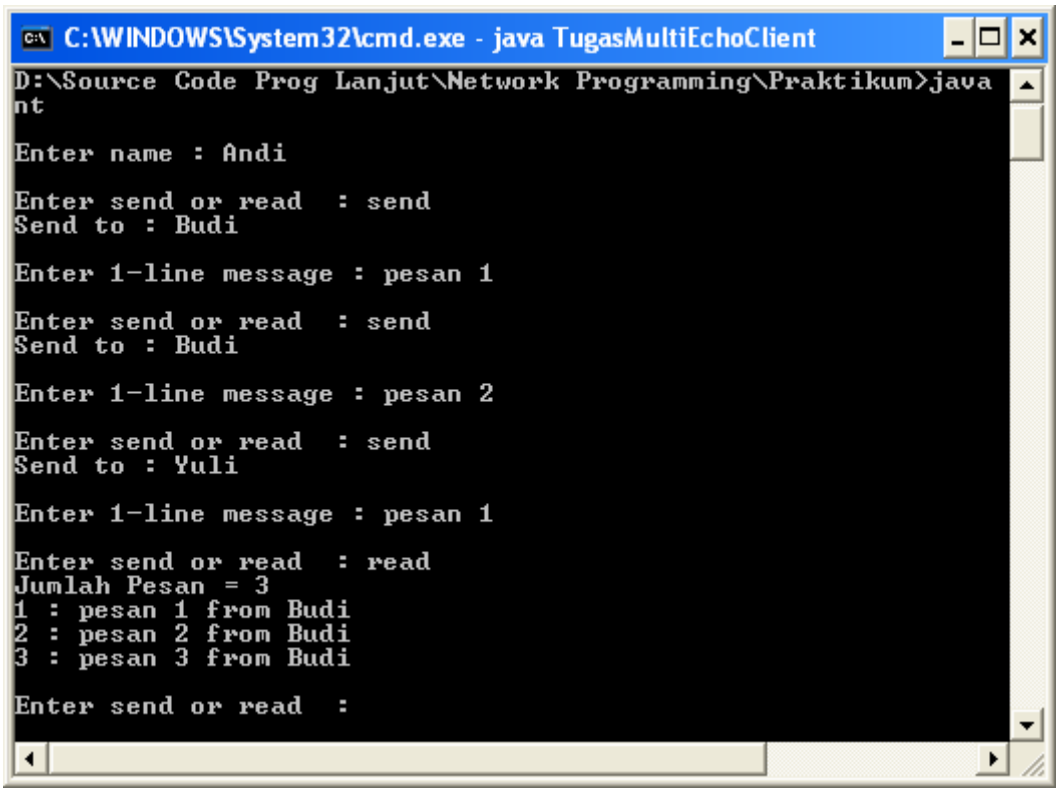


Selanjutnya user Budi melakukan send message ke Andi sebanyak 3 kali, maka mailBox user Andi terdapat 3 message.

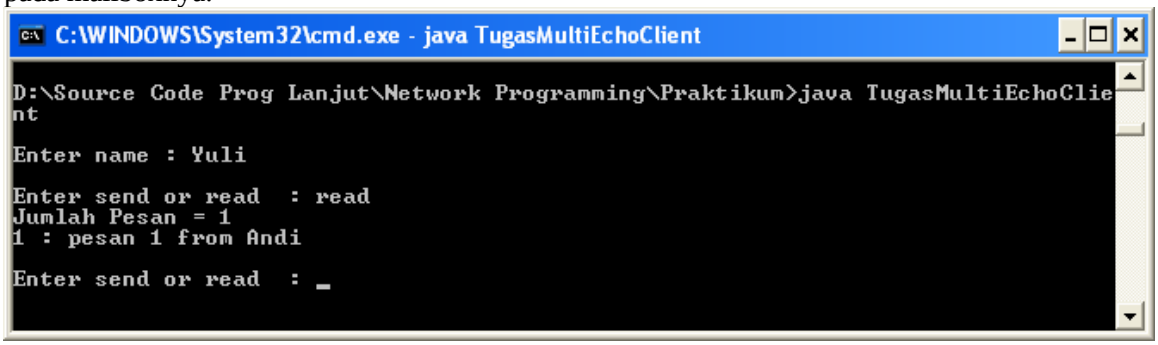


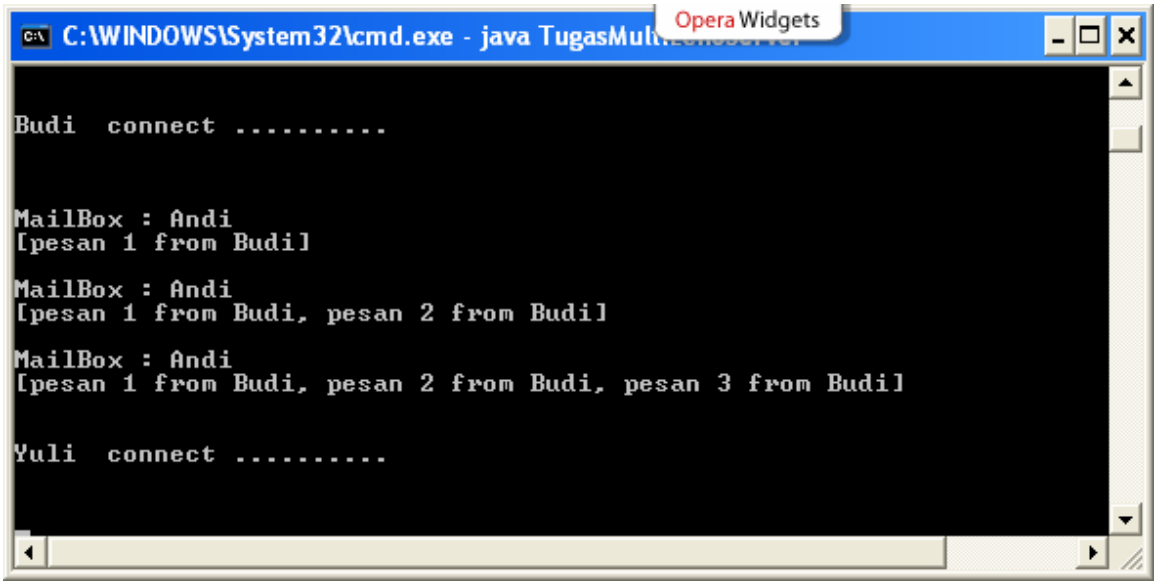


Kembali lagi pada user Andi lakukan read, maka akan tampil 3 message.

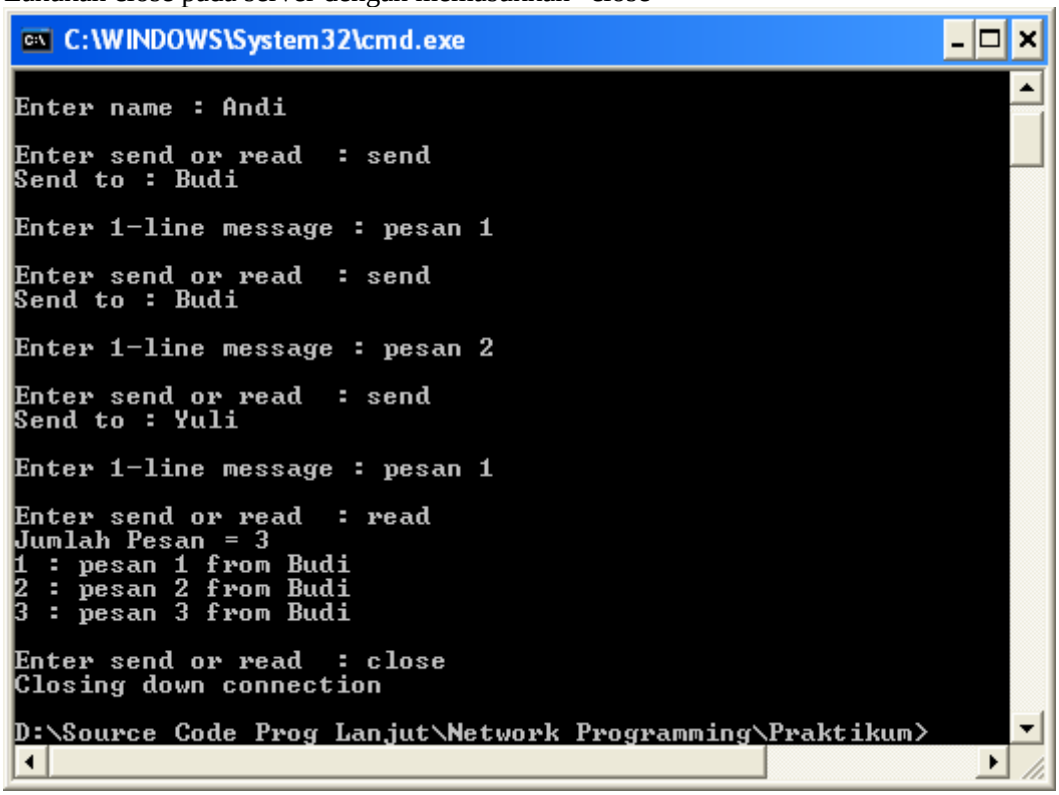


Selanjutnya user Yuli terhubung dengan server. User yuli melakukan read, terdapat 1 message pada mailboxnya.





Lakukan close pada server dengan memasukkan ”close”



PROGRAM

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

public class TugasMultiEchoServer{
   private static ServerSocket  servSock;
   private static final int PORT = 50000 ;
```

```
//untuk menampung user
private static Vector user = new Vector();
//merupakan mailbox semua user
private static Vector mailBoxUser = new Vector();
private static BufferedReader in ;
private static PrintWriter out ;

//untuk mendapatkan mailbox dari user.
//Pada objek user (objek user terbuat dari class Vector), Andi tersimpan
 //dalam indek ke-1, untuk mendapatkan mailbox Andi maka gunakan method
 //getMailBoxUser (1)
public static Vector getMailBoxUser(int index){

}
//untuk mengeset mailbox user menjadi kosong
//pada saat user Andi melakukan read, maka setelah itu mailboxnya manjadi 0
 //message
public static void setMailBoxUser(int noUser){

}

//memasukkan pesan (String message) pada mailBox user tertentu (int noUser).
public static void setMailBoxUser(String message,int noUser){

}

//Untuk menampilkan semua user
public static void showUser(){
    System.out.println("Show user");
    for(Iterator i=user.iterator() ;i.hasNext();)
    {
        System.out.println("user : " + i.next());
    }
}
//untuk menambahkan user pada objek user
//tapi jangan lupa membuat mailbox user juga
public static void addUser(String u){

}

//untuk mendapatkan nama user berdasarkan noUser
public static String getUser(int noUser){
       }

//untuk mencari user berdasarkan nama jika ditemukan maka mengembalikan no
 //indeks user, jika tidak mengembalikan -1
public static int searchUser(String name){

}

public static void main(String args[]) throws IOException{
   System.out.println("Opening Port.....\n");
   try{

      servSock = new ServerSocket(PORT);
   }catch(IOException e){
      System.out.println("Unable to attach to port");
      System.exit(1);
   }

   do{
      Socket link= servSock.accept();
      in = new BufferedReader(new InputStreamReader(link.getInputStream()));
```

```
         out = new PrintWriter(link.getOutputStream(),true);
         //menerima dari client nama user yang akan terhubung dengan server
        //yang akan menjadi masukan dari konstruktor  ClientHandler

         ClientHandler handler = new ClientHandler(link, name);
         //lakukan pencarian user apakah user sudah tersimpan dalam objek user,
        //jika belum maka tambahkan user.

         handler.start();

      }while(true);
   }
}
```

```
class ClientHandler extends Thread{
   private Socket client ;
   private BufferedReader in ;
   private PrintWriter out ;
   private String name ;


   public ClientHandler(Socket socket, String name){
      client = socket ;
      this.name = name ;
      System.out.println("\n\n" + this.name + "  connect ..........\n\n");
      try{
         in = new BufferedReader(new
InputStreamReader(client.getInputStream()));
         out = new PrintWriter(client.getOutputStream(),true);
      }catch(IOException e){
         e.printStackTrace();
      }
   }

   //untuk membaca message pada mailbox, yang diketahui adalah nama user
   public void doRead() throws IOException{
       //cari dulu no indeks user pada objek user
       //dapatkan jumlah message yang ada pada mailbox user tersebut
       //kirimkan ke client jumlah message
       //lakukan pengiriman message ke client secara berulang sejumlah message
       tersebut.
       //set mailbox user tersebut menjadi kosong
   }

   //untuk menerima message dari client
   public void doSend() throws IOException{
         //menerima masukan dari client message tsb akan dikirim ke siapa
(sendTo)
         //cari no indeks yang akan menjadi tujuan message tersebut
         int x = TugasMultiEchoServer.searchUser(sendTo);

         if (x == -1) {
            //jika user yang menjadi tujuan dari pesan tsb belum tersimpan di
            objek user, maka tambahkan pada objek user
            //membaca message dari client
            //masukkan ke mailbox dari user yang dituju

            //MailBox user tujuan di tampilkan di server
             System.out.println("\nMailBox : " +
            TugasMultiEchoServer.getUser(no));
             Vector vc = TugasMultiEchoServer.getMailBoxUser(no);
             System.out.println(vc);
```

```
         }
         Else{
             //jika user sudah ada
             //menerima message dari client

            //MailBox user di cetak
            System.out.println("\nMailBox : " +
TugasMultiEchoServer.getUser(x));
            Vector vc = TugasMultiEchoServer.getMailBoxUser(x);
            System.out.println(vc);

        }
   }

   public void run(){
      try{
         String message="" ;
         do{
             //menerima masukan dari client read or send
             if (message.equalsIgnoreCase("read"))
                doRead();
             if (message.equalsIgnoreCase("send")){
                doSend();
             }
         }while(!message.equalsIgnoreCase("close"));


      }catch(IOException e){
         e.printStackTrace();
      }
      finally{
         try{
            if (client != null){
               System.out.println("Closing down connection");
               client.close();
            }
         }catch(IOException e){
            e.printStackTrace();
         }
      }
   }
}
```

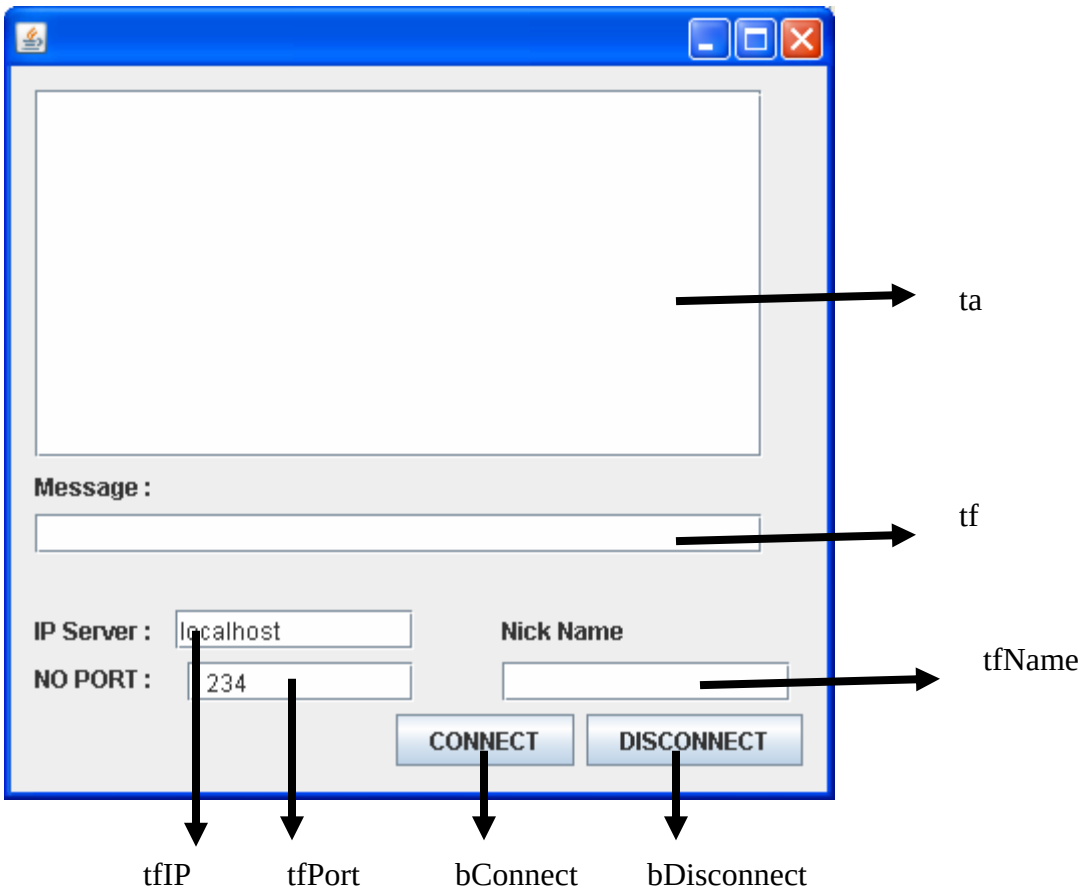
```
import java.io.*;
import java.net.*;

public class TugasMultiEchoClient{
   private static InetAddress host ;
   private static final int PORT = 50000;
   private static Socket link ;
   private static BufferedReader in ;
   private static PrintWriter out ;
   private static BufferedReader userEntry ;
   private static String name="";

   //untuk megirimkan message ke server
   private static void doSend() throws IOException{
      //ketikkan pesan akan dikirim ke siapa
      //kirim ke server
      //ketikkan message yang akan dikirim
      //kirim message tersebut ke server
```

```
   }

   //untuk membaca message
   private static void doRead() throws IOException{
     //menerima pesan dari server jumlah message yang ada pada mailbox, tapi
     masih dalam bentuk string
     //ubah ke int
     //lakukan perulangan sebanyak jumlah message untuk menerima message dari
     server.
   }
   public static void main(String args[]){
      try{
         host = InetAddress.getLocalHost();
         link = new Socket(host,PORT);

         in = new BufferedReader(new InputStreamReader(link.getInputStream()));
         out = new PrintWriter(link.getOutputStream(),true);

         userEntry = new BufferedReader(new InputStreamReader(System.in));

         String  sendRead="" ;

         //masukkan nama user
         //kirim ke server

         do{
           //masukkan send or read
           //kirim ke server
           if (sendRead.equalsIgnoreCase("send"))
              doSend();
           if (sendRead.equalsIgnoreCase("read"))
              doRead();
         }while(!sendRead.equalsIgnoreCase("close"));
      }catch(UnknownHostException e){
          System.out.println("Host ID not found!");
      }
      catch(IOException e){
         e.printStackTrace();
      }
      finally{
         try{
            if (link != null){
                System.out.println("Closing down connection");
                link.close();
            }
         }
         catch(IOException e){
             e.printStackTrace();
         }
      }
   }
}
```

Contoh program tentang Chat Client-Server bisa di download di http://breakdesign.blogspot.com/2008/01/simple-java-chat-client-server.html

Praktikum 4 : Aplikasi Chat Multi Client

Buatlah aplikasi chat multi client, program ini merupakan pengembangan dari MultiEchoServer dan MultiEchoClient. Output pogram untuk client adalah sebagai berikut :



Pertama kali tekan button Connect supaya terhubung ke server, jika ingin mengganti no IP Server dan no Port masukkan text field yang telah disediakan. Tentu saja, untuk no Port harus sama dengan no Port server. Untuk default IP Server adalah localhost dan no Port adalah 1234. Jangan lupa memasukkan nick name Anda supaya dikenali pada saat chatting.

Program Server

```
import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.Vector;

public class server {
   private static ServerSocket  servSock;
   private static final int PORT = 1234 ;

   //untuk menyimpan socket-socket yang terhubung ke server
   static Vector clients = new Vector();
   public static void main(String args[]) throws IOException{
      System.out.println("Opening Port.....\n");
      try{
         servSock = new ServerSocket(PORT);
      }catch(IOException e){
         System.out.println("Unable to attach to port");
```

```
            System.exit(1);
         }

         do{
            Socket client = servSock.accept();
            cThread handler = new cThread(client);
            clients.add(handler);
            handler.start();
         }while(true);
      }

}
```

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.Socket;

class cThread extends Thread{
   private Socket client ;
   private BufferedReader in ;
   private PrintWriter out ;

   public cThread(Socket socket){
      client = socket ;
      try{
         in = new BufferedReader(new
InputStreamReader(client.getInputStream()));
         out = new PrintWriter(client.getOutputStream(),true);
      }catch(IOException e){
         e.printStackTrace();
      }
   }
   public void send(String message){
       out.println(message);
   }
   public boolean equals(cThread ct){
       //jika socket ct dengan socket pada class ini sama return true
       // jika tidak return false ;
   }
   public void run(){
      try{
         String received ;
         do{
            //baca dari client
            //tampilkan pada Server

            //message yang diterima oleh server di broadcast ke semua client
            for(int i=0;i<server.clients.size();i++)
            {
                cThread client = (cThread) server.clients.get(i);
                client.send(received);
            }
         }while(!received.equals("QUIT"));
      }catch(IOException e){
         e.printStackTrace();
      }
      finally{
         try{
            if (client != null){
               System.out.println("Closing down connection");
```

```
               client.close();
            }
         }catch(IOException e){
            e.printStackTrace();
         }
      }
   }
}
```

+

Program Client

```
public class ChatClient extends javax.swing.JFrame implements Runnable {
   private InetAddress host ;
   private int PORT = 1234;
   private Socket link ;
   private BufferedReader in ;
   private PrintWriter out ;
   private Thread listener;

   public ChatClient() {
      initComponents();
   }

//untuk menutup koneksi ke server
private void bDisConnectActionPerformed(java.awt.event.ActionEvent evt) {
      try{

        //for ini untuk : sebelum link.close(), remove socket yang tersimpan di
Vector clients
        for(int i=0;i<server.clients.size();i++)
        {    //dapatkan data socket client yang terkoneksi dengan server pada
indeks ke-i
             //cek apakah sama dengan socket yang akan di close pada saat ini
             //jika sama lakukan link.close(); dan lakukan remove dari Vector
Client
        }
        bConnect.setEnabled(true) ;
        }catch(IOException e){

        }
}

   //untuk melakukan koneksi ke server
   private void bConnectActionPerformed(java.awt.event.ActionEvent evt) {
        try{
            //isilah host dan no Port
            // buat koneksi ke server
            // set in
            // set out
        }catch(IOException e){

        }
        listener = new Thread(this);
        listener.start();
        bConnect.setEnabled(false);
    }

   public void run () {
    try {
      while (true) {
        //baca dari server
```

```
      //tampilkan di TextArea
    }
  } catch (IOException ex) {
    ex.printStackTrace ();
  } finally {
    listener = null;

    validate ();
    try {
      out.close ();
    } catch (Exception ex) {
      ex.printStackTrace ();
    }
  }

}
```

```
 private void tfActionPerformed(java.awt.event.ActionEvent evt) {
    try{
      String message="", response;
        if (!message.equals("QUIT")){
            // simpan message dari textfield tf ke var message
            // simpan nickName user ke var name
           //kirim ke server name + message
           // set text field tf dengan nilai = ""
        }
      }catch(Exception e){

      }
  }
```

******* *Selamat Mengerjakan* *********