University of Dhaka

Department of Computer Science and Engineering

CSE-3111 : Computer Networking Lab

Lab Report 2 : Introduction to Socket Programming — Exercises on Simple Client-Server Communication

Submitted By:

Afser Adil Olin

Roll No : 47

Anika Tabassum

Roll No : 61

Submitted On :

January 24, 2023

Submitted To :

Dr. Md. Abdur Razzaque

Md Mahmudur Rahman

Md. Ashraful Islam

Md. Fahim Arefin

Contents

1 Introduction

The preliminary objective of this lab is to create a simple file creating TCP connections using Socket Programming to establish client requests for a file from server and server sends corresponding file to the client.

1.1 Objectives

Creating TCP Connections using Socket Programming

- Establishing a TCP connection between a server process running on host A and a client process running on host B can be done using socket programming
  - Small letter to capital conversion for a line of text: The client process can send a request to the server process with a line of text, and the server process can convert all small letters to capital letters and send the converted text as a response to the client process.
  - Checking whether a number is prime or not: The client process can send a request to the server process with a number, and the server process can check whether the number is prime or not and send the result as a response to the client process.
- To handle failure of request messages, failure of response messages and process execution failures, a non-idempotent operation using exactly-once semantics can be implemented.
  - An application-level protocol can be designed that allows a user's card and password to be verified, the account balance to be queried, and an account withdrawal to be made. The protocol can include messages such as:
    - Request for verification of card and password
    - Request for account balance
    - Request for withdrawal
    - Confirmation of withdrawal
    - Error message if there is not enough money in the account
  - To handle errors related to both request and response messages, the protocol can include a mechanism for resending messages in case of failures, and an acknowledgement system to confirm

the receipt of messages. The protocol can also include a time-out mechanism to handle cases where a response is not received within a certain period. In case of process execution failures, the protocol can include a mechanism to restart the process.

2 Theory

Socket programming is a method of inter-process communication (IPC) that allows processes to communicate with each other across a network using sockets. A socket is a endpoint for sending or receiving data across a computer network. It is a combination of an IP address and a port number.

In socket programming, a client creates a socket and connects to a server using the server's IP address and port number. The server then creates a socket and binds it to a specific IP address and port number. Once the connection is established, the client and server can send and receive data through the socket.

2.1 Server

In the server side when we turn it on it will wait for client requests. If it gets any request then it will establish a connection.

2.2 Client

Here our client side is any host. We will enter IP address of our server and the port number. Then the client connection request will be sent from our client to the server.

3 Methodology

Creating TCP (Transmission Control Protocol) connections using socket programming involves several steps, including creating a socket, connecting to a server, and sending and receiving data.

After setting up the connection, it receives an http query corresponding to which file client requested. Then it will read bytes from that file and send it to the client.

3.1 Creating a socket

The first step is to create a socket using the socket() function. This function takes three arguments: the address family, the type of socket, and the protocol.

3.2 Connecting to a server

Once the socket is created, the next step is to connect to a server using the connect() function. This function takes three arguments: the socket descriptor, the server's IP address, and the server's port number.

3.3 Sending and receiving data

Once the connection is established, the client and server can send and receive data through the socket.

3.4 Closing the connection

Once the data exchange is complete, the connection can be closed using the close() function. This function takes one argument: the socket descriptor.

3.5 Idempotent Operation

In computing, an idempotent operation is one that has no additional effect if it is called more than once with the same input parameters. For example, removing an item from a set can be considered an idempotent operation on the set.

3.6 Exactly-once Semantics

As its name suggests, exactly-once semantics means that each message is delivered precisely once. The message can neither be lost nor delivered twice (or more times). Exactly-once is by far the most dependable message delivery guarantee.

4 Experimental result

4.1 Some operation by the server process requested by the client and send responses from the server:

- Small letter to capital conversion for a line of text:

Java program for a Client

```
// A Java program for a Client
import java.io.*;
import java.net.*;
import java.util.Scanner;

public class Client {
    // initialize socket and input output streams
    private Socket socket = null;
    private Socket socketout = null;
    private ServerSocket server = null;
    private DataInputStream input = null;
    private DataOutputStream out = null;
    private DataInputStream in = null;

    // constructor to put ip address and port
    public Client(String address, int port) {
        Scanner scanner = new Scanner(System.in);
        // establish a connection
        try {
            server = new ServerSocket(5000);
            socket = new Socket(address, port);
            System.out.println("Connected");
            socketout = server.accept();

            // takes input from terminal
            input = new DataInputStream(System.in);

            // sends output to the socket
            out = new DataOutputStream(
                    socket.getOutputStream());
            in = new DataInputStream(socketout.
                getInputStream());
        } catch (UnknownHostException u) {
            System.out.println(u);
            return;
        } catch (IOException i) {
```

```
36              System.out.println(i);
37              return;
38          }
39
40          // string to read message from input
41          String line = "Hello";
42
43          // keep reading until "Over" is input
44          while (true) {
45              try {
46                  line = input.readLine();
47                  out.writeUTF(line);
48                  line = in.readUTF();
49                  System.out.println(line);
50
51              } catch (IOException i) {
52                  System.out.println(i);
53                  break;
54              }
55          }
56
57          // close the connection
58          try {
59              input.close();
60              out.close();
61              socket.close();
62          } catch (IOException i) {
63              System.out.println(i);
64          }
65      }
66
67      public static void main(String args[])
68      {
69          Client client = new Client("10.33.2.89", 5000);
70      }
71  }
```

Java program for a Server

```
72  // A Java program for a Server
73  import java.net.*;
74  import java.io.*;
75
76  public class Server
77  {
78      //initialize socket and input stream
79      private Socket       socket = null;
80      private ServerSocket server = null;
81      private Socket       socketsent = null;
82      private DataInputStream in   = null;
83      private DataOutputStream out = null;
84      private int traxid ;
85      // constructor with port
86      public Server(int port)
87      {
88          // starts server and waits for a connection
89          try
90          {
91              //10.33.2.89
92              server = new ServerSocket(port);
93              System.out.println("Server started");
94
95              System.out.println("Waiting for a client
                    ...");
96
97              socket = server.accept();
98              System.out.println("Client accepted");
99              socketsent = new Socket("10.33.2.90",5000);
100
101             // takes input from the client socket
102             in = new DataInputStream(
103                     new BufferedInputStream(socket.
                            getInputStream()));
104             out = new DataOutputStream(
105                     socketsent.getOutputStream());
106
107             String line = "";
108
109             // reads message from client until "Over"
                    is sent
110             while (true)
111             {
```

```
                try
                {

                    line = in.readUTF();
                    //lower to upper
                    System.out.println(line);
                    out.writeUTF(line.toUpperCase());

                }
                catch(IOException i) {
                    System.out.println(i);
                    break;
                }
            }
            System.out.println("Closing connection");
            // close connection
            socket.close();
            socketsent.close();
            in.close();
            out.close();
        }
        catch(IOException i)
        {
            System.out.println(i);
        }
    }
    public static void main(String args[])
    {
        Server server = new Server(5000);
    }
}
```

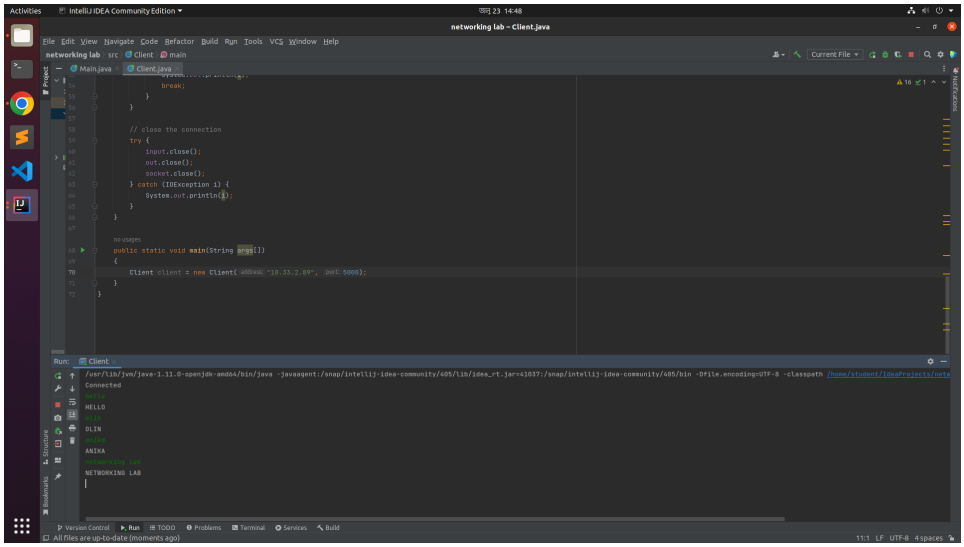

Figure 1: Content of Client

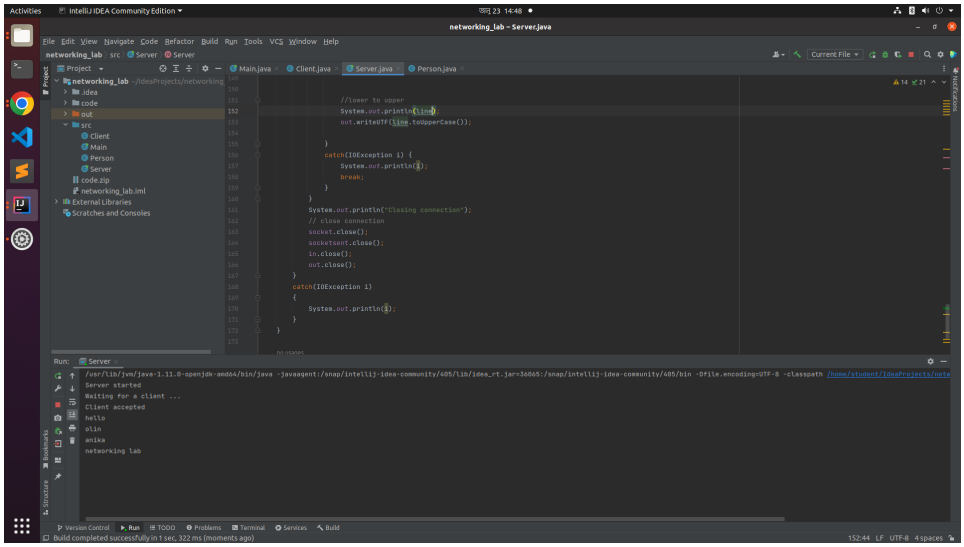

Figure 2: Content of Server

- ii. Checking whether a number is prime or not:

Java program for a Client

```
143 // A Java program for a Client
144 import java.io.*;
145 import java.net.*;
146 import java.util.Scanner;
147
148 public class Client {
149     // initialize socket and input output streams
150     private Socket socket = null;
151     private Socket socketout = null;
152     private ServerSocket server = null;
153     private DataInputStream input = null;
154     private DataOutputStream out = null;
155     private DataInputStream in = null;
156
157     // constructor to put ip address and port
158     public Client(String address, int port) {
159         Scanner scanner = new Scanner(System.in);
160         // establish a connection
161         try {
162             server = new ServerSocket(5000);
163             socket = new Socket(address, port);
164             System.out.println("Connected");
165             socketout = server.accept();
166
167             // takes input from terminal
168             input = new DataInputStream(System.in);
169
170             // sends output to the socket
171             out = new DataOutputStream(
172                     socket.getOutputStream());
173             in = new DataInputStream(socketout.
                    getInputStream());
174         } catch (UnknownHostException u) {
175             System.out.println(u);
176             return;
177         } catch (IOException i) {
178             System.out.println(i);
179             return;
180         }
181
182         // string to read message from input
```

```
183             String line = "Hello";
184
185             // keep reading until "Over" is input
186             while (true) {
187                 int x;
188                 x=scanner.nextInt();
189                 try {
190                     //line = input.readLine();
191                     out.writeUTF(""+x);
192                     line = in.readUTF();
193                     System.out.println(line);
194
195                 } catch (IOException i) {
196                     System.out.println(i);
197                     break;
198                 }
199             }
200
201             // close the connection
202             try {
203                 input.close();
204                 out.close();
205                 socket.close();
206             } catch (IOException i) {
207                 System.out.println(i);
208             }
209         }
210
211         public static void main(String args[])
212         {
213             Client client = new Client("10.33.2.89", 5000);
214         }
215 }
```

Java program for a Server

```
216 // A Java program for a Server
217 import java.net.*;
218 import java.io.*;
219
220 public class Server
221 {
222     //initialize socket and input stream
223     private Socket       socket = null;
```

```
224     private ServerSocket server = null;
225     private Socket       socketsent = null;
226     private DataInputStream in   = null;
227     private DataOutputStream out = null;
228     private int traxid ;
229     // constructor with port
230     public Server(int port)
231     {
232         // starts server and waits for a connection
233         traxid = 10000414;
234         try
235         {
236             //10.33.2.89
237             server = new ServerSocket(port);
238             System.out.println("Server started");
239
240             System.out.println("Waiting for a client ...");
241
242             socket = server.accept();
243             System.out.println("Client accepted");
244             socketsent = new Socket("10.33.2.90",5000);
245
246             // takes input from the client socket
247             in = new DataInputStream(
248                     new BufferedInputStream(socket.
                            getInputStream()));
249             out = new DataOutputStream(
250                     socketsent.getOutputStream());
251
252             String line = "";
253
254             // reads message from client until "Over" is
                    sent
255             while (true)
256             {
257                 try
258                 {
259
260                     line = in.readUTF();
261                     //prime checker
262                     System.out.println(line);
263                     int x = new Integer(line);
264                     boolean ok = false;
265                     for(int i=2;i*i<=x;i++)
266                     {
```

```
                        if(x%i==0)
                        {
                            ok = true;
                        }
                    }
                    if(ok)
                    {
                        out.writeUTF("Not Prime");
                    }
                    else
                    {
                        out.writeUTF("Prime");
                    }

                }
                catch(IOException i) {
                    System.out.println(i);
                    break;
                }
            }
            System.out.println("Closing connection");
            // close connection
            socket.close();
            socketsent.close();
            in.close();
            out.close();
        }
        catch(IOException i)
        {
            System.out.println(i);
        }
    }

    public static void main(String args[])
    {
        Server server = new Server(5000);
    }
}
```

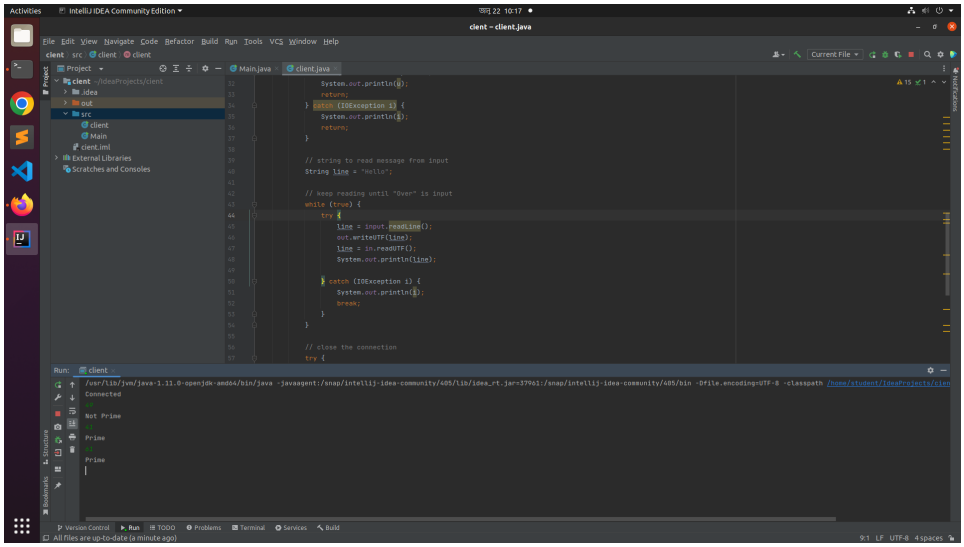

Figure 3: Content of Client

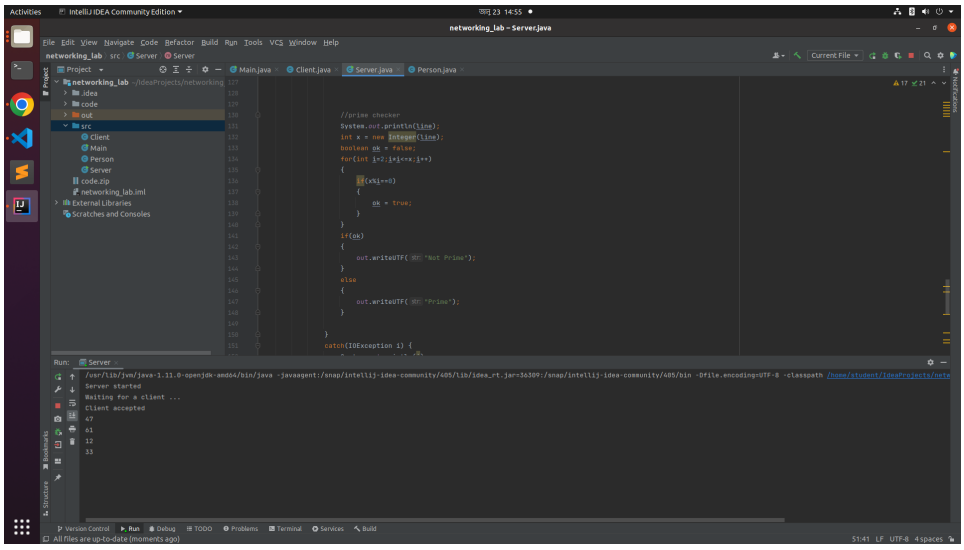

Figure 4: Content of Server

4.2 Implementation of a non-idempotent operation using exactly-once semantics that can handle failure of request messages, failure of response messages and process execution failures:

- Application-level Banking protocol:

Java program for a Client

```
306         // A Java program for a Client
307         import java.io.*;
308         import java.net.*;
309         import java.security.spec.RSAOtherPrimeInfo;
310         import java.util.Scanner;
311
312 public class Client {
313     // initialize socket and input output streams
314     private Socket socket = null;
315     private Socket socketout = null;
316     private ServerSocket server = null;
317     private DataInputStream input = null;
318     private DataOutputStream out = null;
319     private DataInputStream in = null;
320
321     // constructor to put ip address and port
322     public Client(String address, int port) {
323         Scanner scanner = new Scanner(System.in);
324         // establish a connection
325         try {
326             server = new ServerSocket(5000);
327             socket = new Socket(address, port);
328             System.out.println("Connected");
329             socketout = server.accept();
330
331             // takes input from terminal
332             input = new DataInputStream(System.in);
333
334             // sends output to the socket
335             out = new DataOutputStream(
336                     socket.getOutputStream());
337             in = new DataInputStream(socketout.
                    getInputStream());
338         } catch (UnknownHostException u) {
339             System.out.println(u);
```

```
340             return;
341         } catch (IOException i) {
342             System.out.println(i);
343             return;
344         }
345
346         // string to read message from input
347         String line , line2, line3;
348
349         // keep reading until "Over" is input
350         while (true) {
351             try {
352                 System.out.println("Enter Username:");
353                 line = input.readLine();
354                 out.writeUTF(line);
355
356                 System.out.println("Enter Password:");
357                 line2 = input.readLine();
358                 out.writeUTF(line2);
359
360                 line3 = in.readUTF();
361                 if(line3.equals("ok"))
362                 {
363                     System.out.println("Proceed");
364                     int choose;
365                     do
366                     {
367                         System.out.println("Press 1 for
                              Check balance\nPress 2 for
                             Credit balance\nPress 3 for
                             debit balance\nPress 0 for
                             exit");
368                         choose = scanner.nextInt();
369                         //System.out.println(choose);
370                         if(choose ==1 )
371                         {
372                             out.writeUTF("1");
373                             line = in.readUTF();
374                             System.out.println(line);
375                         }
376                         if(choose ==2)
377                         {
378                             out.writeUTF("2");
379                             String money = input.
                                    readLine();
```

```
380                             out.writeUTF(money);
381                             line = in.readUTF();
382                             if(line.equals("1"))
383                             {
384                                 line = in.readUTF();
385                                 System.out.println(line
                                        );
386                                 System.out.println("
                                        Your transection is
                                        successful");
387                             }
388                             else {
389                                 System.out.println("
                                        insufficient balance
                                        ");
390                             }
391
392                         }
393
394                         if(choose ==3)
395                         {
396                             out.writeUTF("3");
397                             String money = input.
                                    readLine();
398                             out.writeUTF(money);
399                             line = in.readUTF();
400                             if(line.equals("1"))
401                             {
402                                 line = in.readUTF();
403                                 System.out.println(line
                                        );
404                                 System.out.println("
                                        Your transection is
                                        successful");
405                             }
406                         }
407
408                     }
409                     while (choose!=0);
410                 }
411                 else System.out.println("Invalid
                        username or password");
412
413             } catch (IOException i) {
414                 System.out.println(i);
```

```
                break;
            }
        }

        // close the connection
        try {
            input.close();
            out.close();
            socket.close();
        } catch (IOException i) {
            System.out.println(i);
        }
    }

    public static void main(String args[])
    {
        Client client = new Client("10.33.2.89", 5000);

    }
}
```

Java program for a Server

```
// A Java program for a Server
import java.net.*;
import java.io.*;

public class Server
{
    //initialize socket and input stream
    private Socket       socket = null;
    private ServerSocket server = null;
    private Socket       socketsent = null;
    private DataInputStream in   = null;
    private DataOutputStream out = null;

    // constructor with port
    public Server(int port)
    {
        // starts server and waits for a connection
        try
        {
            //10.33.2.89
            server = new ServerSocket(port);
            System.out.println("Server started");
```

```
457
458             System.out.println("Waiting for a client
                    ...");
459
460             socket = server.accept();
461             System.out.println("Client accepted");
462             socketsent = new Socket("10.33.2.90",5000);
463
464             // takes input from the client socket
465             in = new DataInputStream(
466                     new BufferedInputStream(socket.
                            getInputStream()));
467             out = new DataOutputStream(
468                     socketsent.getOutputStream());
469
470             Person [] person =  new Person[4];
471             person[0] = new Person("Olin" ,"olinone","
                    123456", 500000 );
472             person[1] =   new Person("Anika", "anika",
                    "12345678", 11000100);
473             person[2] =    new Person("Ryana","lali", "
                    fardin",10000);
474             person[3] =   new Person("Fardin","magu", "
                    halamadrid",80000);
475
476             String line = "";
477
478             // reads message from client until "Over"
                    is sent
479             while (true)
480             {
481                 try
482                 {
483
484                     line = in.readUTF();
485                     String username = line;
486                     System.out.println(line);
487                     line = in.readUTF();
488                     String pass = line;
489                     System.out.println(line);
490                     boolean credok = false;
491                     for(int i=0;i<4;i++)
492                     {
493                         int totmoney = 0;
494                         if(person[i].check(username,
```

```
                    pass))
495             {
496                 credok = true;
497                 out.writeUTF("ok");
498                 do {
499                     line = in.readUTF();
500                     System.out.println(line
                            );
501                     if(line.equals("1"))
502                     {
503                         out.writeUTF("Your
                                current balance
                                is : " + person[
                                i].balance);
504                     }
505                     if(line.equals("2"))
506                     {
507                         line = in.readUTF()
                                ;
508                         int money = new
                                Integer(line);
509                         if(money<=person[i
                                ].balance &&
                                totmoney + money
                                <=20000)
510                         {
511                             person[i].
                                    balance-=
                                    money;
512                             out.writeUTF("1
                                    ");
513                             out.writeUTF("
                                    Your current
                                     balance is
                                    : " + person
                                    [i].balance)
                                    ;
514                             totmoney+=money
                                    ;
515                         }
516                         else
517                         {
518                             out.writeUTF("0
                                    ");
519                         }
```

```
520                                 }
521                                 if(line.equals("3"))
522                                 {
523                                     line = in.readUTF()
                                        ;
524                                     int money = new
                                        Integer(line);
525                                     System.out.println(
                                        money);
526                                     person[i].balance+=
                                        money;
527                                     out.writeUTF("1");
528                                     out.writeUTF("Your
                                        current balance
                                        is : " + person[
                                        i].balance);
529                                 }
530                                 if(line.equals("0"))
531                                 {
532                                     break;
533                                 }
534                             }while (true);
535
536                         }
537                     }
538                     if(!credok)
539                     {
540                         System.out.println("Username or
                                password is invalid");
541                         out.writeUTF("not ok");
542                     }
543                 }
544                 catch(IOException i) {
545                     System.out.println(i);
546                     break;
547                 }
548             }
549             System.out.println("Closing connection");
550             // close connection
551             socket.close();
552             socketsent.close();
553             in.close();
554             out.close();
555         }
556         catch(IOException i)
```

```
557             {
558                 System.out.println(i);
559             }
560         }
561
562         public static void main(String args[])
563         {
564             Server server = new Server(5000);
565         }
566 }
```

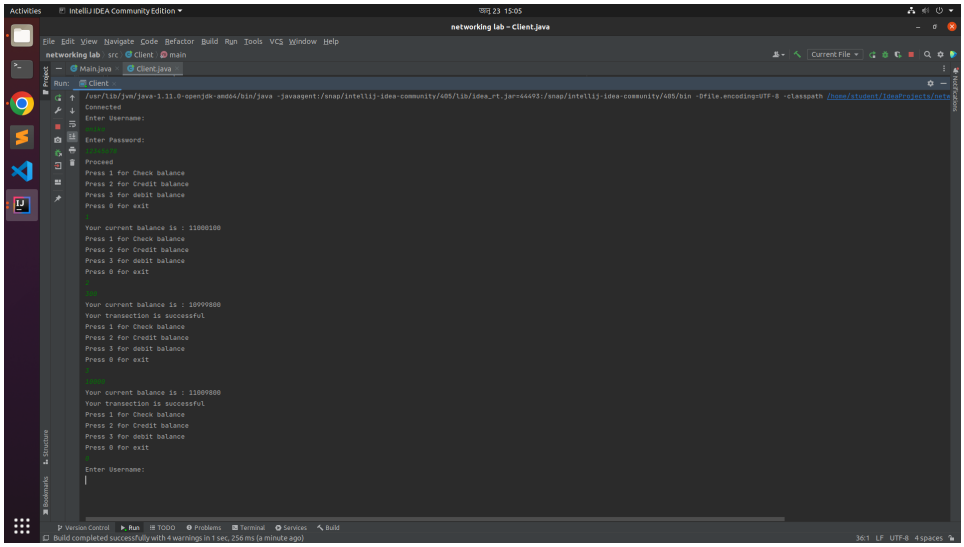

Figure 5: Content of Client

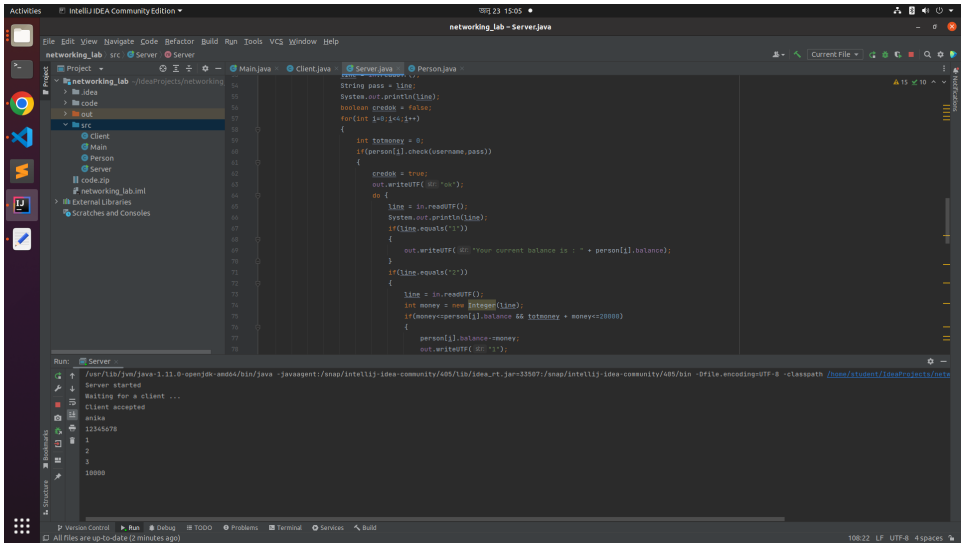

Figure 6: Content of Server

- Enhance the above protocol so that it can handle errors related to both request and response messages to and from the server.:

Java program for a Client

```
567 // A Java program for a Client
568         import java.io.*;
569         import java.net.*;
570         import java.security.spec.RSAOtherPrimeInfo;
571         import java.util.Scanner;
572
573 public class Client {
574     // initialize socket and input output streams
575     private Socket socket = null;
576     private Socket socketout = null;
577     private ServerSocket server = null;
578     private DataInputStream input = null;
579     private DataOutputStream out = null;
580     private DataInputStream in = null;
581
582     // constructor to put ip address and port
583     public Client(String address, int port) {
584         Scanner scanner = new Scanner(System.in);
585         // establish a connection
```

```
586         try {
587             server = new ServerSocket(5000);
588             socket = new Socket(address, port);
589             System.out.println("Connected");
590             socketout = server.accept();
591 
592             // takes input from terminal
593             input = new DataInputStream(System.in);
594 
595             // sends output to the socket
596             out = new DataOutputStream(
597                     socket.getOutputStream());
598             in = new DataInputStream(socketout.
                    getInputStream());
599         } catch (UnknownHostException u) {
600             System.out.println(u);
601             return;
602         } catch (IOException i) {
603             System.out.println(i);
604             return;
605         }
606 
607         // string to read message from input
608         String line , line2, line3;
609         int traxid =845743;
610         int serverid;
611 
612         // keep reading until "Over" is input
613         while (true) {
614             try {
615                 System.out.println("Enter Username:");
616                 line = input.readLine();
617                 out.writeUTF(line);
618 
619                 System.out.println("Enter Password:");
620                 line2 = input.readLine();
621                 out.writeUTF(line2);
622 
623                 line3 = in.readUTF();
624 
625                 if(line3.equals("ok"))
626                 {
627                     System.out.println("Proceed");
628                     int choose;
629                     do
```

```
630                     {
631                         System.out.println("Press 1 for
                                Check balance\nPress 2 for
                               Credit balance\nPress 3 for
                               debit balance\nPress 0 for
                               exit");
632                         choose = scanner.nextInt();
633                         //System.out.println(choose);
634                         if(choose ==1 )
635                         {
636                             out.writeUTF("1");
637                             line = in.readUTF();
638                             System.out.println(line);
639                         }
640                         if(choose ==2) {
641                             out.writeUTF("2");
642
643                             traxid += 1;//send trax id
644                             out.writeUTF("" + traxid);
645
646                             line = in.readUTF();//got
                                    server id
647                             serverid = new Integer(line
                                    );
648                             System.out.println("Server
                                    id: "+serverid);
649                             System.out.println("trax id
                                    :"+traxid);
650
651                             String money = input.
                                    readLine();//money send
652                             out.writeUTF(money);
653                             //System.out.println(money)
                                    ;
654
655                     while(true)
656                     {
657                         line=in.readUTF();
658                         //System.out.println(line);
659                         if(line.charAt(0)=='0')
660                         {
661                             System.out.println("
                                    insufficient balance");
662                             break;
663                         }
```

```
664                         else
665                         {
666                             int tmp = new Integer(line.
                                substring(2,line.length
                                ()));
667                             //System.out.println(tmp);
668                             //System.out.println(traxid
                                );
669                             if(tmp!=traxid)
670                             {
671                                 out.writeUTF("mile nai"
                                    );
672                             }
673                             else {
674                                 out.writeUTF(""+
                                    serverid);
675                                 line = in.readUTF();
676                                 if(line.equals("1"))
677                                 {
678                                     line = in.readUTF()
                                        ;
679                                     System.out.println(
                                        line);
680                                     break;
681                                 }
682                             }
683                         }
684                     }
685
686
687                 }
688
689                 if(choose ==3)
690                 {
691                     out.writeUTF("3");
692                     String money = input.readLine();
693                     out.writeUTF(money);
694                     line = in.readUTF();
695                     if(line.equals("1"))
696                     {
697                         line = in.readUTF();
698                         System.out.println(line);
699                         System.out.println("Your
                            transection is successful");
700                     }
```

```
701                     }
702
703                 }
704                 while (choose!=0);
705             }
706             else System.out.println("Invalid username or
                    password");
707
708         } catch (IOException i) {
709             System.out.println(i);
710             break;
711         }
712 }
713
714 // close the connection
715 try {
716     input.close();
717     out.close();
718     socket.close();
719 } catch (IOException i) {
720     System.out.println(i);
721 }
722 }
723
724     public static void main(String args[])
725     {
726         Client client = new Client("10.33.2.89", 5000);
727
728     }
729 }
```

Java program for a Server

```
730 // A Java program for a Server
731 import java.net.*;
732 import java.io.*;
733 import java.util.Random;
734
735 public class Server
736 {
737     //initialize socket and input stream
738     private Socket       socket = null;
739     private ServerSocket server = null;
740     private Socket       socketsent = null;
741     private DataInputStream in   = null;
742     private DataOutputStream out = null;
743     int clientid , myid ;
744
745     // constructor with port
746     public Server(int port)
747     {
748         // starts server and waits for a connection
749         try
750         {
751             //10.33.2.89
752             myid = 345239;
753             server = new ServerSocket(port);
754             System.out.println("Server started");
755
756             System.out.println("Waiting for a client ...");
757
758             socket = server.accept();
759             System.out.println("Client accepted");
760             socketsent = new Socket("10.33.2.90",5000);
761
762             // takes input from the client socket
763             in = new DataInputStream(
764                     new BufferedInputStream(socket.
                            getInputStream()));
765             out = new DataOutputStream(
766                     socketsent.getOutputStream());
767
768             Person [] person =  new Person[4];
769             person[0] = new Person("Olin" ,"olinone","
                    123456", 500000 );
770             person[1] =   new Person("Anika", "anika", "123
```

```
                ", 11000100);
771             person[2] =    new Person("Ryana","lali", "
                    fardin",10000);
772             person[3] =   new Person("Fardin","magu", "
                    halamadrid",80000);
773
774             String line = "";
775
776             // reads message from client until "Over" is
                    sent
777     while (true)
778     {
779         try
780         {
781
782             line = in.readUTF();
783             String username = line;
784             System.out.println(line);
785             line = in.readUTF();
786             String pass = line;
787             System.out.println(line);
788             boolean credok = false;
789             for(int i=0;i<4;i++)
790             {
791                 int totmoney = 0;
792                 if(person[i].check(username,pass))
793                 {
794                     credok = true;
795                     out.writeUTF("ok");
796                     do {
797                         line = in.readUTF();
798                         System.out.println(line);
799                         if(line.equals("1"))
800                         {
801                             out.writeUTF("Your current
                                balance is : " + person[i].
                                balance);
802                         }
803                         if(line.equals("2"))
804                         {
805                             myid++;
806                             line = in.readUTF();
807                             clientid = new Integer(line);
808                             out.writeUTF(""+myid);
809
```

```
                    line = in.readUTF();
                    int money = new Integer(line);
                    int err;
                    if(money<=person[i].balance && totmoney
                        +money<=20000)
                    {

                        do {
                            out.writeUTF("1*"+clientid);

                            line = in.readUTF();
                            Random rd = new Random(); //
                                creating Random object
                            System.out.println(rd.nextInt()
                                );
                            err = Math.abs(rd.nextInt())
                                %100;
                            System.out.println("error = "+
                                err);
                            if(line.charAt(0) == 'm')
                            {
                                continue;
                            }
                            int x = new Integer(line);
                            System.out.println("error = "+
                                err);
                            if(x==myid)
                            {
                                if(err<30)
                                {
                                    out.writeUTF("1");
                                    break;
                                }
                                else {
                                    out.writeUTF("0");
                                }
                            }
                        }while (true);
                        person[i].balance-=money;
                        out.writeUTF("Your current balance
                            is : "+person[i].balance);
                        totmoney+=money;
                    }
                    else
                    {
```

```
                        out.writeUTF("0");
                    }
                }
                if(line.equals("3"))
                {
                    line = in.readUTF();
                    int money = new Integer(line);
                    System.out.println(money);
                    person[i].balance+=money;
                    out.writeUTF("1");
                    out.writeUTF("Your current balance is :
                        " + person[i].balance);
                }
                if(line.equals("0"))
                {
                    break;
                }
            }while (true);

        }
    }
    if(!credok)
            {
                System.out.println("Username or password is
                    invalid");
                out.writeUTF("not ok");
            }
        }
        catch(IOException i) {
            System.out.println(i);
            break;
        }
    }
    System.out.println("Closing connection");
    // close connection
    socket.close();
    socketsent.close();
    in.close();
    out.close();
}
catch(IOException i)
{
    System.out.println(i);
}
}
```

```
891
892         public static void main(String args[])
893         {
894             Server server = new Server(5000);
895         }
896 }
```

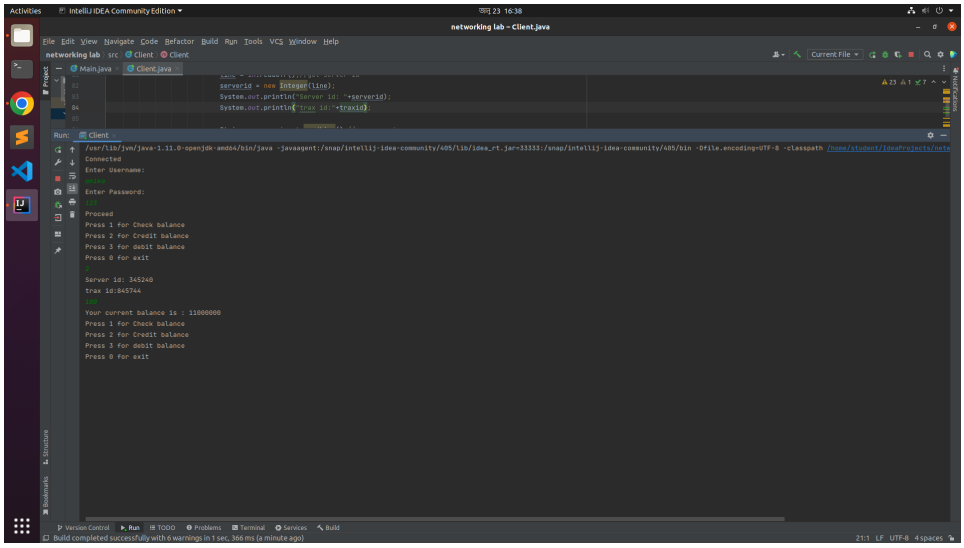

Figure 7: Content of Client

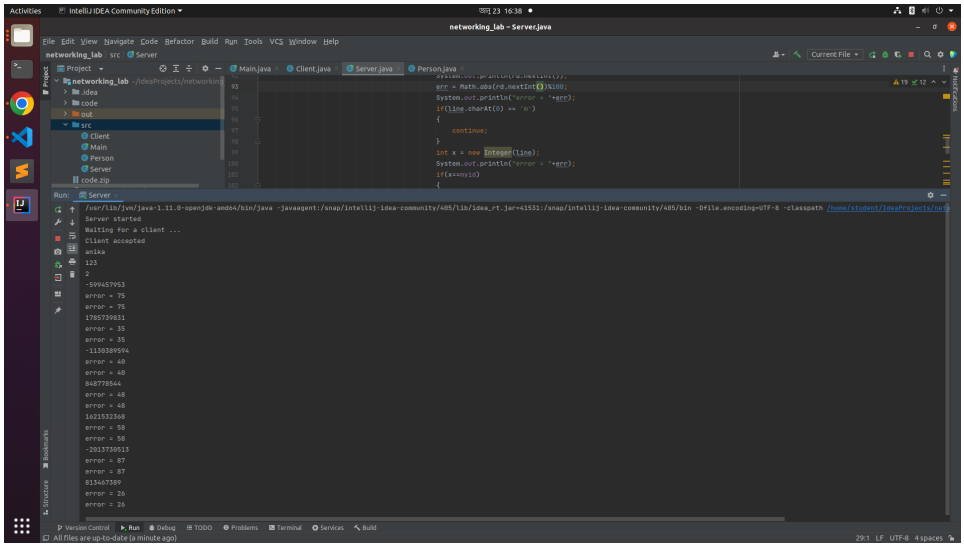

Figure 8: Content of Server

5 Experience

1. We had to see some examples of how to use Server-Client package in Java
2. We had to compile both of them on two different terminals or tabs
3. We had to run the Server program first. Then run the Client program
4. Then we had to type messages in the Client Window which will be received and shown by the Server Window simultaneously.
5. Then we had to type Over to end the program.

References

[1] Socket programming in java. *GeeksforGeeks*, may 31 2016. [Online; accessed 2023-01-24].

[2] Establishing the two-way Communication between Server and Client in Java. *GeeksforGeeks*, oct 14 2019. [Online; accessed 2023-01-24].