ASSIGNMENT-1a

Aim and Objective:

To develop any distributed application through implementing client-server communication programs based on Java Sockets.

Tools / Environment:

Java Programming Environment, jdk 1.8, Eclipse IDE.

CODES

Server.java

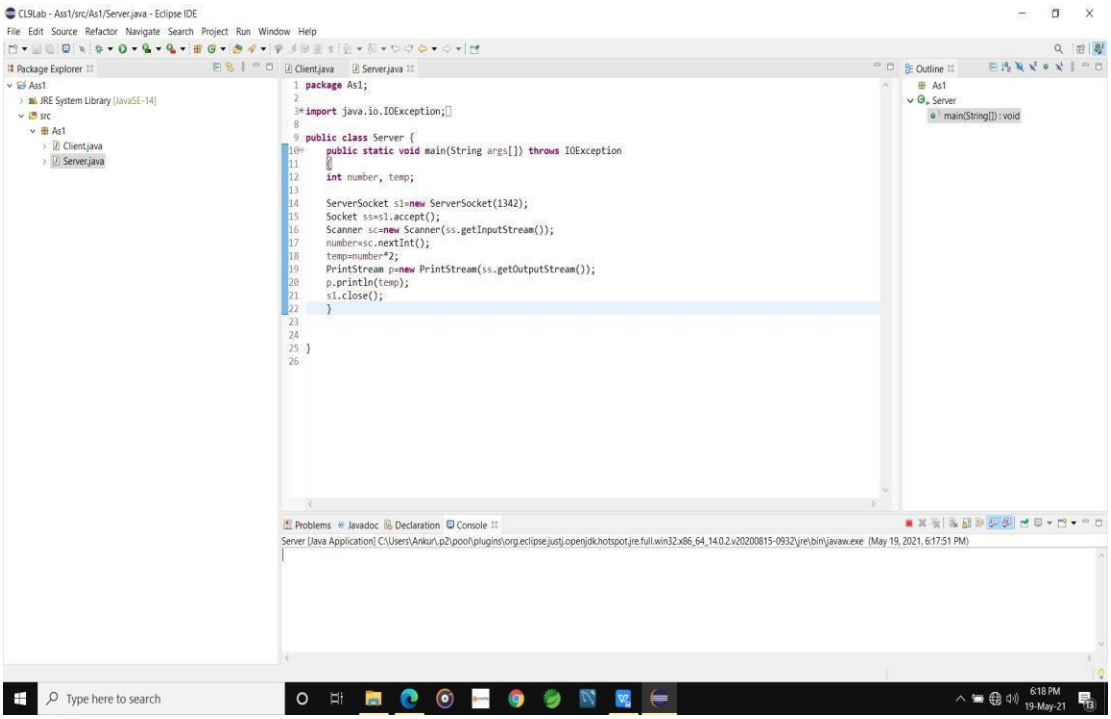
```
package Ass1;

import java.io.IOException;
import java.io.PrintStream;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.Scanner;

public class Server {
	public static void main(String args[]) throws IOException
	{
	int number, temp;

	ServerSocket s1=new ServerSocket(1342);
	Socket ss=s1.accept();
	Scanner sc=new Scanner(ss.getInputStream());
	number=sc.nextInt();
	temp=number*2;
	PrintStream p=new PrintStream(ss.getOutputStream());
	p.println(temp);
	s1.close();
	}
}
```

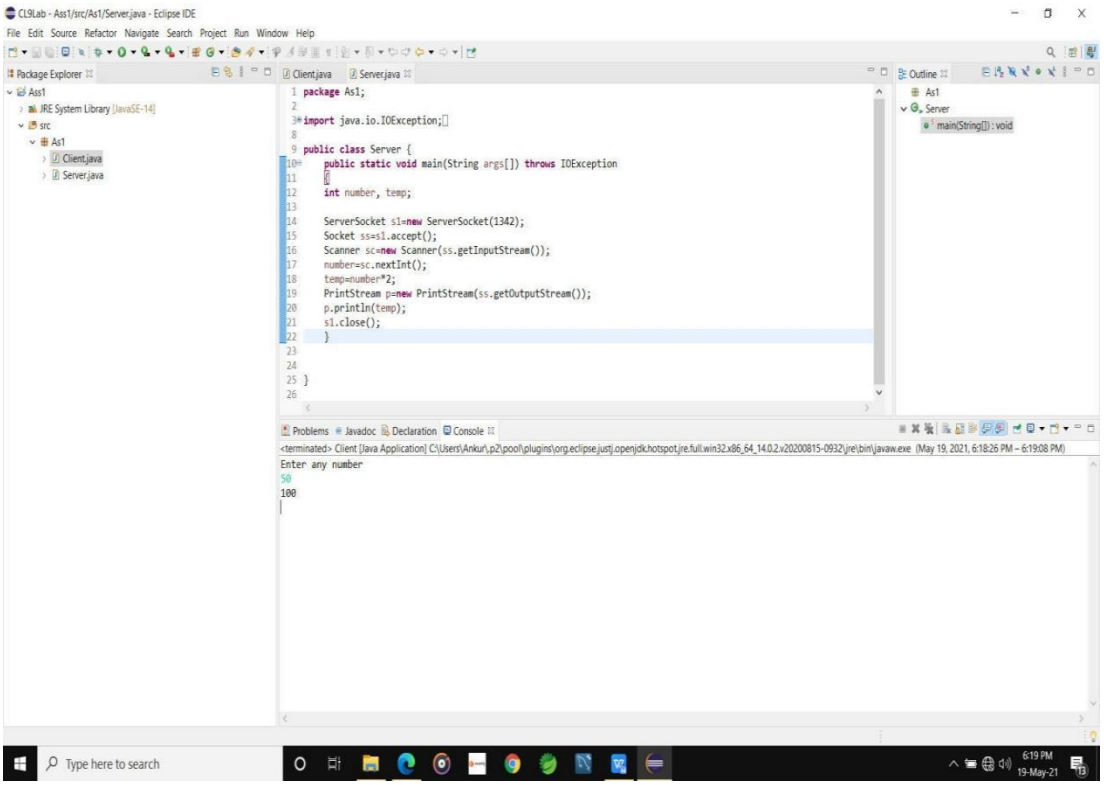
Client.java

```
package Ass1;

import java.io.IOException;
import java.io.PrintStream;
import java.net.Socket;
import java.net.UnknownHostException;
import java.util.Scanner;

public class Client {
	public static void main(String args[]) throws UnknownHostException ,IOException
	{
	int number,temp;
	Scanner sc =new Scanner(System.in);
```

```
        Socket s =new Socket("127.0.0.1",1342);
        Scanner sc1 =new Scanner(s.getInputStream());
        System.out.println("Enter any number");
        number =sc.nextInt();
        PrintStream p =new PrintStream(s.getOutputStream());
        p.println (number) ;
        temp=sc1.nextInt();
        System.out.println(temp);
        sc.close();
        s.close();
        }
}
```

Outputs

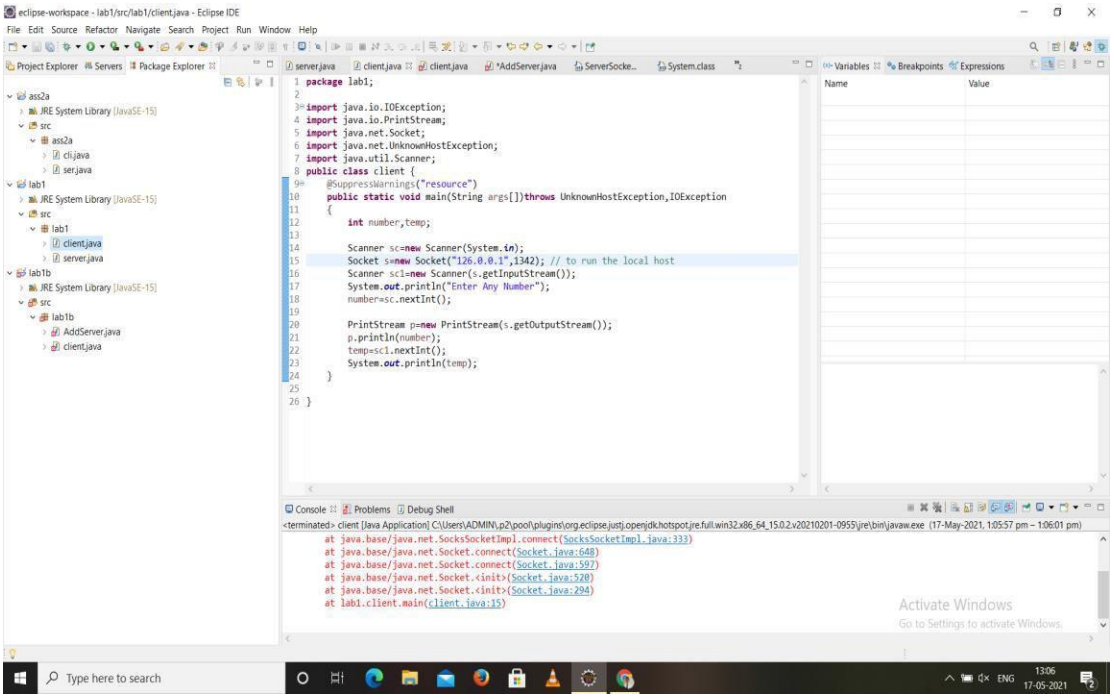




To run the code
First select project >build automatically
Then click on server program and run
We get output

//to create a program select
i)window>perpespective/if not present select other >newperpestive>jave
ii)click on file >new>project>select wizard >javaproject>project name>next>createproject>no>finish.
iii) right click on src file>new>package>lab>finish.
iv) right click on lab>new>class>name .server>finish;
in same way create client also
v) right click on lab>new>class>name.client>finish;

when we take 126 as input instead of 127



CalCulator finction

```
importjava.io.DataInputStream;
                        import java.io.DataOutputStream;
                        import java.io.IOException;
                        import java.io.PrintWriter;
                        import java.net.ServerSocket;
                        import java.net.Socket;

                        public class AdditionServer {
                               public static void main(String []arg)
                        throws IOException{


                                        try{

                                                while(true){
                                                        ServerSocket
                        serverSocket = new ServerSocket(8689);
                                                        Socket sc =
                        serverSocket.accept();
                                                        try{

                               System.out.println("Getting Addition
                        Request.. ");
                                                        DataInputStream
                        inFromClient = new
                         DataInputStream(sc.getInputStream());
                                                         int first =
                        inFromClient.readInt();
                                                        int second =
```

```
                                    inFromClient.readInt();
                                                PrintWriter pr = new
                                    PrintWriter(sc.getOutputStream(),true);
                                                int
                                    answer=first+second ;

                                                DataOutputStream
                                    outFromClient = new
                                    DataOutputStream(sc.getOutputStream());


                                          outFromClient.writeUTF("Answer of
                                    Addition operation is :"+answer);
                                                }
                                                catch(Exception e){

                                          e.printStackTrace();
                                                }
                                                finally{
                                                      sc.close();

                                          serverSocket.close();
                                                }
                                                }
                                          }
                                          catch(Exception ex){
                                                ex.printStackTrace();
                                          }

                                    }
                              }
```

Client.java

```
      import java.io.DataInputStream;
      import java.io.DataOutputStream;
      import java.io.IOException;
      import java.net.Socket;
      import java.net.UnknownHostException;
      import java.util.Scanner;

      public class Client {
            public static void main(String arg[]) throws UnknownHostException,
      IOException{

                  try{

                        boolean isContinue =true;

                        while(isContinue)
                        {
                              //Asking user for operation Selection
                              System.out.println("Which operation do you
      want to perform ? Please enter index number: ");
                              boolean isValid =false;
                              Scanner scanner = new Scanner(System.in);
                              String selected= null;
                              while(!isValid){
```

```
                                    System.out.println("1. Addition");
                                    System.out.println("2. Subtraction");
                                    System.out.println("3.
Multiplication");
                                    System.out.println("4.  Division");

                                    int selection = scanner.nextInt();

                                    switch(selection){

                                     case 1 :
                                           selected ="add";
                                           isValid=true; break;
                                     case 2 :
                                           selected ="sub";
                                           isValid=true; break;
                                     case 3 :
                                           selected ="mul";
                                           isValid=true; break;
                                     case 4 :
                                           selected ="div";
                                           isValid=true; break;
                                     default :
                                           System.out.println("Invalid

                                           isValid=false;
                                           break;
                                    }
choice : ");                }

                            //Asking two parameters for selected
operations
                            System.out.println("Please enter two
parameters for this operation:");
                            System.out.println("Param 1 :");
                            int first = scanner.nextInt();
                            scanner.nextLine();
                            System.out.println("Param 2 :");
                            int second = scanner.nextInt();

                            //Connecting with Naming Server.
                            Socket opSocket = new
Socket("134.154.76.104", 8694);
                            DataOutputStream opToClient = new
DataOutputStream(opSocket.getOutputStream());
                            opToClient.writeInt(first);
                            opToClient.writeInt(second);
                            opToClient.writeUTF(selected);

                            //Getting ans from Naming Server
                            DataInputStream opInStream = new
DataInputStream(opSocket.getInputStream());
                            String ans = opInStream.readUTF();
```

```
                              //Printing Answer and asking for
      continuation
                              System.out.println(ans);
                              System.out.println("Enter 1 if you want to
      continue or 0 to exit :");
                              int operation = scanner.nextInt();
                              if(operation==0)
                                    isContinue=false;
                        }
                        System.out.println("Exit performed successfully.");
                  }
                  catch(Exception ex){
                        ex.printStackTrace();
                  }
                  finally{
                        //opSocket.close();
                  }
            }

      }
```

Output

```
Enter the equation in the form: 'operand operator operand'
5 * 6
Answer=30
Enter the equation in the form: 'operand operator operand'
5 + 6
Answer=11
Enter the equation in the form: 'operand operator operand'
9 / 3

Answer=3
```

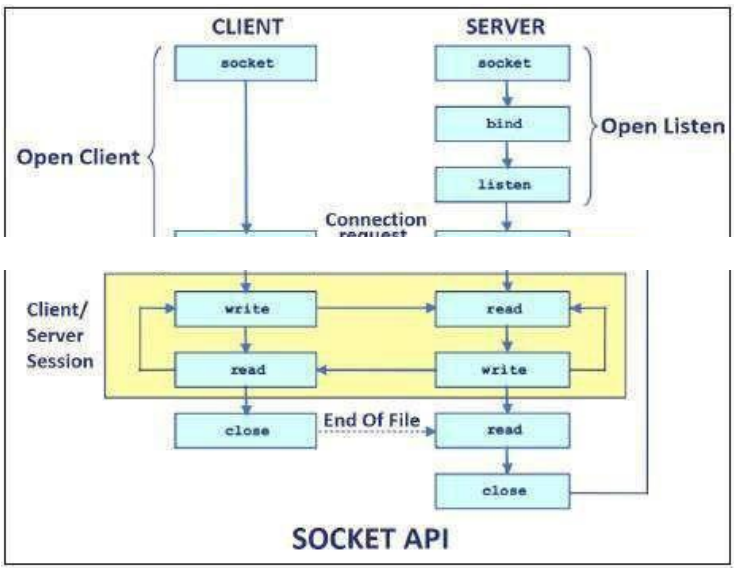
Related Theory:

Socket: In distributed computing, network communication is one of the essential parts of any system, and the socket is the endpoint of every instance of network communication. In Java communication, it is the most critical and basic object involved. A socket is a handle that a local program can pass to the networking API to connect to another machine. It can be defined as the terminal of a communication link through which two programs /processes/threads running on the network can communicate with each other. The TCP layer can easily identify the application location and access information through the port number assigned to the respective sockets. During an instance of communication, a client program creates a socket at its end and tries to connect it to the socket on the server. When the connection is made,

the server creates a socket at its end and then server and client communication is established.

The java.net package provides classes to facilitate the functionalities required for networking. The socket class programmed through Java using this package has the capacity of being independent of the platform of execution; also, it abstracts the calls specific to the operating system on which it is invoked from other Java interfaces. The ServerSocket class offers to observe connection invocations, and it accepts such invocations from different clients through another socket. High-level wrapper classes, such as URLConnection and URLEncoder, are more appropriate. If you want to establish a connection to the Web using a URL, then these classes will use the socket internally. The following steps occur when establishing a TCP connection between two computers using sockets –

- The server instantiates a ServerSocket object, denoting which port number communication is to occur on.
- The server invokes the accept() method of the ServerSocket class. This method waits until a client connects to the server on the given port.
- After the server is waiting, a client instantiates a Socket object, specifying the server name and the port number to connect to.
- The constructor of the Socket class attempts to connect the client to the specified server and the port number. If communication is established, the client now has a Socket object capable of communicating with the server.
- On the server side, the accept() method returns a reference to a new socket on the server that is connected to the client's socket.
- After the connections are established, communication can occur using I/O streams. Each socket has both an OutputStream and an InputStream. The client's OutputStream is connected to the server's InputStream, and the client's InputStream is connected to the server's OutputStream.
- TCP is a two-way communication protocol, hence data can be sent across both streams at the same time. Following are the useful classes providing complete set of methods to implement sockets.



Conclusion:

In this assignment, the students learned about client-server communication through different protocols and sockets. They also learned about Java support through the socket API for TCP and UDP programming. Also implemented TCP is a two-way communication protocol, hence data can be sent across both streams at the same time. Following are the useful classes providing complete set of methods to implement socket