Module 2
Lab Assignment 2

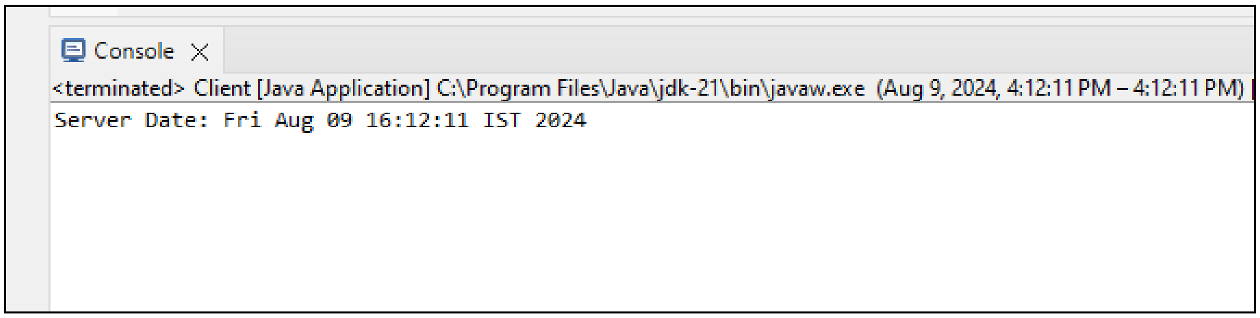
Aim: 1.Write a program to Implement a Date time server containing date( ) and time( ) using UDP.

Theory:
A UDP date and time server provides clients with the current date or time upon request using the User Datagram Protocol (UDP). It operates in a connectionless manner, allowing for fast, simple communication without delivery guarantees. Clients send requests (e.g., "date" or "time") to the server, which responds with the requested information. This approach is efficient for applications where speed is prioritized over reliability.

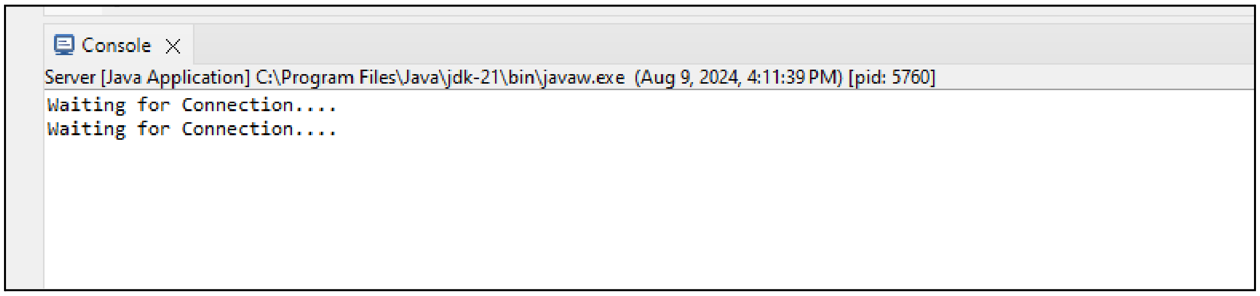
Code:
```
package assignment2;
import java.io.DataOutputStream;
import java.net.*;
import java.net.*;
import java.util.Date;
public class Server {
        public static void main(String[] args) throws Exception {
                // TODO Auto-generated method stub
ServerSocket ss = new ServerSocket(1234);
while(true) {
        System.out.println("Waiting for Connection....");
        Socket soc = ss.accept();
        DataOutputStream out = new DataOutputStream(soc.getOutputStream());
        out.writeBytes("Server Date: "+(new Date()).toString()+"\n");
        out.close();
        soc.close();
}
        }
}
```

```
package assignment2;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.InetAddress;
import java.net.Socket;
public class Client {
        public static void main(String[] args) throws Exception {
                // TODO Auto-generated method stub
Socket soc= new Socket(InetAddress.getLocalHost(),1234);
BufferedReader in = new BufferedReader(new InputStreamReader(soc.getInputStream()));
System.out.println(in.readLine());
        }
}
```

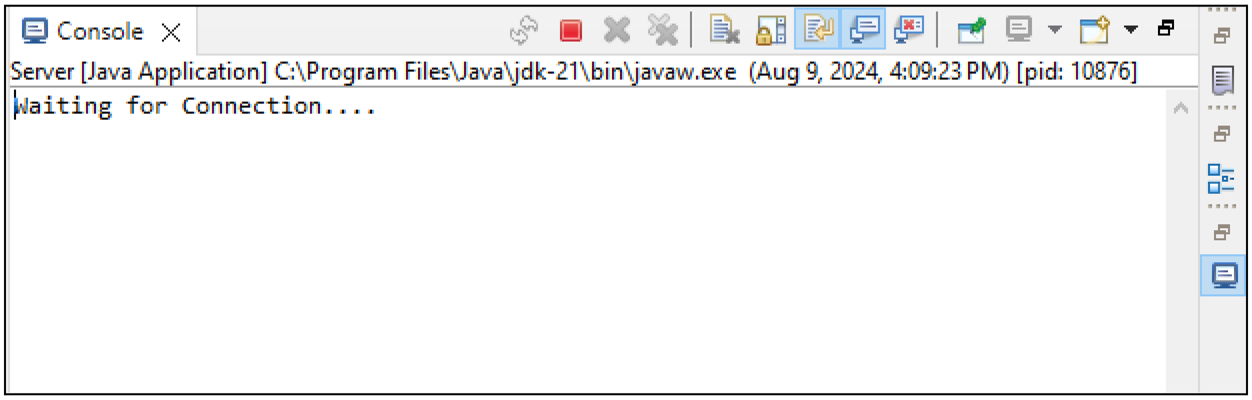
Output:



```
<terminated> Client [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (Aug 9, 2024, 4:12:11 PM – 4:12:11 PM)
Server Date: Fri Aug 09 16:12:11 IST 2024
```



```
Server [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (Aug 9, 2024, 4:11:39 PM) [pid: 5760]
Waiting for Connection....
Waiting for Connection....
```



```
Server [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (Aug 9, 2024, 4:09:23 PM) [pid: 10876]
Waiting for Connection....
```

Aim: 2.Write a program to implement a Server calculator containing ADD( ) , MUL( ) , SUB( ) , DIV( ). Implement using RPC(UDP).

Theory:
A server calculator using RPC over UDP allows clients to perform basic arithmetic operations (addition, subtraction, multiplication, division) by sending requests to the server. The server processes these requests using UDP for fast, connectionless communication. Each operation is handled as a remote procedure call, where the server computes the result and sends it back to the client. This approach leverages UDP's low overhead while simplifying network interactions.

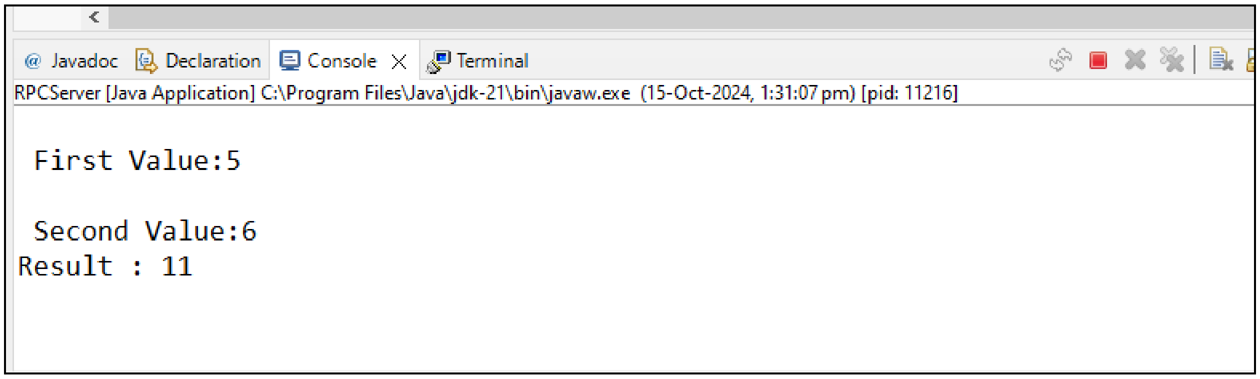
Code:
RPCServer.java
```
package siddhi;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.util.StringTokenizer;
public class RPCServer {
        DatagramSocket ds;
        DatagramPacket dp;
        String str, methodName,result;
        int val1,val2;
        RPCServer(){
                try {
                        ds = new DatagramSocket(1200);
                        byte b[] = new byte[4096];
                        while(true) {
                                dp = new DatagramPacket(b,b.length);
                                ds.receive(dp);
                                str = new String(dp.getData(),0,dp.getLength());
                                if(str.equalsIgnoreCase("quit")) {
                                        System.exit(1);
                                }
                                else {
                                StringTokenizer st = new StringTokenizer(str," ");
                                        int i =0;
                                        while(st.hasMoreTokens()) {
                                                String token = st.nextToken();
                                                methodName = token;
                                        val1 = Integer.parseInt(st.nextToken());
                                        val2 = Integer.parseInt(st.nextToken());
                                        }
                                }
                        System.out.println("\nClient Selected\""+str+"\"Method:");
                                System.out.println("\n First Value:"+val1);
                                System.out.println("\n Second Value:"+val2);
                                InetAddress ia = InetAddress.getLocalHost();
                                if(methodName.equalsIgnoreCase("add"))
                                        result=" " + add(val1,val2);
                                else if(methodName.equalsIgnoreCase("sub"))
                                result = "" + sub(val1,val2);
                                else if(methodName.equalsIgnoreCase("mul"))
                                result = "" + mul(val1,val2);
                                else if(methodName.equalsIgnoreCase("div"))
                                result = "" + div(val1,val2);
                                byte b1[] = result.getBytes();
                                DatagramSocket ds1 = new DatagramSocket();
                                DatagramPacket dp1 = new
DatagramPacket(b1,b1.length,InetAddress.getLocalHost(),1400);
                                System.out.println("Result :"+result+"\n");
                                ds1.send(dp1);

                        }
                }
                catch(Exception e) {
                        e.printStackTrace();
                }
        }
        public int add(int val1,int val2) {
                return val1 + val2;
        }
        public int sub(int val1,int val2) {
                return val1 - val2;
        }
        public int mul(int val1,int val2) {
                return val1 * val2;
        }
        public int div(int val1,int val2) {
                return val1 / val2;
        }
        public static void main(String[] args) {
                // TODO Auto-generated method stub
                new RPCServer();
        }
}
```

RPCClient.java

```
package siddhi;
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
public class RPCClient {
        RPCClient()
        {
        try
        {
        InetAddress ia = InetAddress.getLocalHost();
        DatagramSocket ds = new DatagramSocket();
        DatagramSocket ds1 = new DatagramSocket(1800);
        System.out.println("----------\n");
        System.out.println("Enter Method Name with Parameter like add 3 4\n"); while (true)
        {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in)); String
str = br.readLine();
        byte b[] = str.getBytes();
        DatagramPacket dp = new DatagramPacket(b,b.length,ia,1200); ds.send(dp);
        dp = new DatagramPacket(b,b.length);
        ds1.receive(dp);
        String s = new String(dp.getData(),0,dp.getLength());
        System.out.println("\nResult = " + s + "\n");
        System.out.println("\n\nEnter Method Name with Parameter like add 3 4\n"); }
        }
        catch (Exception e)
        {
        e.printStackTrace();
        }
        }
        public static void main(String[] args) {
                // TODO Auto-generated method stub
                 new RPCClient();
        }
}
```

Output:



```
RPCServer [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe  (15-Oct-2024, 1:31:07 pm) [pid: 11216]

 First Value:5

 Second Value:6
Result : 11
```

Aim: 3.
A. Implement a Date Time Server containing date() and time() using Socket.
B. Implement a Date Time Server containing date() and time() using Datagram.
Theory:
A Date Time Server provides clients with current date and time information through network communication.

1. **Using Sockets:** In a socket-based implementation (TCP), the server establishes a connection with clients, allowing reliable data transfer. Clients can request the date or time, and the server responds with the requested information.
2. **Using Datagram:** In a datagram-based implementation (UDP), the server sends date and time information without establishing a persistent connection. This method is faster and suitable for applications where speed is prioritized over reliability, allowing clients to send requests and receive responses independently.

Code:
A. Implement a Date Time Server containing date() and time() using Socket.
Server_Time.java
```
package siddhi;
import java.io.*;
import java.net.*;
import java.util.Date;
public class Server_Time
{
  public static void main(String args[]) throws Exception
  {
     ServerSocket s=new ServerSocket(1234);
     while(true)
     {
        System.out.println("Waiting For Connection ...");
        Socket soc=s.accept();
        DataOutputStream out=new DataOutputStream(soc.getOutputStream());
        out.writeBytes("Server Date: " + (new Date()).toString() + "\n");
        out.close();
        soc.close();
     }
  }
}
```

Client_Time.java
```
package siddhi;
import java.io.*;
import java.net.*;
public class Client_Time
{
  public static void main(String args[]) throws Exception
  {
     Socket soc = new Socket(InetAddress.getLocalHost(),1234);
     BufferedReader in=new BufferedReader(new InputStreamReader(soc.getInputStream()  ));
     System.out.println(in.readLine());
  }
}
```

B. Implement a Date Time Server containing date() and time() using Datagram.
DateServer.java

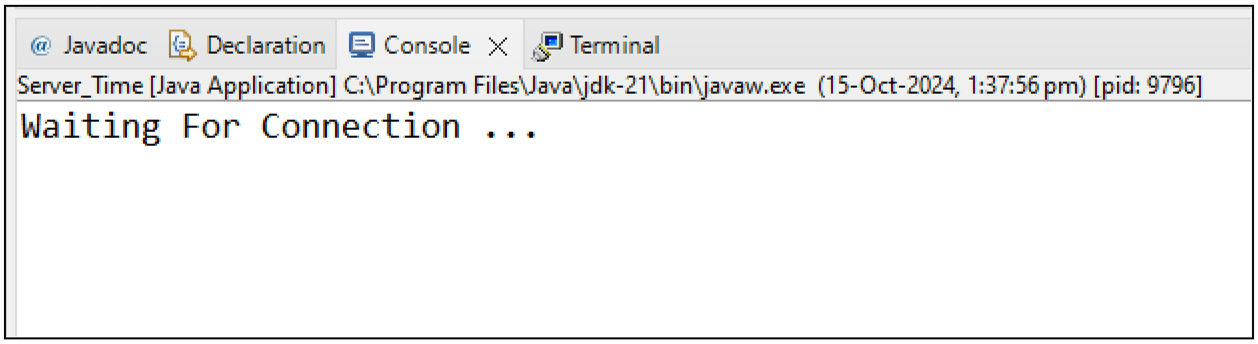
```
package siddhi;
import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.text.SimpleDateFormat;
import java.util.Date;
public class DateServer {
  public static void main(String args[]) throws Exception {
     DatagramSocket socket = new DatagramSocket(1234);
     System.out.println("Date Time Server is running...");
     byte[] buffer = new byte[4096];
     while (true) {
        DatagramPacket packet = new DatagramPacket(buffer, buffer.length);
        socket.receive(packet);
        String request = new String(packet.getData(), 0, packet.getLength());
        String response;
        if (request.equalsIgnoreCase("date")) {
           response = new SimpleDateFormat("yyyy-MM-dd").format(new Date());
        } else if (request.equalsIgnoreCase("time")) {
           response = new SimpleDateFormat("HH:mm:ss").format(new Date());
        } else {
           response = "Invalid request. Please send 'date' or 'time'.";
        }
        InetAddress clientAddress = packet.getAddress();
        int clientPort = packet.getPort();
        socket.send(new DatagramPacket(response.getBytes(), response.length(),
clientAddress, clientPort));
     }
  }
}
```

DateClient.java

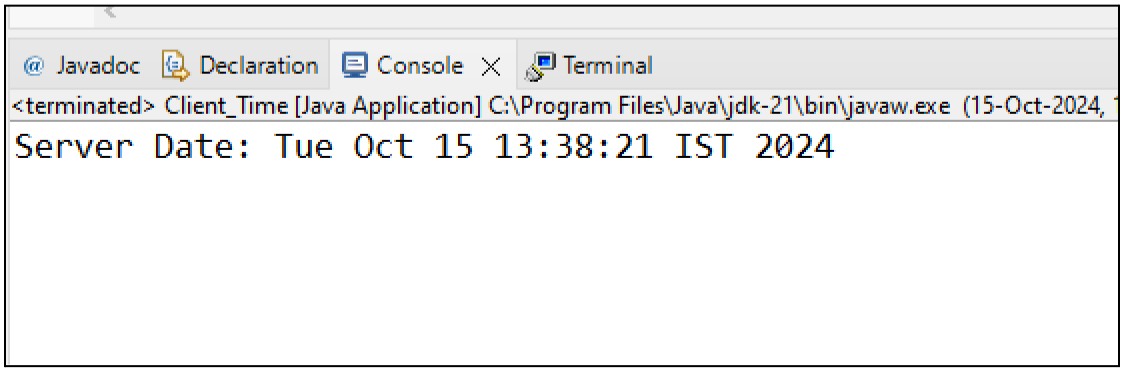
```
package siddhi;
import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.util.Scanner;
public class DateClient {
  public static void main(String args[]) throws Exception {
      DatagramSocket socket = new DatagramSocket();
      InetAddress serverAddress = InetAddress.getByName("localhost");
      Scanner scanner = new Scanner(System.in);
      while (true) {
         System.out.print("Enter 'date' or 'time' (or 'quit' to exit): ");
         String request = scanner.nextLine();
         if (request.equalsIgnoreCase("quit")) break;
         socket.send(new DatagramPacket(request.getBytes(), request.length(), serverAddress,
1234));
         byte[] responseBuffer = new byte[4096];
         DatagramPacket responsePacket = new DatagramPacket(responseBuffer,
responseBuffer.length);
         socket.receive(responsePacket);
         System.out.println("Response from server: " + new String(responsePacket.getData(), 0,
responsePacket.getLength()));
      }
      scanner.close();
      socket.close();
  }
}
```

Output:

A. Implement a Date Time Server containing date() and time() using Socket.

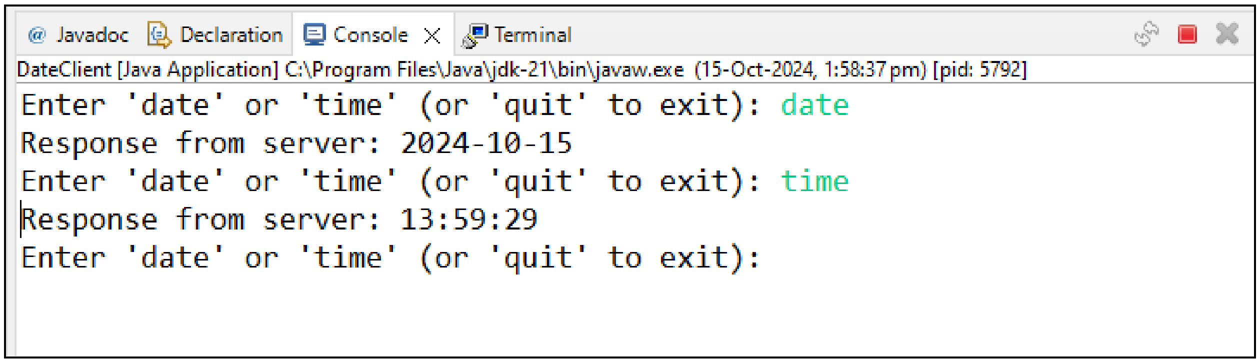


```
Server_Time [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe  (15-Oct-2024, 1:37:56 pm) [pid: 9796]
Waiting For Connection ...
```



```
<terminated> Client_Time [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe  (15-Oct-2024,
Server Date: Tue Oct 15 13:38:21 IST 2024
```

B. Implement a Date Time Server containing date() and time() using Datagram.

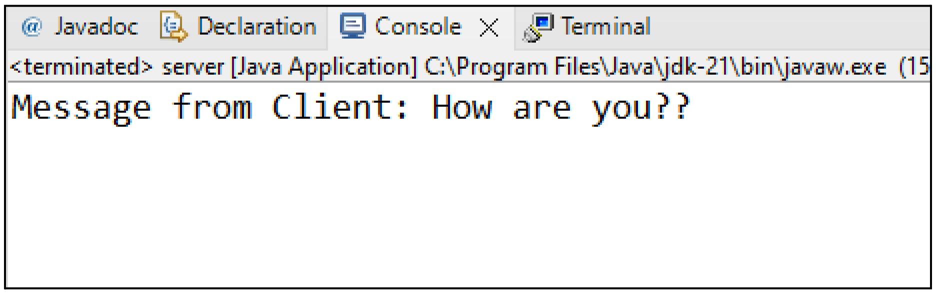


```
DateClient [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe  (15-Oct-2024, 1:58:37 pm) [pid: 5792]
Enter 'date' or 'time' (or 'quit' to exit): date
Response from server: 2024-10-15
Enter 'date' or 'time' (or 'quit' to exit): time
Response from server: 13:59:29
Enter 'date' or 'time' (or 'quit' to exit):
```

Aim: 4. Write a program for one-way client and server communication using Datagram Socket.
Theory:
One-way client-server communication using Datagram Sockets involves the transmission of data from a client to a server without expecting a response. This method employs the User Datagram Protocol (UDP), which is connectionless and allows for fast data transfer with minimal overhead. The client sends a message to the server using a DatagramPacket, while the server listens for incoming packets on a specified port. Once the server receives the data, it can process it as needed, but it does not send any acknowledgment or response back to the client
Code:
server.java
```
package siddhi;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
public class server {
	public static void main(String[] args) throws Exception{
		// TODO Auto-generated method stub
		DatagramSocket ds=new DatagramSocket(2221);
		byte[] b=new byte[1024];
		DatagramPacket p=new DatagramPacket(b,1024);
		ds.receive(p);
		String msg = new String(p.getData(),0,p.getLength());
		System.out.println("Message from Client: "+msg);
	}
}
```

client .java
```
package siddhi;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
public class client {
	public static void main(String[] args) throws Exception{
		// TODO Auto-generated method stub
		String s="How are you??";
		DatagramSocket ds=new DatagramSocket();
		InetAddress ip=InetAddress.getByName("localhost");
		DatagramPacket p=new DatagramPacket(s.getBytes(),s.length(),ip,2221);
		ds.send(p);
	}
}
```

Output:



```
<terminated> server [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (15
Message from Client: How are you??
```