RAMAIAH UNIVERSITY OF APPLIED SCIENCES

ASSIGNMENT

Course Code	19CSC313A
Course Name	Distributed and Cloud Computing
Programme	B. Tech.
Department	Computer Science and Engineering
Faculty	FET

Name of the Student	Deepak R
Reg. No	18ETCS0020041
Semester/Year	6th /2018
Course Leader/s	Jishmi Jos Choondal

Declaration Sheet			
Student Name	Deepak R		
Reg. No	18ETCS002041		
Programme	B. Tech.	Semester/Year	6th /2018
Course Code	19CSC313A		
Course Title	Distributed and Cloud Computing		
Course Date		to	
Course Leader	Jishmi Jos Choondal		

Declaration

The assignment submitted herewith is a result of my own investigations and that I have conformed to the guidelines against plagiarism as laid out in the Student Handbook. All sections of the text and results, which have been obtained from other sources, are fully referenced. I understand that cheating and plagiarism constitute a breach of University regulations and will be dealt with accordingly.

Signature of the Student		Date	
Submission date stamp (by Examination & Assessment Section)			

Signature of the Course Leader and date	Signature of the Reviewer and date

Faculty of Engineering & Technology			
Ramaiah University of Applied Sciences			
Department	Computer Science and Engineering	**Programme**	B.Tech in CSE
Semester/Batch	06/2019		
Course Code	19CSC313A	**Course Title**	Distributed and Cloud Computing
Course Leader(s)	Jishmi , Prakash, Chaitra		

Assignment-1			
Register No.	18ETCS002041	Name of Student	Deepak R

Sections		**Marking Scheme**	**Max Marks**	**First Examiner Marks**	**Second Examiner Marks**
Part-A	A.1	Need for Distributed Systems in Modern Computing	03		
	A.2	Select any conference paper on any one distributed system application and give a summary	07		
	A.3	Stance taken and justification with the support of conference paper	05		
		Part-A Max Marks	**15**		
Part-B B.1	B.1.1	Design algorithm(s)/flowchart(s) for theproblem	04		
	B.1.2	Implementation	04		
	B.1.3	Results and Analysis	02		
		B.1 Max Marks	**10**		

Course Marks Tabulation				
Component- 1(B) Assignment	**First Examiner**	**Remarks**	**Second Examiner**	**Remarks**
A				
B1				
Marks (out of 25)				
Signature of First Examiner				Signature of Second Examiner

Assignment-1

Instructions to students:

1. The assignment consists of **2** questions: Part A –**1** Question, Part B- **1**Question.
2. Maximum marks are **25**.
3. The assignment has to be neatly word processed as per the prescribed format.
4. The maximum number of pages should be restricted to **10**.
5. The printed assignment must be submitted to the course leader.

   Submission Date: 29th May 2021

7. **Submission after the due date is not permitted.**
8. **IMPORTANT**: It is essential that all the sources used in preparation of the assignment must be suitably referenced in the text.

Marks will be awarded only to the sections and subsections clearly indicated as per the problem statement/exercise/question

Solution to Part A

"Is Distributed Systems Knowledge Essential for Modern Computing Professionals?"

Solution for A1.1 Need for Distributed Systems in Modern Computing

Today's applications are marvels of distributed systems development. Each function or service that makes up an application may be executing on a different system, based upon a different system architecture, that is housed in a different geographical location, and written in a different computer language. Components of today's applications might be hosted on a powerful system carried in the owner's pocket and communicating with application components or services that are replicated in data centers all over the world.

We need distributed systems in Modern Computing for several reasons:

Fault tolerance, in the event of machine failures or data center failures or entire region wipeout due to natural disasters, you want to preserve user's information

Reduce Latency, Users are spread all over the world and you want to provide them best user experience by reducing the response time.

Availability, in the event when one machine is slow or down, another nearby machine should be able to serve the request to avoid any request failure.

Load Balancing etc.

What's amazing about this, is that individuals using these applications typically are not aware of the complex environment that responds to their request for the local time, local weather, or for directions to their hotel.

Solution for A1.2 Select any conference paper on any one distributed system application and give a summary.

Selected Conference Paper - Mona Shah et al, / (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 6 (3) , 2015, 3010-3013.

Role of Distributed Systems in Data Mining

Summary of What I understood by Reading above Stated Paper.

Distribution of data and computation allows for solving larger problems and execute applications that are distributed in nature. Data mining technology has emerged as a means for identifying patterns and trends from large quantities of data. The Data Mining technology normally adopts data integration method to generate Data warehouse, on which to gather all data into a central site, and then run an algorithm against that data to extract the useful Module Prediction and knowledge evaluation. Applications from various domains have adopted this technique to perform data analysis efficiently. Several issues need to be addressed when such techniques apply on data these are bulk at size and geographically distributed at various sites. The system contains modules for secure distributed communication, database connectivity, organized data management and efficient data analysis for generating a global mining model. Performance evaluation of the system is also carried out and presented. New technologies are emerging to make big data analytics possible and costeffective. The above paper described system architecture and distributed data mining, also known as multi agent based distributed data mining, in terms of significance, system overview, existing systems, and research trends.

The following depicts a **5-Step approach** on predictive maintenance for distributed systems:

• **Step 1** - Data Gathering (logging, online monitoring): This step provides the basis for building the model of the system. Observational data from the operational system is gathered either in an online or an off-line setting. Using data from the real system for model building allows for much more realistic models than a-priori static models or simulations.

• **Step 2** - Model Building: This step is based on the data mining analysis of the monitoring data. E.g., feature-selection and classification algorithms can be used to determine which parameters of the system are most correlated to failures or performance problems

• **Step 3** - Online-Monitoring and Prediction: Given that a model of the system gained in Step2 exists, selected parameters can be monitored and short term prediction on potential problems can be made on the basis of this model. Also, predictions regarding parameter settings of the distributed system become possible.

• **Step 4** - Preventive Measures: In case that the model predicts reliability or performance problems based on the actual observations, measures might be taken to prevent problems, instance system parameters might be changed or load might be reduced

• **Step 5** - Adaptation of the model): As the actual system might emerge over the time, adaptations of the model itself might become necessary. Basically, the model building process taking place in Step 2 on the data from Step 1 will have to be checked against long term changes in the actual monitored data. The model adaptation might require human interaction, but also could be foreseen automatically, for instance when re-adjusting thresholds or expected delays.

Distributed data mining (DDM) is a fast-growing area which deals with the problem of finding data patterns in an environment with distributed data and computation. In current era most of the data analysis systems require centralized storage of data, the increasing merger of computation with communication is more demand data mining environments that can utilize the full advantage of distributed computation. Reliability of distributed no longer can be assured by static design because distributed systems are increasingly large, heterogeneous and dynamic. On the one side, large-scale computing grids, clouds and clusters provide computing and data resources with hundreds or even up to thousands of nodes.

Distributed Systems today grow dynamically in terms of new applications, hardware and network components, users and workload changes. Complex interactions between the different layers of a distributed system make systems and effects of faults hard to understand such that faulty behavior and poor performance cannot always be distinguished. Our approach to reliable operation of distributed systems is based on building a dynamic model for the distributed systems from monitored system data.

A fundamental challenge for DDM is to develop mining techniques without having to communicate data unnecessarily. Such functionality is required for reasons of efficiency, accuracy and privacy. In addition, appropriate protocols, languages, and network services are required for mining distributed data to handle the required metadata and mapping.

Solution for A1.3 Stance taken and justification with the support of conference paper

By exploiting the Grid services features it is possible to develop data mining services accessible every time and everywhere. This approach may result in

• Service-based distributed data mining applications

• Data mining services for virtual organizations.

• Distributed data analysis services on demand.

• A sort of knowledge discovery eco-system formed of a large numbers of decentralized data analysis services

In this paper they have highlighted the problem of the increase in complexity, diversity and scale of data. They introduced a separation of concerns between data mining and integration (DMI) process development and the mapping, optimization and enactment of these processes. They postulated this separation of concerns will allow handling separately the user and application diversity and the system diversity and complexity issues simultaneously.

Distributed systems can accomplish the task of choosing from a pool of solutions to achieve the result and choosing the best possible solution according to the requirement. Application specific methods can be offered for the execution of a task or partition of a task in two subtasks.

They introduced an initial architecture for observing the distributed systems and algorithm executions that allows for model building and on-line monitoring based on predicting upcoming reliability and performance problems with previously generated models. Users are enabled to take preventive measures for increased reliability or performance. **Finally, they presented a concrete scenario of applying this approach in the field of distributed data mining.**

Question 2

B.1.1 Design algorithm(s)/flowchart(s) for the problem

Client Side done using UDP

```
Step 1:Create the socket object for carrying the data
    DatagramSocket ds = new DatagramSocket();
    InetAddress ip = InetAddress.getLocalHost();
    Initialize byte buf[] = null;
Step 2 :loop while user not enters "bye"
    while (true) {
      Display "Enter the equation in the format:"
      Display "'operand1 operator operand2'"
      String inp = sc.nextLine();
      buf = new byte[65535];
Step 3:convert the String input into the byte array.
      buf = inp.getBytes();
Step 4:Create the datagramPacket for sending the data.
      DatagramPacket DpSend= new DatagramPacket(buf, buf.length, ip, 1234);
Step 5:invoke the send call to actually send the data.
      ds.send(DpSend);
Step 6:break the loop if user enters "bye"
      if (inp.equals("bye")) {
        break;
      } buf = new byte[65535];
      DatagramPacket DpReceive= new DatagramPacket(buf, buf.length);
     ds.receive(DpReceive);
      Display"Answer = "+ new String(buf, 0, buf.length) }}}
Step 7:Stop
```

Server Side done using UDP

Step 1: Create a socket to listen at port 1234

```
DatagramSocket ds = new DatagramSocket(1234);
Initialize byte[] buf = null;
Initialize DatagramPacket DpReceive = null;
Initialize DatagramPacket DpSend = null;
while (true)
{buf = new byte[65535];
```

Step 2:create a DatgramPacket to receive the data.

```
    DpReceive = new DatagramPacket(buf, buf.length);
```

Step 3:receive the data in byte buffer.

```
    ds.receive(DpReceive);
    String inp = new String(buf, 0, buf.length);
```

Step 4: To remove extra spaces.

```
    inp=inp.trim();
    Display "Equation Received:- " + inp;
```

Step 5: Exit the server if the client sends "bye"

```
    if (inp.equals("bye"))
    {Display "Client sent bye.....EXITING"
        break;}
```

Step 6: Use StringTokenizer to break the equation into operand and operation

```
    StringTokenizer st = new StringTokenizer(inp);
     int oprnd1 = Integer.parseInt(st.nextToken());
    String operation = st.nextToken();
    int oprnd2 = Integer.parseInt(st.nextToken());
```

Step 7: perform the required operation.

```
                if (operation.equals("+"))
                        result = oprnd1 + oprnd2;
                else if (operation.equals("-"))
                        result = oprnd1 - oprnd2;


                else if (operation.equals("*"))
                        result = oprnd1 * oprnd2;
                else
                        result = oprnd1 / oprnd2;
                Display ("Sending the result...");
                String res = Integer.toString(result);
```

Step 8:lear the buffer after every message.

```
                buf = res.getBytes();
```

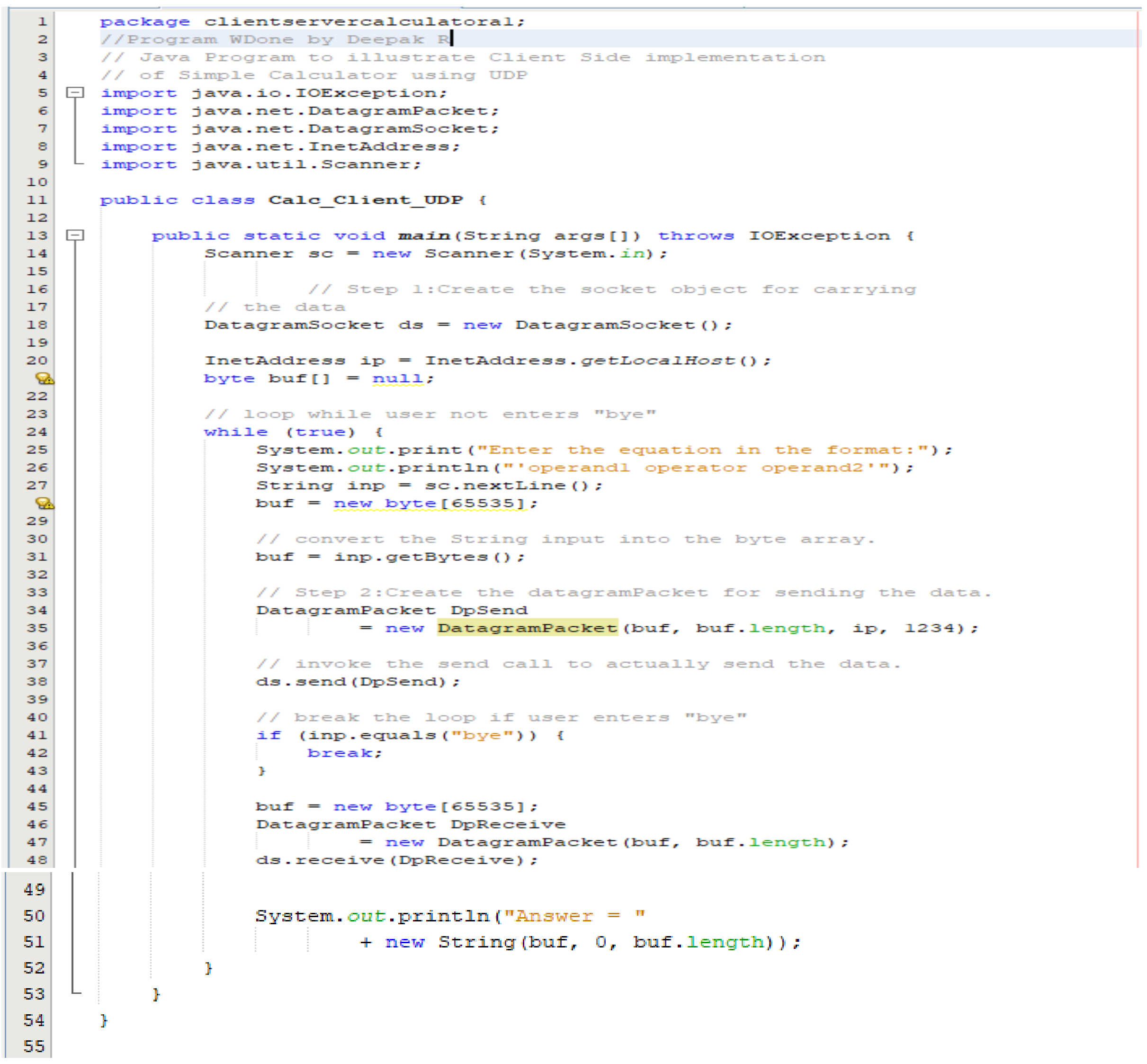
Step 9:get the port of client.

```
                int port = DpReceive.getPort();
                DpSend = new DatagramPacket(buf, buf.length,
                                        InetAddress.getLocalHost(), port);
                ds.send(DpSend);}}}
```

Step 10: Stop

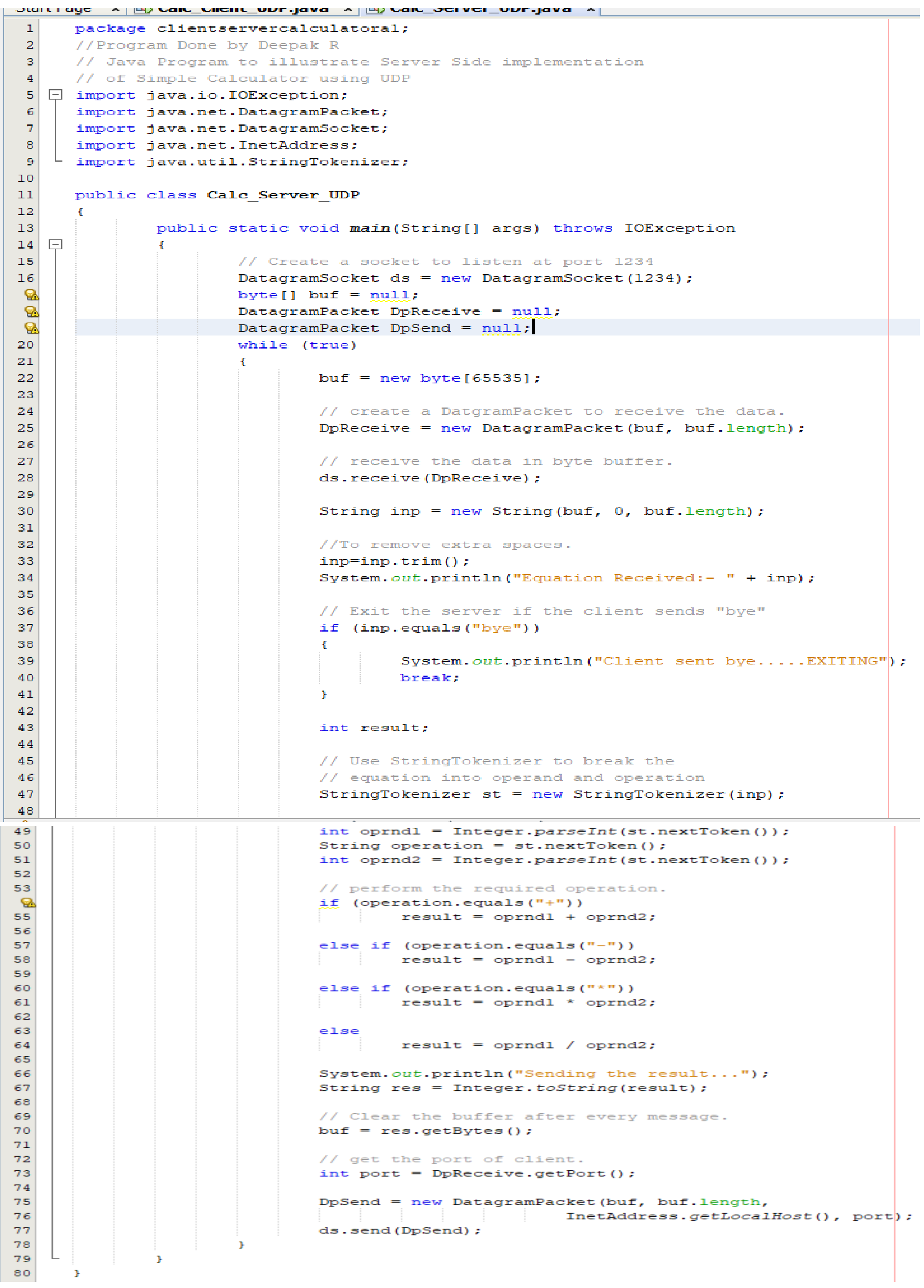
B.1.2 Implementation

Client side



```java
package clientservercalculatoral;
//Program WDone by Deepak R
// Java Program to illustrate Client Side implementation
// of Simple Calculator using UDP
import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.util.Scanner;

public class Calc_Client_UDP {

    public static void main(String args[]) throws IOException {
        Scanner sc = new Scanner(System.in);

                // Step 1:Create the socket object for carrying
        // the data
        DatagramSocket ds = new DatagramSocket();

        InetAddress ip = InetAddress.getLocalHost();
        byte buf[] = null;

        // loop while user not enters "bye"
        while (true) {
            System.out.print("Enter the equation in the format:");
            System.out.println("'operand1 operator operand2'");
            String inp = sc.nextLine();
            buf = new byte[65535];

            // convert the String input into the byte array.
            buf = inp.getBytes();

            // Step 2:Create the datagramPacket for sending the data.
            DatagramPacket DpSend
                    = new DatagramPacket(buf, buf.length, ip, 1234);

            // invoke the send call to actually send the data.
            ds.send(DpSend);

            // break the loop if user enters "bye"
            if (inp.equals("bye")) {
                break;
            }

            buf = new byte[65535];
            DatagramPacket DpReceive
                    = new DatagramPacket(buf, buf.length);
            ds.receive(DpReceive);

            System.out.println("Answer = "
                    + new String(buf, 0, buf.length));
        }
    }
}
```

Server Side

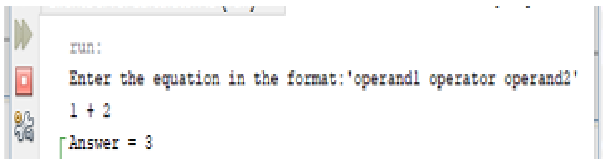


```java
package clientservercalculatoral;
//Program Done by Deepak R
// Java Program to illustrate Server Side implementation
// of Simple Calculator using UDP
import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.util.StringTokenizer;

public class Calc_Server_UDP
{
        public static void main(String[] args) throws IOException
        {
                // Create a socket to listen at port 1234
                DatagramSocket ds = new DatagramSocket(1234);
                byte[] buf = null;
                DatagramPacket DpReceive = null;
                DatagramPacket DpSend = null;
                while (true)
                {
                        buf = new byte[65535];

                        // create a DatgramPacket to receive the data.
                        DpReceive = new DatagramPacket(buf, buf.length);

                        // receive the data in byte buffer.
                        ds.receive(DpReceive);

                        String inp = new String(buf, 0, buf.length);

                        //To remove extra spaces.
                        inp=inp.trim();
                        System.out.println("Equation Received:- " + inp);

                        // Exit the server if the client sends "bye"
                        if (inp.equals("bye"))
                        {
                                System.out.println("Client sent bye.....EXITING");
                                break;
                        }

                        int result;

                        // Use StringTokenizer to break the
                        // equation into operand and operation
                        StringTokenizer st = new StringTokenizer(inp);

                        int oprnd1 = Integer.parseInt(st.nextToken());
                        String operation = st.nextToken();
                        int oprnd2 = Integer.parseInt(st.nextToken());

                        // perform the required operation.
                        if (operation.equals("+"))
                                result = oprnd1 + oprnd2;

                        else if (operation.equals("-"))
                                result = oprnd1 - oprnd2;

                        else if (operation.equals("*"))
                                result = oprnd1 * oprnd2;

                        else
                                result = oprnd1 / oprnd2;

                        System.out.println("Sending the result...");
                        String res = Integer.toString(result);

                        // Clear the buffer after every message.
                        buf = res.getBytes();

                        // get the port of client.
                        int port = DpReceive.getPort();

                        DpSend = new DatagramPacket(buf, buf.length,
                                                InetAddress.getLocalHost(), port);
                        ds.send(DpSend);
                }
        }
}
```

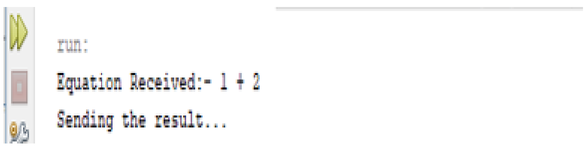
B.1.3 Results and Analysis

Test Cases

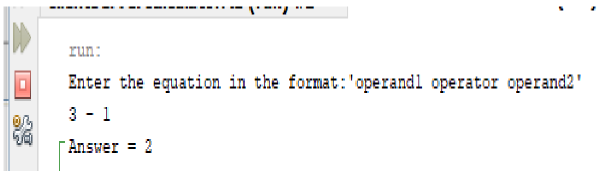
Client Side | **Server Side**



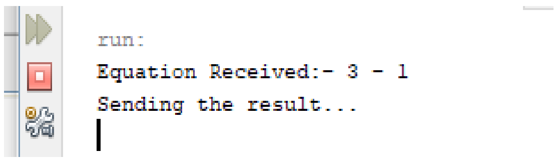
```
run:
Enter the equation in the format:'operand1 operator operand2'
1 + 2
Answer = 3
```



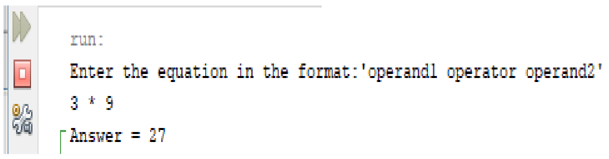
```
run:
Equation Received:- 1 + 2
Sending the result...
```



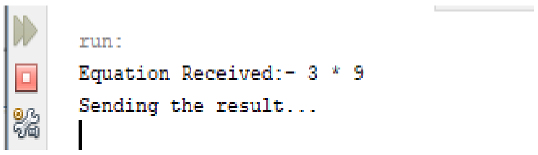
```
run:
Enter the equation in the format:'operand1 operator operand2'
3 - 1
Answer = 2
```



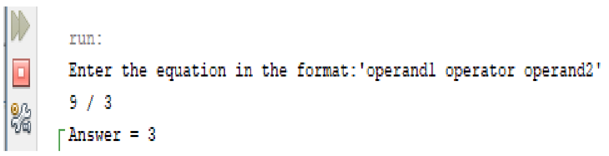
```
run:
Equation Received:- 3 - 1
Sending the result...
```



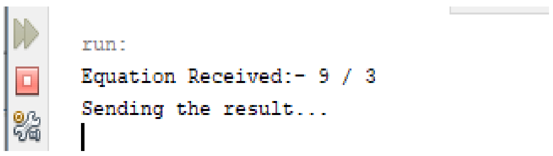
```
run:
Enter the equation in the format:'operand1 operator operand2'
3 * 9
Answer = 27
```



```
run:
Equation Received:- 3 * 9
Sending the result...
```



```
run:
Enter the equation in the format:'operand1 operator operand2'
9 / 3
Answer = 3
```



```
run:
Equation Received:- 9 / 3
Sending the result...
```

The calculator server accepts requests to add, subtract, multiple, and divide a pair of integers. :We run the server program first and then the client one to run Program Successfully. When we enter our operands in client-side Server calculate the given operation and send back result to client and Result gets Displayed on Client Side.

References

1. Mona Shah et al, / (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 6 (3) , 2015, 3010-3013.