RAMAIAH UNIVERSITY OF APPLIED SCIENCES

ASSIGNMENT - 1

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

Name of the Student	K Srikanth
Reg. No	17ETCS002124
Semester/Year	6th / 3rd Year
Course Leader/s	Ms. Jishmi Choondal

Declaration Sheet				
Student Name	K Srikanth			
Reg. No	17ETCS002124			
Programme	B.Tech		Semester/Year	6th/3rd Year
Course Code	19CSC313A			
Course Title	Distributed and Cloud Computing			
Course Date	18/03/2021	to	06/07/2021	
Course Leader	Ms. Jishmi 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	K Srikanth	Date	29-05-2021
Submission date stamp (by Examination & Assessment Section)			

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

RAMAIAH UNIVERSITY OF APPLIED SCIENCES

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.	17ETCS002124	Name of Student	K Srikanth

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

RAMAIAH UNIVERSITY OF APPLIED SCIENCES

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.
6. **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.
9. Marks will be awarded only to the sections and subsections clearly indicated as per the problem statement/exercise/question

Name: K Srikanth Registration Number: 17ETCS002124

Part A

A-1.1)

Introduction

Distributed system is a collection of independent components located on different machines that share messages with each other in order to achieve common goals. Such that the distributed system appears as if it is **one interface or computer to the end-user**. The Plan for doing this is that the system can **maximize resources and information while minimizing failures**, as if one system fails, it won't **affect the availability of the service.**

Now the question is how does it work? why do we need it? Right? So, let's take a deeper dive into how distributed systems work and their advantages in modern distributed computing environment

Okay so first lets first talk about architectures in modern distributed computing environment,

1. **Three- Tier**

   In this architecture, the clients need not worry about **the delivery and sending the data as they can rely on a middle tier to do the processing and decision making**. Most of the web applications are under this category. The middle tier could be called an agent that receives requests from **clients, that could be stateless, processes the data and then forwards it on to the servers. Just like a waiter who can take your order and deliver it to chef.**
2. **Multi – Tier**

   Enterprise web services use n-tier or multi-tier systems architectures. **This application servers that contain the business logic and interacts both with the data tiers and presentation tiers.**
3. **Peer to Peer**

   There are no centralized or special machine that does the heavy work in this architecture. All the **decision making and responsibilities are split up amongst the machines involved and each could take on client or server roles.** Blockchain is the example for this application architecture.

Moving on let's talk about the functions of distributed computing,

1. **Resource sharing - Resource sharing** is one of the primary functions of distributed computing whether it's the hardware, software or data they can be shared among each other with other machines.
2. **Openness – Openness** is next in line functions to show how open is the software designed to be developed and shared with each other
3. **Concurrency** – This helps us achieve the process output with multiple machines as they process the same function at the same time.
4. **Scalability – Scaling is also one of the primary functions where** the computing and processing power are multiplied when extended to many machines
5. **Fault tolerance -** let's say if one server goes down, others could still serve the users.

Now comes the part where we talk about the advantages of modern distributed computing environment,

- **Unlimited Horizontal Scaling** – which means that we can add machines whenever required.
- **Low Latency** – These machines have very low latency as they are spread all over the region and therefore close to serve user.
- **Fault Tolerance** – let's say if one server goes down, others could still serve the users .
-

A-1.2)

The Conference paper that I've selected to talk about Distributed Systems Applications
Is
"Docker and Kubernetes"

Turn to Next Page….

Name: K Srikanth Registration Number: 17ETCS002124

Introduction to Docker and Kubernetes

Before we talk about Docker and how Kubernetes work let's get to know about **"Micro Services" and why Micro Services over Monolith Servers.**

- **What are Micro Services?**

  Microservices are an architectural approach to building applications. As an architectural framework, microservices are distributed and loosely coupled, so one team's changes won't break the entire app. The benefit to using microservices is that development teams are able to rapidly build new components of apps to meet changing requirements.

- **Why do we need them?**

  We have seen problems in monolithic applications which is hard to scale, hard to maintain, and new releases take time which results in lack of moving quickly and easily, lack of innovation and frustrated customer. In today's fast moving world quickly and easily is one of the key components and therefore parallel development is one of the key aspects for the Modern computing . At the same time, the smaller code base is easy to maintain as long as you can manage multiple code bases and monitor this. Also, customers are paying high prices especially when the code is running in the cloud without utilizing resources effectively. Microservice is the way to rescue.

  1. Microservice is a variant of SOA architectural style that structures an application as a collection of loosely coupled services.
  2. Service should be fine-grained
  3. The protocol should be light weight

- **Advantages**

  i. Allow the architecture of individual service to evolve through continuous refactoring and enable continuous delivery and better quickly and easily
  ii. Improves modularity, easier to understand, parallelize development scale independently
  iii. Faster time to market and reduce the frustration of end user
  iv. Decreased Cost – reduce the cost to add more products, customers or business solutions

Name: K Srikanth Registration Number: 17ETCS002124

Okay so we have spoken about how and why "Micro Services", So let's jump into Docker.
Now let's talk about docker and see how it works with orchestration i.e., Kubernetes,
So, Docker is a tool designed to make it easier to create, deploy, and run applications by using containers. Containers allow a developer to package up an application with all of the parts it needs, such as libraries and other dependencies, and deploy it as one package. By doing so, thanks to the container, the developer can rest assured that the application will run on any other Linux machine regardless of any customized settings that machine might have that could differ from the machine used for writing and testing the code.

Now let's properly understand what are containers?
Containers can be thought of as in three categories of software,

- **Builder: technology used to build a container.**
- **Engine: technology used to run a container.**
- **Orchestration: technology used to manage many containers.**

One of the appeals of using containers is their ability to **die gracefully and respawn upon demand.** Whether a c**ontainer's demise is caused by a crash or because it's simply no longer needed when server traffic is low,** containers are **cheap** to start, and they're designed to seamlessly appear and disappear. **Because containers are meant to be short time spawn and to spawn new instances as often as required, it's expected that monitoring and managing them is not done by a human in real-time, but is instead automated**. **Linux containers** have **facilitated high-availability computing**, and there are many toolsets out there to help you run services (or even your entire operating system) in containers. Docker is one option among many, as defined by **Open Container Initiative (OCI),**
If you decide to run services in containers, then you probably need software designed to host and manage those containers. **This is broadly known as container orchestration. The Kubernetes provides container orchestration for a variety of container runtimes.**

Architecture

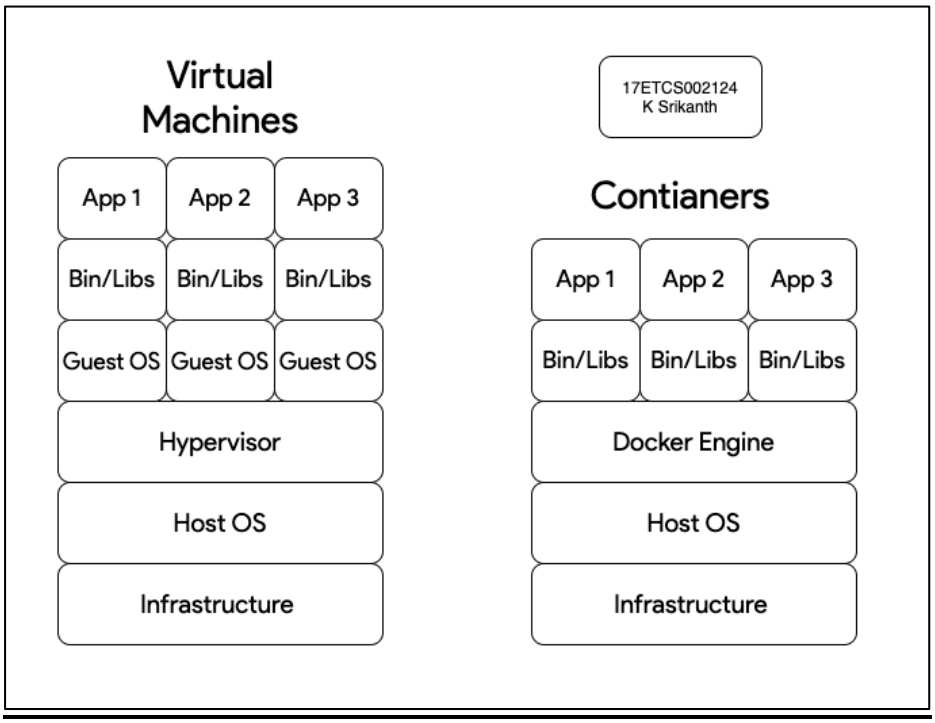

Figure 1 High Level Architecture Diagram of VM and Docker

The New Way is to deploy containers based on operating-system-level virtualization rather than hardware virtualization. These containers are isolated from each other and from the host: they have their own file systems, they can‘t see each other‘s processes, and their computational resource usage can be bounded. They are easier to build than VMs, and because they are decoupled from the underlying infrastructure and from the host file system, they are portable across clouds and OS distributions to start working with them.

Now that we have an idea of how docker works then let’s start orchestrating it with Kubernetes,

Working with Kubernetes

In layman terms Kubernetes are like an orchestrator who can orchestrate our docker containers now what can **Kubernetes do? you ask?**

1. Replication of components
2. Auto-scaling
3. Load balancing
4. Rolling updates
5. Logging across components

6. Monitoring and health checking
7. Service Discovery
8. Authentication

Kubernetes can **schedule and run application containers on clusters of physical or virtual machines.** However, **Kubernetes also allows developers to cut the cord' to physical and virtual machines, moving from a host- centric infrastructure to a container-centric infrastructure,** which provides the full advantages and benefits inherent to containers. **Kubernetes provides the infrastructure to build a truly container-centric development environment.**

Moving parts of Kubernetes (PODS)

1. May contain multiple containers
2. The life cycle of these containers bound together
3. Containers in the pod can see each other on localhost

Pods are not intended to live long. They are created, destroyed and re-created on demand, based on the state of the server and the service itself.

What are all the component that Kubernetes has and provides us to orchestrate our Docker Image
Kubernetes consists of several parts, some of them optional, some mandatory for the whole system.

1. Master Node
2. API Server
3. Replication Controller
4. Scheduler
5. ETDC storage
6. Controller Manager
7. Worker Node
8. Docker

Now let's take a dive and see how some of the components work,

Name: K Srikanth Registration Number: 17ETCS002124

1. **Master Node**

   The master node is responsible for management of Kubernetes cluster. The following components which are from here belong to the master node.

2. **API Server**

   API server is the entry points for all REST commands to control the cluster. It processes the rest requests, validates them, and executes the bound business logic. The resulting state has to be persisted somewhere, and that brings us to the next component of the master node.

3. **Etc/D Storage ("etc Distributed")**

   Etc/D Storage is a simple, distributed, consistent key-value store. It's mainly used for shared configuration and service discovery. It provides a REST API for CRUD operations as well as an interface to register watchers on specific nodes, which enables a reliable way to notify the rest of the cluster about configuration changes.

4. **Scheduler**

   The deployment of configured pods and services onto the nodes happens by scheduler component. The scheduler has the information regarding resources available on the members of the cluster, as well as the ones required for the configured service to run and hence is able to decide where to deploy a specific service.

5. **Docker**

   Docker runs on each of the worker nodes and runs the configured pods. It takes care of downloading the images and starting the containers.

Name: K Srikanth Registration Number: 17ETCS002124

Summary

Docker is a tool designed to make it easier to create, deploy, and run applications by using containers. **Containers allow a developer to package up an application with all of the parts it needs, such as libraries and other dependencies, and deploy it as one package.** There are multiple ways one can deploy container service. Some of the popular ways are to use open source frame work like **Docker Swarm, Kubernetes, Mesos or cloud provided service like AWS ECS**. One can choose any framework but considering above discussion points, **i suggest to use Kubernetes framework because of its key features like out of the box service discovery, portability, namespace configuration, features like config map, a secret map, ingress load balancer, local deployment support with minikube and persistence volume support**.

A-1.3)

The Reason i choose this **research article was because the modern cloud computing and distributed system architecture** its way far ahead of **over monolith servers as most of the dockers containers use micro services** to full fill their user requirements**.** as **micro services are the future of distributed systems and cloud computers** as they are light and faster than usual monolith servers.

Let's take a scenario to get a clear picture of docker containers with kubernetes**,**
Let's consider that our docker images with kubernetes are working in a real time environment like many industries **which use real-time systems that are distributed locally and globally**. The best example would be **Uber where they use it real-time tracking systems**. Now let's say that our **Uber Backend Server has a Micro Service**.

Micro Service for Tracking all the on-going trips: This handles all the incoming and outgoing data which is real time for on-going trips that **are happening at the current moment** so where the server **has to keep a check on where all the trips are going all with the path and also let the user know about it** in their mobile application. So now if it was a monolith server if there are say "**n" number of real time requests coming in then I would say I would be in really tough spo**t to scale as it is supports vertical scaling say that it was high

of demand so the number of **real time requests were high** now you can't manage the server after it served its purpose all the scaling you did would have costed you fortune whereas in case of **docker images they are really light to run say some are around 80Mb** as the number of **real time requests** go up with help of **kubernetes we can actually monitor all that is happening and server the users with those requests** and when the **real time requests gets low** we can **delete the pods when it served its purpose** so see ? there is not point of improving the hardware here when it comes to Docker and Kubernetes they stay minimal and efficient and save you lot of money.

Now let's go through the Benefits of using containers

1. **Agile application creation and deployment**: Increased ease and efficiency of container image creation compared to VM image use.
2. **Continuous development, integration, and deployment**: Provides for reliable and frequent container image build and deployment with quick and easy rollbacks (due to image immutability).
3. **Dev and Ops separation of concerns**: Create application container images at build/release time rather than deployment time, thereby decoupling applications from infrastructure.
4. **Environmental consistency across development, testing, and production:** Runs the same on a laptop as it does in the cloud.
5. **Cloud and OS distribution portability**: Runs on Ubuntu, RHEL, CoreOS, on-prem, Google Container Engine, and anywhere else.
6. **Resource isolation**: Predictable application performance.
7. **Resource utilization**: High efficiency and density. In today's world when the customer is moving to cloud, effective resource utilization is one key aspect to reduce operational cost. Micro-service will help the customer to achieve that goal by deploying multiple micro-services into a virtual machine so that effective resource utilization is much better and it will reduce infra-structure cost also.

Now let's talk about the why did we chose this orchestrator over those other two (tbh google made this even better with their people)

Kubernetes has become the **standard orchestration platform for containers**. All the major cloud providers support it, making it the logical choice for organizations looking to move more applications to the cloud.

Kubernetes provides a common framework to **run distributed systems so development teams have consistent, immutable infrastructure from development to production for every project**. Kubernetes can **manage scaling requirements, availability, failover, deployment patterns, and more.**

Name: K Srikanth Registration Number: 17ETCS002124

Kubernetes has many powerful and advanced capabilities as Kubernetes delivers,

- **Availability.** Kubernetes clustering has very high fault tolerance built-in, allowing for extremely large scale operations.
- **Auto-scaling.** Kubernetes can scale up and scale down based on traffic and server load automatically.
- **Extensive Ecosystem.** Kubernetes has a strong ecosystem around Container Networking Interface (CNI) and Container Storage Interface (CSI) and inbuilt logging and monitoring tools.

However, Kubernetes' complexity is overwhelming for a lot of people who are using it for the first time. The early adopters of Kubernetes have been sophisticated, used mostly by developers from larger scale organizations.. As the developing world begins to look at adopting Kubernetes internally, this approach is often what is referenced in the broader community today. But this approach may not be right for every organization with respects to their plan of design ..

Finally my justification would be I totally support this conference paper so in short words docker rocks with kubernetes for sure as this would be heading in future of distributed systems.

Name: K Srikanth Registration Number: 17ETCS002124

Part B

B-2.1)

Reviewing the given scenario, the problem is single client single server. The client creates a socket bound to the port which server used to register its server socket, and makes a connect request to initiate conversation with the server. the server has a task of watching the queue for new connection requests, process if there are any and go back to its normal task.

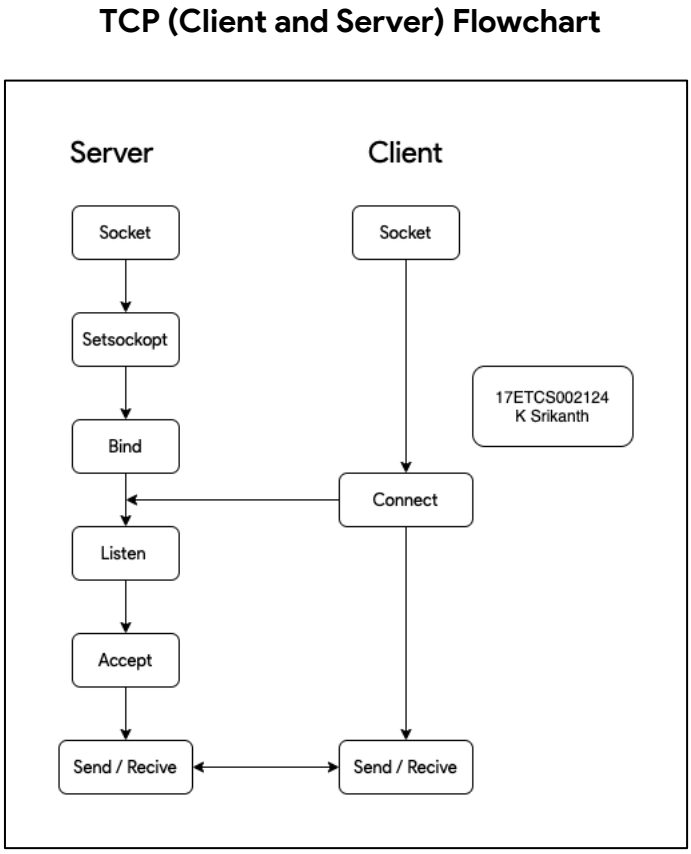

Figure 2 Flowchart for TCP (Client and Server)

Algorithm (Server Side)

1. Start
2. Create a socket
3. Update socket details
4. Bind the socket to the local host
5. Listen passively, if any client is requesting a connection
6. Accept the connection
7. Get the input Datastream

8. Get the output Datastream
9. Repeat forever (While Loop)
    1. Read data from the DataInputStream (Input 1 , Input 2 and Choice)
    2. Convert the received data from ascii to integer: Input 1
    3. Convert the received data from ascii to integer: Input 2
    4. Convert the received data from ascii to integer: Choice
    5. If Choice == '1'
        - **i.** Compute their **sum** and store it i.e. **result = Input 1 + Input 2**
        - ii. Print the message at server's screen
        - iii. Write back the result into the DataOutputStream
    6. If Choice == '2'
        - i. Compute their **sub** and store it i.e. **result = Input 1 - Input 2**
        - ii. Print the message at server's screen
        - iii. Write back the result into the DataOutputStream
    7. If Choice == '3'
        - **i.** Compute their **mull** and store it i.e. **result = Input 1 + Input 2**
        - ii. Print the message at server's screen
        - iii. Write back the result into the DataOutputStream
    8. If Choice == '4'
        - i. Compute their **div** and store it i.e. **result = Input 1 / Input 2**
        - ii. Print the message at server's screen
        - iii. Write back the **r**esult into the DataOutputStream
    9. If Choice == '5"'
        - i. Print the message at server's screen
        - ii. Write back the "Bye String "into the DataOutputStream
10. End

Name: K Srikanth Registration Number: 17ETCS002124

Algorithm (Client Side)

1. Start
2. Create a socket
3. Update socket details
4. Establish a connection
5. Get the input Datastream
6. Get the output Datastream
7. Repeat forever
    1. Read the operation in following format operand1 operator operand2
    2. Send it over the DataOutputStream
    3. If Choice == '1'
        1. Compute their **sum** and store it i.e. **result = Input 1 + Input 2**
        2. Send it over the DataOutputStream
    4. If Choice == '2'
        1. Compute their **sub** and store it i.e. **result = Input 1 - Input 2**
        2. Send it over the DataOutputStream
    5. If Choice == '3'
        1. Compute their **mull** and store it i.e. **result = Input 1 + Input 2**
        2. Send it over the DataOutputStream
    6. If Choice == '4'
        1. Compute their **div** and store it i.e. **result = Input 1 / Input 2**
        2. Send it over the DataOutputStream
    7. If Choice == '5"'
        1. Send it over the DataOutputStream
8. Wait for server to reply
9. Read the information(response) received in socket from dataInputStream
10. Stop

B-2.2)

Java Code for Server (Using TCP Protocol)

```java
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.StringTokenizer;

public class Assignment_Server {
    public static void main(String args[]) throws IOException {

        ServerSocket socketobj = new ServerSocket(8080);
        Socket soccon = socketobj.accept();
        DataInputStream data_input_stream = new DataInputStream(soccon.getInputStream());
        DataOutputStream data_output_stream = new DataOutputStream(soccon.getOutputStream());
        System.out.println("");
        System.out.println("****** 17ETCS002124 K Srikanth **************");
        System.out.println("****** Server Side **************");

        while (true) {
            String in1 = data_input_stream.readUTF();
            String in2 = data_input_stream.readUTF();
            String choice = data_input_stream.readUTF();
            System.out.println("Received Data from Client");
            int result = 0;
            int Input_1 = Integer.parseInt(in1);
            System.out.println("Srikanth's Input 1 is : " + Input_1);
            int Input_2 = Integer.parseInt(in2);
            System.out.println("Srikanth's Input 2 is : " + Input_2);
            int switch_choice = Integer.parseInt(choice);
            System.out.println("Srikanth's Choice is : " + switch_choice);
            System.out.println("");
            System.out.println("");
            switch (switch_choice) {
                case 1:
                    result = Input_1 + Input_2;
                    data_output_stream.writeUTF(String.valueOf(result));
                    break;
                case 2:
                    result = Input_1 - Input_2;
                    data_output_stream.writeUTF(String.valueOf(result));
                    break;
                case 3:
                    result = Input_1 * Input_2;
                    data_output_stream.writeUTF(String.valueOf(result));
                    break;
                case 4:
                    result = Input_1 / Input_2;
                    data_output_stream.writeUTF(String.valueOf(result));
                    break;
                case 5:
                    String bye_String = "Good Bye From Server Hahah XD";
                    data_output_stream.writeUTF(bye_String);
                    break;
                default:
                    System.out.println("Invalid Input Please Try Again");

            }
        }

    }
}

```

Figure 3 Java Program for Server Side Calculator Problem

Java Code for Client (Using TCP Protocol)

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

public class Assignment_Client {
    public static void main(String[] args) throws IOException {
        InetAddress ip_address = InetAddress.getLocalHost();
        int port_num = 8080;
        Scanner scanobj = new Scanner(System.in);
        Socket socketobj = new Socket(ip_address, port_num);
        DataInputStream data_input_stream = new DataInputStream(socketobj.getInputStream());
        DataOutputStream data_output_stream = new DataOutputStream(socketobj.getOutputStream());
        System.out.println("");
        System.out.println("****** 17ETCS002124 K Srikanth **************");
        System.out.println("****** Client Side **************");

        while (true) {
            System.out.println("");
            System.out.print("Enter the Inputs\n");
            System.out.print("***************************************\n");
            System.out.print("Enter the First Number\n");
            String input_1 = scanobj.nextLine();
            System.out.print("Enter the Second Number\n");
            String input_2 = scanobj.nextLine();
            System.out.print("********* Calculator *********\n");
            System.out.print("Press 1 For Addition\n");
            System.out.print("Press 2 For Subtraction\n");
            System.out.print("Press 3 For Multiplication\n");
            System.out.print("Press 4 For Division\n");
            System.out.print("Press 5 to exit\n");
            String choice = scanobj.nextLine();
            data_output_stream.writeUTF(input_1);
            data_output_stream.writeUTF(input_2);
            if (choice.equals("1")) {
                data_output_stream.writeUTF(choice);
                String result = data_input_stream.readUTF();
                System.out.println("The Addition you're looking for is " + result);
                continue;
            }
            if (choice.equals("2")) {
                data_output_stream.writeUTF(choice);
                String result = data_input_stream.readUTF();
                System.out.println("The Subtraction you're looking for is " + result);
                continue;
            }
            if (choice.equals("3")) {
                data_output_stream.writeUTF(choice);
                String result = data_input_stream.readUTF();
                System.out.println("The Multiplication you're looking for is " + result);
                continue;
            }
            if (choice.equals("4")) {
                data_output_stream.writeUTF(choice);
                String result = data_input_stream.readUTF();
                System.out.println("The Division you're looking for is " + result);
                continue;
            }
            if (choice.equals("5")) {
                data_output_stream.writeUTF(choice);
                String result = data_input_stream.readUTF();
                System.out.println("The resultwer you're looking for is " + result);
                break;
            }

        }

    }

}

```

Figure 4 Java Program for Client Side Calculator Problem

Name: K Srikanth Registration Number: 17ETCS002124

B-2.3)

Few terminologies that have been used in the code,

1. **Socket:** This class is for use by a pair of processes with a connection. The client uses a constructor to create a socket, specifying the DNS hostname and port of a server.

2. The Socket class provides the methods **getInputStream and getOutputStream** for accessing the two streams associated with a socket. The return types of these methods are **InputStream and OutputStream,** respectively. They are the abstract classes that define methods for reading bytes from Input stream and writing bytes to output stream.

3. There is an **InputStream and OutputStream** one to push data and latter one to receive data.

4. **DataInputStream and DataOutputStream**: for passing normal data. (i.e. UTF, String, int, float, double etc.) Java serialization is the process of converting data into a stream of bytes so we can do stuff like store it on disk or send it over the network. Deserialization is the reverse process – converting a stream of bytes into an object in memory.

5. **ServerSocket:** This class is for listening to port and accepting requests from the clients. Server uses the server port to create **ServerSocket by specifying port number of the server**. It provides accept method to accept requests from the clients and returns file descriptor for the client socket. **Using its InputStream and OutputStream server will be able to interact with the client.**

Name: K Srikanth Registration Number: 17ETCS002124

Results from Client Side

1. Addition

```
****** 17ETCS002124 K Srikanth **************
****** Client Side **************

Enter the Inputs
****************************************
Enter the First Number
2
Enter the Second Number
3
********* Calculator *********
Press 1 For Addition
Press 2 For Subtraction
Press 3 For Multiplication
Press 4 For Division
Press 5 to exit
1
The Addition you're looking for is 5
```

Figure 5 Java Program Output for Client Side for Addition Choice

2. Subtraction

```
Enter the Inputs
****************************************
Enter the First Number
2
Enter the Second Number
3
********* Calculator *********
Press 1 For Addition
Press 2 For Subtraction
Press 3 For Multiplication
Press 4 For Division
Press 5 to exit
2
The Subtraction you're looking for is -1
```

Figure 5 Java Program Output for Client Side for Subtraction Choice

3. Multiplication

```
Enter the Inputs
****************************************
Enter the First Number
23
Enter the Second Number
17
********* Calculator *********
Press 1 For Addition
Press 2 For Subtraction
Press 3 For Multiplication
Press 4 For Division
Press 5 to exit
3
The Multiplication you're looking for is 391
```

Figure 6 Java Program Output for Client Side for Multiplication Choice

4. Division

```
Enter the Inputs
****************************************
Enter the First Number
22
Enter the Second Number
7
********* Calculator *********
Press 1 For Addition
Press 2 For Subtraction
Press 3 For Multiplication
Press 4 For Division
Press 5 to exit
4
The Division you're looking for is 3
```

Figure 7 Java Program Output for Client Side for Division Choice

Results from Server Side

```
****** 17ETCS002124 K Srikanth **************
****** Server Side **************
Received Data from Client
Srikanth's Input 1 is : 2
Srikanth's Input 2 is : 3
Srikanth's Choice is : 1


Received Data from Client
Srikanth's Input 1 is : 2
Srikanth's Input 2 is : 3
Srikanth's Choice is : 2


Received Data from Client
Srikanth's Input 1 is : 23
Srikanth's Input 2 is : 17
Srikanth's Choice is : 3


Received Data from Client
Srikanth's Input 1 is : 22
Srikanth's Input 2 is : 7
Srikanth's Choice is : 4

Received Data from Client
Srikanth's Input 1 is : 2
Srikanth's Input 2 is : 2
Srikanth's Choice is : 5
```

Figure 8 Java Program Output for Server Side for all the user choices

References

Link to conference paper: https://ieeexplore.ieee.org/document/8862654/authors#authors

Authors

Nikhil Marathe

Computer Science and Engineering, Parul University, Vadodara, India

Ankita Gandhi

Computer Science and Engineering, Parul University, Vadodara, India

Jaimeel M Shah

Computer Science and Engineering, Parul University, Vadodara, India