SCHOOL OF COMPUTER SCIENCE ENGINEERING

AND INFORMATION SYSTEMS

FALL SEMESTER 2024-2025

PMCA502P – JAVA PROGRAMMING LAB

LAB ASSESSMENT – 5

SUBMITTED ON: 15 – NOV - 2024

SUBMITTED BY-

AKASH KUMAR BANIK

PROGRAM: MCA

REGISTER No.: 24MCA0242

01. Develop a TCP based client –server application using java.net.*. Let the client transfers the operation in the format of 'A + B' and server in turn interprets the operator, perform the requested operation using the operands and transmits the result to get displayed in client.

CODE:

Server.java:

```java
package tcp_client_server;

import java.io.*;

import java.net.*;

public class Server {

    public static void main(String[] args) {

    try (ServerSocket serverSocket = new ServerSocket(6789)) {

       System.out.println("Server Connected");

       System.out.println("Server is listening on port 6789");

       while (true) {

          try (Socket socket = serverSocket.accept()) {

             BufferedReader      input      =      new      BufferedReader(new InputStreamReader(socket.getInputStream()));

                PrintWriter output = new PrintWriter(socket.getOutputStream(), true);

                String operation = input.readLine();

                String[] parts = operation.split(" ");

                int num1 = Integer.parseInt(parts[0]);

                String operator = parts[1];

                int num2 = Integer.parseInt(parts[2]);

                int result = 0;

                switch (operator) {
```

```java
            case "+":

                result = num1 + num2;

                break;

            case "-":

                result = num1 - num2;

                break;

            case "*":

                result = num1 * num2;

                break;

            case "/":

                result = num1 / num2;

                break;

            default:

                output.println("Invalid operator");

                continue;

            }

            output.println("Result: " + result);

        } catch (IOException | NumberFormatException e) {

            System.out.println("Server exception: " + e.getMessage());

        }

    }

} catch (IOException e) {

    System.out.println("Could not listen on port 6789");

}

}
```

}

Client.java:

```java
package tcp_client_server;

import java.io.*;

import java.net.*;

public class Client {

    public static void main(String[] args) {

        try (Socket socket = new Socket("localhost", 6789)) {

            PrintWriter output = new PrintWriter(socket.getOutputStream(), true);

            BufferedReader         input         =         new         BufferedReader(new
InputStreamReader(socket.getInputStream()));

            BufferedReader         consoleInput         =         new         BufferedReader(new
InputStreamReader(System.in));

            System.out.println("Enter operation (e.g., 5 + 3): ");

            String operation = consoleInput.readLine();

            output.println(operation);

            String response = input.readLine();

            System.out.println(response);

        } catch (IOException e) {

            System.out.println("Client exception: " + e.getMessage());

        }

    }

}
```

OUTPUT:

02. Design a UDP application in which sender broadcasts the message and the receiver computes count of each character in a message.

CODE:

UDPSender.java:

```
package udp_application;

import java.net.DatagramPacket;

import java.net.DatagramSocket;

import java.net.InetAddress;

public class UDPSender {

    public static void main(String[] args) {

    try {

       DatagramSocket socket = new DatagramSocket();

       socket.setBroadcast(true);

       String message = "Hello JAVA";

       byte[] buffer = message.getBytes();
```

```java
        InetAddress broadcastAddress = InetAddress.getByName("255.255.255.255");

        DatagramPacket packet = new DatagramPacket(buffer, buffer.length, broadcastAddress, 9876);

        socket.send(packet);

        socket.close();

        System.out.println("Message broadcasted: " + message);

    } catch (Exception e) {

        e.printStackTrace();

    }

  }

}
```

UDPReceiver.java:

```java
package udp_application;

import java.net.DatagramPacket;

import java.net.DatagramSocket;

public class UDPReceiver {

    public static void main(String[] args) {

        try {

            DatagramSocket socket = new DatagramSocket(9876);

            byte[] buffer = new byte[1024];

            DatagramPacket packet = new DatagramPacket(buffer, buffer.length);

            System.out.println("Waiting for broadcast...");

            socket.receive(packet);

            String message = new String(packet.getData(), 0, packet.getLength());

            System.out.println("Message received: " + message);
```

```java
        int[] charCount = new int[256];

        for (char c : message.toCharArray()) {

            charCount[c]++;

        }

        for (int i = 0; i < charCount.length; i++) {

            if (charCount[i] > 0) {

                System.out.print((char) i + ": " + charCount[i]);

            }

        }

        socket.close();

    } catch (Exception e) {

        e.printStackTrace();

    }

  }

}
```
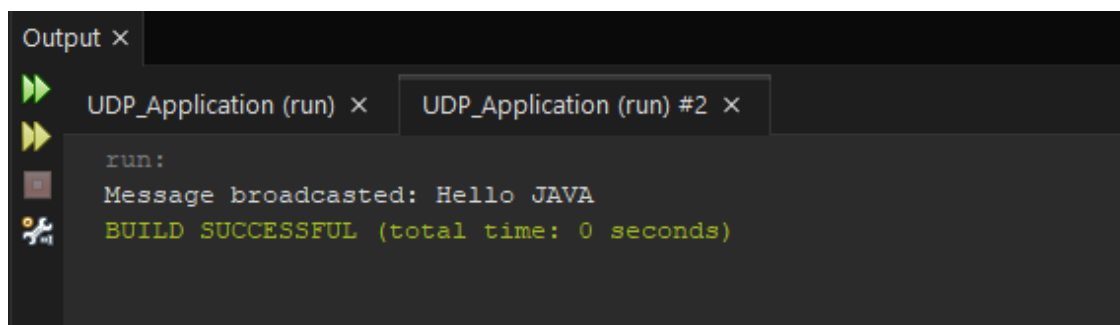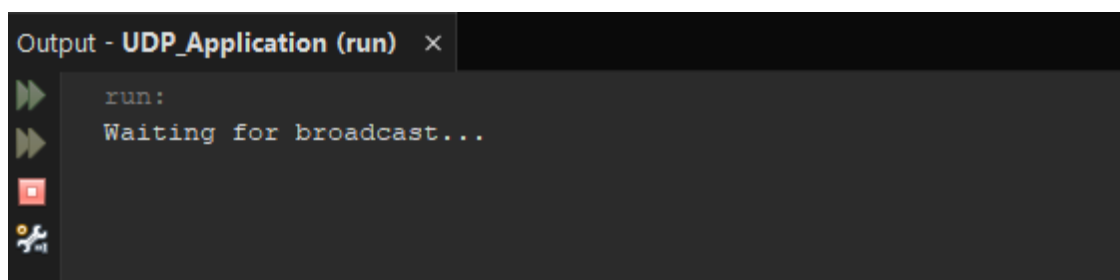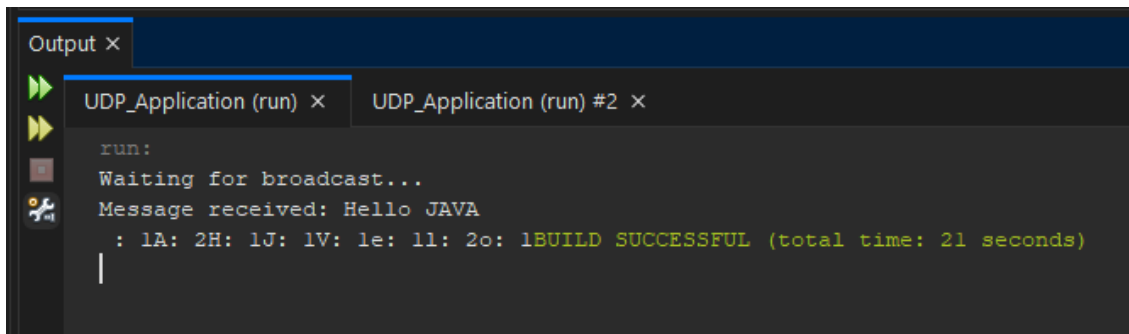
OUTPUT:

03. Implement an RMI application to invoke a factorial method with appropriate parameter deployed in server and display the result in client.

CODE:

Factorial_Impl.java:

import java.rmi.RemoteException;

import java.rmi.server.UnicastRemoteObject;

public class Factorial_Impl extends UnicastRemoteObject implements Factorial {

  protected Factorial_Impl() throws RemoteException {

    super();

  }

  @Override

  public long calculateFactorial(int number) throws RemoteException {

    if (number <= 1) {

      return 1;

    } else {

      return number * calculateFactorial(number - 1);

    }

  }

}

FactorialServer.java:

```java
import java.rmi.Naming;

import java.rmi.registry.LocateRegistry;

public class FactorialServer {

    public static void main(String[] args) {

        try {

            LocateRegistry.createRegistry(1099);

            Factorial_Impl factorial = new Factorial_Impl();

            Naming.rebind("rmi://localhost/FactorialService", factorial);

            System.out.println("Factorial Service is running...");

        } catch (Exception e) {

            e.printStackTrace();

        }

    }

}
```

FactorialClient.java:

```java
import java.rmi.Naming;

public class FactorialClient {

    public static void main(String[] args) {

        try {

            Factorial factorial = (Factorial) Naming.lookup("rmi://localhost/FactorialService");

            int number = 5;

            long result = factorial.calculateFactorial(number);

            System.out.println("Factorial of " + number + " is " + result);

        } catch (Exception e) {
```
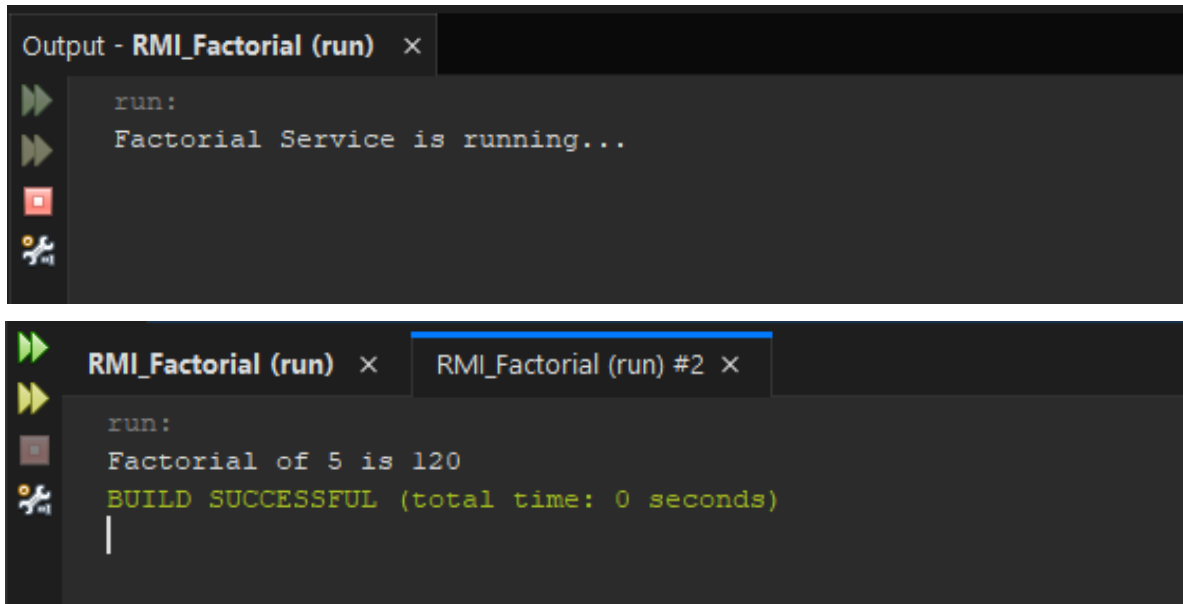
```
        e.printStackTrace();

    }

  }

}
```

OUTPUT: