Bhuvana Kanakam
SE21UCSE035
CS4101 Lab 03
09.06.2024

note : in class work - question 1 and question 2, assignment to evaluate is question3

1 Simple Inter-process Communication Using Sockets

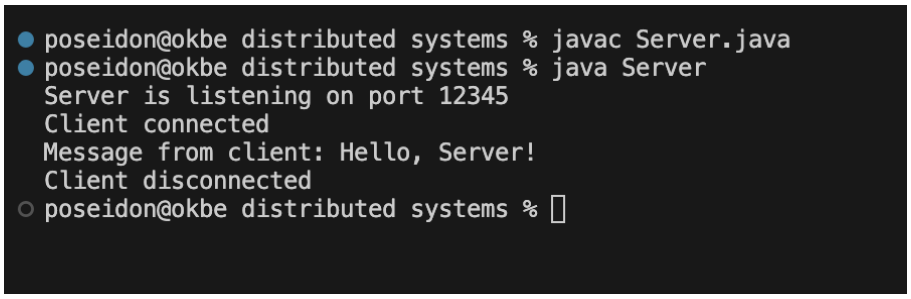
Server.java Code

```
import java.io.*;
import java.net.*;

public class Server {
    public static void main(String[] args) {
        int port = 12345;
        try (ServerSocket serverSocket = new ServerSocket(port)) {
            System.out.println("Server␣is␣listening␣on␣port␣" + port);
            Socket socket = serverSocket.accept();
            System.out.println("Client␣connected");

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

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

            String clientMessage = reader.readLine();
            System.out.println("Message␣from␣client:␣" + clientMessage);

            String response = "Server␣received:␣" + clientMessage;
            writer.println(response);

            socket.close();
            System.out.println("Client␣disconnected");
        } catch (IOException ex) {
            System.out.println("Server␣exception:␣" + ex.getMessage());
            ex.printStackTrace();
        }
    }
}
```

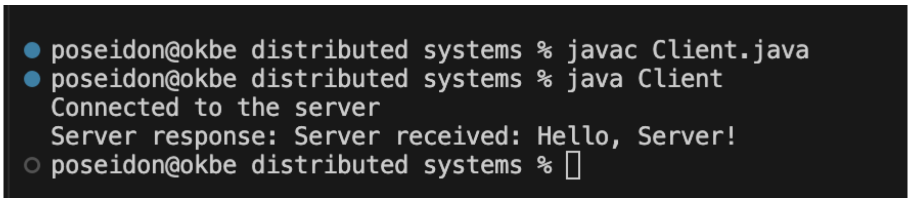
Client.java Code

```
import java.io.*;
import java.net.*;

public class Client {
    public static void main(String[] args) {
        String hostname = "localhost";
        int port = 12345;

        try (Socket socket = new Socket(hostname, port)) {
            System.out.println("Connected␣to␣the␣server");
            OutputStream output = socket.getOutputStream();
            PrintWriter writer = new PrintWriter(output, true);

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

            String message = "Hello, Server!";
            writer.println(message);

            String response = reader.readLine();
            System.out.println("Server response: " + response);

            socket.close();
        } catch (UnknownHostException ex) {
            System.out.println("Server not found: " + ex.getMessage());
        } catch (IOException ex) {
            System.out.println("I/O error: " + ex.getMessage());
        }
    }
}
```

Output



```
● poseidon@okbe distributed systems % javac Server.java
● poseidon@okbe distributed systems % java Server
  Server is listening on port 12345
  Client connected
  Message from client: Hello, Server!
  Client disconnected
○ poseidon@okbe distributed systems % ▯
```



```
● poseidon@okbe distributed systems % javac Client.java
● poseidon@okbe distributed systems % java Client
  Connected to the server
  Server response: Server received: Hello, Server!
○ poseidon@okbe distributed systems % ▯
```

2 IPC using java for RMI

Implement a simple Remote Method Invocation system where the client calls a method on the server to add two numbers. The server should return the sum, and the client should display the result.

CalculatorInterface.java Code

```java
import java.rmi.Remote;
import java.rmi.RemoteException;

public interface CalculatorInterface extends Remote {
    int add(int a, int b) throws RemoteException;
}
```

CalculatorServer.java Code

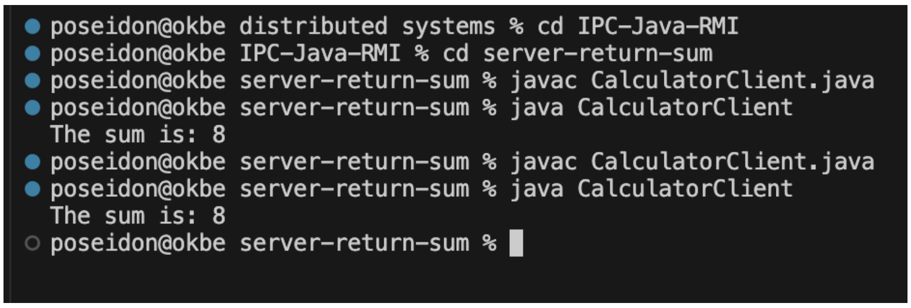
```java
import java.rmi.Naming;
import java.rmi.RemoteException;
import java.rmi.registry.LocateRegistry;
import java.rmi.server.UnicastRemoteObject;

public class CalculatorServer extends UnicastRemoteObject implements
CalculatorInterface {

    public CalculatorServer() throws RemoteException {
        super();
    }

    @Override
    public int add(int a, int b) throws RemoteException {
        return a + b;
    }

    public static void main(String[] args) {
        try {
            LocateRegistry.createRegistry(1100);

            Naming.rebind("//localhost:1100/CalculatorService",
            new CalculatorServer());
            System.out.println("Calculator Server is ready.");
        } catch (Exception e) {
            System.err.println("Server failed: " + e);
            e.printStackTrace();
        }
    }
}
```

CalculatorClient.java Code

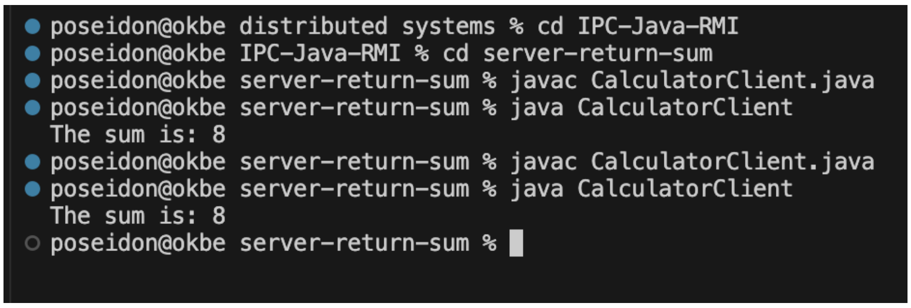
```java
import java.rmi.Naming;

public class CalculatorClient {
    public static void main(String[] args) {
        try {
            CalculatorInterface calculator = (CalculatorInterface)
            Naming.lookup("rmi://localhost:1100/CalculatorService");

            int result = calculator.add(5, 3);
```

```
                System.out.println("The sum is: " + result);
            } catch (Exception e) {
                System.err.println("Client exception: " + e);
                e.printStackTrace();
            }
        }
}
```

Output



```
● poseidon@okbe distributed systems % cd IPC-Java-RMI
● poseidon@okbe IPC-Java-RMI % cd server-return-sum
● poseidon@okbe server-return-sum % javac CalculatorClient.java
● poseidon@okbe server-return-sum % java CalculatorClient
  The sum is: 8
● poseidon@okbe server-return-sum % javac CalculatorClient.java
● poseidon@okbe server-return-sum % java CalculatorClient
  The sum is: 8
○ poseidon@okbe server-return-sum % █
```



```
● poseidon@okbe distributed systems % cd IPC-Java-RMI
● poseidon@okbe IPC-Java-RMI % cd server-return-sum
● poseidon@okbe server-return-sum % javac CalculatorClient.java
● poseidon@okbe server-return-sum % java CalculatorClient
  The sum is: 8
● poseidon@okbe server-return-sum % javac CalculatorClient.java
● poseidon@okbe server-return-sum % java CalculatorClient
  The sum is: 8
○ poseidon@okbe server-return-sum % █
```

3 Multithread Client-Server Communication using Sockets

Write a multithread java server that can handle multiple clients simultaneously using tcp sockets. each client should send a message, and the server should respond by sending the clients message in reverse

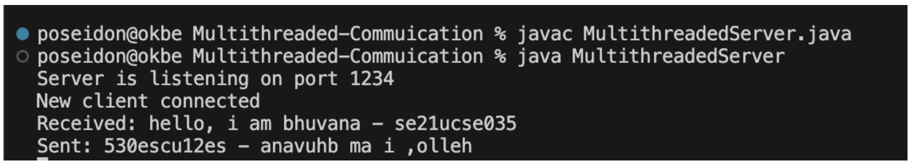
MultithreadedServer.java code

```
import java.io.*;
import java.net.*;

public class MultithreadedServer {
    public static void main(String[] args) {
        int port = 1234; // Port number for the server to listen on
        try (ServerSocket serverSocket = new ServerSocket(port)) {
            System.out.println("Server is listening on port " + port);

            while (true) {
                // Accept a client connection
                Socket clientSocket = serverSocket.accept();
                System.out.println("New client connected");

                new ClientHandler(clientSocket).start();
            }
        } catch (IOException e) {
            System.err.println("Server exception: " + e.getMessage());
            e.printStackTrace();
        }
    }
}

class ClientHandler extends Thread {
    private Socket clientSocket;

    public ClientHandler(Socket socket) {
        this.clientSocket = socket;
    }

    @Override
    public void run() {
        try (
            BufferedReader in = new BufferedReader(new
            InputStreamReader(clientSocket.getInputStream()));
            PrintWriter out = new PrintWriter(
            clientSocket.getOutputStream(), true)
        ) {
            String message = in.readLine();
            System.out.println("Received: " + message);

            String reversedMessage = new StringBuilder(message).reverse().toString();
            out.println(reversedMessage);
            System.out.println("Sent: " + reversedMessage);
        } catch (IOException e) {
            System.err.println("Client handler exception: " + e.getMessage());
            e.printStackTrace();
        } finally {
            try {
                clientSocket.close();
            } catch (IOException e) {
                System.err.println("Failed to close client socket: "
                + e.getMessage());
            }
        }
    }
}
```

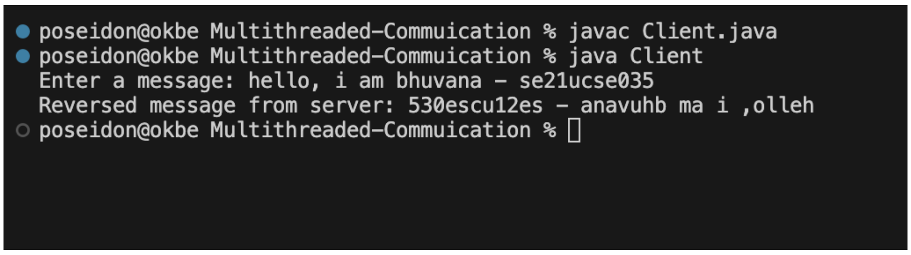
Client.java code

```java
import java.io.*;
import java.net.*;

public class Client {
    public static void main(String[] args) {
        String hostname = "localhost";
        int port = 1234;

        try (
            Socket socket = new Socket(hostname, port);

            BufferedReader in = new BufferedReader(new
            InputStreamReader(socket.getInputStream()));
            PrintWriter out = new PrintWriter(socket.getOutputStream(), true);
            BufferedReader userInput = new BufferedReader(new
            InputStreamReader(System.in))
        ) {
            System.out.print("Enter a message: ");
            String message = userInput.readLine();

            out.println(message);

            String response = in.readLine();
            System.out.println("Reversed message from server: " + response);
        } catch (UnknownHostException e) {
            System.err.println("Server not found: " + e.getMessage());
        } catch (IOException e) {
            System.err.println("I/O error: " + e.getMessage());
        }
    }
}
```

Output



```
poseidon@okbe Multithreaded-Commuication % javac MultithreadedServer.java
poseidon@okbe Multithreaded-Commuication % java MultithreadedServer
Server is listening on port 1234
New client connected
Received: hello, i am bhuvana - se21ucse035
Sent: 530escu12es - anavuhb ma i ,olleh
```



```
poseidon@okbe Multithreaded-Commuication % javac Client.java
poseidon@okbe Multithreaded-Commuication % java Client
Enter a message: hello, i am bhuvana - se21ucse035
Reversed message from server: 530escu12es - anavuhb ma i ,olleh
poseidon@okbe Multithreaded-Commuication %
```