Implementasi RPC Menggunakan Java

Dua program client (**RPCClient.java**) dan server (**RPCServer.java**) di bawah ini memperlihatkan bagaimana RPC (Remote procedure call) disimplementasikan di dalam Bahasa Pemrogaraman Java menggunakan pustaka bawaan Socket, SocketServer dan yang terkait.

RPCServer.java:

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintStream;
import java.net.InetAddress;
import java.net.ServerSocket;
import java.net.Socket;


public class RPCServer {

    private final ServerSocket serverSocket;

    @SuppressWarnings("CallToThreadStartDuringObjectConstruction")
    public RPCServer(int port) throws IOException {
        serverSocket = new ServerSocket(port);

        String localIP = InetAddress.getLocalHost().getHostAddress();

        System.out.println("Server is running on " + localIP + ":" + port);

        while (true) {

            Socket rpcClient = serverSocket.accept();
            String address = rpcClient.getRemoteSocketAddress().toString();

            System.out.println("New client connected : " + address);

            new Thread(() -> {
                try {
                    addHook(rpcClient);
                } catch (IOException ex) {
                    System.err.println("Client disconnected " + address);
                }
            }).start();
        }
    }

    private void addHook(Socket rpcClient) throws IOException {

        BufferedReader reader = new BufferedReader(new
InputStreamReader(rpcClient.getInputStream()));
        String line;

        while ((line = reader.readLine()) != null) {

            System.out.println("Client request : " + line);
            String[] commands = line.split(":", 3);
```

```java
            int result;
            int operand1 = Integer.parseInt(commands[1]);
            int operand2 = Integer.parseInt(commands[2]);

            String message = "";

            switch (commands[0]) {

                case "add":
                    result = (operand1 + operand2);
                    message = operand1 + " + " + operand2 + " = " + result;
                    break;

                case "sub":
                    result = (operand1 - operand2);
                    message = operand1 + " - " + operand2 + " = " + result;
                    break;

                case "mul":
                    result = (operand1 * operand2);
                    message = operand1 + " * " + operand2 + " = " + result;
                    break;

                case "div":
                    result = (operand1 / operand2);
                    message = operand1 + " / " + operand2 + " = " + result;
                    break;

                case "mod":
                    result = (operand1 % operand2);
                    message = operand1 + " % " + operand2 + " = " + result;
                    break;

            }

            PrintStream printStream = new PrintStream(rpcClient.getOutputStream(), true);
            printStream.println(message);
        }
    }

    public static void main(String[] args) throws Exception {
        RPCServer server = new RPCServer(3000);
    }
}
```

RPCClient.java:

```java
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintStream;
import java.net.Socket;
import java.util.Scanner;


public class RPCClient {

    private final PrintStream printStream;

    @SuppressWarnings("CallToThreadStartDuringObjectConstruction")
    public RPCClient(String ipAddress, int port) throws IOException {

        Socket rpcClient = new Socket(ipAddress, port);

        new Thread(() -> {
            try {
                BufferedReader reader = new BufferedReader(new
InputStreamReader(rpcClient.getInputStream()));
                String line;

                while ((line = reader.readLine()) != null) {
                    System.out.println("Server response : " + line);
     System.out.print("\nCommands [add, sub, mul, div, mod, exit] : ");
                }
            } catch (IOException ex) {
                System.err.println("\nDisconnected!!");
                System.exit(0);
            }
        }).start();

        printStream = new PrintStream(rpcClient.getOutputStream(), true);

    }

    public void sendMessage(String operation) {
        Scanner scan = new Scanner(System.in);
        System.out.print("\nEnter 1st number : ");
        int f1 = scan.nextInt();

        System.out.print("Enter 2nd number : ");
        int s1 = scan.nextInt();

        printStream.println(operation + ":" + f1 + ":" + s1);

    }

    public static void main(String[] args) {

        try {
   Scanner scan = new Scanner(System.in);

   System.out.print("Enter server ip address : ");
   String ipAddress = scan.nextLine();

   System.out.print("Enter connection port : ");
   int port = scan.nextInt();

            RPCClient client = new RPCClient(ipAddress, port);
            System.out.println("\nConnected to server\n");

   System.out.print("Commands [add, sub, mul, div, mod, exit] : ");

            while (true) {

    scan = new Scanner(System.in);

                String command = scan.nextLine();

                if (command.equals("exit")) {
                    System.exit(0);
                }
                client.sendMessage(command);

    System.out.print("\n");

            }
        } catch (IOException ex) {
            System.err.println("\nUnable to connected!");
        }

    }

}
```

Server Output:

```
Server is running on 172.20.52.46:3000
New client connected : /172.20.52.46:52843
Client request : add:10:20
Client request : sub:50:20
Client request : mul:10:2
Client request : div:100:5
Client request : mod:1234:10
Client disconnected /172.20.52.46:52843
```

Client Output:

```
Enter server ip address : 172.20.52.42
Enter connection port : 3000

Connected to server
Commands [add, sub, mul, div, mod, exit] :
add
Enter 1st number : 10
Enter 2nd number : 20
Server response : 10 + 20 = 30

sub
Enter 1st number : 50
Enter 2nd number : 20
Server response : 50 - 20 = 30

mul
Enter 1st number : 10
Enter 2nd number : 2
Server response : 10 * 2 = 20

div
Enter 1st number : 100
Enter 2nd number : 5
Server response : 100 / 5 = 20
```

```
mod
Enter 1st number : 1234
Enter 2nd number : 10
Server response : 1234 % 10 = 4

exit
```