1. flow chart (architecture)

<Server>

1) welcoming socket을 만든다.

2) 아래 2-1 부터 2-3까지를 반복한다.

2-1) Client를 기다리고 새로운 연결을 한다.

2-2) 연결된 Client와 소통한다.

a) Client로부터 입력 받은 명령어와 숫자를 출력한다.

b) Client로부터 입력 받은 명령어와 숫자를 이용해 계산한다.

c) 계산한 결과를 출력한다.

2-3) 연결을 종료한다.

<Client>

1) socket을 만들고 Server와 연결한다.

2) Server와 소통한다.

2-1) 계산 명령어와 숫자를 입력한다.

3)연결을 종료한다.

2. message format

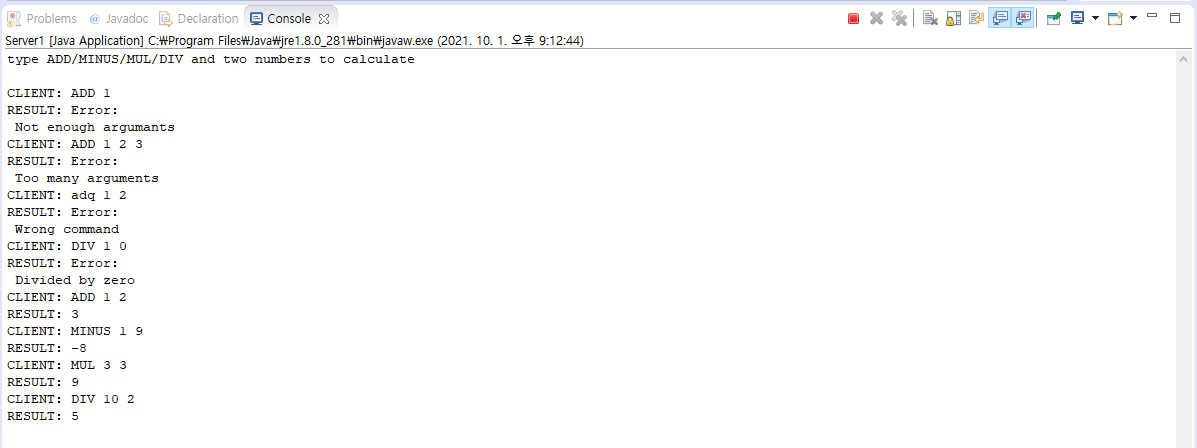
(계산 명령어) (계산할 숫자) (계산할 숫자)

계산 명령어에는 ADD, MINUS, MUL, DIV 중 하나가 들어가야 한다.

계산할 숫자는 무조건 두 개가 입력되어야 하고, 2개 미만이거나 2개 초과일 경우 에러 메세지가 출력된다.

계산 명령어와 계산할 첫 번째 숫자와 계산할 두 번째 숫자 사이에는 띄어쓰기를 한 번씩 해야 한다.

3. test set



4. source code

**package** myPackage;

**import** java.io.BufferedReader;

**import** java.io.DataOutput;

**import** java.io.DataOutputStream;

**import** java.io.InputStreamReader;

**import** java.net.ServerSocket;

**import** java.net.Socket;

**public** **class** Server1 {

**public** **static** **void** main(String[] args) **throws** Exception {

String clientSentence = **null**;

String[] calCommand;

**int** num1, num2;

String calResult = **null**;

ServerSocket welcomeSocket = **new** ServerSocket(6789); //create welcoming socket

System.***out***.println("type ADD/MINUS/MUL/DIV and two numbers to calculate\n ");

**while**(**true**) {

Socket connectionSocket = welcomeSocket.accept(); //wait on welcoming socket for client

BufferedReader inFromClient = **new** BufferedReader(

**new** InputStreamReader(connectionSocket.getInputStream())); //communicate with client

DataOutputStream outToClient =

**new** DataOutputStream(connectionSocket.getOutputStream()); //communicate with client

clientSentence = inFromClient.readLine();

System.***out***.println("CLIENT: " + clientSentence); //print client's input

calCommand = clientSentence.split(" "); //split calculate command and numbers

**if**(calCommand.length>3)

calResult = "Error:\n Too many arguments"; //error message when client input has more than 2 number

**else** **if**(calCommand.length<3)

calResult = "Error:\n Not enough argumants"; //error message when client input has less than 2 number

**else**{

num1 = Integer.*parseInt*(calCommand[1]);

num2 = Integer.*parseInt*(calCommand[2]);

**if**(calCommand[0]!="ADD" && calCommand[0]!="MINUS" && calCommand[0]!="MUL" && calCommand[0]!="DIV")

calResult = "Error:\n Wrong command"; //error message when client input has wrong calculate command

**if**(calCommand[0].equals("DIV")) {

**if**(num2==0) calResult = "Error:\n Divided by zero"; //error message when divided by zero

**else** calResult = Integer.*toString*(num1/num2);

}

**if**(calCommand[0].equals("ADD"))

calResult = Integer.*toString*(num1+num2);

**if**(calCommand[0].equals("MINUS"))

calResult = Integer.*toString*(num1-num2);

**if**(calCommand[0].equals("MUL"))

calResult = Integer.*toString*(num1\*num2);

}

((DataOutput) outToClient).writeBytes(calResult);

System.***out***.println("RESULT: " + calResult); //print the result of calculation

}

}

}

**package** myPackage;

**import** java.io.BufferedReader;

**import** java.io.DataOutput;

**import** java.io.DataOutputStream;

**import** java.io.InputStreamReader;

**import** java.net.Socket;

**public** **class** Client1 {

**public** **static** **void** main(String[] args) **throws** Exception {

String calculate;

String result;

BufferedReader inFromUser =

**new** BufferedReader(**new** InputStreamReader(System.***in***)); //communicate with Server

Socket clientSocket = **new** Socket("127.0.0.1", 6789); //create client socket

DataOutputStream outToServer =

**new** DataOutputStream(clientSocket.getOutputStream()); //communicate with Server

BufferedReader inFromServer = **new** BufferedReader

(**new** InputStreamReader(clientSocket.getInputStream())); //communicate with Server

calculate = inFromUser.readLine(); //store input

((DataOutput) outToServer).writeBytes(calculate + '\n'); //send input to server

result = inFromServer.readLine();

System.***out***.println(result); //print calculate result

clientSocket.close(); //close client socket

}

}