정, 승호

# github URL (optional):

|  |  |
| --- | --- |
|  | [과제의 목적을 간단하게 설명] |

# #개요

|  |  |
| --- | --- |
|  | 1, server구현 : multithreading을 사용하여 동시에 여러 클라이언트 요청을 처리 할 수 있는 서버구현  client로부터 수신된 계산식을 평가하고, 결과를 client에게 다시 전송.  2, client구현 : 서버에 연결할 수 있는 client 구현.  user에게 계산식을 입력 받아 server에게 메시지를 전송 후, 서버의 메시지를 수신하여 user에게 결과를 출력.  3, 프로토콜 정의 : server와 client 간에 메시지를 주고 받기 위하여 프로토콜을 정의, 계산식을 보내는 양식, 결과를 받는 타입을 포함.  4, 예외 처리 : 프로토콜 정의에 위반하는 양식으로 메시지를 보내거나, 0으로 정수를 나누는 것과같이 예외상황에 대한 처리를 구현 |

**Requirements**

* < 요구조건 #1: 간략한 핵심 요구 조건 설명>

# REPORT

1, 구조도텍스트, 스크린샷, 도표, 평행이(가) 표시된 사진

자동 생성된 설명

# PRotocol

|  |  |
| --- | --- |
|  | 1, calculation request를 보내는 양식 (정수)(공백)(calculator)(공백)(정수) 형식, 이형식을 위반하고 공백을 넣지 않고 request를 요청하면 client에게 connection error이라는 메시지를 response.  2, calculation request를 보낼 때 나눗셈 연산자를 사용하지만 정수를 0으로 나누는 경우, connection error이라는 메시지를 response하고 server의 작동을 중지.  3, calculation request를 보낼 때 protocol의 정의보다 더 많은 문자가 들어가는 경우 Incorrect : too many argument라는 메시지를 response하고 server의 작동을 중지.  4, calculation request를 보낼 때 양식이 정확하여 정상작동이 되면, sever 는 received메세지 출력, 계산 후 result값이 정확히 response되면 sending 메시지 출력.  5, server 와 client간의 connection이 제대로 되지 않으면 server에서 Error 메시지를 출력.  6, calculation request 메시지를 받고 연산을 수행한후 결과를 response할 때 correct answer = result의 형태로 string 타입으로 client에게 반환.  7, client가 end라는 메시지를 server에게 보냈을 경우 server와 client간의 connection을 끝냄.  8, client 가 server에 연결하기위해 사용하는 port number는 7777, ip address는 localhost(127.0.0.1).  9, client가 server에 요청하는 calculation request 메시지에는 +, \*, /, -만 사용 될 수 있음. |

## SOURCE CODES

|  |
| --- |
| **package** HW1\_src;  //server program  **import** java.io.\*;  **import** java.net.\*;  **import** java.util.\*;  //server 정의  **public** **class** server {  //socket 생성  **private** Socket socket = **null**;  //port number 를 받는 server 정의  **public** server(**int** port)  {  **try** {  //server socket 생성  ServerSocket serversocket = **new** ServerSocket(port);  Socket socket = serversocket.accept();//server socket 이 accept 했을때 socket 생성    DataInputStream in = **new** DataInputStream(socket.getInputStream());//input stream 정의  DataOutputStream out = **new** DataOutputStream(socket.getOutputStream());//데이터를 보내기위한 output stream 정의  **while** (**true**) {//end 라는 메세지를 받기전까지 client 의 calculation request 메세지 수행  String input = in.readUTF();//calculation request 메세지를 읽어옴  **if** (input.equals("end"))//end 를 입력 받았을 경우 종료  **break**;  System.***out***.println("received");  **int** result = 0;  StringTokenizer string = **new** StringTokenizer(input);//읽어온 메세지를 프로토콜기반으로 해석하기위하여 토큰을 사용    **int** integer1 = Integer.*parseInt*(string.nextToken());//첫번째 정수를 읽어옴  String operation = string.nextToken();//연산자를 읽어옴  **int** integer2 = Integer.*parseInt*(string.nextToken());//두번째 정수를 읽어옴  **if**(string.hasMoreTokens()) {//프로토콜에 정의에 따라서 위반되는 더 많은 데이터가 들어온 경우 그에맞는 결과를 반환 후 종료  out.writeUTF("Incorrect : Too many arguments");  **break**;  }    **if** (operation.equals("+")) {//덧셈 연산자  result = integer1 + integer2;  }  **else** **if** (operation.equals("-")) {//빼기 연산자  result = integer1 - integer2;  }  **else** **if** (operation.equals("\*")) {//곱셈 연산자  result = integer1 \* integer2;  }  **else** **if** (operation.equals("/")) {//나눗셈 연산자  result = integer1 / integer2;  }  **if** (operation.equals("/") && integer2 == 0) {//나눗셈 연산자에서 나누는 수가 0인경의 예외처리  out.writeUTF("Incorrect : divided by zero");  **break**;  }      System.***out***.println("sending");  out.writeUTF("correct answer = " + Integer.*toString*(result));//string 의 형태로 프로토콜에 맞춰 result 값을 client 에게 반환  }  }  **catch** (Exception e) {//예외처리  System.***out***.println("Error");  }    }  **public** **static** **void** main(String args[])  {  server server = **new** server(7777);//포트번호가 7777 인 server 를 생성  }  } |
| **package** HW1\_src;  **import** java.io.\*;  **import** java.net.\*;  **import** java.util.\*;  //client program  **public** **class** client {  **private** Socket socket = **null**;  **public** client(String address, **int** port)  {  **try** {    Scanner scanner = **new** Scanner(System.***in***);//값을 입력받기위한 scanner 정의  socket = **new** Socket(address, port);//socket 정의    System.***out***.println("OK");//소켓이 잘 생성되면 user 에게 ok 메세지 출력  DataInputStream in = **new** DataInputStream(socket.getInputStream());//input stream 정의  DataOutputStream out = **new** DataOutputStream(socket.getOutputStream());//output stream 정의    **while** (**true**) {  System.***out***.println("enter <int> <operator> <int>");//프로토콜 형식을 user 에게 알려줌  String tmp = scanner.nextLine();//user 에게 값을 입력 받음  **if** (tmp.equals("end"))//user 가 end 를 입력했을경우 client 종료  **break**;  out.writeUTF(tmp);//calculation request  String answer = in.readUTF();//correct result  System.***out***.println(answer);//user 에게 correct result 값을 출력  }  }  **catch** (Exception e) {//예외 처리  System.***out***.println("connection error");  }  }  **public** **static** **void** main(String args[])  {    client client = **new** client("127.0.0.1", 7777);//port number 가 7777 인 서버에 연결하는 client 생성  }  } |

## OUTPUT (Screen Shots)

|  |  |
| --- | --- |
| #1 | [설명] |
| 1,    2,  텍스트, 폰트, 스크린샷, 라인이(가) 표시된 사진  자동 생성된 설명  3,  텍스트, 폰트, 스크린샷, 화이트이(가) 표시된 사진  자동 생성된 설명  4,  텍스트, 폰트, 스크린샷, 화이트이(가) 표시된 사진  자동 생성된 설명  5,  텍스트, 폰트, 스크린샷, 화이트이(가) 표시된 사진  자동 생성된 설명  6,    7, | |