ㅣ

Lee, HAEUN (이하은)

# github URL (optional):

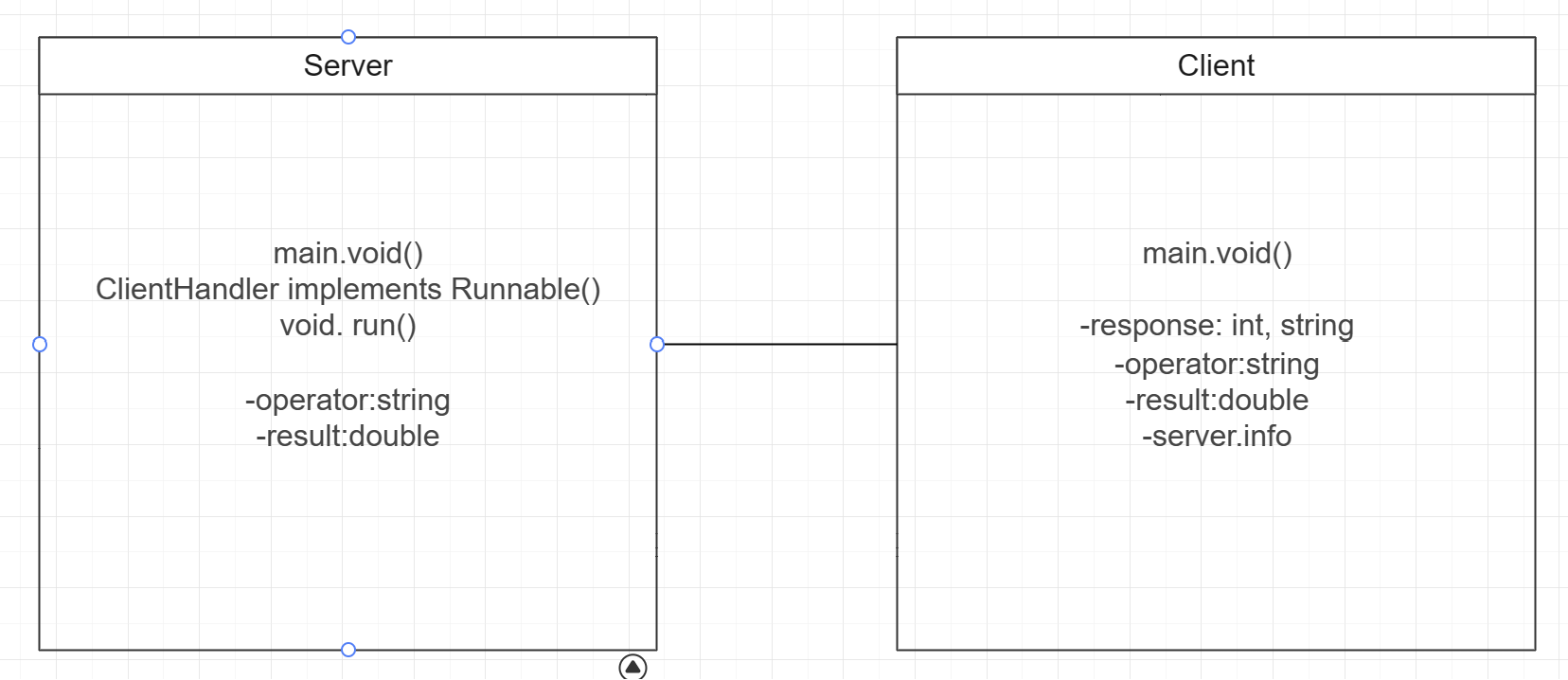
|  |  |
| --- | --- |
|  | https://github.com/amou6518/Calculator |

# #개요

|  |  |
| --- | --- |
|  | 이 프로그램은 Java 소켓 API를 사용하여 구현된 계산기 애플리케이션입니다.  아래는 프로그램 실행 메커니즘과 요구 사항에 대한 간략한 설명입니다.  **프로그램 실행 메커니즘**  **1.서버-클라이언트 연결**  - 서버는 클라이언트의 연결을 수락하고, 클라이언트로부터 받은 식을 해석하여 계산 결과를 클라이언트에게 전송합니다.  **2. 클라이언트가 서버로 식 전송**  - 클라이언트는 사용자로부터 수식을 입력받고, 이를 서버로 전송합니다.  **3.서버에서 수신한 수식의 형식 확인**  - 서버는 수신한 수식이 올바른 형식인지 확인합니다.  **3-1올바른 수식일 경우**   * 서버는 수식을 계산하고 정확한 답을 클라이언트에게 전송합니다.   **3-2부적절한 수식일 경우**   * 서버는 오류 정보를 클라이언트에게 전송합니다.   **프로그램 요구 사항**  **1.명령/응답 프로토콜 정의**  - 프로그램은 통신 메시지 형식을 정의하는 프로토콜을 가집니다.  **2.네 가지 연산 지원**  - 덧셈, 뺄셈, 곱셈, 나눗셈을 지원합니다.  **3.서버 응답**  - 서버 응답은 수식의 정답 또는 오류 메시지입니다.  **4.다중 클라이언트 처리**  - 서버는 ThreadPool과 Runnable 인터페이스를 이용하여 동시에 여러 클라이언트를 처리합니다.  **5.서버 정보 파일**  - 클라이언트는 서버의 IP와 포트 번호 정보를 담은 텍스트 파일을 읽어 연결을 시도합니다.  **추가 설명**  서버 정보 파일은 "server\_info.dat"으로 가정되며, 클라이언트에서 읽어들여 동적으로 서버 정보를 얻을 수 있습니다. |

# REPORT

# architecture design diagram



# PRotocol

|  |  |
| --- | --- |
|  | **프로토콜**  **1.클라이언트에서 서버로의 요청 메시지:**  - 형식: 수학 표현식(예: "2 + 3")을 포함하는 문자열  - 예시: "2 + 3"  - 설명: 클라이언트는 서버에 계산을 요청하기 위해 수학 표현식을 문자열로 보냅니다.  **2. 서버에서 클라이언트로의 응답 메시지:**  - 형식: 계산 결과 또는 오류 메시지를 포함하는 문자열  - 예시 1 (성공): "Result: 5.0"  - 예시 2 (오류): "Error: Division by zero"  - 설명: 서버는 계산 결과 또는 오류 메시지를 클라이언트에게 문자열로 응답합니다.  **3. 주의사항**  - 수학 표현식 포맷:  현재 예제에서는 두 피연산자와 연산자 사이에 공백이 있는 간단한 표현식만 지원합니다.  - 서버 응답: 서버 응답은 성공적인 결과일 경우 "Result: 계산값"으로,  오류가 발생한 경우 "Error: 오류 메시지"로 구분됩니다.  - 클라이언트 요청: 클라이언트는 서버에게 계산을 요청하기 위해 유효한 수학 표현식을 전송합니다. |

## SOURCE CODES (Server)

|  |
| --- |
| **package** calculator;  **import** java.io.BufferedReader;  **import** java.io.IOException;  **import** java.io.InputStreamReader;  **import** java.io.PrintWriter;  **import** java.net.ServerSocket;  **import** java.net.Socket;  **import** java.util.concurrent.ExecutorService;  **import** java.util.concurrent.Executors;  **public** **class** Server {  **private** **static** **final** **int** ***PORT*** = 12345;  **public** **static** **void** main(String[] args) {  // Create a thread pool to handle multiple clients concurrently  ExecutorService executorService = Executors.*newFixedThreadPool*(10);  **try** (ServerSocket serverSocket = **new** ServerSocket(***PORT***)) {  System.***out***.println("Server is listening on port " + ***PORT***);  // Accept incoming client connections indefinitely  **while** (**true**) {  // Accept a client connection  Socket clientSocket = serverSocket.accept();  System.***out***.println("Accepted connection from " + clientSocket.getInetAddress());  // Submit the client handling task to the thread pool  executorService.submit(**new** ClientHandler(clientSocket));  }  } **catch** (IOException e) {  e.printStackTrace();  } **finally** {  // Shut down the thread pool when the server is done  executorService.shutdown();  }  }  }  // Runnable task to handle a connected client  **class** ClientHandler **implements** Runnable {  **private** Socket clientSocket;  **public** ClientHandler(Socket clientSocket) {  **this**.clientSocket = clientSocket;  }  @Override  **public** **void** run() {  **try** (  BufferedReader reader = **new** BufferedReader(**new** InputStreamReader(clientSocket.getInputStream()));  PrintWriter writer = **new** PrintWriter(clientSocket.getOutputStream(), **true**)  ) {  // Read the arithmetic expression from the client  String expression = reader.readLine();  System.***out***.println("Received expression from client: " + expression);  // Parse and calculate the expression, then send the result or an error message back to the client  **try** {  **double** result = calculateExpression(expression);  writer.println("Result: " + result);  } **catch** (ArithmeticException ex) {  writer.println("Error: " + ex.getMessage());  }  } **catch** (IOException e) {  e.printStackTrace();  }  }  // Helper method to parse and calculate the arithmetic expression  **private** **double** calculateExpression(String expression) {  // Implement expression parsing and calculation logic here    String[] tokens = expression.split(" ");  **double** operand1 = Double.*parseDouble*(tokens[0]);  String operator = tokens[1];  **double** operand2 = Double.*parseDouble*(tokens[2]);  **switch** (operator) {  **case** "+":  **return** operand1 + operand2;  **case** "-":  **return** operand1 - operand2;  **case** "\*":  **return** operand1 \* operand2;  **case** "/":  **if** (operand2 == 0) {  **throw** **new** ArithmeticException("Division by zero");  }  **return** operand1 / operand2;  **default**:  **throw** **new** IllegalArgumentException("Invalid operator: " + operator);  }  }  } |

## SOURCE CODES (Client)

|  |
| --- |
| **package** calculator;  **import** java.io.BufferedReader;  **import** java.io.IOException;  **import** java.io.InputStreamReader;  **import** java.io.PrintWriter;  **import** java.net.Socket;  **public** **class** Client {  **private** **static** **final** String ***SERVER\_HOST*** = "localhost";  **private** **static** **final** **int** ***SERVER\_PORT*** = 12345;  **public** **static** **void** main(String[] args) {  **try** (Socket socket = **new** Socket(***SERVER\_HOST***, ***SERVER\_PORT***);  BufferedReader userInputReader = **new** BufferedReader(**new** InputStreamReader(System.***in***));  BufferedReader serverReader = **new** BufferedReader(**new** InputStreamReader(socket.getInputStream()));  PrintWriter writer = **new** PrintWriter(socket.getOutputStream(), **true**)  ) {  System.***out***.println("Connected to server.");  // Prompt the user to enter an arithmetic expression  System.***out***.print("Enter arithmetic expression (e.g., 2 + 3): ");  String expression = userInputReader.readLine();  // Send the expression to the server  writer.println(expression);  // Receive and print the server's response  String response = serverReader.readLine();  System.***out***.println("Server response: " + response);  } **catch** (IOException e) {  e.printStackTrace();  }  }  } |

## OUTPUT (Screen Shots)

|  |  |
| --- | --- |
| #1 | 4 x 5의 연산 |
|  | |
| #2 | **2023 / 11 의 연산** |
|  | |
| #3 | **문자열을 입력한 경우** |
|  | |