ㅣ

PARK, JEONGWAN (이름)

# github URL (optional):

|  |  |
| --- | --- |
|  | [과제의 목적을 간단하게 설명]  https://github.com/musicmalgon |

# #개요

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

**Requirements**

* < 요구조건 #1: 간략한 핵심 요구 조건 설명>
* 어플리케이션의 구조를 그려야함 메시지 포맷을 설명해야함

# REPORT

* 과제의 기본 요구 사항을 이곳에 원하는 만큼 작성 (e.g. 구조도, protocol 내용 상세 설명 등)
* Procotol에 대한 설명
* 소스코드는 마지막에 제공하는 양식을 이용 하여 제출 (zip파일로도 제출)

# PRotocol

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

애플리케이션 계층은 컴퓨터 네트워크에서 인터넷 프로토콜 컴퓨터 네트워크를 통하는 프로세스 간 통신 접속을 위해 설계되어 통신 프로토콜과 방식을 위해 보유된 추상 계층이다.

TCP/IP 네트워크는 애플리케이션 계층에 2개의 전송 프로토콜 , TCP와 UDP를 사용하며 TCP를 더 많이 이용한다.

## SOURCE CODES

|  |
| --- |
| import java.io.\*;  import java.net.\*;  import java.util.Properties;  public class CalculatorClient {      public static void main(String[] args) throws IOException {          Properties config = new Properties();          //config.load(new FileInputStream("server\_info.dat"));          String serverAddress = config.getProperty("server.address", "localhost");          int serverPort = Integer.parseInt(config.getProperty("server.port", "1234"));          try (Socket socket = new Socket(serverAddress, serverPort)) {              BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()));              PrintWriter out = new PrintWriter(socket.getOutputStream(), true);              out.println("ADD 20 30");              System.out.println(in.readLine());          }      }  }  import java.io.\*;  import java.net.\*;  import java.util.concurrent.\*;  public class CalculatorServer {      private static final int PORT = 1234;      public static void main(String[] args) throws IOException {          ExecutorService pool = Executors.newFixedThreadPool(20);          try (ServerSocket server = new ServerSocket(PORT)) {              while (true) {                  Socket connection = server.accept();                  Callable<Void> task = new CalculatorTask(connection);                  pool.submit(task);              }          }      }  }  import java.io.\*;  import java.net.\*;  import java.util.Scanner;  import java.util.StringTokenizer;  import java.util.concurrent.\*;  class CalculatorTask implements Callable<Void> {      private Socket connection;      CalculatorTask(Socket connection) {          this.connection = connection;      }      @Override      public Void call() {          try {              try {                  InputStream in = connection.getInputStream();                  OutputStream out = connection.getOutputStream();                  try (Scanner scanner = new Scanner(in)) {                      PrintWriter writer = new PrintWriter(out, true);                      while (scanner.hasNextLine()) {                          writer.println(evaluate(scanner.nextLine()));                      }                  }              } finally {                  connection.close();              }          } catch (IOException ex) {              System.err.println(ex);          }          return null;      }      private static String evaluate(String input) {          StringTokenizer st = new StringTokenizer(input);          String operator = st.nextToken();          double operand1 = Double.parseDouble(st.nextToken());          double operand2 = Double.parseDouble(st.nextToken());          double result;          switch (operator) {              case "ADD":                  result = operand1 + operand2;                  break;              case "SUB":                  result = operand1 - operand2;                  break;              case "MUL":                  result = operand1 \* operand2;                  break;              case "DIV":                  if (operand2 != 0) {                      result = operand1 / operand2;                  } else {                      return "Error: Division by zero";                  }                  break;              default:                  return "Error: Unknown operator";          }          return "Result: " + result;      }  }  Server\_info.dat  server.address=localhost  server.port=1234 |

// 위 내용 지우고 사용. source 코드 별로 위 양식을 복사하여 사용

## OUTPUT (Screen Shots)

|  |  |
| --- | --- |
| #1 | [설명] |
|  | |

//(본인이 실행한 화면을 출력)  
//(최소 3개 이상 또는 필요한 만큼 더 생성하여 실행 결과를 잘 파악할 수 있도록 제시)   
// (Screen shot이 없는 경우 생략 가능)