ㅣ

KIm, DONG HYEON (김동현)

# github URL (optional):

|  |  |
| --- | --- |
|  | https://github.com/devxlofi/quizgame |

# #개요

|  |  |
| --- | --- |
|  | Java Swing과 소켓 프로그래밍을 활용하여 개발한 실시간 멀티플레이어 네트워크 퀴즈 게임입니다. 이 과제는 멀티스레드 기반의 서버-클라이언트 아키텍처와 직관적인 사용자 인터페이스를 구현하고, 효율적인 네트워크 통신 프로토콜을 설계하는 것을 목표로 합니다. |

**Requirements**

실시간 퀴즈 게임 시스템 사용자 인터페이스: Java Swing을 이용하여 직관적이고 반응성이 좋은 UI 구현

네트워크 통신: Socket Programming과 멀티스레딩을 활용하여 안정적인 클라이언트-서버 통신 설계

멀티플레이 지원: 최대 10명의 동시 접속 및 게임 진행 가능

정답 피드백 시스템: 즉각적인 정답 여부 확인 및 최종 점수 집계

# REPORT

**과제의 기본 요구 사항**

**시스템 아키텍처**

src/

├── App.java # 메인 애플리케이션 진입점

├── QuizServer.java # 서버 로직 및 문제 관리

├── ClientHandler.java # 클라이언트 세션 관리

├── QuizClient.java # 클라이언트 네트워크 로직

├── QuizGameGUI.java # 사용자 인터페이스

└── Question.java # 문제 데이터 모델

**주요 기능 설명**

* **서버 (QuizServer)**
  + 최대 10명의 동시 접속 클라이언트 지원
  + 기본 포트: 1234
  + 문제 풀 관리 및 문제 배포
  + 클라이언트 응답 처리 및 채점
* **클라이언트 (QuizClient + QuizGameGUI)**
  + 스플래시 화면 제공
  + 실시간 퀴즈 인터페이스 구현
  + 정답 제출 및 즉각적인 피드백
  + 최종 결과 표시 및 게임 재시작 기능

# PRotocol

|  |  |
| --- | --- |
|  | **클라이언트 → 서버**   * START: 게임 시작 요청 * ANSWER:{답변}: 사용자 답변 제출   **서버 → 클라이언트**   * QUESTION:{문제}: 새로운 문제 전송 * CORRECT:{정답}: 정답 피드백 * INCORRECT:{정답}: 오답 피드백 * FINAL\_SCORE:{점수}/{총문제수}: 최종 결과   텍스트, 스크린샷, 도표, 폰트이(가) 표시된 사진  자동 생성된 설명 |

## SOURCE CODES

|  |
| --- |
| **public class App {**  **public static void main(String[] args) {**  **if (args.length > 0 && args[0].equals("server")) {**  **QuizServer server = new QuizServer();**  **server.start();**  **} else {**  **QuizClient client = new QuizClient();**  **client.start();**  **}**  **}**  **}** |
| **import java.io.BufferedReader;**  **import java.io.PrintWriter;**  **import java.io.InputStreamReader;**  **import java.util.List;**  **import java.net.Socket;**  **import java.io.IOException;**  **class ClientHandler implements Runnable {**  **private Socket clientSocket;**  **private List<Question> questions;**  **private PrintWriter out;**  **private BufferedReader in;**  **public ClientHandler(Socket socket, List<Question> questions) {**  **this.clientSocket = socket;**  **this.questions = questions;**  **}**  **@Override**  **public void run() {**  **try {**  **out = new PrintWriter(clientSocket.getOutputStream(), true);**  **in = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));**  **String command;**  **int currentQuestion = 0;**  **int score = 0;**  **while ((command = in.readLine()) != null) {**  **if (command.equals("START")) {**  **currentQuestion = 0;**  **score = 0;**  **// 첫 문제 전송**  **out.println("QUESTION:" + questions.get(currentQuestion).getQuestion());**  **} else if (command.startsWith("ANSWER:")) {**  **String answer = command.substring(7).trim();**  **Question question = questions.get(currentQuestion);**    **if (answer.equalsIgnoreCase(question.getAnswer())) {**  **score++;**  **out.println("CORRECT:" + question.getAnswer());**  **} else {**  **out.println("INCORRECT:" + question.getAnswer());**  **}**    **currentQuestion++;**    **// 모든 문제를 다 풀었는지 확인**  **if (currentQuestion >= questions.size()) {**  **out.println("FINAL\_SCORE:" + score + "/" + questions.size());**  **} else {**  **// 잠시 대기 후 다음 문제 전송**  **Thread.sleep(2000);**  **out.println("QUESTION:" + questions.get(currentQuestion).getQuestion());**  **}**  **}**  **}**  **} catch (IOException | InterruptedException e) {**  **System.err.println("클라이언트 처리 중 에러: " + e.getMessage());**  **}**  **}**  **}** |
| **class Question {**  **private String question;**  **private String answer;**  **public Question(String question, String answer) {**  **this.question = question;**  **this.answer = answer;**  **}**  **public String getQuestion() {**  **return question;**  **}**  **public String getAnswer() {**  **return answer;**  **}**  **}** |
| **import java.io.\*;**  **import java.net.\*;**  **import javax.swing.SwingUtilities;**  **public class QuizClient {**  **private static final String SERVER\_ADDRESS = "localhost";**  **private static final int SERVER\_PORT = 1234;**  **private Socket socket;**  **private PrintWriter out;**  **private BufferedReader in;**  **private QuizGameGUI gui;**  **public QuizClient() {**  **gui = new QuizGameGUI(this);**  **}**  **public void start() {**  **try {**  **socket = new Socket(SERVER\_ADDRESS, SERVER\_PORT);**  **out = new PrintWriter(socket.getOutputStream(), true);**  **in = new BufferedReader(new InputStreamReader(socket.getInputStream()));**    **// 서버로부터 메시지를 받는 스레드 시작**  **new Thread(this::receiveMessages).start();**  **} catch (IOException e) {**  **System.err.println("서버 연결 실패: " + e.getMessage());**  **}**  **}**  **private void receiveMessages() {**  **try {**  **String message;**  **while ((message = in.readLine()) != null) {**  **processMessage(message);**  **}**  **} catch (IOException e) {**  **System.err.println("서버로부터 메시지 수신 실패: " + e.getMessage());**  **}**  **}**  **private void processMessage(String message) {**  **if (message.startsWith("QUESTION:")) {**  **String question = message.substring(9);**  **SwingUtilities.invokeLater(() -> gui.setQuestion(question));**  **} else if (message.startsWith("CORRECT:") || message.startsWith("INCORRECT:")) {**  **boolean isCorrect = message.startsWith("CORRECT:");**  **String correctAnswer = message.substring(message.indexOf(":") + 1);**  **SwingUtilities.invokeLater(() -> gui.showAnswer(isCorrect, correctAnswer));**  **} else if (message.startsWith("FINAL\_SCORE:")) {**  **String score = message.substring(12);**  **SwingUtilities.invokeLater(() -> gui.showResultScreen(score));**  **}**  **}**  **public void startGame() {**  **out.println("START");**  **gui.showQuizScreen();**  **}**  **public void sendAnswer(String answer) {**  **out.println("ANSWER:" + answer);**  **}**  **public void requestNextQuestion() {**  **out.println("NEXT");**  **}**  **public void restartGame() {**  **startGame();**  **}**  **}** |
| **import javax.swing.\*;**  **import java.awt.\*;**  **import java.awt.event.\*;**  **public class QuizGameGUI {**  **private JFrame mainFrame;**  **private CardLayout cardLayout;**  **private JPanel cardPanel;**  **private JPanel splashPanel;**  **private JPanel quizPanel;**  **private JPanel resultPanel;**    **private JLabel questionLabel;**  **private JTextField answerField;**  **private JButton submitButton;**  **private JLabel resultLabel;**  **private JLabel scoreLabel;**  **private QuizClient client;**  **public QuizGameGUI(QuizClient client) {**  **this.client = client;**  **initializeGUI();**  **}**  **private void initializeGUI() {**  **mainFrame = new JFrame("네트워크 퀴즈 게임");**  **mainFrame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);**  **mainFrame.setSize(600, 400);**  **mainFrame.setLocationRelativeTo(null);**  **cardLayout = new CardLayout();**  **cardPanel = new JPanel(cardLayout);**  **createSplashScreen();**  **createQuizPanel();**  **createResultPanel();**  **cardPanel.add(splashPanel, "SPLASH");**  **cardPanel.add(quizPanel, "QUIZ");**  **cardPanel.add(resultPanel, "RESULT");**  **mainFrame.add(cardPanel);**  **showSplashScreen();**  **}**  **private void createSplashScreen() {**  **splashPanel = new JPanel(new BorderLayout());**  **splashPanel.setBackground(new Color(41, 128, 185));**  **JLabel titleLabel = new JLabel("네트워크 퀴즈 게임", SwingConstants.CENTER);**  **titleLabel.setFont(new Font("맑은 고딕", Font.BOLD, 32));**  **titleLabel.setForeground(Color.WHITE);**  **JButton startButton = new JButton("게임 시작");**  **startButton.setFont(new Font("맑은 고딕", Font.BOLD, 18));**  **startButton.addActionListener(e -> client.startGame());**  **JPanel buttonPanel = new JPanel();**  **buttonPanel.setOpaque(false);**  **buttonPanel.add(startButton);**  **splashPanel.add(titleLabel, BorderLayout.CENTER);**  **splashPanel.add(buttonPanel, BorderLayout.SOUTH);**  **}**  **private void createQuizPanel() {**  **quizPanel = new JPanel(new BorderLayout(20, 20));**  **quizPanel.setBorder(BorderFactory.createEmptyBorder(30, 30, 30, 30));**  **questionLabel = new JLabel("", SwingConstants.CENTER);**  **questionLabel.setFont(new Font("맑은 고딕", Font.BOLD, 24));**  **answerField = new JTextField();**  **answerField.setFont(new Font("맑은 고딕", Font.PLAIN, 18));**    **submitButton = new JButton("제출");**  **submitButton.setFont(new Font("맑은 고딕", Font.BOLD, 16));**    **submitButton.addActionListener(e -> {**  **String answer = answerField.getText().trim();**  **if (!answer.isEmpty()) {**  **client.sendAnswer(answer);**  **}**  **});**  **answerField.addActionListener(e -> {**  **String answer = answerField.getText().trim();**  **if (!answer.isEmpty()) {**  **client.sendAnswer(answer);**  **}**  **});**  **JPanel inputPanel = new JPanel(new BorderLayout(10, 0));**  **inputPanel.add(new JLabel("A. "), BorderLayout.WEST);**  **inputPanel.add(answerField, BorderLayout.CENTER);**  **inputPanel.add(submitButton, BorderLayout.EAST);**  **quizPanel.add(questionLabel, BorderLayout.CENTER);**  **quizPanel.add(inputPanel, BorderLayout.SOUTH);**  **}**  **private void createResultPanel() {**  **resultPanel = new JPanel(new BorderLayout(20, 20));**  **resultPanel.setBorder(BorderFactory.createEmptyBorder(30, 30, 30, 30));**  **scoreLabel = new JLabel("", SwingConstants.CENTER);**  **scoreLabel.setFont(new Font("맑은 고딕", Font.BOLD, 32));**  **JButton restartButton = new JButton("다시 시작");**  **restartButton.setFont(new Font("맑은 고딕", Font.BOLD, 18));**  **restartButton.addActionListener(e -> client.restartGame());**  **JPanel buttonPanel = new JPanel();**  **buttonPanel.add(restartButton);**  **resultPanel.add(scoreLabel, BorderLayout.CENTER);**  **resultPanel.add(buttonPanel, BorderLayout.SOUTH);**  **}**  **public void showSplashScreen() {**  **cardLayout.show(cardPanel, "SPLASH");**  **mainFrame.setVisible(true);**  **}**  **public void showQuizScreen() {**  **cardLayout.show(cardPanel, "QUIZ");**  **}**  **public void showResultScreen(String score) {**  **scoreLabel.setText("최종 점수: " + score);**  **cardLayout.show(cardPanel, "RESULT");**  **}**  **public void setQuestion(String question) {**  **questionLabel.setText("Q. " + question);**  **answerField.setText("");**  **answerField.setEnabled(true);**  **submitButton.setEnabled(true);**  **answerField.requestFocus();**  **}**  **public void showAnswer(boolean isCorrect, String correctAnswer) {**  **answerField.setEnabled(false);**  **submitButton.setEnabled(false);**    **if (isCorrect) {**  **questionLabel.setText("정답입니다! 👍\n정답: " + correctAnswer);**  **} else {**  **questionLabel.setText("틀렸습니다! 😢\n정답: " + correctAnswer);**  **}**  **}**  **public JTextField getAnswerField() {**  **return answerField;**  **}**  **public JButton getSubmitButton() {**  **return submitButton;**  **}**  **}** |
| **import java.io.\*;**  **import java.net.\*;**  **import java.util.\*;**  **import java.util.concurrent.\*;**  **public class QuizServer {**  **private static final int DEFAULT\_PORT = 1234;**  **private static final int THREAD\_POOL\_SIZE = 10;**  **private List<Question> questions;**  **private ExecutorService executorService;**  **public QuizServer() {**  **questions = initializeQuestions();**  **executorService = Executors.newFixedThreadPool(THREAD\_POOL\_SIZE);**  **}**  **private List<Question> initializeQuestions() {**  **List<Question> questions = new ArrayList<>();**  **questions.add(new Question("대한민국의 수도는?", "서울"));**  **questions.add(new Question("1 + 1 = ?", "2"));**  **questions.add(new Question("지구의 위성은?", "달"));**  **// 더 많은 질문 추가 가능**  **return questions;**  **}**  **public void start() {**  **try (ServerSocket serverSocket = new ServerSocket(DEFAULT\_PORT)) {**  **System.out.println("서버가 포트 " + DEFAULT\_PORT + "에서 시작되었습니다.");**    **while (true) {**  **Socket clientSocket = serverSocket.accept();**  **executorService.execute(new ClientHandler(clientSocket, questions));**  **}**  **} catch (IOException e) {**  **System.err.println("서버 에러: " + e.getMessage());**  **}**  **}**  **}** |

## OUTPUT (Screen Shots)

|  |  |
| --- | --- |
| #1 | 메인 스플래시 화면 |
| 텍스트, 스크린샷, 운영 체제, 소프트웨어이(가) 표시된 사진  자동 생성된 설명 | |
| #2 | **첫 퀴즈 질문** |
| 텍스트, 스크린샷, 멀티미디어, 폰트이(가) 표시된 사진  자동 생성된 설명 | |
| #3 | **정답을 맞출경우** |
| 텍스트, 스크린샷, 멀티미디어, 소프트웨어이(가) 표시된 사진  자동 생성된 설명 | |
| #4 | **정답을 틀릴경우** |
| 텍스트, 스크린샷, 멀티미디어, 소프트웨어이(가) 표시된 사진  자동 생성된 설명 | |
| #5 | **최종점수** |
| 텍스트, 스크린샷, 폰트, 운영 체제이(가) 표시된 사진  자동 생성된 설명 | |
| #6 | | **서버 터미널** |
| 텍스트, 스크린샷, 폰트이(가) 표시된 사진  자동 생성된 설명 | | |