Computer Networking
Socket Program Assignment #1

Quiz Game over the Cloud

Due:

2024.11.16 (Sat) 23:59

HW#1: Quiz Game over the Cloud (30pt)

- Create your first network application

- **Develop a simple quiz game application using Java sockets, where the client can connect to the server to play a text-based quiz.**

- Define an "application-layer" protocol (communication message formats) for this application

Details

- Overview
  - Create a client-server quiz game where the server hosts a set of questions, and the client answers each question in real-time.
  - The server evaluates responses and keeps track of the score.

- Protocol for Command/Response Interaction
  - You should define ASCII-code based message formats
  - Commands and responses formats
    - Refer (참조): HTTP request/response formats
      - https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods
      - https://developer.mozilla.org/en-US/docs/Web/HTTP/Status

Details

- **Client-Side:**
  - Connect to the server and start the quiz.
  - Receive questions one at a time from the server and provide answers.
  - Display feedback from the server (e.g., “Correct!” or “Incorrect”) after each answer.
  - At the end of the quiz, receive the final score from the server.

- **Server-Side:**
  - Store a set of questions and answers (could be as simple as an array or list).
  - Send questions to the client and wait for responses.
  - Evaluate each response, update the client’s score, and send feedback on the answer.
  - After all questions are answered, send the final score to the client and close the connection.

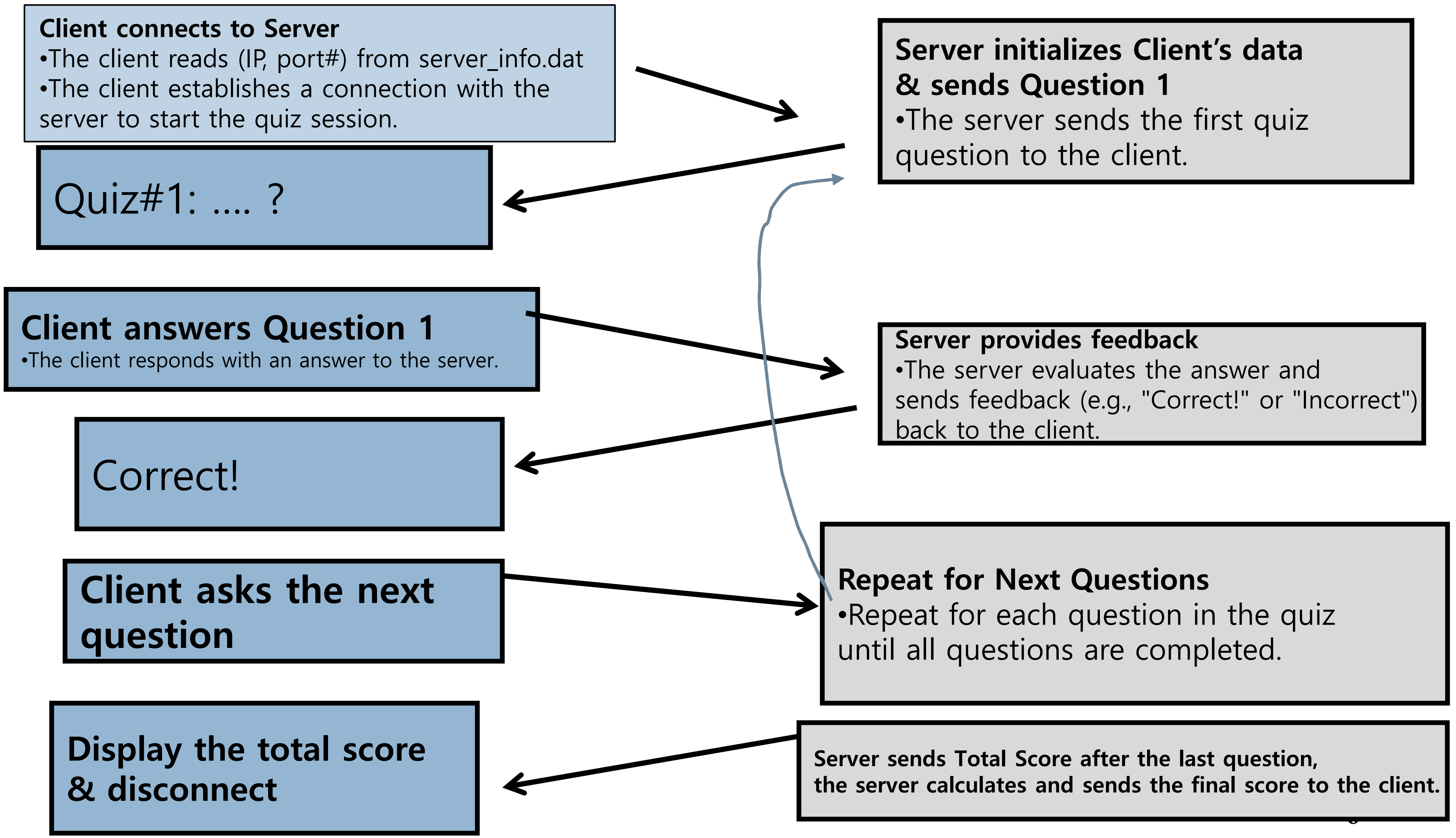
Requirements

- **Client-side configuration:**
  - Server connection details (IP and port) are stored in a configuration file (e.g., server_info.dat).
  - The client program reads these details from the file when the program starts.
  - If the file is missing, default values (e.g., localhost and port 1234) are used.

- **(Optional) Multi-client Support (자기 주도 학습):**
  - **The server <u>can handle multiple clients at a time</u>**
    - **Hint:** Use **ThreadPool** & **Runable** interface

Example Scenarios

- Client :
- Server:



Client connects to Server
- The client reads (IP, port#) from server_info.dat
- The client establishes a connection with the server to start the quiz session.

Server initializes Client's data & sends Question 1
- The server sends the first quiz question to the client.

Quiz#1: .... ?

Client answers Question 1
- The client responds with an answer to the server.

Server provides feedback
- The server evaluates the answer and sends feedback (e.g., "Correct!" or "Incorrect") back to the client.

Correct!

Client asks the next question

Repeat for Next Questions
- Repeat for each question in the quiz until all questions are completed.

Display the total score & disconnect

Server sends Total Score after the last question, the server calculates and sends the final score to the client.

Grading: 40 points

세부 기능 : 상 / 중 / 하 배점

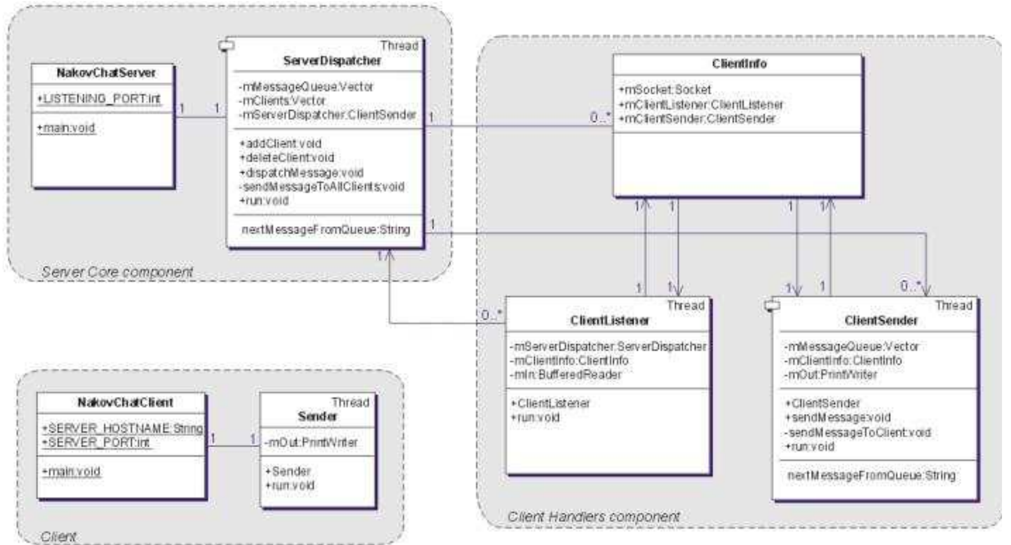
- 기본 작동/연결 : **12** / 6 / 0 points
  - 서버 Thread 처리: 보너스 +6점
  - 프로토콜 기반 메시지 전송/수신/해석 처리: 6점
- Quiz 전달&정답, 예외처리: **8** / 4 / 0 points
  - 기능: 4점, 예외처리 여부: 4점
- **Protocol 정의 문서: 8 / 4 / 2 points**
- **Report (구조도, 주석, 스크린샷) : 12 / 6 / 3 points**
- Any violation of submission guideline
  - File name or file format
  - -5 points
- No late submission is allowed
  - Late submission : 0 point

What should you do also ?

- 1. Report (written in Korean)
  - Draw the architecture design diagram of your application
  - Explain the message formats (protocol for your application)
  - Should include:
    - **all your source codes be attached on the tail of the template document**
    - **Screenshots (at least three output cases)**

- 2. Comments
  in source files

Example of an architecture diagram:



Submission Guideline

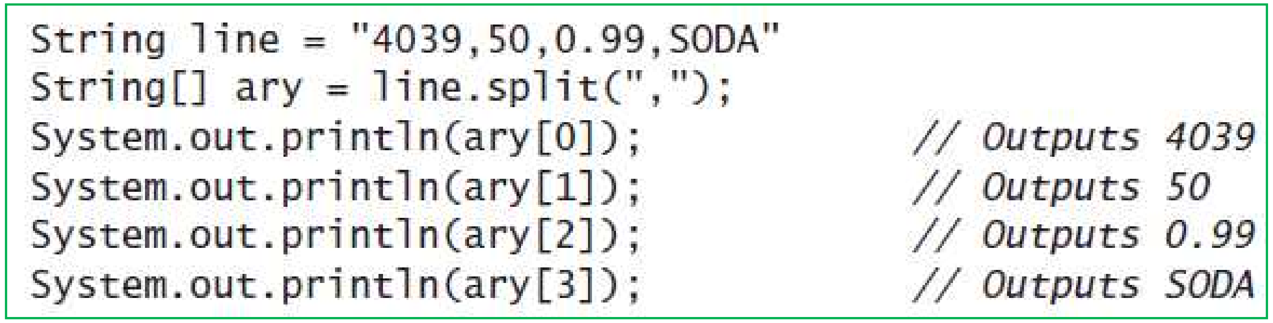
- Submission
  - 1. One report file (템플릿 제공)
    - HW1_report_YOURNAME.docx
  - 2. **All source code should be compressed into a single zip file**
    - HW1_src_YourNAME.zip
  - 3. **1,2를 하나의 zip 파일로 압축해서 제출 /**
    compress 1 and 2 into a single zip file for submission
    - HW1_YourNAME.zip

- Option: GitHub for version control of the project (~10pt)
  - Use Github for your project and Create a wiki
  - In this case, you will be eligible for a maximum of 10 bonus points
  - Report 첫 줄에 github 주소 기재
    Include the GitHub address on the first line of the report.

End.

Appendix: Processing a CSV File

- View program that calculates total sales, listing 10.4 **`class TransactionReader`**
- Uses the **`split`** method (in String class) which puts strings separated by a delimiter into an array



```
String line = "4039,50,0.99,SODA"
String[] ary = line.split(",");
System.out.println(ary[0]);           // Outputs 4039
System.out.println(ary[1]);           // Outputs 50
System.out.println(ary[2]);           // Outputs 0.99
System.out.println(ary[3]);           // Outputs SODA
```

Appendix: Reading Words in a String: Using `StringTokenizer` Class

- **`StringTokenizer`** can be used to parse a line into words
  - import `java.util.*`
  - some of its useful methods are shown in the text
    - e.g. test if there are more tokens
  - you can specify *delimiters* (the character or characters that separate words)
    - the default delimiters are "white space" (space, tab, and newline)

Appendix:
Example: StringTokenizer

- Display the words separated by any of the following characters: space, new line (₩n), period (.) or comma (,).

```
String inputLine = keyboard.nextLine();
StringTokenizer wordFinder =
            new StringTokenizer(inputLine, " \n.,");
//the second argument is a string of the 4 delimiters
while(wordFinder.hasMoreTokens())
{
   System.out.println( wordFinder.nextToken() );
}
```

Entering "`Question,2b.or !tooBee.`" gives this output:

```
Question
2b
or
!tooBee
```