Trivia Game Application

Maier Georgiana Ștefaina

1. Project Overview

- **Description**: A client-server trivia game where players answer random general knowledge questions. The server distributes questions, checks answers, and provides feedback with the correct answers.

2. Used Technologies

- **Python**: Server application. (Linux)
- **Java**: Client application. (Windows)
- **Docker**: Containerizes the server for consistent deployment.
- **JSON**: Stores questions and answers in `questions.json`.
- **Batch Script (`.bat`)**: Automates Docker commands for starting, stopping, and rebuilding the Docker container.

3. Application Architecture

- **Client-Server Model**: Server distributes questions to each client, validates answers, and provides feedback.
- **Random Question Selection**: Each session delivers 5 random questions from the full pool.
- **Communication Protocol**: Uses TCP sockets for communication between the server and the client.

4. Files and Directory Structure

- **TriviaServer.py**: The server code in Python.
- **TriviaClient.java**: The client code in Java.
- **questions.json**: JSON file containing the questions and answers.
- **Dockerfile**: Dockerfile to create the Docker image.
- **run_trivia_server.bat**: Batch script to automate Docker container management.

Example structure:

```
/trivia-game
├── TriviaServer.py
├── TriviaClient.java
├── questions.json
├── Dockerfile
├── run_trivia_server.bat
```

5. Setup Instructions

Prerequisites:

- Docker, Python 3.x and Java are required.

Server Setup Using the `.bat` Script:

- A batch script `run_trivia_server.bat` is provided to automate server setup in Docker on Windows.

run_trivia_server.bat:

```
@echo off
echo Stopping and removing any existing trivia-server containers...

docker ps -q --filter "name=trivia-server" | findstr . >nul
&& docker stop trivia-server
docker ps -aq --filter "name=trivia-server" | findstr . >nul
&& docker rm trivia-server

echo Building the trivia-server Docker image...
docker build -t trivia-server .

echo Running the trivia-server Docker container...
docker run -d -p 12345:12345 --name trivia-server trivia-server
```

Steps:

1. **Run the `.bat` Script**:
   - Double-click `run_trivia_server.bat` or run it from Command Prompt:

     ```
     .\run_trivia_server.bat
     ```

   - This will stop and remove any existing `trivia-server` container, rebuild the Docker image, and start the container.
2. **Prepare the Client**:
   - Compile the Java client:

     ```
     javac TriviaClient.java
     ```

   - Run the client:

     ```
     java TriviaClient
     ```

6. Usage Instructions

- **Gameplay**:

- After the client connects, it receives five questions.
- For each question, the client provides an answer and receives feedback with the correct answer.
- The final score is displayed at the end.

7. Code Explanation

- **TriviaServer.py**:

```
import socket
import threading
import time
import json
import random



class TriviaServer:
    def __init__(self, host='0.0.0.0', port=12345,
     question_file='questions.json'):
        self.server_socket = socket.socket(socket.AF_INET,
         socket.SOCK_STREAM)
        self.server_socket.bind((host, port))
        self.server_socket.listen(5)
        self.questions = self.load_questions(question_file)
        print("Server started and waiting for connections...")

    def load_questions(self, filename):
        with open(filename, 'r', encoding='utf-8') as file:
            return json.load(file)

    def handle_client(self, client_socket, client_address):
        print(f"Connected with {client_address}")
        score = 0

        selected_questions = random.sample(self.questions, 5)

        for q in selected_questions:
            print(f"Sending question: {q['question']}")
            client_socket.sendall((q["question"] + "\n").encode())
            print("Question sent, awaiting response...")

            try:
                response = client_socket.recv(1024).decode().strip()
                print(f"Response received: {response}")
                if response.lower() == q["answer"].lower():
                    score += 1
                    client_socket.sendall("Correct!\n".encode())
                else:
                    client_socket.sendall("Incorrect!\n".encode())
            except socket.error as e:
                print(f"Error receiving response: {e}")
                break

            time.sleep(1)

        client_socket.sendall(f"Final score: {score} out of 5\n".encode())
        client_socket.close()
        print(f"Connection with {client_address} has been closed")

    def run(self):
        while True:
            client_socket, client_address = self.server_socket.accept()
            client_thread = threading.Thread(target=self.handle_client,
             args=(client_socket, client_address))
            client_thread.start()


if __name__ == "__main__":
    server = TriviaServer()
    server.run()
```

- `load_questions`: Loads questions from `questions.json`.
- `handle_client`: Manages a client session, sends questions, checks responses, and gives feedback.
- `run`: Starts the server and waits for client connections.
- **TriviaClient.java**:

```
import java.io.*;
import java.net.Socket;
import java.nio.charset.StandardCharsets;
import java.util.Scanner;

public class TriviaClient {
    public static void main(String[] args) {
        String host = "127.0.0.1";
        int port = 12345;

        try (Socket socket = new Socket(host, port);
             BufferedReader in = new BufferedReader
             (new InputStreamReader(socket.getInputStream(),
              StandardCharsets.UTF_8));
             BufferedWriter out = new BufferedWriter
             (new OutputStreamWriter(socket.getOutputStream(),
              StandardCharsets.UTF_8));
             Scanner scanner = new Scanner(System.in)) {
```

```
            System.out.println("Connected to server,
            waiting for questions...");

            String question;
            while ((question = in.readLine()) != null) {

                if (question.startsWith("Final score:")) {
                    System.out.println(question);
                    System.out.println("Game over. " + question);
                    break;
                }

                System.out.println("Question: " + question);

                System.out.print("Your answer: ");
                String answer = scanner.nextLine();
                out.write(answer + "\n");
                out.flush();

                String feedback = in.readLine();
                if (feedback != null) {
                    System.out.println(feedback);
                } else {
                    break;
                }
            }

            System.out.println("Connection to server has been closed.");

        } catch (IOException e) {
            System.out.println("Connection to server failed: "
             + e.getMessage());
        }
    }
}
```

- Connects to the server on the specified host and port.
- Reads questions, accepts user input, and sends responses back to the server.
- Receives and displays feedback and the final score.

---

8. Dockerfile Explanation

The `Dockerfile` is used to build the Docker image for the trivia server, which encapsulates the server's code and dependencies, ensuring consistent deployment.

Dockerfile Contents:

```
FROM python:3.8-slim

WORKDIR /app

COPY TriviaServer.py .
COPY questions.json .

EXPOSE 12345

CMD ["python", "TriviaServer.py"]
```

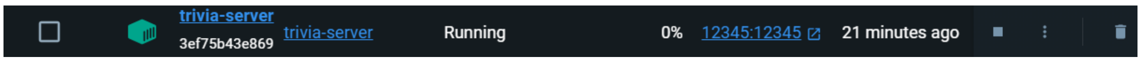
Explanation:

- `FROM python:3.8-slim`: This specifies the base image for the container, using a minimal version of Python 3.8 to keep the image lightweight.
- `WORKDIR /app`: Sets `/app` as the working directory inside the container. All subsequent commands will run from this directory.
- `COPY TriviaServer.py .` and `COPY questions.json .`: Copies `TriviaServer.py` and `questions.json` from your local directory to the `/app` directory inside the container.
- `EXPOSE 12345`: Informs Docker that the container listens on port `12345` at runtime.
- `CMD ["python", "TriviaServer.py"]`: Sets the command to start the Python trivia server when the container runs.

---

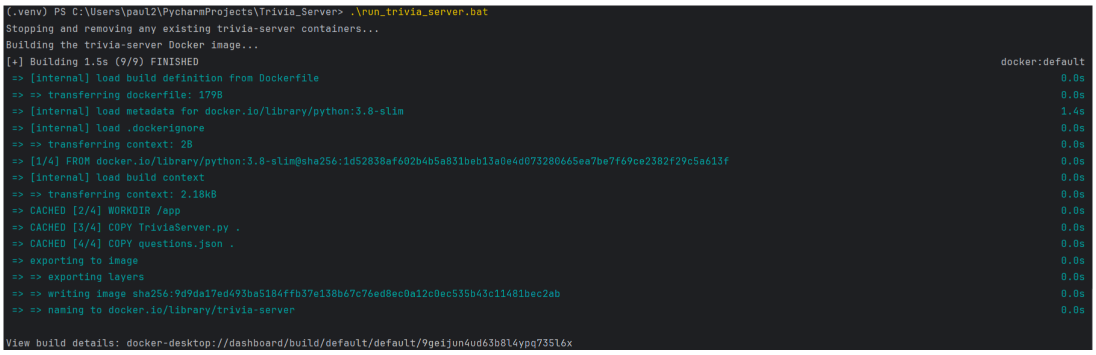
9. Screenshots

1. **Docker Container:**



```
trivia-server
3ef75b43e869   trivia-server   Running   0%   12345:12345   21 minutes ago
```

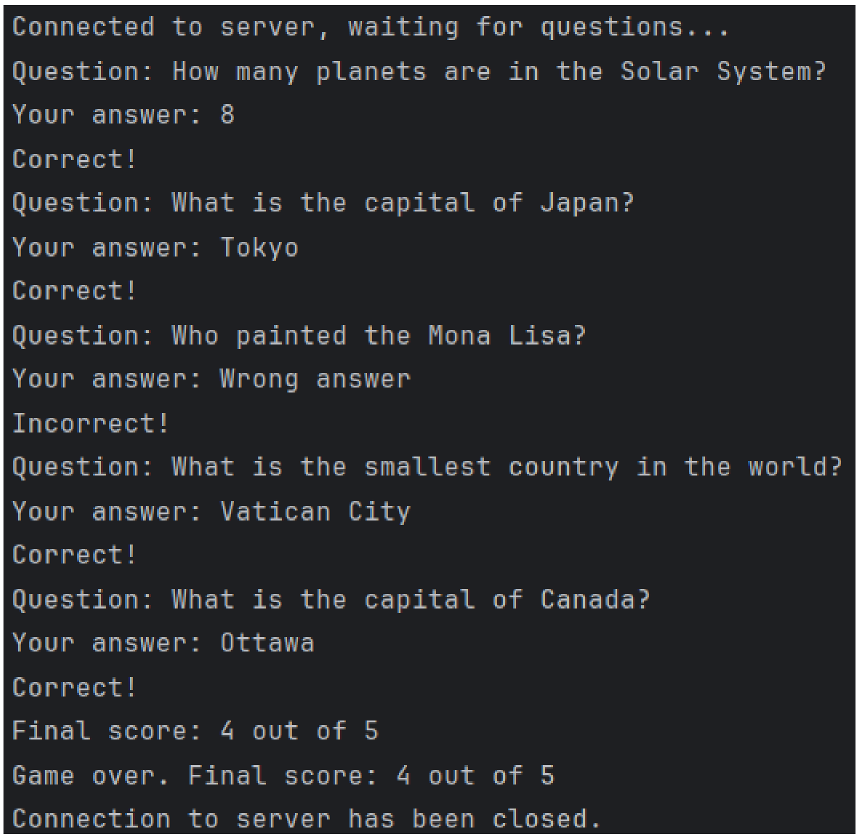
2. **Server Setup Screenshot**:



```
(.venv) PS C:\Users\paul2\PycharmProjects\Trivia_Server> .\run_trivia_server.bat
Stopping and removing any existing trivia-server containers...
Building the trivia-server Docker image...
[+] Building 1.5s (9/9) FINISHED                                                    docker:default
 => [internal] load build definition from Dockerfile                                          0.0s
 => => transferring dockerfile: 179B                                                          0.0s
 => [internal] load metadata for docker.io/library/python:3.8-slim                            1.4s
 => [internal] load .dockerignore                                                             0.0s
 => => transferring context: 2B                                                               0.0s
 => [1/4] FROM docker.io/library/python:3.8-slim@sha256:1d52838af602b4b5a831beb13a0e4d073280665ea7be7f69ce2382f29c5a613f  0.0s
 => [internal] load build context                                                             0.0s
 => => transferring context: 2.18kB                                                           0.0s
 => CACHED [2/4] WORKDIR /app                                                                 0.0s
 => CACHED [3/4] COPY TriviaServer.py .                                                       0.0s
 => CACHED [4/4] COPY questions.json .                                                        0.0s
 => exporting to image                                                                        0.0s
 => => exporting layers                                                                       0.0s
 => => writing image sha256:9d9da17ed493ba5184ffb37e138b67c76ed8ec0a12c0ec535b43c11481bec2ab  0.0s
 => => naming to docker.io/library/trivia-server                                              0.0s

View build details: docker-desktop://dashboard/build/default/default/9geijun4ud63b8l4ypq735l6x
```

3. **Client Connection and Gameplay**:


```
Connected to server, waiting for questions...
Question: How many planets are in the Solar System?
Your answer: 8
Correct!
Question: What is the capital of Japan?
Your answer: Tokyo
Correct!
Question: Who painted the Mona Lisa?
Your answer: Wrong answer
Incorrect!
Question: What is the smallest country in the world?
Your answer: Vatican City
Correct!
Question: What is the capital of Canada?
Your answer: Ottawa
Correct!
Final score: 4 out of 5
Game over. Final score: 4 out of 5
Connection to server has been closed.
```