OBJECT ORIENTED PROGRAMMING

QUIZ #4, #5, #6

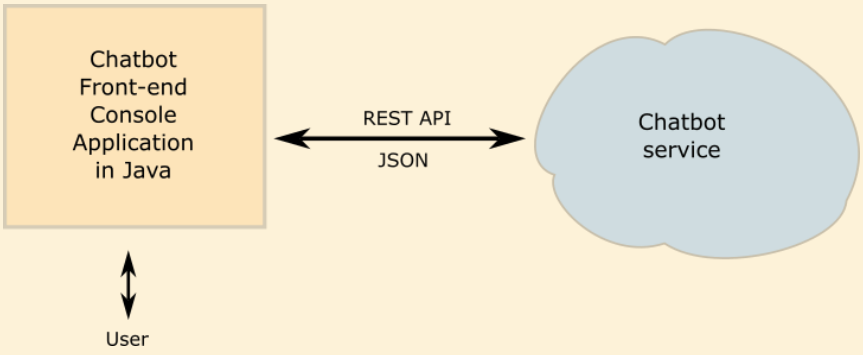
7.5 points in total

Rules:

- Answers should be submitted as GitHub repository link via email. The repository should be alive until the quizzes are evaluated.
- You have 3 hours in total. Late answers will not be evaluated. Please note that file upload dates will be checked in your repository, all files should be uploaded during the quiz.
- You are allowed to view materials in Google Classroom, use Search Engines such as Google, Bing, etc. and ask AI tools such as ChatGPT, Bard, etc.
- You are not allowed to have any type of communication (Mobile, Email, Social Network, etc.) with other humans during the quiz.
- The quiz will not be evaluated in case of cheating.

Assignment

Create a chatbot front-end console application in java. The application should be able to get the users messages from the console and answer them on the console as well. The application should use the remote chatbot service via REST API for getting the answers to the user messages. Assume that chatbot remote service is up and running already, you do not have to implement it.



To get the answer to the user message, the application should send the user message and the whole past chat history to the remote service of the chatbot. Communication should be implemented using POST request of the REST API.
See the example of the conversation below:

Chat Example	***Description for implementing the console application for chatbot front-end***
`User: Hello`	*the application should send this message to the chatbot remote service using POST method*
`Chatbot: Good day`	*the response of the POST request should be displayed on the screen with "Chatbot: " prefix*
`User: What time is it?`	*the application should send "What time is it?" and the past chat with "hello" and "Good day"*
`Chatbot: 9:00 AM`	*the response "9:00 AM" should be displayed on the screen with "Chatbot: " prefix*
`User: I should go!`	*the application should send "I should go!" and the all the past chat using the POST method*
`Chatbot: Wait for me!`	*the response "Wait for me!" should be displayed on the screen with "Chatbot: " prefix*

Quiz #4

2.5 points

1. Create and describe the JSON format for POST requests for the remote service of the chatbot. You can take any format that will be compatible for the assignment condition. (1.5 points).
2. Provide an example of the chat conversation and corresponding JSON representations for each of the POST request of the conversation. (1 point).

Upload the PDF file as the answer to this quiz.

Note that the format should be elaborated independently and if several students will use one and the same structure, the quiz will be left not evaluated for all of them.

Quiz #5

2.5 points

Create classes `UsrerInteractionManager` and `CommunicationManager`.
Class `UsrerInteractionManager` should interact with user. This class should use the class `CommunicationManager` to send and receive the information to and from the remote service of the chatbot. The JSON data that you have created for Quiz# 4 should be accepted by the `CommunicationManager` from the `UsrerInteractionManager`. Interaction with remote service should be implemented by the `CommunicationManager` using the REST API.

Tip: create `DummyCommunicationManager` in order to test your chat bot as there is no running service for the chatbot service.

All the classes for this quiz should be in the package `quiz5`.

Quiz #6

2.5 points

Create the `SpecialCommunicationManager` class with two endpoint strings - `commonServiceUrl` and `specialServiceUrl`. If user message or the conversation history contains the user message “help”, the request should be directed to the special service instead of the common chatbot service.

The class should be uploaded in the package `quiz6`.

Good Luck!