TP : Client/Serveur

Exercice 1: Knock-Knock Joke

The goal of this exercise is to implement a knock-knock joke server and corresponding client. Through the client, users can ask for knock-knock jokes and can teach the server new jokes.

Provision: The Knock-Knock protocol

The server will be started first and will simply listen for a connection request from a client. (For this assignment, we have only one client.) Once the client connection is established, the server enters a 'tell/learn/quit' mode. The description of the protocol assumes that anything sent by the server is received by the client and likewise by the server for anything sent by the client. Interactions must follow this pattern exactly:

1. client sends one of tell, learn, or quit
2. if tell:
    1. server sends Knock knock
    2. client sends Who's there?
    3. server sends the first part of a Knock-knock joke randomly selected from the server's joke-base
    4. client sends whatever the server sent with who? appended
    5. server sends punchline of joke
    6. server returns to tell/learn/quit mode
3. if learn:
    1. server sends ready
    2. client sends Knock knock
    3. server sends Who's there?
    4. client sends the first part of a Knock-knock joke
    5. server sends whatever the client sent with who? appended
    6. client sends punchline of joke
    7. server creates a new Knock-knock joke instance and adds it to the joke-base
    8. server returns to tell/learn/quit mode
4. if quit then close any open sockets and exit the program

Requirements

1. *Write a KnockKockJoke class*. This class need only model the two key features of a knock-knock joke. A knock-knock joke has a standard introduction; the agent telling the joke says "Knock-knock", and the other agent says "Who's there?". This exchange is irrelevant to any instance of a knock-knock joke as it is always the same. So the class you write only needs to capture the next two utterances of the joke-telling agent (as the response prior to the punchline is entirely scripted). You should provide a constructor that consumes the two String arguments that comprise a KnockKnockJoke instance.
2. *Create a KnockKnockServer class*. In the javadoc comments at the top of your class, include the details of our protocol. As described above, your server should listen for a connection from a client. Once a connection is established, the server should enter an indefinite loop where on each iteration one of three services is available: tell a joke, learn a joke, or quit. The client should signal which service is desired and the server should then provide the appropriate service. The server must include some type of collection for mananging its joke-base.
3. *Create a KnockKnockClient class*. Your client should make a connection with a server at a designated host. The host designation must be a *command-line input*; you can access this as the first element of the array of strings that is the argument in the main method. When you run your client program, you should provide the host name (or IP address) of the computer that is running your server. For development, you should probably run both the server and client on your own machine;

in this case the host name argument should be “localhost”. Your client should prompt the user for input to select one of the three services, present appropriate messages sent from the server, prompt the user for required input, and forward user input to the server as needed

Exercice 2: Knock-Knock Joke multi-client version

Exercice 3: Médiathèque

1. Définir un protocole de communication claire entre le serveur de la médiathèque et les différents kindles.
2. Implémenter le protocole que vous avez définit.