Instituto Politécnico Nacional

Escuela Superior de Cómputo

Programación Orientada a Objetos

Reporte de Práctica #6

Sockets Servidores

Profesor: Roberto Tecla Parra

Alumno: Calva Hernández José Manuel

Grupo: 2CM3

ChatBot

Codificar un cliente y un servidor que interactúen del siguiente modo el cliente envía una pregunta al servidor y el servidor envía una respuesta al cliente. El servidor puede almacenar al menos 10 preguntas y 10 respuestas predefinidas (se pueden usar 2 arreglos o un HashMap).

Ejemplos de preguntas tipo y respuestas tipo

En que ciudad vives?

D.F

Cuántos años tienes?

20

En que escuela estudias?

ESCOM

Objetivos

- ➢ Entender la funcionalidad del servidor como intermediario para el recibo y envío de objetos.
- ➢ Entender cada uno de los pasos del protocolo para socket cliente, con el objetivo de realizar una correcta conexión entre el servidor y el cliente.
- ➢ Mediante dos arreglos de tipo String guardar la información a evaluar en el Servidor, por medio de los objetos (cadenas) que enviará el Cliente para la conexión establecida entre ambos.
- ➢ Entender el manejo de flujos de entrada y salida para el traspaso de información y objetos.
- ➢ Hacer uso del manejo de errores mediante el catch y ejecutar todo lo que se requiera hacer en el try, funcionando como código "en un mundo ideal".

Desarrollo

Servidor:

Se empezó por importar las librerías necesarias para el desarrollo de la práctica:

```
import java.io.*;
import java.net.*;
import java.util.*;
```

java.net-Con los sockets de flujo, un proceso mantiene una comunicación con otro proceso. El flujo que se establece entre estos dos procesos es continuo, este tipo

de sockets proporcionan un servicio orientado a conexiones utilizando el protocolo TCP.

java.io- En Java podemos tener un flujo de entrada (en ingles: input stream) por el cual recibimos datos desde el exterior de nuestro programa, o un flujo de salida (en ingles: output stream) el cual envía nuestros datos hacia un cierto destino.

Luego se crearía el método main en la cual se declaró un HashMap que contendrá tanto preguntas como respuestas del chat bot, para que el Cliente enviase cualquier pregunta idéntica a la que está presente en el arreglo. Además, en el cual se crearían otras variables para el paso de información entre el Cliente y el propio Servidor, entre las cuales destacan el socket del servidor, el lector que sería propiamente un BufferedReader y serviría para leer la entrada de la cadena enviada por el cliente, el flujo y encontrada que sería un booleano para ver si la pregunta que envió el cliente se encontraba en el arreglo de preguntas del servidor.

```
public static void main(String[] args) {
    ServerSocket theServer;
    Socket theClient;
    BufferedReader in;
    PrintStream out;

    HashMap<String,String> map = new HashMap<>();
    map.put("HOLA" , "HOLA, MUCHO GUSTO");
    map.put("ADIOS" , "ADIOS, FUE UN PLACER");
    map.put("COMO TE LLAMAS?" , "ME LLAMO MANUEL");
    map.put("CUANTOS AÑOS TIENES?" , "22");
    map.put("DONDE ESTUDIAS?" , "EN ESCOM");
    map.put("QUE CLASE ES ESTA?" , "PROGRAMACION ORIENTADA A OBJETOS");
    map.put("QUE LENGUAJE UTILIZAS?" , "JAVA");
    map.put("COLOR FAVORITO?" , "ROJO");
    map.put("QUE MUSICA ESCUCHAS?" , "ROCK");
    map.put("A QUE HORA SALES?" , "3 DE LA TARDE");
    map.put("DONDE NACISTE?" , "CIUDAD DE MEXICO");
    map.put("COMIDA FAVORITA?" , "MOLE CON POLLO");
```

Después se haría el código de lo que se desea hacer dentro del try, el cual consistía en crear un nuevo socket para el servidor con el puerto 1025, luego se aceptaría la conexión en cuanto la hubiese con el cliente, y se bloquea el programa, se leería posteriormente la cadena que fue enviada por el cliente, para después hacer una búsqueda por medio de la key asociada a la pregunta, en caso de no haber respuesta asociada, se devuelve por defecto un: “LO SIENTO, NO TE ENTIENDO”.

```
try{
    theServer = new ServerSocket(answerPort);
    try{
        theClient = theServer.accept();
        System.out.println("Connection Established.\n");
        in = new BufferedReader(new InputStreamReader(theClient.getInputStream()));
        out = new PrintStream(theClient.getOutputStream());
        String theQuestion, theAnswer;
        while(true){
            theQuestion = in.readLine();
            if(theQuestion.equalsIgnoreCase("ADIOS")){
                out.println(map.get(theQuestion.toUpperCase()));
                break;
            }
            else if(map.containsKey(theQuestion.toUpperCase())){
                out.println(map.get(theQuestion.toUpperCase()));
            }
            else{
                out.println("LO SIENTO, NO TE ENTIENDO");
            }
        }
        theClient.close();
    } catch (IOException e){
        theServer.close();
        System.err.println(e);
    }
} catch (IOException e){
    System.err.println(e);
}
```

Cliente:

Para el cliente se importarían las mismas librarías que el Servidor y además la de util utilizando el método Scanner. Se crearía la clase cliente donde iniciando se crearía el método main que tendría el socket del cliente, el host y el flujo de salida. Se crearía el bloque try en el cual igualaríamos el socket del cliente a 1025 (el del servidor), crearíamos una variable de tipo impresión llamada escribe, crearíamos una variable de tipo String llamada entradaTeclado inicializada (""), para que posteriormente fuese cambiada por el cliente, que escribiría en la consola la pregunta deseada, esto con entradaEscaner que sería de tipo Scanner, así entradaTeclado se igualaría a lo que el cliente haya escrito en pantalla para mandarla a la variable escribe y así el lector (BufferedReader) leería lo que hay ahí y lo pondría en una variable de entrada getInputStream, finalmente se imprimiría en pantalla lo la respuesta que el servidor enviaría.

```java
public class answerClient{
    public final static int answerPort = 1025;

    public static void main(String[] args) {
        Socket theConnection;
        String hostname = "localhost";
        BufferedReader in;
        PrintStream out;

        try{
            theConnection = new Socket(hostname, answerPort);
            Scanner scan = new Scanner(System.in);
            out = new PrintStream(theConnection.getOutputStream());
            in = new BufferedReader(new InputStreamReader(theConnection.getInputStream()));
            String theQuestion, theAnswer;
            while(scan.hasNextLine()){
                theQuestion = scan.nextLine();
                out.println(theQuestion);
                theAnswer = in.readLine();
                System.out.println("The server says: "+theAnswer+"\n");
                if (theAnswer.length() == 20) {
                    return;
                }
            }
        } catch (UnknownHostException e) {
            System.err.println(e);
        } catch (IOException e) {
            System.err.println(e);
        }
    }
}
```

Conclusiones

Para lograr una buena comunicación entre el cliente y el servidor su host debe ser el mismo y a su vez el proceso de flujos de entrada los llevará acabo el cliente, los de salida el servidor. El manejo de envió de objetos es importante y por ello se debe establecer una comunicación entre cliente y servidor, en la cual se establecerá la conexión cuando haya un cliente que se haya podido comunicar y se cerrará cuando el proceso haya acabado.