/\*--------------------------------------------------------

1. Name / Date: Arturo Chaidez III September 24, 2020

2. Java version used, if not the official version for the class:

openjdk version "15" 2020-09-15

OpenJDK Runtime Environment (build 15+36-1562)

OpenJDK 64-Bit Server VM (build 15+36-1562, mixed mode, sharing)

3. Precise command-line compilation examples / instructions:

> javac JokeServer.java

> javac JokeClient.java

> javac JokeClientAdmin.java

4. Precise examples / instructions to run this program:

Run in 3 separate terminals:

> java JokeServer.java

> java JokeClient.java

> java JokeClientAdmin.java

Cannot run a secondary server

In separate shell windows:

> java JokeServer

> java JokeClient

> java JokeClientAdmin

5. List of files needed for running the program.

1. JokeServer.java

2. JokeClient.java

3. JokeClientAdmin.java

4. JokeLog.txt

5. checklist-joke.html.

5. Notes:

Do not have a secondary server. When switching between jokes/proverbs, it does

save and returns to the proper list order. None of the MultipleServers checklist has been done.

----------------------------------------------------------\*/

import java.io.\*;

import java.net.\*;

import java.util.Arrays;

import java.util.HashMap;

import java.util.Random;

//format goes:

//From lectures, using HashMaps to store jokes and proverbs

/\*Suggestions from classmates on D2L, Store on a HashMap. \*/

//store jokes/proverbs on Server. Maybe it can be done in ServerThread?

public class JokeServer {

/\* Setting- allows admin to switch between jokes and proverbs

\* Name, id- store client info on HashMap. Use large random number \*/

static String *setting* = "Joke";

static String *name*;

static String *id* = "";

//need arrays to store told jokes/proverbs

static int[] *toldJokes*;

static int[] *toldProverbs*;

//create a class to store client data

static ClientData *client*;

//HashMaps to store info from client. Classmate on D2L suggested HashMaps

static HashMap<String, String> *jokes* = new HashMap<>();

static HashMap<String, String> *proverbs* = new HashMap<>();

static HashMap<String, ClientData> *clientFiles* = new HashMap<>();

//get jokes

public static String getJoke(String key) {

//NBA Hall of Famer Charles Barkley

*jokes*.put("JA", "You got to believe in yourself. I believe" +

" I'm the best-looking guy in the world and I might be right.");

//Seinfeld quote

*jokes*.put("JB", "Did you know that the original title for War " +

"and Peace was War, What Is It Good For?");

//Quote from the show Parks and Rec

*jokes*.put("JC", "Jogging is the worst. I know it keeps you healthy, but God, at what cost?");

*jokes*.put("JD", "I googled your symptoms and it says here you could " +

"have network connectivity problems.");

/\*key should be JA, JB, etc. + Client name + joke \*/

//CheckList point requires us to return it in this format

return key + " " + *name* + "- " + *jokes*.get(key);

}

//get proverbs

public static String getProverb(String key) {

/\*From a classmate from CSC447. He was explaining

\* prof wants us to complete this method without builtin functions\*/

*proverbs*.put("PA", "Sometimes we need to push a marble " +

"across the floor with our noses.");

/\*From former NY Yankees outfielder Oscar Gamble. Rumor is this was his response to

\* allegations that there is racism being a ballplayer, or that playing for the Yankees

\* feels like a circus. I think its insightful but funny.\*/

*proverbs*.put("PB", "They don't think it be like it is, but it do.");

//Famous Ali quote

*proverbs*.put("PC", "Float like a butterfly and sting like a bee.");

//Baseball HOF Yogi Berra

*proverbs*.put("PD", "Baseball is ninety percent mental. The other half is physical.");

/\*key should be PA, PB, etc. + Client name + proverb \*/

//CheckList point requires us to return it in this format

return key + " " + *name* + "- " + *proverbs*.get(key);

}

public static void main(String[] args) throws IOException {

Socket sock;

int queueLen = 6; //how many requests our server can handle at a time

//ports normally in 45750-55000 range, never below 1025

int port = 4545; //prof wants 4545 port

//from Prof

AdminLooper ad = new AdminLooper(); // create a thread

Thread t = new Thread(ad);

t.start(); // start it, waits for input from administration

//create server socket using ints above

ServerSocket servSock = new ServerSocket(port, queueLen);

// print out correct port number

System.*out*.println("Arturo Chaidez's Joke Server 1.0 , using port " + port);

while (true) {

sock = servSock.accept();

new JokeWorker(sock).run();

}

}

}

//IntelliJ not recognizing I made this class?

//It is allowing me to use JavaServer.ClientData,however...

/\*Figured out why!!! I had my { } all wrong and ClientData was not its own

\* class....\*/

class ClientData {

String name;

String id;

int[] toldJokes = {0, 0, 0, 0};

int[] toldProverbs = {0, 0, 0, 0};

//Store info as objects/arrays

//allows server to keep track what jokes have been said and avoid reusing

public void setToldJokes(int[] toldJokes) {

this.toldJokes = toldJokes;

}

public int[] getToldJokes() {

return toldJokes;

}

public int[] getToldProverbs() {

return toldProverbs;

}

public void setName(String name) {

this.name = name;

}

public void setId(String id) {

this.id = id;

}

public void setToldProverbs(int[] toldProverbs) {

this.toldProverbs = toldProverbs;

}

}

//to toggle between joke/proverb

class AdminLooper implements Runnable {

//All of this straight from prof example. Similar to what we have in main

public static boolean *adminSwitch* = true;

public void run() {

System.*out*.println("In the admin looper thread.");

int queueLen = 6;

int port = 5050; // Prof wants 5050 for admin client

Socket sock;

try {

ServerSocket servsock = new ServerSocket(port, queueLen);

while (*adminSwitch*) {

// wait for the next ADMIN client connection:

sock = servsock.accept();

new AdminWorker(sock).start();

}

} catch (IOException ioe) {

System.*out*.println(ioe);

}

}

}

//thread for Admin worker. Nearly the same as Worker thread in InetServer

class AdminWorker extends Thread {

Socket adminSock;

AdminWorker (Socket s) {adminSock = s;}

public void run() {

//same in/out as before

PrintStream adminOut = null;

BufferedReader adminIn = null;

try {

adminIn = new BufferedReader(new InputStreamReader(adminSock.getInputStream()));

adminOut = new PrintStream((adminSock.getOutputStream()));

try {

JokeServer.*name* = adminIn.readLine();

//check if clientAdmin wants to switch between settings.

if (JokeServer.*setting*.equals("Joke")) { JokeServer.*setting* = "Proverb";

} else {

JokeServer.*setting* = "Joke";

}

String settingChanged = "Changing to " + JokeServer.*setting* + " setting.";

//print it out on screens

System.*out*.println(settingChanged);

adminOut.println(settingChanged);

} catch (IOException x) {

System.*out*.println("Sever Error!");

x.printStackTrace(); //tells us where error happened

}

adminSock.close(); //close it down, boys.

} catch (IOException e) {

System.*out*.println(e); //for errors about input and output stream

}

}

}

//again, JokeWorker nearly the same as InetWorker thread

// Lots of code, controls randomizing joke/proverbs and keeping track of it

class JokeWorker extends Thread {

//set up socket and constructor

Socket jokeSock;

JokeWorker(Socket s) { jokeSock = s; }

public void run() {

/\*decided not to call this clientIn and clientOut. Hard to keep track what "client"

\* I am using, too many things named client. Maybe try to rename some stuff?\*/

PrintStream out = null;

BufferedReader in = null;

try {

in = new BufferedReader(new InputStreamReader(jokeSock.getInputStream()));

out = new PrintStream((jokeSock.getOutputStream()));

try {

//read line client gave.

JokeServer.*name* = in.readLine();

System.*out*.println("Getting Client info");

//Reads in ID from Client

JokeServer.*id* = in.readLine();

//if client has been here, get their info!

if (JokeServer.*clientFiles*.containsKey(JokeServer.*id*)) {

JokeServer.*client* = JokeServer.*clientFiles*.get(JokeServer.*id*);

JokeServer.*id* = JokeServer.*clientFiles*.get(JokeServer.*id*).id;

JokeServer.*name* = JokeServer.*clientFiles*.get(JokeServer.*id*).name;

JokeServer.*toldJokes* = JokeServer.*clientFiles*.get(JokeServer.*id*).toldJokes;

JokeServer.*toldProverbs* = JokeServer.*clientFiles*.get(JokeServer.*id*).toldProverbs;

} else {

//No idea why it will not accept ClientData without JokeServer (JokeServer.ClientData)

//made a ClientData class, says it does not exist?

/\*Figured out why!!! I had my { } all wrong and ClientData was not its own

\* class....\*/

ClientData newClient = new ClientData();

newClient.setName(JokeServer.*name*);

newClient.setId(JokeServer.*id*);

JokeServer.*clientFiles*.put(JokeServer.*id*, newClient);

JokeServer.*client* = JokeServer.*clientFiles*.get(JokeServer.*id*);

}

//send name to client

sendToClient(JokeServer.*name*, out);

} catch (IOException x) {

System.*out*.println("Server read error");

x.printStackTrace();

}

jokeSock.close(); //close socket

} catch (

IOException ioe) {

System.*out*.println(ioe);

}

}

//method to determine next joke/proverb

//used in senToClient

//Will not accept ClientData...

/\*Figured out why!!! I had my { } all wrong and ClientData was not its own

\* class....\*/

public Line newLine(ClientData client) {

/\*only need to use one variable to store joke/proverb, since

\* setting is only set to one at a time \*/

String nextLine = "";

HashMap<String, String> jokeMap = new HashMap<>();

HashMap<String, String> proverbMap = new HashMap<>();

jokeMap.put("0", "JA");

jokeMap.put("1", "JB");

jokeMap.put("2", "JC");

jokeMap.put("3", "JD");

proverbMap.put("0", "PA");

proverbMap.put("1", "PB");

proverbMap.put("2", "PC");

proverbMap.put("3", "PD");

Line savedLine = new Line();

Random randomNumber = new Random();

//use a random generator to pick a random joke/proverb

int pickRandom= randomNumber.nextInt((4));

/\* if/else statement. If server is in joke setting, get next joke, make sure we

\* dont need to randomize again, reset jokes. Else, do the same for proverb\*/

if (JokeServer.*setting*.equals("Joke")) {

//set bool to true. Can loop to find if all jokes have been used.

boolean allJokesUsed = true;

int[] toldJokes = client.getToldJokes();

//loop if all Jokes have been used. If one has not been used, set it to false

for (int x = 0; x < toldJokes.length; x++) {

if (toldJokes[x] != 1) {

allJokesUsed = false;

break;

}

}

//if all jokes have been seen, reset array to 0

if (allJokesUsed) {

Arrays.*fill*(toldJokes, 0);

System.*out*.println("Seen all jokes, new random order.");

}

//pick a random joke if they have not been used. Use it as nextLine

if (toldJokes[pickRandom] == 0 && allJokesUsed) {

toldJokes[pickRandom] = 1;

client.setToldJokes(toldJokes);

nextLine = jokeMap.get(String.*valueOf*(pickRandom));

savedLine.nextJoke = nextLine;

return savedLine;

}

/\*if randomly picked joke already used, loop through to find an unused

\* joke. Can't do loop first because then it would not be random\*/

else {

for (int x = 0; x < toldJokes.length; x++) {

if (toldJokes[x] == 0) {

toldJokes[x] = 1;

client.setToldJokes(toldJokes);

nextLine = jokeMap.get(String.*valueOf*(x));

savedLine.nextJoke = nextLine;

break;

}

}

return savedLine;

}

}

//else used for proverbs. Same steps

else {

boolean allProverbsUsed = true;

int[] toldProverbs = client.getToldProverbs();

for (int x = 0; x < toldProverbs.length; x++) {

if (toldProverbs[x] != 1) {

allProverbsUsed = false;

break;

}

}

//reset proverbs

if (allProverbsUsed) {

Arrays.*fill*(toldProverbs, 0);

System.*out*.println("Seen all proverbs, new random order");

}

if (toldProverbs[pickRandom] == 0 && allProverbsUsed) {

toldProverbs[pickRandom] = 1;

client.setToldJokes(toldProverbs);

nextLine = proverbMap.get(String.*valueOf*(pickRandom));

savedLine.nextJoke = nextLine;

return savedLine;

}

/\*if randomly picked proverb already used, loop through to find an unused

\* proverb. Can't do loop first because then it would not be random\*/

else {

for (int x = 0; x < toldProverbs.length; x++) {

if (toldProverbs[x] == 0) {

toldProverbs[x] = 1;

client.setToldProverbs(toldProverbs);

nextLine = proverbMap.get(String.*valueOf*(x));

savedLine.nextJoke = nextLine;

break;

}

}

}

return savedLine;

}

}

class Line {

String jokeList;

String proverbList;

String nextJoke;

//lists for jokes and proverbs

public String getJokeList() {

return jokeList;

}

public String getProverbList() {

return proverbList;

}

}

//Method Sends jokes to client

//I never use String name. Maybe send a one time message?

public void sendToClient(String name, PrintStream out) {

Line savedLine = newLine(JokeServer.*client*);

//if the setting is in Joke Mode, get joke list and send unused joke

if (JokeServer.*setting*.equals("Joke")) {

if (savedLine.getJokeList() != null) {

out.println(savedLine.getJokeList());

}

out.println(JokeServer.*getJoke*(savedLine.nextJoke));

}

//do the same but for proverbs

else {

if (savedLine.getProverbList() != null) {

out.println(savedLine.getProverbList());

}

out.println(JokeServer.*getProverb*(savedLine.nextJoke));

}

}

}

/\*--------------------------------------------------------

1. Name / Date: Arturo Chaidez III September 24, 2020

2. Java version used, if not the official version for the class:

openjdk version "15" 2020-09-15

OpenJDK Runtime Environment (build 15+36-1562)

OpenJDK 64-Bit Server VM (build 15+36-1562, mixed mode, sharing)

3. Precise command-line compilation examples / instructions:

> javac JokeServer.java

> javac JokeClient.java

> javac JokeClientAdmin.java

4. Precise examples / instructions to run this program:

Run in 3 separate terminals:

> java JokeServer.java

> java JokeClient.java

> java JokeClientAdmin.java

Cannot run a secondary server

In separate shell windows:

> java JokeServer

> java JokeClient

> java JokeClientAdmin

5. List of files needed for running the program.

1. JokeServer.java

2. JokeClient.java

3. JokeClientAdmin.java

4. JokeLog.txt

5. checklist-joke.html.

5. Notes:

Do not have a secondary server. When switching between jokes/proverbs, it does

save and returns to the proper list order. None of the MultipleServers checklist has been done.

----------------------------------------------------------\*/

import java.io.\*;

import java.net.\*;

//Similar to InetClient. A lot less work than JokeServer

public class JokeClient {

public static void main(String[] args) {

String serverName;

//Same as InetClient

if (args.length < 1) {

serverName = "localhost";

} else {

serverName = args[0];

}

//prof wants primary port to be 4545

System.*out*.println("Welcome to Arturo Chaidez's Joke server, port 4545.");

//prof wants admin port to be 5050

System.*out*.println(("ClientAdmin port 5050"));

BufferedReader in = new BufferedReader(new InputStreamReader(System.*in*));

try {

//need to get client name and ID

String name;

String input;

int id = 0;

System.*out*.println("You must be pretty bored to connect here..give me your name...");

name = in.readLine();

//randomize id to keep track of user

//IDK how to use UUID, come back to it later. This is the same concept

//This is acceptable according to prof.

id = 0 + (int) (Math.*random*() \* ((10000000 - 0) + 1));

System.*out*.println("Click enter for a joke. Once " +

"you realize this place isn't very funny, (quit) to get out.");

do {

//get info

input = in.readLine();

//make sure client did not quit right away

if (input.indexOf("quit") < 0) {

/\*Send name, unique id, and serverName to Server \*/

*printLine*(name, id, serverName);

}

}

//exit out when told

while (input.indexOf("quit") < 0);

System.*out*.println("I see you got bored. Program ended");

} catch (IOException x) {

x.printStackTrace();

}

}

//since we are not using IP addresses, do not need the toText Prof gave us for Inet

//seems to be working without it

//Similar as printRemoteAddress in Inet. Need to get jokes/proverbs

static void printLine(String name, int id, String serverName) {

Socket sock;

BufferedReader fromServer;

PrintStream toServer;

String textFromServer;

try {

sock = new Socket(serverName, 4545); //it wont let me change setting here

//I am dumb, changing between jokes and proverbs happen in the AdminClient

fromServer = new BufferedReader((new InputStreamReader(sock.getInputStream())));

toServer = new PrintStream(sock.getOutputStream());

//Send info to Server

toServer.println(name);

toServer.println(id);

//get lines from Server. Probably only need 1, but do 5 to be sure

//some jokes proverbs kind of long.

for(int i = 1; i <= 5; i++) {

textFromServer = fromServer.readLine();

if (textFromServer != null) {

System.*out*.println(textFromServer);

}

}

sock.close(); //done, can close socket

} catch (IOException x) { //usual error

System.*out*.println("Error: Socket error.");

x.printStackTrace();

}

}

}

\*--------------------------------------------------------

1. Name / Date: Arturo Chaidez III September 24, 2020

2. Java version used, if not the official version for the class:

openjdk version "15" 2020-09-15

OpenJDK Runtime Environment (build 15+36-1562)

OpenJDK 64-Bit Server VM (build 15+36-1562, mixed mode, sharing)

3. Precise command-line compilation examples / instructions:

> javac JokeServer.java

> javac JokeClient.java

> javac JokeClientAdmin.java

4. Precise examples / instructions to run this program:

Run in 3 separate terminals:

> java JokeServer.java

> java JokeClient.java

> java JokeClientAdmin.java

Cannot run a secondary server

In separate shell windows:

> java JokeServer

> java JokeClient

> java JokeClientAdmin

5. List of files needed for running the program.

1. JokeServer.java

2. JokeClient.java

3. JokeClientAdmin.java

4. JokeLog.txt

5. checklist-joke.html.

5. Notes:

Do not have a secondary server. When switching between jokes/proverbs, it does

save and returns to the proper list order. None of the MultipleServers checklist has been done.

----------------------------------------------------------\*/

import java.io.\*; //used to retrieve the I/O libraries

import java.net.\*; //used to retrieve the Java networking libraries

//THIS is where we change between server

/\*I was trying to do it all on Client, misread instructions. I thought this

\* had to do with primary/secondary servers. \*/

public class JokeClientAdmin {

//switch between jokes and proverbs

private static String *setting* ;

static boolean *changeSetting* = false;

// This is the main method.

public static void main(String[] args) {

//this is the same as Inet

String serverName;

//Same as InetClient

if (args.length < 1) {

serverName = "localhost";

} else {

serverName = args[0];

}

System.*out*.println(("Arturo Chaidez's JokeClientAdmin"));

BufferedReader in = new BufferedReader(new InputStreamReader(System.*in*));

try {

//get input

String adminInput = "";

System.*out*.println(

"Click enter to switch settings, or type (quit) to end program.\n");

do {

//check input from client

adminInput = in.readLine();

//if they do not quit and clicked enter, toggle to other setting

if (adminInput.indexOf("quit") < 0) {

*talkToServer*(adminInput, serverName);

System.*out*.println("Changed setting");

}

} while (adminInput.indexOf("quit") < 0); //user quit

System.*out*.println("Client has ended program.");

} catch (IOException x) {

x.printStackTrace();

}

}

static void talkToServer(String name, String serverName) {

//same as usual

Socket sock;

BufferedReader fromServer;

PrintStream toServer;

String textFromServer;

try {

//make socket

sock = new Socket(serverName, 5050);

fromServer = new BufferedReader((new InputStreamReader(sock.getInputStream())));

toServer = new PrintStream(sock.getOutputStream());

//double check what this does?

//doesn't seen to do anything. It should only be "" anyway

toServer.println(name);

//switch to other setting

if (*changeSetting*) {

//writing name

toServer.println(*setting*);

// Read 3 lines, probably too many

for(int i = 1; i <= 3; i++) {

textFromServer = fromServer.readLine();

if (textFromServer != null) {

System.*out*.println(textFromServer);

}

}

*changeSetting* = false;

}

//complete

sock.close();

} catch (IOException x) { //same error check

System.*out*.println("Error: Socket error.");

x.printStackTrace();

}

}

}