**JokeServer.java**

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

1. Name / Date: Vatsal Parikh, April 16, 2022

2. Java version used (java -version), if not the official version for the class:

java 11.0.12

3. Precise command-line compilation examples / instructions:

> javac JokeServer.java

> javac JokeClient.java

> javac JokeClientAdmin.java

4. Precise examples / instructions to run this program:

In separate shell windows:

> java JokeServer

> java JokeClient

> java JokeClientAdmin

All acceptable commands are displayed below:

> java JokeServer secondary

> java JokeClient

> java JokeClient localhost

> java JokeClient localhost localhost

> java JokeClient 127.0.0.1 localhost

> java JokeClientAdmin

> java JokeClientAdmin localhost

> java JokeClientAdmin localhost localhost

5. List of files needed for running the program.

a. JokeServer.java

b. JokeClient.java

c. JokeClientAdmin.java

5. Notes:

It is a multithreaded server with a secondary server for both Client and Admin.

I didn't transfer objects between servers and clients instead I stored only UUID

at client side and passed it as string between server client adding more security

but also complexity at server side.

I used hashtables for storing state as it is easier than custom classes to iterate over.

Only the shuffle of Joke IDs are stored as state for users and not all jokes/proverbs.

There is only one copy of jokes/proverbs.

I tried adding InetAddress.getByName() in JokeClientAdmin socket but 140.192.1.9 was always

giving error: Cannot assign requested address: NET\_Bind. So except this everything works.

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

// Importing Java Input Output and networking libraries

import java.io.\*;

import java.net.\*;

import java.util.\*;

// This will act as the parent class whose variable can be used in both AdminWorker and ServerWorker

class Worker extends Thread{

// mode\_admin and mode\_client will be used keep track of the current mode and both servers

static Hashtable<Integer, String> mode\_admin = new Hashtable<Integer, String>();

static Hashtable<Integer, String> mode\_client = new Hashtable<Integer, String>();

}

// AdminWorker class

class AdminWorker extends Worker {

ServerSocket servsock;

Socket sock;

// Passsing the serversocket instead of socket as argument in constructor

AdminWorker(ServerSocket servsock) {

this.servsock = servsock;

// Inserting default values

mode\_admin.putIfAbsent(5050, "joke");

mode\_admin.putIfAbsent(5051, "joke");

mode\_client.putIfAbsent(4545, "joke");

mode\_client.putIfAbsent(4546, "joke");

}

// start() method

public void run() {

while(true){

try{

// Connecting to admin client

sock = servsock.accept();

int admin\_port = servsock.getLocalPort();

PrintStream out = null;

BufferedReader in = null;

// Trying to assign in/out to socket's(client) Input/Output stream and returning error if not possible

try {

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

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

// Trying to read client request and if it cannot be read returning an error

try {

// Sending the current mode to admin

out.println(mode\_admin.get(admin\_port));

// Reading the mode returned by admin, changing the old value in mode\_admin and printing it on server side

mode\_admin.replace(admin\_port, in.readLine());

System.out.println("Changed mode for " + admin\_port +" to " + mode\_admin.get(admin\_port));

// For the current port when mode if changed it will be reflected on admin\_port for server worker to use

if(admin\_port==5050){

mode\_client.replace(4545, mode\_admin.get(admin\_port));

}

else if(admin\_port==5051){

mode\_client.replace(4546, mode\_admin.get(admin\_port));

}

} catch (Exception x) {}

// Closing the connection with client, though server is still running

sock.close();

} catch (IOException ioe) {System.out.println(ioe);}

} catch(Exception e){

System.out.println(e);

}

}

}

static String toText (byte ip[]) {

StringBuffer result = new StringBuffer ();

for (int i = 0; i < ip.length; ++ i) {

if (i > 0) result.append (".");

result.append (0xff & ip[i]);

}

return result.toString ();

}

}

// ServerWorker class

class ServerWorker extends Worker {

// sock contains the current client connection

Socket sock;

// Defining jokes and proverbs

static Hashtable<String, String> jokes = new Hashtable<String, String>();

static Hashtable<String, String> proverbs = new Hashtable<String, String>();

// According to server used and user accessing the server, user's joke/proverb sequence will be stored here

static Hashtable<Integer, Hashtable<String, ArrayList<String>>> server\_jokes = new Hashtable<Integer, Hashtable<String, ArrayList<String>>>();

static Hashtable<Integer, Hashtable<String, ArrayList<String>>> server\_proverbs = new Hashtable<Integer, Hashtable<String, ArrayList<String>>>();

// This will contain the shuffle used to randomize the joke sequence

ArrayList<String> shuffle\_jokes;

ArrayList<String> shuffle\_proverbs;

// Worker Constructor

ServerWorker (Socket s) {

// Putting jokes and proverbs in static jokes and proverbs hashtable with respective keys

jokes.put("JA", "A clean house is the sign of a broken computer.");

jokes.put("JB", "CAPS LOCK has been preventing login since 1980.");

jokes.put("JC", "What do you call 8 hobbits? A hobbyte");

jokes.put("JD", "I love the F5 key. It is so refreshing.");

proverbs.put("PA", "I do not fear computers. I fear lack of them.");

proverbs.put("PB", "The computer was born to solve problems that did not exist before.");

proverbs.put("PC", "Computing is not about computers any more. It is about living.");

proverbs.put("PD", "Man is still the most extraordinary computer of all.");

sock = s;

mode\_client.putIfAbsent(4545, "joke");

mode\_client.putIfAbsent(4546, "joke");

}

// Run method will run when start() is called

public void run(){

int client\_port = sock.getLocalPort();

server\_jokes.putIfAbsent(client\_port, new Hashtable<>());

server\_proverbs.putIfAbsent(client\_port, new Hashtable<>());

// Defining I/O streams in/out to null

PrintStream out = null;

BufferedReader in = null;

// Trying to assign in/out to socket's(client) Input/Output stream and returning error if not possible

try {

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

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

// Trying to read client request and if it cannot be read returning an error

try {

// Reading client request with the help of in

String new\_name = in.readLine();

String uuid = in.readLine();

// Checking if the mode is joke and adding the shuffled list if empty to respective user to respective server

if(mode\_client.get(client\_port).equals("joke")){

shuffle\_jokes = Collections.list(jokes.keys());

Collections.shuffle(shuffle\_jokes);

server\_jokes.get(client\_port).putIfAbsent(uuid, shuffle\_jokes);

if(server\_jokes.get(client\_port).get(uuid).isEmpty()){

// When the shuffled list is empty message is sent to client

if(client\_port==4546) out.print("<S2>");

out.println("JOKE CYCLE COMPLETED");

server\_jokes.get(client\_port).put(uuid, shuffle\_jokes);

}

// Calling the printstate function

printState(new\_name, out, client\_port, server\_jokes.get(client\_port).get(uuid).remove(0));

}

// Checking if the mode is joke and adding the shuffled list if empty to respective user to respective server

else if(mode\_client.get(client\_port).equals("proverb")){

shuffle\_proverbs = Collections.list(proverbs.keys());

Collections.shuffle(shuffle\_proverbs);

server\_proverbs.get(client\_port).putIfAbsent(uuid, shuffle\_proverbs);

if(server\_proverbs.get(client\_port).get(uuid).isEmpty()){

// When the shuffled list is empty message is sent to client

if(client\_port==4546) out.print("<S2>");

out.println("PROVERB CYCLE COMPLETED");

server\_proverbs.get(client\_port).put(uuid, shuffle\_proverbs);

}

// Calling the printstate function

printState(new\_name, out, client\_port, server\_proverbs.get(client\_port).get(uuid).remove(0));

}

} catch (IOException x) {

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

x.printStackTrace ();

}

// Closing the connection with client, though server is still running

sock.close();

} catch (IOException ioe) {System.out.println(ioe);}

}

// This static method will print the output on client side with the help of out

static void printState(String name, PrintStream out, int client\_port, String shuffle) {

try {

// When second server is connected <S2> is added

if(client\_port==4546) out.print("<S2> ");

out.print(shuffle + " " + name + ": ");

if(mode\_client.get(client\_port).equals("joke")){

out.println(jokes.get(shuffle));

}

else if(mode\_client.get(client\_port).equals("proverb")){

out.println(proverbs.get(shuffle));

}

} catch(Exception ex) {

out.println ("Failed!" + name);

}

}

}

class ServerLooper implements Runnable {

public static boolean ServerSwitch = true;

public void run(){ // RUNning the Admin listen loop

int q\_len = 6;

int port = 4546;

Socket sock;

try{

ServerSocket servsock\_admin = new ServerSocket(5051, q\_len);

new AdminWorker(servsock\_admin).start();

ServerSocket servsock\_client = new ServerSocket(port, q\_len);

while (ServerSwitch) {

// wait for the next ADMIN client connection:

sock = servsock\_client.accept();

new ServerWorker(sock).start();

}

}catch (IOException ioe){

System.out.println(ioe);

}

}

}

// Server class

public class JokeServer {

// Main method

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

// Number of request at a time is stored in a queue of length 6

int q\_len = 6;

Socket sock;

// Creating a server socket

InetAddress bindAddress = InetAddress.getByName("127.0.0.1");

ServerSocket servsock\_client = new ServerSocket(4545, q\_len, bindAddress);

System.out.println("Vatsal Parikh's Joke Server is starting up, listening at port 4545.\n");

// If the argument "secondary" is added then Serverlooper is called

if(args.length>0){

if(args[0].contains("secondary")){

System.out.println("Vatsal Parikh's Joke Server 2 is starting up, listening at port 4546.\n");

ServerLooper SL = new ServerLooper();

Thread t = new Thread(SL);

t.start();

}

}

// Calling AdminWorker with server socket

ServerSocket servsock\_admin = new ServerSocket(5050, q\_len, bindAddress);

new AdminWorker(servsock\_admin).start();

// Waiting for a client connection

while (true) {

// Accepting requests from clients

sock = servsock\_client.accept();

new ServerWorker(sock).start();

}

}

}

**JokeClient.java**

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

1. Name / Date: Vatsal Parikh, April 16, 2022

2. Java version used (java -version), if not the official version for the class:

java 11.0.12

3. Precise command-line compilation examples / instructions:

> javac JokeServer.java

> javac JokeClient.java

> javac JokeClientAdmin.java

4. Precise examples / instructions to run this program:

In separate shell windows:

> java JokeServer

> java JokeClient

> java JokeClientAdmin

All acceptable commands are displayed below:

> java JokeServer secondary

> java JokeClient

> java JokeClient localhost

> java JokeClient localhost localhost

> java JokeClient 127.0.0.1 localhost

> java JokeClientAdmin

> java JokeClientAdmin localhost

> java JokeClientAdmin localhost localhost

5. List of files needed for running the program.

a. JokeServer.java

b. JokeClient.java

c. JokeClientAdmin.java

5. Notes:

It is a multithreaded server with a secondary server for both Client and Admin.

I didn't transfer objects between servers and clients instead I stored only UUID

at client side and passed it as string between server client adding more security

but also complexity at server side.

I used hashtables for storing state as it is easier than custom classes to iterate over.

Only the shuffle of Joke IDs are stored as state for users and not all jokes/proverbs.

There is only one copy of jokes/proverbs.

I tried adding InetAddress.getByName() in JokeClientAdmin socket but 140.192.1.9 was always

giving error: Cannot assign requested address: NET\_Bind. So except this everything works.

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

// Importing Java Input Output and networking libraries

import java.io.\*;

import java.net.\*;

import java.util.\*;

// JokeClient class

public class JokeClient{

// UUID with respective name is stored in name\_uuid

static Hashtable<String, String> name\_uuid = new Hashtable<String, String>();

static String name;

// Main method

public static void main (String args[]) {

// Defining servernames and port numbers

String serverName1 = "localhost";

String serverName2 = "localhost";

String currentServer = serverName1;

int port\_server1 = 4545;

int port\_server2 = 4546;

int currentPort = port\_server1;

// Switcher will be used to check if user provided multiple arguments(ie multi server)

boolean switcher = false;

// Based on arguments servername will be fetched

if(args.length == 1) {

serverName1 = args[0];

}

System.out.println("Vatsal Parikh's Joke Client\n");

System.out.println("Server one: " + serverName1 + ", Port: 4545");

if (args.length == 2){

serverName2 = args[1];

System.out.println("Server two: " + serverName1 + ", Port: 4546");

System.out.println("\n\*\*(s) to change servers\*\*");

switcher = true;

}

// Taking input from the user using InputStreamReader

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

// Trying to get user inputs until user passes quit as input

try {

do {

// Fetching user input

System.out.print("\nEnter your name or just press enter(if you are the previous user), \n(quit) to end: ");

System.out.flush ();

String temp = in.readLine();

// Changing servers when the input is s and printing accordingly on admin side

if(temp.equals("s")){

// Changing boolean value

if(switcher){

if(currentPort==port\_server1){

currentServer = serverName2;

System.out.println("Now communicating with: localhost, port 4546");

currentPort = port\_server2;

}

else if(currentPort==port\_server2){

currentServer = serverName1;

System.out.println("Now communicating with: localhost, port 4545");

currentPort = port\_server1;

}

continue;

}

// If no secondary server is provided it will provide this output

else{

System.out.println("No secondary server being used");

continue;

}

}

if(!temp.isEmpty()){

name = temp;

}

// If user input is valid, getJoke method is called with user input and server name

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

if(!name\_uuid.keySet().contains(name)){

name\_uuid.put(name, UUID.randomUUID().toString());

}

getJoke(name, name\_uuid.get(name), currentServer, currentPort);

}

} while (name.indexOf("quit") < 0);

System.out.println ("Cancelled by user request.");

} catch (Exception x) {System.out.println(x);}

}

// This static method will fetch the output from server side with the help of fromServer

static void getJoke (String name, String uuid, String serverName, int currentPort){

// sock contains the current client connection

Socket sock;

// Defining I/O streams fromServer/toServer to null

BufferedReader fromServer;

PrintStream toServer;

String textFromServer;

// Trying to make a connection with the server name and port number

try{

sock = new Socket(serverName, currentPort);

// Assigning fromServer/toServer to socket's(server) Input/Output stream

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

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

// Sending name, uuid and cuurent port to server

toServer.println(name+"\n"+uuid+"\n"+currentPort);

toServer.flush();

// Reading the first 2-3 lines from the server

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

textFromServer = fromServer.readLine();

// If the line read is not null, it is printed on client side

if (textFromServer != null) System.out.println(textFromServer);

}

sock.close();

} catch (IOException x) {

System.out.println ("Socket error.");

x.printStackTrace ();

}

}

}

**JokeClientAdmin.java**

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

1. Name / Date: Vatsal Parikh, April 16, 2022

2. Java version used (java -version), if not the official version for the class:

java 11.0.12

3. Precise command-line compilation examples / instructions:

> javac JokeServer.java

> javac JokeClient.java

> javac JokeClientAdmin.java

4. Precise examples / instructions to run this program:

In separate shell windows:

> java JokeServer

> java JokeClient

> java JokeClientAdmin

All acceptable commands are displayed below:

> java JokeServer secondary

> java JokeClient

> java JokeClient localhost

> java JokeClient localhost localhost

> java JokeClient 127.0.0.1 localhost

> java JokeClientAdmin

> java JokeClientAdmin localhost

> java JokeClientAdmin localhost localhost

5. List of files needed for running the program.

a. JokeServer.java

b. JokeClient.java

c. JokeClientAdmin.java

5. Notes:

It is a multithreaded server with a secondary server for both Client and Admin.

I didn't transfer objects between servers and clients instead I stored only UUID

at client side and passed it as string between server client adding more security

but also complexity at server side.

I used hashtables for storing state as it is easier than custom classes to iterate over.

Only the shuffle of Joke IDs are stored as state for users and not all jokes/proverbs.

There is only one copy of jokes/proverbs.

I tried adding InetAddress.getByName() in JokeClientAdmin socket but 140.192.1.9 was always

giving error: Cannot assign requested address: NET\_Bind. So except this everything works.

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

// Importing Java Input Output and networking libraries

import java.io.\*;

import java.net.\*;

// JokeClientAdmin class

public class JokeClientAdmin{

// Main method

public static void main (String args[]) {

// Defining servernames and port numbers

String serverName1 = "localhost";

String serverName2 = "localhost";

String currentServer = serverName1;

int port\_server1 = 5050;

int port\_server2 = 5051;

int currentPort = port\_server1;

// Switcher will be used to check if user provided multiple arguments(ie multi server)

boolean switcher = false;

// Based on arguments servername will be fetched

if(args.length == 1) {

serverName1 = args[0];

}

System.out.println("Vatsal Parikh's Admin Client\n");

System.out.println("Server one: " + serverName1 + ", Port: 5050");

if (args.length == 2){

serverName2 = args[1];

System.out.println("Server two: " + serverName1 + ", Port: 5051");

System.out.println("\n\*\*(s) to change servers\*\*\n");

switcher = true;

}

try {

// Defining I/O streams fromServer/toServer and in

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

BufferedReader fromServer;

PrintStream toServer;

// Trying to get user inputs until user passes quit as input

do {

Socket sock;

System.out.println("Press enter to change modes");

// Waiting for user input and it won't execute further until some kind of input is given

String temp = in.readLine();

// Trying to make a connection with the server name and port number

try{

sock = new Socket(currentServer, currentPort);

// Assigning fromServer/toServer to socket's(server) Input/Output stream

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

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

String mode="joke";

// If the input is quit and breaking the loop and exiting the program

if(temp.toString().contains("quit")){

break;

}

// Changing servers when the input is s and printing accordingly on admin side

if(temp.equals("s")){

// Changing boolean value

if(switcher){

if(currentPort==port\_server1){

currentServer = serverName2;

System.out.println("Now communicating with: localhost, port 5051\n");

currentPort = port\_server2;

}

else if(currentPort==port\_server2){

currentServer = serverName1;

System.out.println("Now communicating with: localhost, port 5050\n");

currentPort = port\_server1;

}

continue;

}

// If no secondary server is provided it will provide this output

else{

System.out.println("No secondary server being used");

continue;

}

}

// Server sends the current mode to admin

mode = fromServer.readLine();

// Based on the current mode admin will change the mode and send that that to server

if (mode.equals("joke")){

mode = "proverb";

}

else if(mode.equals("proverb")){

mode = "joke";

}

toServer.println(mode);

toServer.flush();

// Closing socket

sock.close();

} catch (Exception x) {

System.out.println ("Socket error.");

x.printStackTrace ();

}

} while(true);

} catch (IOException x) {x.printStackTrace();}

}

}

**Output:**

##JokeServer##

D:\Study\DS\Joke>java JokeServer secondary

Vatsal Parikh's Joke Server is starting up, listening at port 4545.

Vatsal Parikh's Joke Server 2 is starting up, listening at port 4546.

Changed mode for 5050 to proverb

Changed mode for 5051 to proverb

##JokeClient##

D:\Study\DS\Joke>java JokeClient localhost localhost

Vatsal Parikh's Joke Client

Server one: localhost, Port: 4545

Server two: localhost, Port: 4546

\*\*(s) to change servers\*\*

Enter your name or just press enter(if you are the previous user),

(quit) to end: vatsal

JA vatsal: A clean house is the sign of a broken computer.

Enter your name or just press enter(if you are the previous user),

(quit) to end: s

Now communicating with: localhost, port 4546

Enter your name or just press enter(if you are the previous user),

(quit) to end: vatsal

<S2> JB vatsal: CAPS LOCK has been preventing login since 1980.

Enter your name or just press enter(if you are the previous user),

(quit) to end:

<S2> PC vatsal: Computing is not about computers any more. It is about living.

Enter your name or just press enter(if you are the previous user),

(quit) to end:

<S2> PB vatsal: The computer was born to solve problems that did not exist before.

Enter your name or just press enter(if you are the previous user),

(quit) to end:

<S2> PD vatsal: Man is still the most extraordinary computer of all.

Enter your name or just press enter(if you are the previous user),

(quit) to end:

<S2> PA vatsal: I do not fear computers. I fear lack of them.

Enter your name or just press enter(if you are the previous user),

(quit) to end:

<S2>PROVERB CYCLE COMPLETED

<S2> PC vatsal: Computing is not about computers any more. It is about living.

##JokeClientAdmin##

D:\Study\DS\Joke>java JokeClientAdmin localhost localhost

Vatsal Parikh's Admin Client

Server one: localhost, Port: 5050

Server two: localhost, Port: 5051

\*\*(s) to change servers\*\*

Press enter to change modes

Press enter to change modes

s

Now communicating with: localhost, port 5051

Press enter to change modes

Press enter to change modes