1) JokeServer:

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

1. Jing Li / Jan 22, 2017

2. Java version used: Version 8 Update 60 (build 1.8.0\_60-b27)

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

5. List of files needed for running the program.

a. JokeServer.java

b. JokeClient.java

c. JokeClientAdmin.java

5. Notes: I saved all the status of clients on the clients side

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

import java.io.\*;

import java.net.\*;

import java.util.HashMap;

import java.util.Arrays;

import java.util.ArrayList;

import java.util.Collections;

/\*

The class JokeServer contains the main function,

it creates a new thread for making connection with clientadmins

and opens a server socket and make it wait to connet to clients.

After connecting to a client, use Worker class to do the task.

\*/

public class JokeServer{

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

int q\_len=6;

int port=4545; //the port number for connecting with clients is at port 4545

Socket sock;

Admin admin=new Admin(); //admin is a new object which is runnable

new Thread(admin).start(); //builds a new thread using a runnable object to control the mode of the server

ServerSocket servsock=new ServerSocket(port, q\_len); //create the server waiting for the clients at port 4545.

System.out.println("\nJokeServer: localhost, port 4545.\n");

while(true){

sock=servsock.accept(); //block and wait to connet to client

new Worker(sock).start(); //use Worker class to do the actual work

}

}

}

/\*

The Worker class which does the actual task for clients.

Deals with clients in a multithreaded way - can talk to multiple clients.

I put all the status in clients, so each time when connected with clients, need the status and cycle tracker for both jokes and proverbs

Every time sent one joke/proverb, add it to the status list, also take it out of the ordered cycle tracker.

\*/

class Worker extends Thread{

Socket sock;

Worker(Socket s){sock=s;}

public void run(){

PrintStream out=null;

BufferedReader in=null;

String fromClient;

String userName;

String[] fromClientSep;

String statusFrom;

String statusFromNew;

String[] status;

ArrayList<String> jokeitems=new ArrayList<String>(); //jokeitems and proverbs items are used to save all the joke names and proverb names

ArrayList<String> proverbitems=new ArrayList<String>(); //also do the shuffle using these two lists.

String cycleFrom;

String[] cycleJoke;

String[] cycleProverb;

ArrayList<String> cycleListJoke=new ArrayList<String>(); //cycleListJoke and cycleListProverb are the cycles which each client would keep track of

ArrayList<String> cycleListProverb=new ArrayList<String>();

jokeitems.add("JA");

jokeitems.add("JB");

jokeitems.add("JC"); //give values to the jokeitems and jokeproverbs

jokeitems.add("JD");

proverbitems.add("PA");

proverbitems.add("PB");

proverbitems.add("PC");

proverbitems.add("PD");

try{

in=new BufferedReader(new InputStreamReader(sock.getInputStream())); //it is a buffered read to read the character-based text coming into the socket

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

fromClient=in.readLine();

fromClientSep=fromClient.split(", ");

userName=fromClientSep[0].substring(1); //get the userName from client

statusFrom=in.readLine();

if (statusFrom.length()<=2){status=new String[0];}

else{

statusFromNew=statusFrom.substring(1,statusFrom.length()-1); //get the status from clietns to keep track of what jokes or proverbs he/she has already seen

status=statusFromNew.split(", ");

}

cycleFrom=in.readLine();

if (cycleFrom.length()<=2){cycleJoke=new String[0];}

else{

cycleJoke=cycleFrom.substring(1,cycleFrom.length()-1).split(", "); //get the cycle track for jokes from the client

}

for (String item: cycleJoke){

cycleListJoke.add(item);

}

cycleFrom=in.readLine();

if (cycleFrom.length()<=2){cycleProverb=new String[0];}

else{

cycleProverb=cycleFrom.substring(1,cycleFrom.length()-1).split(", "); //get the cycle track for proverbs from client

}

for (String item: cycleProverb){

cycleListProverb.add(item);

}

if (Admin.modeControl==true){ //if the mode of server is on the joke mode, do the following, otherwise do the else part

if (cycleListJoke.size()==0){ //if the cycle tracker for jokes is empty, that means we need a new cycle with order re-shuffled

Collections.shuffle(jokeitems); //reshuffle all the jokes

for (String item: jokeitems){ //and give the shuffled jokes to the cycle tracker for jokes

cycleListJoke.add(item);

}

}

out.println(cycleListJoke.get(0)+ " "+userName+" "+ Contents.jokes.get(cycleListJoke.get(0))); //sent the info from server to the client as requsted

System.out.println("To client "+userName+": sent "+cycleListJoke.get(0));

System.out.println(cycleListJoke.get(0)+ " "+userName+" "+ Contents.jokes.get(cycleListJoke.get(0))+"\n");

cycleListJoke.remove(0); //takes out the joke which has already been sent from joke cycle tracker

out.println(cycleListJoke); //also send back the cycle tracker to save on the client side

}else{ //if the mode of server is on the proverb mode, do the following

if (cycleListProverb.size()==0){ //if the cycle tracker for proverbs is empty, that means we need a new cycle with order re-shuffled

Collections.shuffle(proverbitems); //reshuffle all the proverbs

for (String item: proverbitems){

cycleListProverb.add(item); //and give the shuffled proverbs to the cycle tracker for proverbs

}

}

out.println(cycleListProverb.get(0)+ " "+userName+" "+ Contents.proverbs.get(cycleListProverb.get(0))); //sent the info from server to the client as requsted

System.out.println("To client "+userName+": sent "+cycleListProverb.get(0));

System.out.println(cycleListProverb.get(0)+ " "+userName+" "+ Contents.proverbs.get(cycleListProverb.get(0))+"\n");

cycleListProverb.remove(0); //takes out the proverbs which has already been sent from proverb cycle tracker

out.println(cycleListProverb); //also send back the cycle tracker to save on the client side

}

out.println(Admin.modeControl); //send the server mode the server is on to the client to determine which cycle tracker to update

}catch(IOException e){

System.out.println(e);

}

}

}

/\*

Admin class implements Runnable Interface

and waits to be connected with admin clients

\*/

class Admin implements Runnable{

public static boolean modeControl=true; //set the default value of mode to be true

public void run(){

int q\_len=6;

int port=5050; //set the admin client port to be 5050

Socket sock;

try{

ServerSocket servsockadmin=new ServerSocket(port, q\_len); //open up a server on port 5050

while(true){

sock=servsockadmin.accept(); //block and wait for admin clients' connection

new AdminWorker(sock).start(); //use AdminWorker class to control the mode of the server

}

}catch(IOException ioe){

System.out.print(ioe);

}

}

}

/\*

every time connected with a admin client, change the mode of server by adminWorker class

\*/

class AdminWorker extends Thread{

Socket sock;

AdminWorker(Socket s){sock=s;}

public void run(){

PrintStream out;

try{

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

if (Admin.modeControl==true){

Admin.modeControl=false; //if the mode is on joke mode, after getting request from the admin client, change it to proverb mode

System.out.println("The joke Server has been changed by admin client: in PROVERB mode now.\n");

out.println("The joke Server is in PROVERB mode now.");

}else {

Admin.modeControl=true; //if it's on proverb, change it to joke mode

System.out.println("The joke Server has been changed by admin client: in JOKE mode now.\n");

out.println("The joke Server is in JOKE mode now.");

}

sock.close();

}catch(IOException e){

System.out.println("sock closing error");

}

}

}

/\*

A class to store all the information about jokes and proverbs.

Can be used anytime when needed

\*/

class Contents{

public static final HashMap<String, String> jokes=new HashMap<String, String>(); //set jokes and proverbs to be static and final

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

static{

jokes.put("JA", "I wanted to grow my own food but I couldnt get bacon seeds anywhere.");

jokes.put("JB","How do you tell that a crab is drunk? It walks forwards."); //initialize jokes and proverbs

jokes.put("JC","Why do cows wear bells? Their horns don’t work.");

jokes.put("JD","What goes up and down but never moves? - The stairs!");

proverbs.put("PA","You can't wake a person who is pretending to be asleep.");

proverbs.put("PB","The talkative bird can not make a nest.");

proverbs.put("PC","Rumors are carried by haters, spread by fools, and accepted by idiots.");

proverbs.put("PD","It is not work that kills, but worry.");

}

}

2) JokeClient

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

1. Jing Li / Jan 22, 2017

2. Java version used: Version 8 Update 60 (build 1.8.0\_60-b27)

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

5. List of files needed for running the program.

a. JokeServer.java

b. JokeClient.java

c. JokeClientAdmin.java

5. Notes: I saved all the status of clients on the clients side

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

import java.io.\*;

import java.net.\*;

import java.util.ArrayList;

import java.util.HashMap;

public class JokeClient {

public static void main(String[] args){

String serverName;

String userName;

ArrayList<String> status=new ArrayList<String>(); //status keeps track of which jokes or proverbs the client has already seen

ArrayList<String> cycleJoke=new ArrayList<String>(); //keeps track of one cycle of jokes, records which jokes haven't been told in the cycle

ArrayList<String> cycleProverb=new ArrayList<String>(); //keeps track of one cycle of proverbs, records which proverbs haven't been told in the cycle

ArrayList<ArrayList<String>> result=new ArrayList<ArrayList<String>>(); //result is the value retruned from getInfo function, which updates the status and cycle information for the specific client

if (args.length<1){

serverName="localhost"; //if didn't give arguments for the ip address of server, use localhost

}else{serverName=args[0];} //otherwise use the argument

System.out.println("Jokeclient: using server: "+serverName+", port 4545");

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

System.out.println("Please enter your user name: "); //ask for the user name

try{

userName=in.readLine(); //get the user name from user

String name;

do {

name=in.readLine();

if (name.indexOf("quit")<0){ //if hasn't quitted

result=getInfo(serverName, name, userName,status,cycleJoke,cycleProverb); //use getInfo function to update the status of this client after connecting with the server

status=result.get(0); //the first value in the ArrayList represents the status which jokes and proverbs this client has already seen

cycleJoke=result.get(1); //the second value in the ArrayList represents that the jokes haven't been told in this cycle for this client

cycleProverb=result.get(2); //the third value in the ArrayList represents that the proverbs haven't been told in this cycle for this client

} //two of these three values would get updated

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

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

}catch(IOException e){

e.printStackTrace();

}

}

/\*

the function getInfo takes in some information like server name, user name, the status, the joke cycle and proverb cycle

connecting with the server, get a joke or probverb depending on the server mode, and

update the status of the client after the connection

\*/

static ArrayList<ArrayList<String>> getInfo(String serverName, String name, String userName, ArrayList<String> status, ArrayList<String> cycleJoke, ArrayList<String> cycleProverb){

Socket sock;

BufferedReader fromServer;

PrintStream toServer;

String textFromServer;

String cycleFrom;

ArrayList<String> passInfo=new ArrayList<String>();

passInfo.add(userName);

passInfo.add(name);

String newItem;

String[] cycleIn;

ArrayList<String> cycleArray= new ArrayList<String>();

ArrayList<ArrayList<String>> result=new ArrayList<ArrayList<String>>();

String mode;

try{

sock=new Socket(serverName, 4545);

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

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

toServer.println(passInfo); //send the user name to the server

toServer.println(status); //send the jokes and proverbs that the client has already seen to the server

toServer.println(cycleJoke); //send the joke cycle to the server

toServer.println(cycleProverb); //send the proverb cycle to the server

toServer.flush();

textFromServer=fromServer.readLine();

if (textFromServer!=null){

System.out.println(textFromServer); //if the content is not empty, print out the message on the console

}

newItem=textFromServer.split(" ")[0]; //get which joke or proverb the client sees this time

status.add(newItem); //add the joke/proverb the client sees this time to the status

cycleFrom=fromServer.readLine();

if (cycleFrom.length()<=2){cycleIn=new String[0];}

else{

cycleIn=cycleFrom.substring(1,cycleFrom.length()-1).split(", ");

}

for (String item: cycleIn){

cycleArray.add(item); //get the joke cycle or proverb cycle depends on the server mode

}

mode=fromServer.readLine(); //get the server mode

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

result.add(status); //if the server mode is on joke, then update status and joke cycle

result.add(cycleArray);

result.add(cycleProverb);

}else{

result.add(status); //if the server mode is on proverb, update the status and proverb cycle

result.add(cycleJoke);

result.add(cycleArray);

}

sock.close();

}catch(IOException e){

System.out.println("Socket Error!"); //IOException, prints an error message

e.printStackTrace();

}

return result;

}

}

3) JokeClientAdmin

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

1. Jing Li / Jan 22, 2017

2. Java version used: Version 8 Update 60 (build 1.8.0\_60-b27)

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

5. List of files needed for running the program.

a. JokeServer.java

b. JokeClient.java

c. JokeClientAdmin.java

5. Notes: I saved all the status of clients on the clients side

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

import java.io.\*;

import java.net.\*;

public class JokeClientAdmin{

public static void main (String[] args){

String serverName;

if (args.length<1){ //if didn't give arguments for the ip address of server, use localhost

serverName="localhost";

}else{

serverName=args[0]; //otherwise use the argument

}

System.out.println("JokeClientAdmin: using server: "+serverName+", port 5050");

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

try{

String name;

do{

name=in.readLine();

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

getControlInfo(serverName, name); //if requsted, do as the getControlInfo function tells to

}

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

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

}catch(IOException e){

e.printStackTrace();

}

}

/\*

getControlInfo function connects with the server

and switch the server mode

\*/

public static void getControlInfo(String serverName, String name){

Socket sock;

PrintStream toServer;

BufferedReader fromServer;

String textFromServer;

try{

sock=new Socket(serverName, 5050); //connects with the server

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

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

toServer.println(name);

toServer.flush();

textFromServer=fromServer.readLine();

if (textFromServer!=null){

System.out.println(textFromServer); //if the content is not empty, print out the message on the console

}

sock.close();

}catch(IOException e){

System.out.print("Connerction error!");

e.printStackTrace();

}

}

}

4) JokeLog

JokeServer:

JokeServer: localhost, port 4545.

To client jane: sent JC

JC jane Why do cows wear bells? Their horns don’t work.

To client jane: sent JB

JB jane How do you tell that a crab is drunk? It walks forwards.

To client jane: sent JA

JA jane I wanted to grow my own food but I couldnt get bacon seeds anywhere.

The joke Server has been changed by admin client: in PROVERB mode now.

To client jane: sent PA

PA jane You can't wake a person who is pretending to be asleep.

To client Zak: sent PA

PA Zak You can't wake a person who is pretending to be asleep.

To client Zak: sent PB

PB Zak The talkative bird can not make a nest.

The joke Server has been changed by admin client: in JOKE mode now.

The joke Server has been changed by admin client: in PROVERB mode now.

The joke Server has been changed by admin client: in JOKE mode now.

To client Zak: sent JA

JA Zak I wanted to grow my own food but I couldnt get bacon seeds anywhere.

To client Zak: sent JC

JC Zak Why do cows wear bells? Their horns don’t work.

To client jane: sent JD

JD jane What goes up and down but never moves? - The stairs!

To client jane: sent JB

JB jane How do you tell that a crab is drunk? It walks forwards.

To client Sannya: sent JB

JB Sannya How do you tell that a crab is drunk? It walks forwards.

The joke Server has been changed by admin client: in PROVERB mode now.

To client Sannya: sent PA

PA Sannya You can't wake a person who is pretending to be asleep.

To client Sannya: sent PB

PB Sannya The talkative bird can not make a nest.

To client Zak: sent PC

PC Zak Rumors are carried by haters, spread by fools, and accepted by idiots.

To client Zak: sent PD

PD Zak It is not work that kills, but worry.

To client Zak: sent PD

PD Zak It is not work that kills, but worry.

To client jane: sent PB

PB jane The talkative bird can not make a nest.

To client jane: sent PD

PD jane It is not work that kills, but worry.

To client Sannya: sent PD

PD Sannya It is not work that kills, but worry.

The joke Server has been changed by admin client: in JOKE mode now.

To client Sannya: sent JC

JC Sannya Why do cows wear bells? Their horns don’t work.

To client Sannya: sent JD

JD Sannya What goes up and down but never moves? - The stairs!

To client jane: sent JD

JD jane What goes up and down but never moves? - The stairs!

To client jane: sent JA

JA jane I wanted to grow my own food but I couldnt get bacon seeds anywhere.

To client Zak: sent JD

JD Zak What goes up and down but never moves? - The stairs!

To client Zak: sent JB

JB Zak How do you tell that a crab is drunk? It walks forwards.

The joke Server has been changed by admin client: in PROVERB mode now.

To client Sannya: sent PC

PC Sannya Rumors are carried by haters, spread by fools, and accepted by idiots.

JokeClientAdmin:

JokeClientAdmin: using server: localhost, port 5050

The joke Server is in PROVERB mode now.

The joke Server is in JOKE mode now.

The joke Server is in PROVERB mode now.

The joke Server is in JOKE mode now.

The joke Server is in PROVERB mode now.

The joke Server is in JOKE mode now.

The joke Server is in PROVERB mode now.

JokeClient 1:

Jokeclient: using server: localhost, port 4545

Please enter your user name:

jane

JC jane Why do cows wear bells? Their horns don’t work.

JB jane How do you tell that a crab is drunk? It walks forwards.

JA jane I wanted to grow my own food but I couldnt get bacon seeds anywhere.

PA jane You can't wake a person who is pretending to be asleep.

JD jane What goes up and down but never moves? - The stairs!

JB jane How do you tell that a crab is drunk? It walks forwards.

PB jane The talkative bird can not make a nest.

PD jane It is not work that kills, but worry.

JD jane What goes up and down but never moves? - The stairs!

JA jane I wanted to grow my own food but I couldnt get bacon seeds anywhere.

JokeClient 2:

Jokeclient: using server: localhost, port 4545

Please enter your user name:

Zak

PA Zak You can't wake a person who is pretending to be asleep.

PB Zak The talkative bird can not make a nest.

JA Zak I wanted to grow my own food but I couldnt get bacon seeds anywhere.

JC Zak Why do cows wear bells? Their horns don’t work.

PC Zak Rumors are carried by haters, spread by fools, and accepted by idiots.

PD Zak It is not work that kills, but worry.

PD Zak It is not work that kills, but worry.

JD Zak What goes up and down but never moves? - The stairs!

JB Zak How do you tell that a crab is drunk? It walks forwards.

JokeClient 3:

Jokeclient: using server: localhost, port 4545

Please enter your user name:

Sannya

JB Sannya How do you tell that a crab is drunk? It walks forwards.

PA Sannya You can't wake a person who is pretending to be asleep.

PB Sannya The talkative bird can not make a nest.

PD Sannya It is not work that kills, but worry.

JC Sannya Why do cows wear bells? Their horns don’t work.

JD Sannya What goes up and down but never moves? - The stairs!

PC Sannya Rumors are carried by haters, spread by fools, and accepted by idiots.