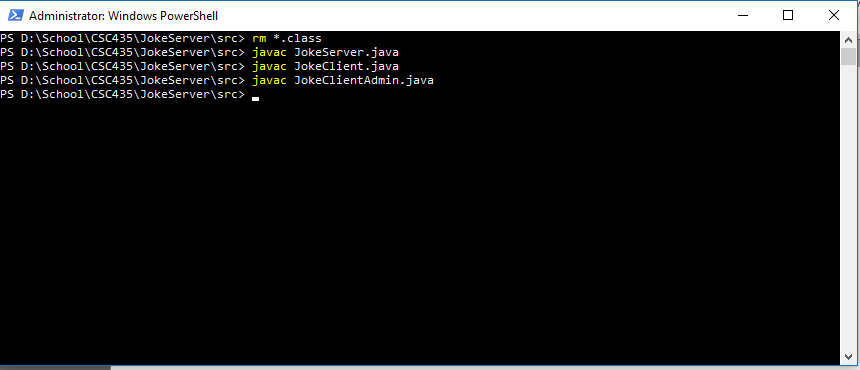
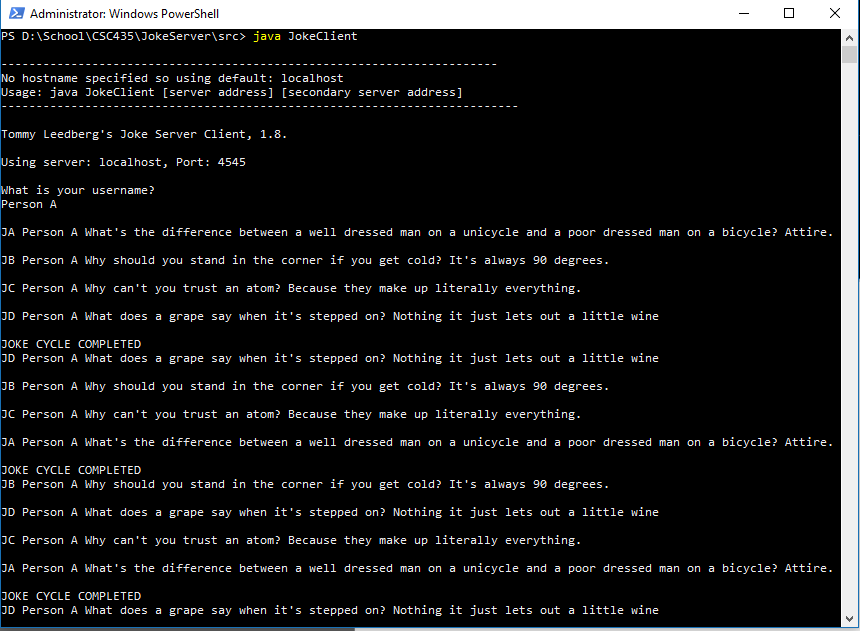
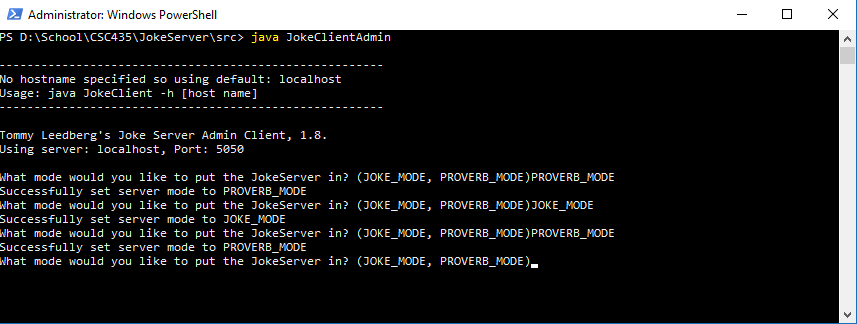
NOTE: JokeServer.java must be compiled first.

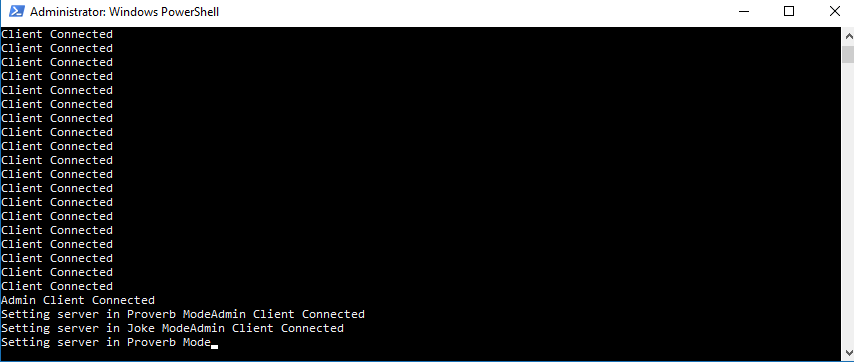


Joke’s are cycled and randomized

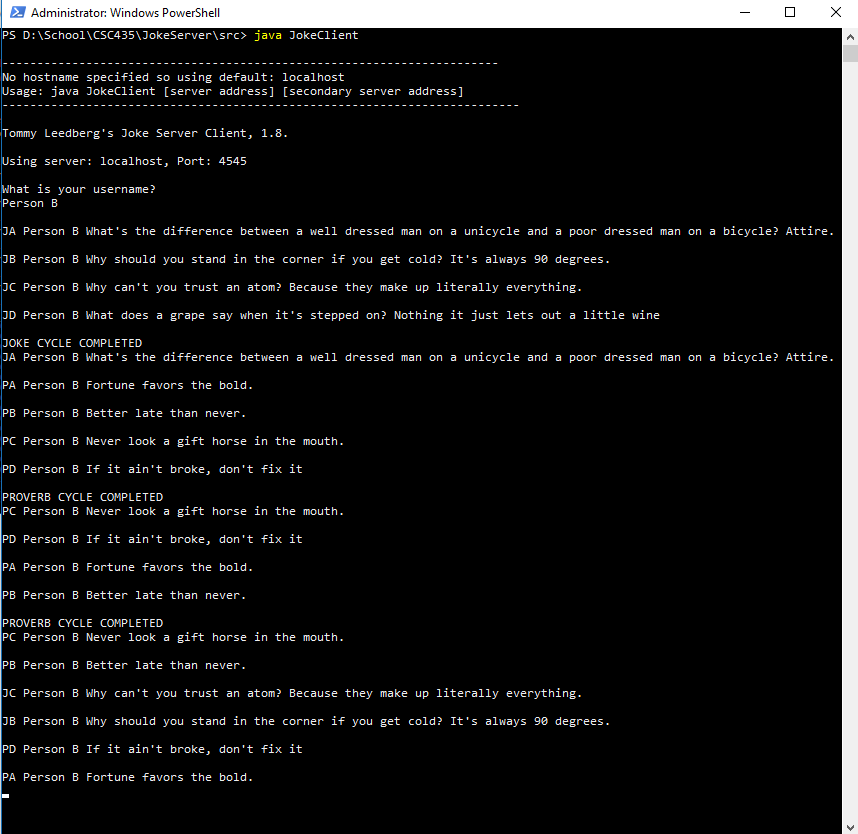


The JokeClientAdmin can switch the server mode, server mode starts in Joke Mode





You can flip flop between modes without losing your place in the cycle



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

Name: Tommy Leedberg

Date: September 12, 2018

Java Version: 1.8.0\_181

Command-Line Examples:

Usage: java JokeServer -p [secondary]

Start only 1 server

java JokeServer

NOTE: Not implemented yet

Start 2 servers

java JokeServer secondary

Instructions:

To Compile:

javac JokeServer.java

NOTE: JokeServer must be compiled first!

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

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.io.PrintStream;

import java.net.ServerSocket;

import java.net.Socket;

import java.util.ArrayList;

import java.util.Collections;

import java.util.LinkedHashMap;

import java.util.Map;

enum ServerModes

{

JOKES,

PROVERBS

}

/\*\*

\* A Data object representing a server request

\*/

class ServerRequest

{

public String userId;

public String clientRequest;

ServerRequest()

{

}

ServerRequest(String clientRequest, String userId)

{

this.clientRequest = clientRequest;

this.userId = userId;

}

ServerRequest(String request)

{

String[] requestProperties = request.split(";");

// There will always( at least in my applications implementations) be at least 1 value in the request

this.clientRequest = requestProperties[0];

// user id's are not required for the admin client so we may not have a value for tone

if (requestProperties.length == 2)

{

this.userId = requestProperties[1];

}

else

{

this.userId = "admin";

}

}

public String toString()

{

return this.clientRequest + ";" + this.userId;

}

}

/\*\*

\* A Data object representing a joke or proverb

\*/

class JokeProverb

{

private String id;

private String body;

JokeProverb(String id, String body)

{

this.id = id;

this.body = body;

}

public String getId()

{

return this.id;

}

public String getBody()

{

return this.body;

}

}

/\*\*

\* ClientWorker thread class

\*/

class ClientWorker extends Thread

{

public static ServerModes ServerMode = ServerModes.JOKES;

// this isn't great, very inefficient but i'm short on time and need to get this done so down the rabbit hole i go

private static ArrayList<JokeProverb> Jokes = new ArrayList<>();

private static ArrayList<JokeProverb> Proverbs = new ArrayList<>();

private static LinkedHashMap<String, LinkedHashMap<String, Boolean>> UsersJokes = new LinkedHashMap<>();

private static LinkedHashMap<String, LinkedHashMap<String, Boolean>> UsersProverbs = new LinkedHashMap<>();

private Socket socket;

ClientWorker(Socket s)

{

this.socket = s;

initializeJokes();

initializeProverbs();

}

/\*\*

\* Run the joke server

\*/

public void run()

{

System.out.println("Client Connected");

PrintStream out = null;

BufferedReader in = null;

try

{

// create an output stream on the specified socket

out = new PrintStream(this.socket.getOutputStream());

// create an input stream on the specified socket

in = new BufferedReader(new InputStreamReader(this.socket.getInputStream()));

try

{

// get the request from the client

ServerRequest request = new ServerRequest(in.readLine());

switch (request.clientRequest)

{

case "GET":

{

if (ServerMode == ServerMode.JOKES)

{

if (!UsersJokes.containsKey(request.userId))

{

this.initializeUsersJokes(request.userId);

}

String jokeId = getNextJoke(request.userId);

if (jokeId == "")

{

this.shuffleJokes(request.userId);

out.println("JOKE CYCLE COMPLETED");

jokeId = this.getNextJoke(request.userId);

}

out.println(buildJokeResponse(request.userId, jokeId));

}

else if (ServerMode == ServerMode.PROVERBS)

{

if (!UsersProverbs.containsKey(request.userId))

{

this.initializeUsersProverbs(request.userId);

}

String proverbId = getNextProverb(request.userId);

if (proverbId == "")

{

this.shuffleProverbs(request.userId);

out.println("PROVERB CYCLE COMPLETED");

proverbId = this.getNextProverb(request.userId);

}

out.println(buildProverbResponse(request.userId, proverbId));

}

break;

}

}

}

catch (Exception e)

{

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

e.printStackTrace();

}

finally

{

this.socket.close();

}

}

catch (IOException e)

{

System.out.println("Error opening i/o pipe on the specified socket: " + e);

}

}

/\*\*

\* Initialize the joke list

\*/

private void initializeJokes()

{

Jokes.add(new JokeProverb("JA", "What's the difference between a well dressed man on a unicycle and a poor dressed man on a bicycle? Attire."));

Jokes.add(new JokeProverb("JB", "Why should you stand in the corner if you get cold? It's always 90 degrees."));

Jokes.add(new JokeProverb("JC", "Why can't you trust an atom? Because they make up literally everything."));

Jokes.add(new JokeProverb("JD", "What does a grape say when it's stepped on? Nothing it just lets out a little wine"));

}

/\*\*

\* Initialize the Proverbs list

\*/

private void initializeProverbs()

{

Proverbs.add(new JokeProverb("PA", "Fortune favors the bold."));

Proverbs.add(new JokeProverb("PB", "Better late than never."));

Proverbs.add(new JokeProverb("PC", "Never look a gift horse in the mouth."));

Proverbs.add(new JokeProverb("PD", "If it ain't broke, don't fix it"));

}

/\*\*

\* Initialize a new user's jokes by adding them to the UsersJokes hashmap

\*

\* @param userId the userId

\*/

private void initializeUsersJokes(String userId)

{

LinkedHashMap<String, Boolean> jokeIds = new LinkedHashMap<>();

for (JokeProverb joke : Jokes)

{

jokeIds.put(joke.getId(), false);

}

UsersJokes.put(userId, jokeIds);

}

/\*\*

\* Initialize a new user's proverbs by adding them to the UsersProverbs hashmap

\*

\* @param userId the userId

\*/

private void initializeUsersProverbs(String userId)

{

LinkedHashMap<String, Boolean> proverbIds = new LinkedHashMap<>();

for (JokeProverb proverb : Proverbs)

{

proverbIds.put(proverb.getId(), false);

}

UsersProverbs.put(userId, proverbIds);

}

/\*\*

\* Get the next joke that hasnt been heard

\*

\* @return the next joke that hasnt been heard

\*/

private String getNextJoke(String userId)

{

String jokeId = "";

for (Map.Entry<String, Boolean> joke : UsersJokes.get(userId).entrySet())

{

if (!joke.getValue())

{

jokeId = joke.getKey();

break;

}

}

if (jokeId != "")

{

UsersJokes.get(userId).replace(jokeId, true);

}

return jokeId;

}

/\*\*

\* Get the next proverb that hasnt been heard

\*

\* @return the next proverb that hasnt been heard

\*/

private String getNextProverb(String userId)

{

String proverbId = "";

for (Map.Entry<String, Boolean> proverb : UsersProverbs.get(userId).entrySet())

{

if (!proverb.getValue())

{

proverbId = proverb.getKey();

break;

}

}

if (proverbId != "")

{

UsersProverbs.get(userId).replace(proverbId, true);

}

return proverbId;

}

private void shuffleJokes(String userId)

{

// All jokes have been heard so shuffle and reset

// Another fairly inefficient process :(

ArrayList<String> jokeIds = new ArrayList<>(UsersJokes.get(userId).keySet());

Collections.shuffle(jokeIds);

UsersJokes.get(userId).clear();

for (String jokeId : jokeIds)

{

UsersJokes.get(userId).put(jokeId, false);

}

}

private void shuffleProverbs(String userId)

{

// All proverbs have been heard so shuffle and reset

// Another fairly inefficient process thats repeated ahhhhh :(

ArrayList<String> proverbIds = new ArrayList<>(UsersProverbs.get(userId).keySet());

Collections.shuffle(proverbIds);

UsersProverbs.get(userId).clear();

for (String proverbId : proverbIds)

{

UsersProverbs.get(userId).put(proverbId, false);

}

}

/\*\*

\* Lookup a Joke from the Joke Directory

\*

\* @param jokeId The id of the joke to lookup

\* @return The joke

\*/

private String lookupJoke(String jokeId)

{

for (JokeProverb joke : Jokes)

{

if (joke.getId() == jokeId)

{

return joke.getBody();

}

}

return "invalid joke id";

}

/\*\*

\* Lookup a Proverb from the Proverb Directory

\*

\* @param proverbId The id of the proverb to lookup

\* @return The proverb

\*/

private String lookupProverb(String proverbId)

{

for (JokeProverb proverb : Proverbs)

{

if (proverb.getId() == proverbId)

{

return proverb.getBody();

}

}

return "invalid proverb id";

}

/\*\*

\* Build the joke response to send to the client

\*

\* @param userId the userId

\* @param jokeId the jokeId to lookup the joke

\* @return a String representing JokeId UserId Joke

\*/

private String buildJokeResponse(String userId, String jokeId)

{

return jokeId + " " + userId + " " + lookupJoke(jokeId);

}

/\*\*

\* Build the proverb response to send to the client

\*

\* @param userId the userId

\* @param proverbId the proverbId to lookup the joke

\* @return a String representing ProverbId UserId Proverb

\*/

private String buildProverbResponse(String userId, String proverbId)

{

return proverbId + " " + userId + " " + lookupProverb(proverbId);

}

}

/\*\*

\* The administration clients worker class

\*/

class AdminWorker extends Thread

{

private Socket socket;

AdminWorker(Socket s)

{

this.socket = s;

}

public void run()

{

System.out.println("Admin Client Connected");

PrintStream out = null;

BufferedReader in = null;

try

{

// create an output stream on the specified socket

out = new PrintStream(this.socket.getOutputStream());

// create an input stream on the specified socket

in = new BufferedReader(new InputStreamReader(this.socket.getInputStream()));

try

{

// The admin client only really makes 1 request, Joke Mode or Server Mode so a string is fine

ServerRequest request = new ServerRequest(in.readLine());

switch (request.clientRequest)

{

case "JOKE\_MODE":

{

ClientWorker.ServerMode = ServerModes.JOKES;

System.out.print("Setting server in Joke Mode");

out.println("Successfully set server mode to " + request.clientRequest);

break;

}

case "PROVERB\_MODE":

{

ClientWorker.ServerMode = ServerModes.PROVERBS;

System.out.print("Setting server in Proverb Mode");

out.println("Successfully set server mode to " + request.clientRequest);

break;

}

default:

{

out.println("Invalid Server Mode.");

}

}

}

catch (Exception e)

{

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

e.printStackTrace();

}

finally

{

this.socket.close();

out.close();

in.close();

}

}

catch (IOException e)

{

System.out.println("Error opening i/o pipe on the specified socket: " + e);

}

}

}

/\*\*

\* The administration clients thread

\*/

class AdminReceiver implements Runnable

{

public void run()

{

int q\_len = 6;

// the Admin Client listens on port 5050

int port = 5050;

Socket socket;

try

{

ServerSocket serverSocket = new ServerSocket(port, q\_len);

System.out.println(String.format("Listening for the admin client on port %s.", port));

while (true)

{

// wait for the next ADMIN client connection:

socket = serverSocket.accept();

// Once a connection has come in start the admin worker

new AdminWorker(socket).start();

}

}

catch (IOException e)

{

System.out.println("Failed to start client admin worker with exception: " + e);

}

}

}

class SecondaryServer implements Runnable

{

public void run()

{

int q\_len = 6;

// the Admin Client listens on port 5050

int port = 4546;

Socket socket;

try

{

ServerSocket serverSocket = new ServerSocket(port, q\_len);

System.out.println(String.format("Listening for the secondary server connections on port %s.", port));

while (true)

{

// wait for the next ADMIN client connection:

socket = serverSocket.accept();

// Once a connection has come in start the Client worker

new ClientWorker(socket).start();

}

}

catch (IOException e)

{

System.out.println("Failed to start client admin worker with exception: " + e);

}

}

}

/\*\*

\* Joke Server class

\*/

public class JokeServer

{

public static void main(String args[])

{

int q\_len = 6;

int port = 4545;

boolean useSecondaryServer = false;

Socket socket;

if (args.length == 0)

{

System.out.println("\n-------------------------------------------------------");

System.out.println("Usage: java JokeServer [secondary]");

System.out.println("-------------------------------------------------------\n");

}

if (args.length == 1 && args[1] == "secondary")

{

useSecondaryServer = true;

}

System.out.println("Tommy Leedberg's Joke server 1.8 starting up");

try

{

// create a new admin client thread to listen on port 5050

AdminReceiver adminReceiverThread = new AdminReceiver();

Thread aThread = new Thread(adminReceiverThread);

aThread.start();

}

catch (Exception e)

{

System.out.println("Failed to create client admin thread with exception: " + e);

}

// We want to use a secondary server so start a new thread to allow multiple servers to run

if (useSecondaryServer)

{

try

{

// create a new admin client thread to listen on port 5050

SecondaryServer secondaryServerThread = new SecondaryServer();

Thread bThread = new Thread(secondaryServerThread);

bThread.start();

}

catch (Exception e)

{

System.out.println("Failed to create secondary server with exception: " + e);

}

}

try

{

// create a socket on the given port for clients to connect to

ServerSocket serverSocket = new ServerSocket(port, q\_len);

System.out.println(String.format("Listening for clients on port %s.", port));

while (true)

{

// Wait for the incoming connection

socket = serverSocket.accept();

// Kick off the client working thread

new ClientWorker(socket).start();

}

}

catch (IOException e)

{

System.out.println("Failed to start client worker with exception: " + e);

}

}

}

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

Name: Tommy Leedberg

Date: September 12, 2018

Java Version: 1.8.0\_181

Command-Line Examples:

Usage: java JokeClient [server address] [server address]

Connect to 1 server only

java JokeClient localhost

NOTE: Not implemented yet

Start to 2 servers

java jokeClient localhost localhost

Instructions:

To Compile:

javac JokeClient.java

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

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.io.PrintStream;

import java.net.Socket;

import java.net.UnknownHostException;

/\*\*

\* The Joke Server client

\*/

public class JokeClient

{

/\*\*

\* The main entry point of the JokeClient

\* @param args The command Line Arguments

\*/

public static void main(String args[])

{

if (args.length == 0)

{

System.out.println("\n-----------------------------------------------------------------------");

System.out.println("No hostname specified so using default: localhost");

System.out.println("Usage: java JokeClient [server address] [secondary server address]");

System.out.println("--------------------------------------------------------------------------\n");

}

// The default server address is localhost

String serverAddress = "localhost";

// Check if there is a command line argument supplied specifying the server address

if (args.length == 1)

{

serverAddress = args[0];

}

try

{

// create the primary client thread to listen on port 4545

JokeClientThread primaryClientThread = new JokeClientThread(4545, serverAddress);

Thread aThread = new Thread(primaryClientThread);

aThread.start();

}

catch (Exception e)

{

System.out.println("Failed to create primary client thread. Exception: " + e);

}

// Check if we want to start a secondary server connection

if (args.length == 2)

{

serverAddress = args[1];

}

/\*try

{

// create a secondary client thread to listen on port 5050

JokeClientThread secondaryJokeClientThread = new JokeClientThread(4546, serverAddress);

Thread aThread = new Thread(secondaryJokeClientThread);

aThread.start();

}

catch (Exception e)

{

System.out.println("Failed to create second client thread. Exception: " + e);

}\*/

}

}

/\*\*

\* The Joke Client thread to start up instances of the joke client

\*/

class JokeClientThread implements Runnable

{

private int port;

private String serverAddress;

/\*\*

\* Creates an instance of the JokeClientThread

\*

\* @param port The port to use when starting the worker

\* @param serverAddress The server address to use when starting the worker

\*/

JokeClientThread(int port, String serverAddress)

{

this.port = port;

this.serverAddress = serverAddress;

}

public void run()

{

try

{

new JokeClientWorker(this.port, this.serverAddress).start();

}

catch (Exception e)

{

System.out.println("Failed to start client admin worker for server " + this.serverAddress + ". Exception: " + e);

}

}

}

/\*\*

\* The Joke Client Worker

\*/

class JokeClientWorker extends Thread

{

private int port;

private String serverAddress;

private String userToken;

private Socket client;

/\*\*

\* Joke Client Worker Constructor

\* @param port The port to open a connection on

\* @param serverAddress The server address to open a connection to

\*/

JokeClientWorker(int port, String serverAddress)

{

this.port = port;

this.serverAddress = serverAddress;

}

/\*\*

\* The threads run method

\*/

public void run()

{

System.out.println("Tommy Leedberg's Joke Server Client, 1.8.\n");

System.out.println(String.format("Using server: " + this.serverAddress + ", Port: %s\n", this.port));

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

try

{

// Ask for the user credentials

generateUserToken(in);

String request;

while (!(request = in.readLine()).contains("quit"))

{

// The server only accepts 1 command "GET" so we aren't worried about the input

writeServerRequest("GET");

}

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

}

catch (IOException e)

{

System.out.println("Error getting user input. Exception: " + e);

}

}

/\*\*

\* Generate a user token( in this case it's a simple user name )

\* @param in The buffered reader's input stream

\*/

private void generateUserToken(BufferedReader in)

{

try

{

System.out.println("What is your username?");

System.out.flush();

this.userToken = in.readLine();

if (this.userToken.length() == 0)

{

System.out.println("You must enter a valid username.");

this.generateUserToken(in);

}

}

catch (IOException e)

{

System.out.println("Error reading username. Exception: " + e);

}

}

/\*\*

\* Sends the command to the joke server

\* @param commandString The command to send to the Joke Server

\*/

private void writeServerRequest(String commandString)

{

BufferedReader fromServer;

PrintStream toServer;

String textFromServer;

try

{

// Open a connection to server, if you can't let the user know and wait for futher input

if (!openConnection())

{

System.out.println("Waiting on Server Connection...");

return;

}

ServerRequest request = new ServerRequest(commandString, this.userToken);

// create an output stream on the specified socket

toServer = new PrintStream(this.client.getOutputStream());

// create an input stream on the specified socket

fromServer = new BufferedReader(new InputStreamReader(this.client.getInputStream()));

// Send the joke server the command

toServer.println(request.toString());

toServer.flush();

// read in and then print out the response from the server

while ((textFromServer = fromServer.readLine()) != null && textFromServer.length() != 0)

{

System.out.println(textFromServer);

}

}

catch (IOException e)

{

System.out.println("Socket error. Exception: " + e);

e.printStackTrace();

}

}

/\*\*

\* Open a new connection to the server on the specified port

\* due to limitations on files code is duplicated

\*

\* @return A value indicating whether or not a connection was able to be made

\*/

private Boolean openConnection()

{

try

{

this.client = new Socket(this.serverAddress, this.port);

return true;

}

catch (UnknownHostException e)

{

System.out.println("Invalid Host. Exception " + e);

return false;

}

catch (IOException e)

{

System.out.println( "Failed to connect to socket. Exception: " + e);

return false;

}

}

}

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

Name: Tommy Leedberg

Date: September 12, 2018

Java Version: 1.8.0\_181

Command-Line Examples/Instructions:

Command-Line Examples:

Usage: java JokeClientAdmin [server address] [server address]

Connect to 1 server only

java JokeClientAdmin localhost

NOTE: Not implemented yet

Start to 2 servers

java jokeClientAdmin localhost localhost

Instructions:

To Compile:

javac JokeClientAdmin.java

To change mode enter either JOKE\_MODE or PROVERB\_MODE

depending on the mode you want to switch to

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

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.io.PrintStream;

import java.net.Socket;

import java.net.UnknownHostException;

/\*\*

\* The Joke Client Admin

\*/

public class JokeClientAdmin

{

public static void main(String args[])

{

if (args.length == 0)

{

System.out.println("\n-------------------------------------------------------");

System.out.println("No hostname specified so using default: localhost");

System.out.println("Usage: java JokeClient -h [host name]");

System.out.println("-------------------------------------------------------\n");

}

// The default server address is localhost

String serverAddress = "localhost";

// Check if there is a command line argument supplied specifying the server address

if (args.length == 1)

{

serverAddress = args[0];

}

try

{

// create the primary admin client thread to listen on port 5050

AdminClientThread primaryAdminClientThread = new AdminClientThread(5050, serverAddress);

Thread aThread = new Thread(primaryAdminClientThread);

aThread.start();

}

catch (Exception e)

{

System.out.println("Failed to create primary admin client thread. Exception: " + e);

}

// Check if we want to start a secondary server connection

/\* if (args.length == 2)

{

serverAddress = args[1];

}

try

{

// create a secondary admin client thread to listen on port 5051

AdminClientThread secondaryAdminClientThread = new AdminClientThread(5051, serverAddress);

Thread aThread = new Thread(secondaryAdminClientThread);

aThread.start();

}

catch (Exception e)

{

System.out.println("Failed to create second admin client thread. Exception: " + e);

}\*/

}

}

/\*\*

\* The admin client thread to start instances of the admin client

\*/

class AdminClientThread implements Runnable

{

private int port;

private String serverAddress;

/\*\*

\* Creates an instance of the AdminClientThread

\*

\* @param port The port to use when starting the worker

\* @param serverAddress The server address to use when starting the worker

\*/

AdminClientThread(int port, String serverAddress)

{

this.port = port;

this.serverAddress = serverAddress;

}

public void run()

{

try

{

new AdminClientWorker(this.port, this.serverAddress).start();

}

catch (Exception e)

{

System.out.println("Failed to start client admin worker for server " + this.serverAddress + ". Exception: " + e);

}

}

}

/\*\*

\* The Admin Client worker

\*/

class AdminClientWorker extends Thread

{

private int port;

private String serverAddress;

private Socket client;

/\*\*

\* Joke Client Worker Constructor

\* @param port The port to open a connection on

\* @param serverAddress The server address to open a connection to

\*/

AdminClientWorker(int port, String serverAddress)

{

this.port = port;

this.serverAddress = serverAddress;

}

/\*\*

\* The threads run method

\*/

public void run()

{

System.out.println("Tommy Leedberg's Joke Server Admin Client, 1.8.");

System.out.println(String.format("Using server: " + this.serverAddress + ", Port: %s\n", this.port));

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

try

{

String command = "";

while (!command.contains("quit"))

{

System.out.print("What mode would you like to put the JokeServer in? (JOKE\_MODE, PROVERB\_MODE)");

System.out.flush();

command = in.readLine();

if (command.contains("quit"))

{

continue;

}

// If the connection isn't available yet do not try to send a message.

if (!openConnection())

{

System.out.println("Waiting on Server Connection...");

continue;

}

writeServerRequest(command);

System.out.flush();

this.client.close();

}

System.out.println("Connection cancelled by user request.");

}

catch (IOException x)

{

x.printStackTrace();

}

}

/\*\*

\* Sends the admin command to the Joke Server

\* @param command The command to send

\*/

private void writeServerRequest(String command)

{

PrintStream toServer;

BufferedReader fromServer;

String textFromServer;

try

{

ServerRequest request = new ServerRequest();

request.clientRequest = command;

// create an output stream on the specified socket

toServer = new PrintStream(this.client.getOutputStream());

// create an input stream on the specified socket

fromServer = new BufferedReader(new InputStreamReader(this.client.getInputStream()));

// Send the joke server the command

toServer.println(request.toString());

toServer.flush();

// read in and then print out the response from the server

while ((textFromServer = fromServer.readLine()) != null && textFromServer.length() != 0)

{

System.out.println(textFromServer);

}

}

catch (IOException e)

{

System.out.println("Error writing error. Exception: " + e);

e.printStackTrace();

}

catch (Exception e)

{

System.out.println("Socket error. Exception: " + e);

}

}

/\*\*

\* Open a new connection to the server on the specified port

\* due to limitations on files code is duplicated

\*

\* @return A value indicating whether or not a connection was able to be made

\*/

private Boolean openConnection()

{

try

{

this.client = new Socket(this.serverAddress, this.port);

return true;

}

catch (UnknownHostException e)

{

System.out.println("Invalid Host. Exception " + e);

return false;

}

catch (IOException e)

{

System.out.println( "Failed to connect to socket. Exception: " + e);

return false;

}

}

}