In The Name Of God

Student Name : Narges Habibi – Fateme Sharifi
Student ID Number : 8004311 – 8203593

1- Development Environment)

1-1 OS : Windows XP Professional
1-2 Editor : Eclipse V.3
1-3 Java : JDK 1.5

(**Note!** : in description commands <> means variable)

2- Usage Documentation)

2-1 Registration Server

Registration Server is used by all buddies to know who is online. The server will run as a separate program, exist on a separate machine of a known IP address, and will list all registered buddies. Clients will be able to ask the server who is online so that they can initiate chat sessions. The registration server's main function is to tell clients the IP addresses of other clients.
For run registration server type : >server.exe
This command runs registration server part of project on port 54321.

2-2 Server Commands (all commands are case sensitive)

- For view all online clients (clients that registered with this server) :
  !!!show_buddies
- For shutting down server :
  !!!quit

Server announces each new client connection by print : " new connection.
<buddy_name> : IP " .

2-3 Client

A peer to peer **Chat Client** that, given the location of a remote buddy, can connect to and chat with a remote buddy. It will be possible to chat with more than one buddy at a time from one command prompt. There will be commands to send messages to all buddies at once or to just one buddy. There will also be commands to stop chatting with a buddy, stop chatting with all buddies, and to terminate the chat application.
For run a client type : >client.exe buddy_name registration_server_IP or registration_server_domain_name
If <buddy_name> is longer than one word, it must be between "".
This command runs client on port 12345.

2-4 Client Commands (commands are case sensitive)

- For view all online buddies that they registered with server :
  !!!show_buddies
  Without any additional characters.
- For start chat with a buddy :
  !!!start_chat any number of space or tab <buddy_name> | buddy IP
- For exit :
  !!!quit

Without any additional character.

- For disconnect from all connected buddies :
  !!!disconnect_all
  Without any additional character.
- For disconnect from one connected buddy :
  !!!disconnect any number of space or tab <buddy_name> | buddy IP
- For sending message to one buddy :
  !!!<buddy_name> any number of space : any number of space Message

3- Overview of Coding Design)

3-1 Project contains 10 java file (1 interface, 9 class) :
ConstantValues
YahooBuddyServer – ServerUserCommands – ServerNetRequest
YahooChat – ServerCinnection – OnlineBuddies – ListenToBuddies – ClientUserCommands – BuddyHandler
We used **BufferedReader** class for read and **DataOutputStream** class for write to/from stream. (standard in/out and socket). BuferedReader has a readLine() method that reads until it sees a '\n'. Then we write '\n' at the end of input.

3-2 registration Server

We defined an application level protocol between client and server. Client and server have interaction in *3 ways* :
1- register) at the start of connection client sends : "register\n<buddy_name>\n".
Server returns "ok\n" or "Duplicated ID, request rejected.\n".
2- show online buddies) client sends : "show_buddies\n", server first sends the number of online clients (except target client) and then name,IP and port of them : "<size>\n<buddy_name1>:<IP>:<port>\n ... \n".
3- get information of buddy) client sends : "buddy_name:<buddy_name>\n" .
Server responses : "<buddy_name>:<IP>:<port>\n" or "" if it is not online.

Server keeps (ID , IP: port) pairs of clients in a Hashtable, because of fast and convenient search.
YahooBuddyServer class : at the start, two methods **responseToCommands()** and **responseToNet()** are called.
responseToCommands() creates an instance of **ServerUserCommands** thread.
ServerUserCommands receives user commands until it sees "!!!quit" command.
For each command it class corresponding method. **Showbuddies()** method prints "ID:IP:port" of all clients in Hashtable.
responseToNet() in main thread creates a ServerSocket and for each incoming connection , creates a new thread of **ServerNetRequest** class and passes it's constructor accepted socket.
ServerNetRequest reads requests from client and handles them. 3 handler method are:

- registerClient() : if ID is already registered, returns and destroys this thread.
  If incoming IP is equals to loopback address, server saves self IP as it's IP.
- showBuddiesToClient()
- wrtiteBuddyInformation(String ID)

3-2 Client

The protocol we defined between clients, contains two type interaction:

1- at the first of connection, requester client sends it's budduy_name.
2- target buddy also writes it's ID.
3- after it, each client simply writes it's message.

Client uses of **OnlineBuddies** class (extends Hashtable) for keeping buddies that has connection to them in form of : (<buddy_name> , socket). It's methods are :

addBuddy (ID , socket)
removeBuddy (ID)
removeAll ()
disconnect (ID)
getIDs ()
writeToBuddy (ID , Message)

YahooChat class : first creates an instance of OnlineBuddies class. Then processes command line arguments. If user doesn't want server or connection is not possible, sets **withServer** variable to false and calls **start()** method.

start() calls helper methods : **registerWithServer()** and **responseToCommands()**.

registerWithServer() : this method assume that command-line argument is IP of server. if it fails, tries it as domain name. it uses from **ServerConnection** class open connection with registration server and read and write to/from it.

responseToCommands() : creates an instance of **ClientUserCommands** class that work in main thread. Client UserCommands class appropriate method for handle user command. For example **startChat(ID)** method , assume that ID is IP of a buddy. if connection fails in this way , asks from registration server. If ID is online, gets response from server, creates a connection with corresponding IP and adds it to OnlineBuddies. Finally, creates a thread for listen to connected buddy.

start() also creates an instance of **ListenToBuddies** thread for startup it's server. for each new connection creates a thread of **BuddyHandler** class.

BuddyHandler records (ID , socket) of incoming connection in OnlineBuddies.