دانشکده کامپیوتر دانشگاه

موضوع پروژه: ایجاد یک پیام رسان ساده به زبان جاوا با استفاده از *Socket*

استاد:

به تلاش: پیمان برجوییان

شماره شناسایی:

معرفی:

برنامه حاضر یک پیام رسان بوده که دارای قابلیت های پشتیبانی از چند کاربر و ارسال فایل را دارد.

این برنامه تماما بر اساس جاوا بوده و از ۲ بخش:

۱. *jServer* یا برنامه سمت سرویس دهنده
۲. *jMessenger* یا برنامه سمت سرویس گیرنده

ویژگی ها:

۱. پشتیبانی از چند کاربر در یک زمان
۲. پشتیبانی از پیغام های عمومی و خصوصی
۳. پشتیبانی از ایجاد و استفاده از حساب کاربری (پایگاه داده)
۴. پشتیبانی از ارسال و دریافت فایل

اجرای برنامه:

ابتدا فایل های *jServer.jar* و *jMessenger.jar* را اجرا کرده و مراحل زیر را انجام دهید:

۱. در *jServer* فایل *data.xml* را به عنوان پایگاه داده انتخاب کنید. این فایل نام های کاربری و رمزهای عبور را در بر می گیرد.
۲. در *jMessenger* اگر سرویس دهنده پیدا نشد فایروال خود را به منظور دادن اجازه دسترسی تنظیم نمایید.

ساختار پیام ها:

هر پیام در *jMessenger* دارای ۴ قسمت می باشد:

- *type*: این قسمت نوع پیام را در بر میگیرد و مقادیری مانند: *login, newuser, message* و ... را می پذیرد.
- *sender*: این قسمت شامل نام کاربری فرستنده می باشد.
- *content*: این قسمت محتوای پیام را در بر می گیرد.
- *recipient*: این قسمت شامل نام کاربری گیرنده می باشد.

jServer:

۲ کلاس اصلی در *jServer* برای ایجاد اتصال ها و پیام ها داریم. در ابتدا *SocketServer* توسط یک *Thread* جدا گونه اجرا می شود. وظیفه ی این کلاس منتظر بودن برای ایجاد اتصال می باشد و به ازاء هر درخواست اتصال *Thread* از جنس *ServerThread* را شروع می کند. وقتی که اتصال برقرار شد *ServerThread* منتظر دریافت پیام می ماند و زمانی که پیامی دریافت کند آن را جهت پردازش مجددا به *SocketServer* پاس می دهد.

مهمترین پردازشی که انجام می شود فرستادن پیام به گیرنده آن می باشد.

```
 // In ServerThread read the incoming message and hand it to SocketServer

Message msg = (Message) streamIn.readObject();
server.handle(ID, msg);
.......


// In SocketServer process the messages based on their type

public synchronized void handle(int ID, Message msg){
         if(msg.type.equals("login")){
                  ....
         }
         else id(msg.type.equals("message")){
                  if(msg.recipient.equals("All")){ Announce("message", msg.sender,
msg.content); }
                  else{
                           // Find the thread of recipient and forward it to him
                  }
         }
.......
```

jMessenger:

jMessenger در مرحله اول بوسیله *IP* و *Port* مشخص شده به *jServer* متصل می شود. پیام هایی که می رسند همراه با نام فرستنده آن ها در *jMessenger* نمایش داده می شوند.

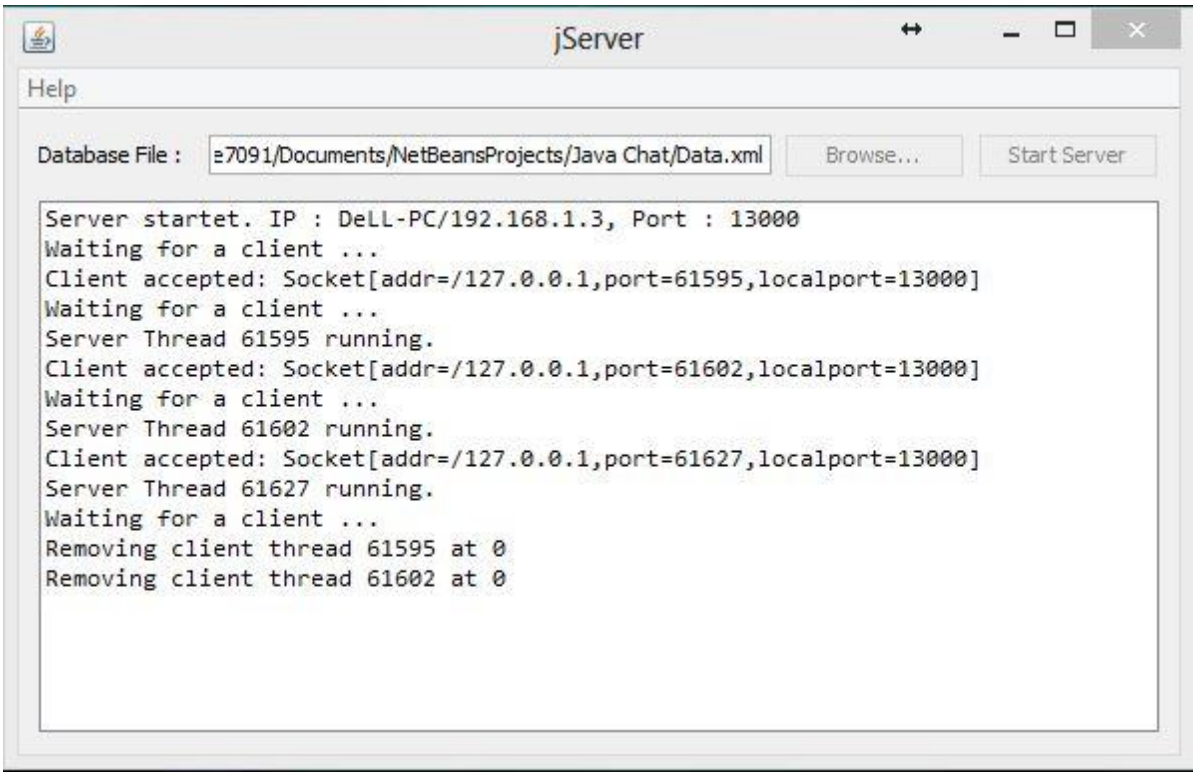
زمانی که کاربر بخواهد فایل ارسال کند ابتدا یک در خواست ارسال از طریق یک پیام از نوع *upload_req* به گیرنده ارسال می شود. سپس در گیرنده مراحل زیر انجام می شود:

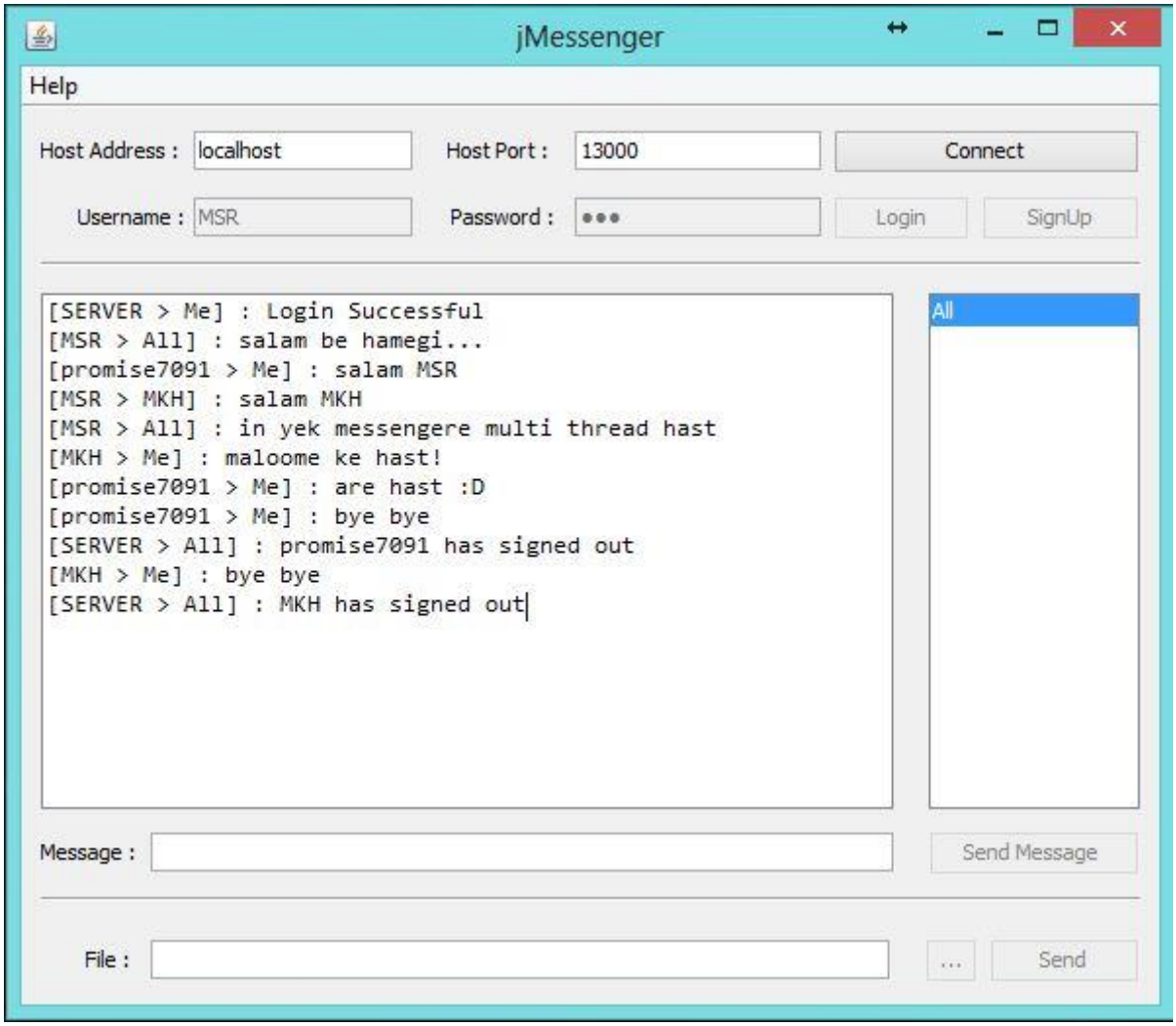
۱. گیرنده توسط پیامی از نوع *upload_res* به درخواست پاسخ می دهد.
۲. اگر گیرنده با درخواست موافقت کرده باشد یک *Port* جدید برای دریافت فایل ایجاد می کند.
۳. اگر در گیرنده با در خواست موافقت شود *IP* و *Port* آن به فرستنده ارسال می شود.
۴. فرستنده با دریافت پاسخ مثبت از سوی گیرنده توسط *Socket* مشخص شده عملیات *upload* را آغاز می کند.

از مزایای این کار امکان چت کردن و ارسال فایل به صورت همزمان می باشد و بر خلاف پیام ها فایل ها از طریق سرویس دهنده ارسال نمیشوند بلکه عمل ارسال به صورت مستقیم و نظیر به نظیر انجام می شود.

```
// On recipient side, start a new thread for download

Download dwn = new Download(....);
Thread t = new Thread(dwn);
t.start();
send(new Message("upload_res", ui.username, dwn.port, msg.sender));
// Reply to sender with IP address and port number
.........

// On sender side, start a new thread for file upload

// Connect to the port specified in reply
Upload upl = new Upload(addr, port, ui.file, ui);
Thread t = new Thread(upl);
t.start();
```

نمای تصویری از محیط برنامه:




jMessenger
Help
Host Address : localhost
Host Port : 13000
Connect
Username : MSR
Password : •••
Login
SignUp
[SERVER > Me] : Login Successful
[MSR > All] : salam be hamegi...
[promise7091 > Me] : salam MSR
[MSR > MKH] : salam MKH
[MSR > All] : in yek messengere multi thread hast
[MKH > Me] : maloome ke hast!
[promise7091 > Me] : are hast :D
[promise7091 > Me] : bye bye
[SERVER > All] : promise7091 has signed out
[MKH > Me] : bye bye
[SERVER > All] : MKH has signed out
All
Message :
Send Message
File :
...
Send