**VOIP Project Technical Report**

We used the JVoIP-Java SIP library to build the project. The Mizu Java VoIP SDK (JVoIP) is a SIP client implemented as a platform independent java library. Since it is based on the open standard [Session Initiation Protocol](http://en.wikipedia.org/wiki/Session_Initiation_Protocol), it can inter-operate with any other SIP-based device (servers and clients). We used the JVoIP java library to be able to make calls to different peers by providing the username of the peer we want to call. The VOIP server we’re using is “voip.mizu-voip.com”.

**Most important used Functions**

* API\_SetParameter (String param, String value):

Most of the parameters can be set with this function except gui parameters (like the colors). Some parameters can take effect only when JVoIP is reinitialized.

This function should be used only in special cases. You should be able to control the java sip client without using this function.

* API\_Start():

This has to be called only if you use JVoIP as and SDK or as a java voip library. It shouldn’t be called from JavaScript.

* API\_Call(int line, String peer):

Initiate call to a number or sip username, If the peer parameter is empty, then will redial the last number.

* API\_Hangup(int line, String reasontext):

Disconnect current call(s). If you set -2 for the line parameter, then all calls will be disconnected (in case if there are multiple calls in progress).

* API\_Accept(int line, int calltype):

Connect incoming call. You should call this function when there is an incoming ringing call if you wish to connect the call. The calltype parameter accepts the following values:

 -1: default

 0: audio only

 1: with video

 2: force video

* API\_CallEx(int line, String peer, int calltype)

Initiate call to a number or sip username.

If the peer parameter is empty, then will redial the last number.

The calltype can have the following values:

0: initiate voice call

1: initiate video call

2: initiate screensharing session

* API\_Reject(int line)

Disconnect incoming call.

**Code Flow**

We start by importing the JVoIP library in our project but we need to make sure that the JVoIP.jar executable file is in our project’s directory. Then we create a new class called VoIPProject and initialize our variables. Then we create a constructor for our class where we call the start\_Jvoip() method. We create our main method in which we create a new instance of the VoIPProject class and we use try and catch to print the message of any exception(if any) that may occur. We create the SIPNotifications class where we use it to process and display notifications. These notifications are very important to handle different events.

In the start method, we create a JVoIP instance using the webphone class, then we create the SIPNotifications object to catch the events from JVoIP then we call the sipnotifications.Start() method to start receiving the SIP notifications. Then we set some parameters using the API\_SetParameter() method such as webphoneobj.API\_SetParameter("loglevel", "1"); to set the loglevel, webphoneobj.API\_SetParameter("serveraddress", "voip.mizu-voip.com"); to set the voip server address we are going to use, webphoneobj.API\_SetParameter("transport", "2"); to set the transport protocol to TLS, webphoneobj.API\_SetParameter("username", username); to set the username, webphoneobj.API\_SetParameter("password", pwd); to set the password and etc. We built a GUI (Graphical User Interface) so that we can use it to login, start a session, make and receive calls, accept or reject incoming calls, hang-up an already ongoing call, specify the username of the peer we want to make a call to, ending the session(unregister), and enable or disable voice recording. The user writes his credentials (username and password) that he wants to sign in with, and the user can check or uncheck the “Voice Record” checkbox which if checked, will auto record the voice calls and save them to the specified path, and if unchecked, will disable the voice recording feature where the we use the voice recording feature in the project to ensure that the call was made successfully and the voice on both ends was sent and received successfully. Then the user can click the “Start” button to register to the server, start the session, and start the sip stack. We use the API\_Start() method to start the SIP stack and we use Thread.sleep(200); to wait a bit until the sip stack is fully initialized. When the SIP stack is initialized, we will see “SIP client registered” message in the console. We use the API\_VoiceRecord() method to auto record the voice calls if the Voice Record checkbox is checked. Now, we can make a call to any peer registered on our VOIP server by typing the username of the peer in the destination text field then clicking the “Call” button where we make the call using the API\_CallEx(-1, String peer, 0) method to make an audio call. In the SIPNotifications java class that we created; we process all the notifications. If there is an incoming call we can click the “Answer” button to answer(accept) the incoming call. We use the API\_Accept(-1,0) to accept an incoming call and we give it 0 value for call type parameter indicating that we are making audio only call. We can click the “Reject” button to reject the incoming call. We use the API\_Reject() to reject an incoming call. We can click the “Hangup” button to hangup(disconnect) an already ongoing call. We use the API\_Hangup() method to disconnect an ongoing call. We can click the “End” button to end the session, sign out and exit the program. We use API\_Stop() to stop the sip stack and unregister, and we use the sipnotifications.Stop() method to stop the JVoIP notification listener.

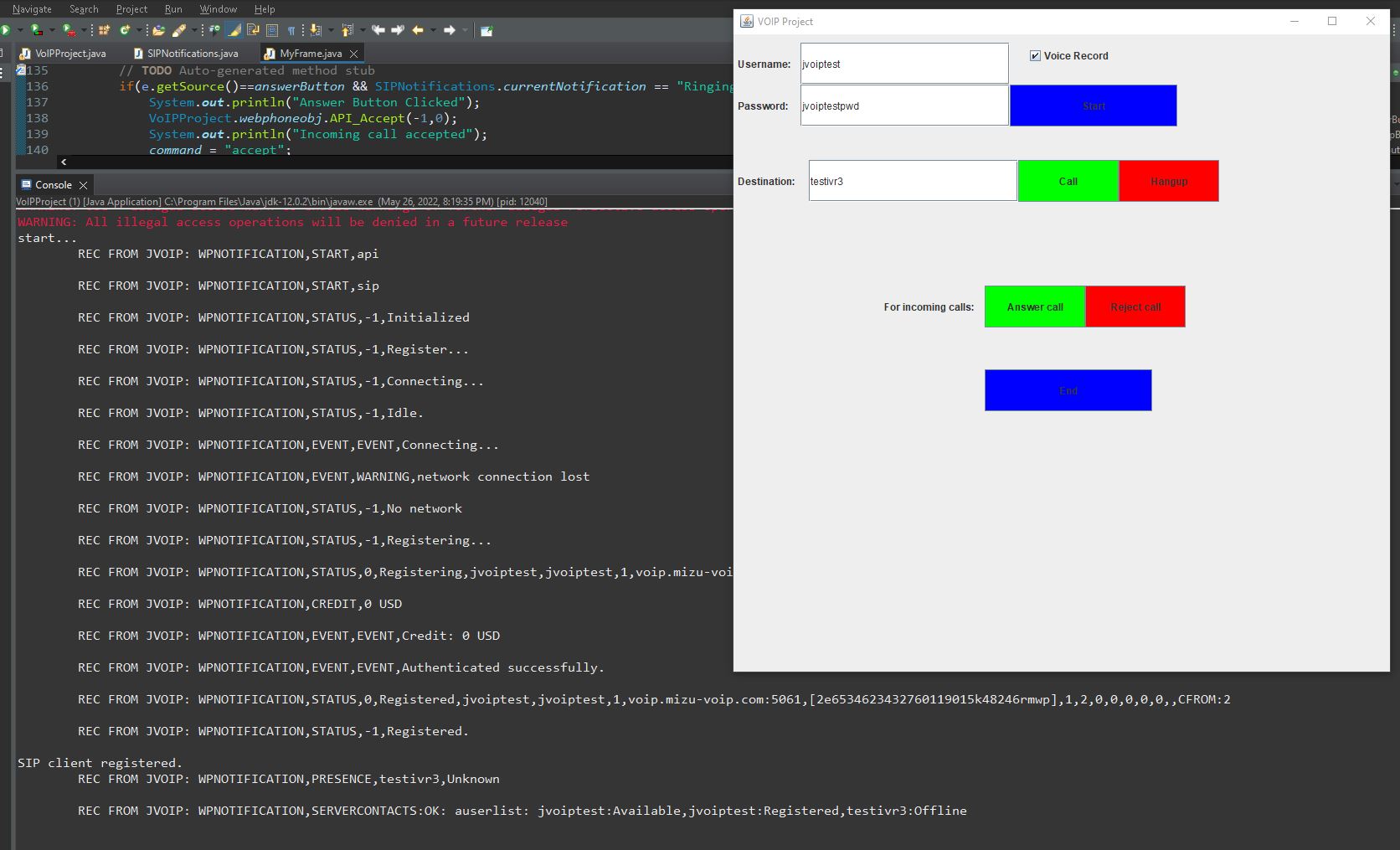
**Performance**

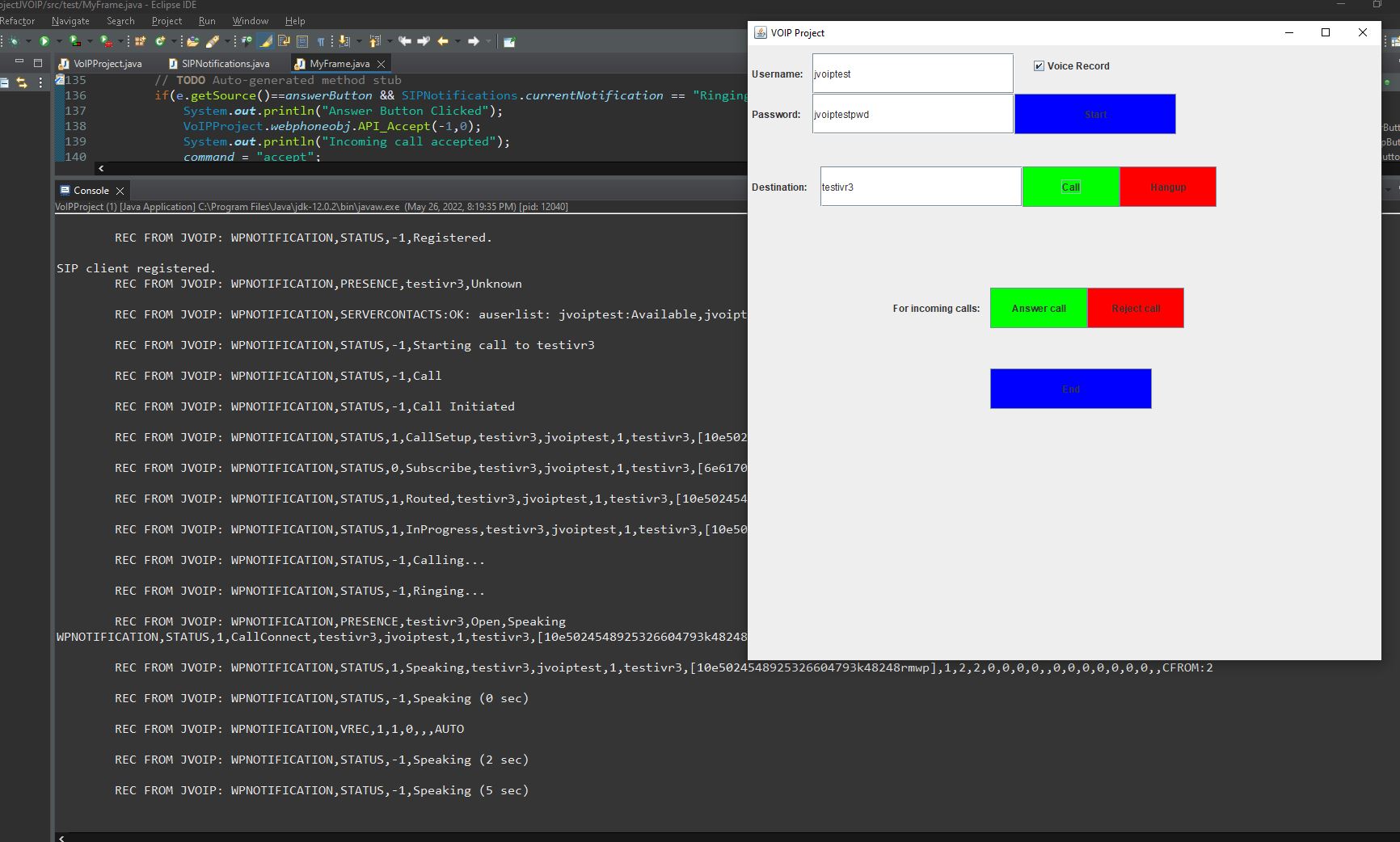
The program is fast and has really good performance as we used JVoIP which is very fast and well written library. The GUI we implemented is very clear, easy to use, and contains all the buttons and inputs that we need. The quality of the sound is not so good and that could be due to multiple reasons including the used input and output devices (headphones and microphones) or the VOIP server that we’re using or etc.

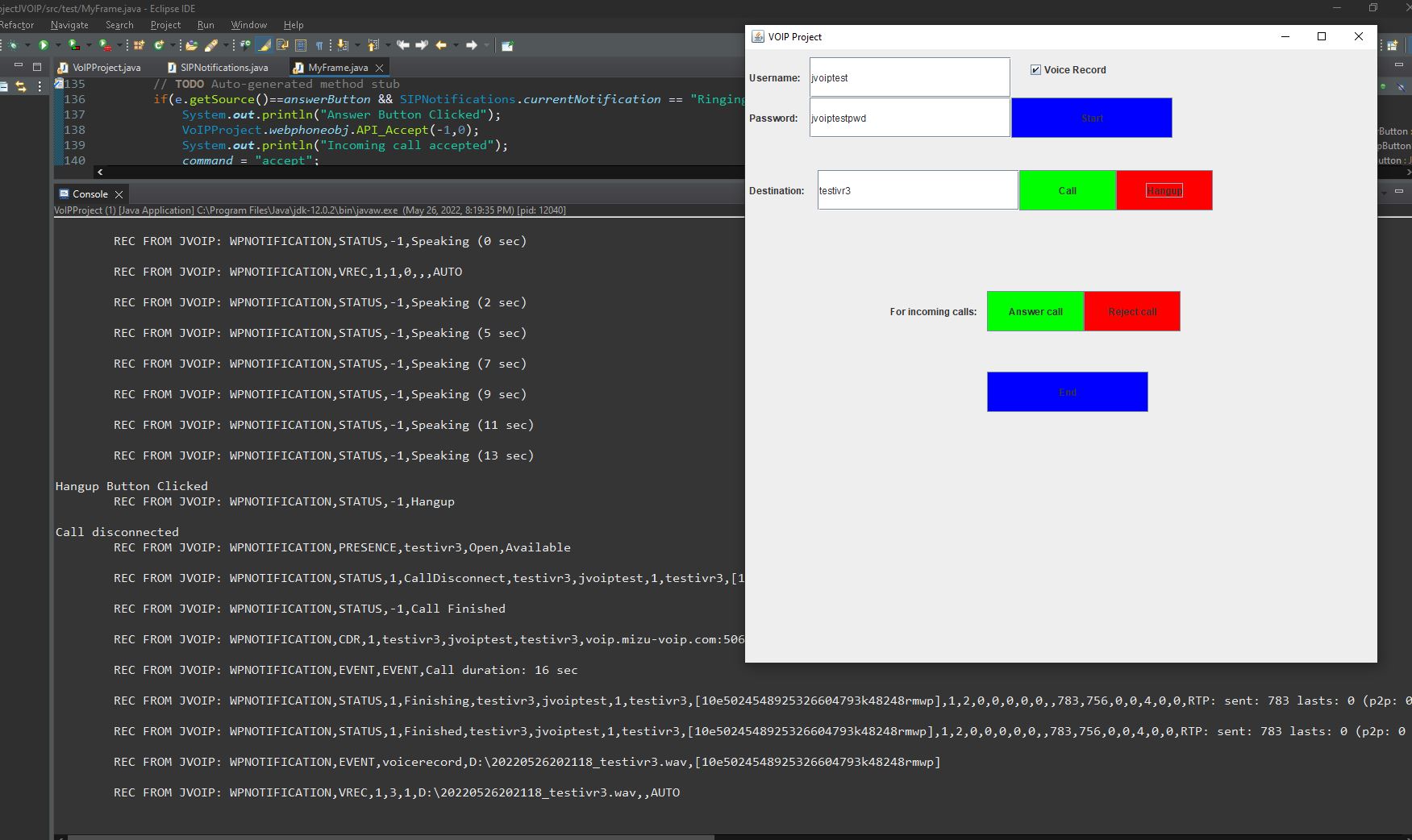
**Testing**

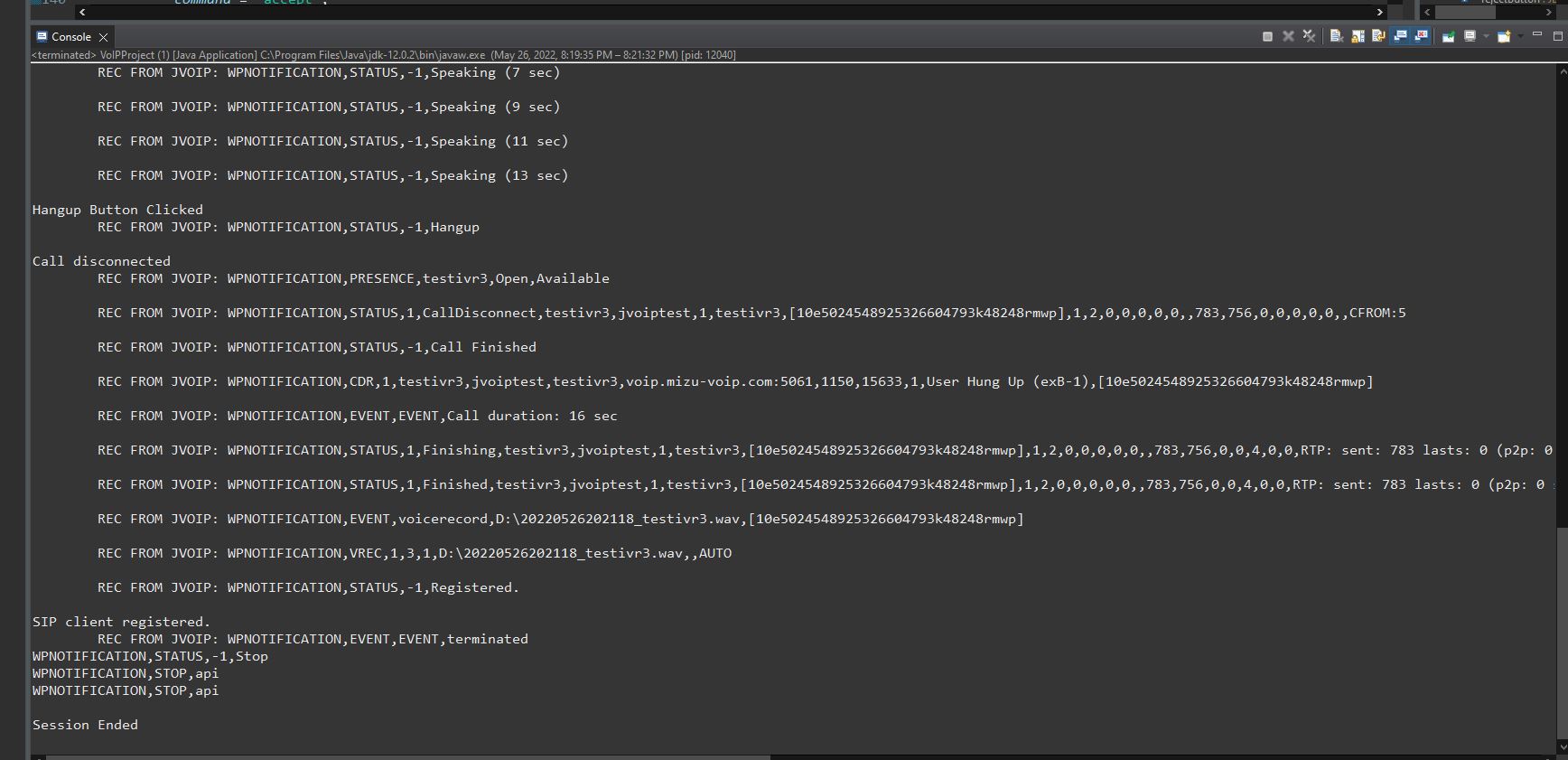
We have tested making outgoing calls by calling a user with username called “testivr3” which JVoIP provides for testing. Calling “testivr3” will make him play a recorded music that we can hear if the call was made successfully. We have run the project on one laptop and on the PC, we logged with different accounts on each device and made successful voice calls between the two devices. We also have added new users such as “username: joe123” and “password: joe123” then we used the JVoIP.jar softphone for testing. For example, registering on the softphone using the account with credentials: “username: joe123” and “password: joe123” and make a call to the account that we are using in our code which has the credentials: “username: jvoiptest” and “password: jvoiptestpwd” then we made calls from our code with the account “jvoiptest” to the softphone with the account “joe123” and vice versa and we were able to make the calls, initialize, and end the sessions successfully.

**Making a call from user “jvoiptest” to user “testivr3”**

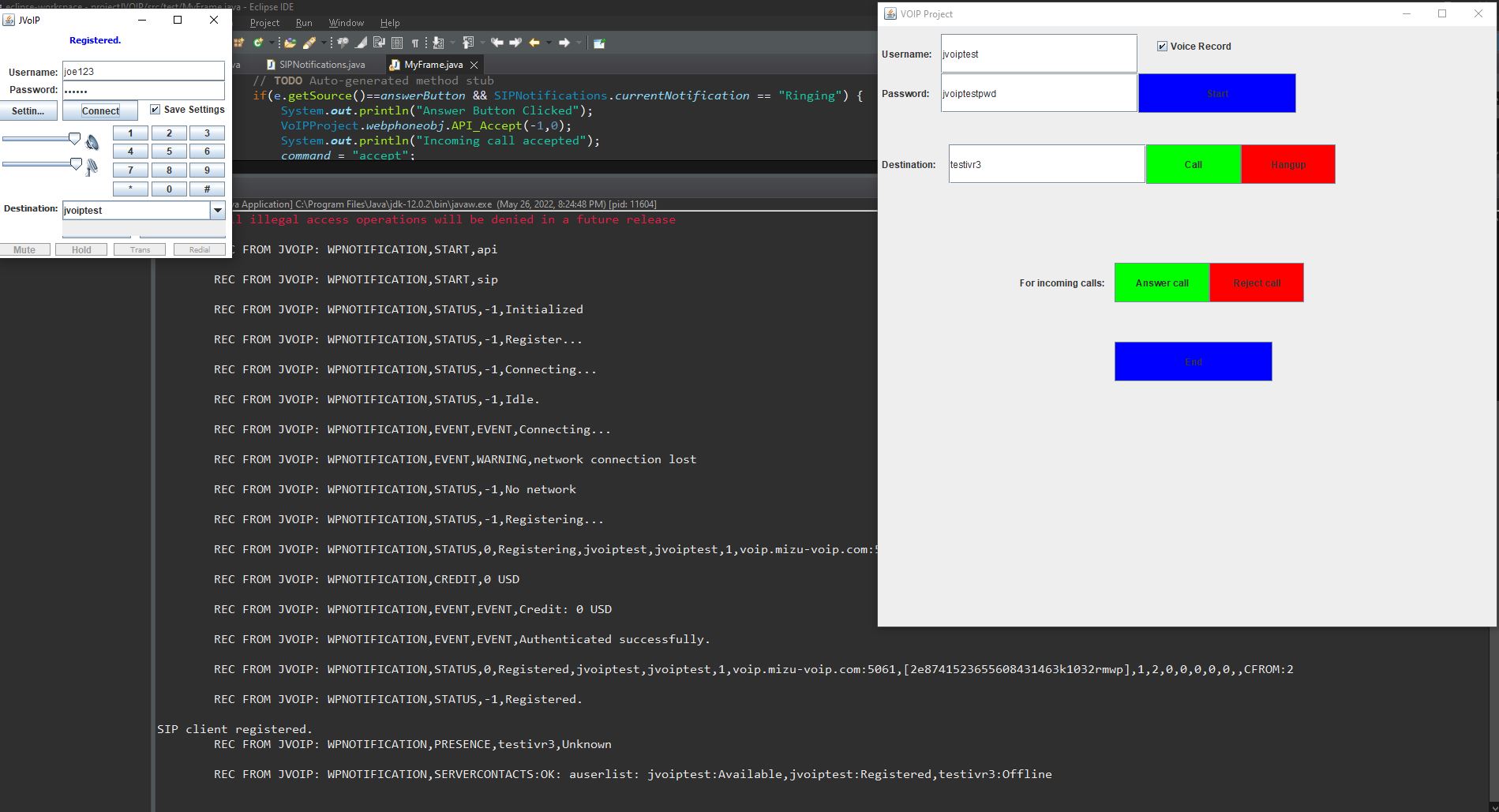
****

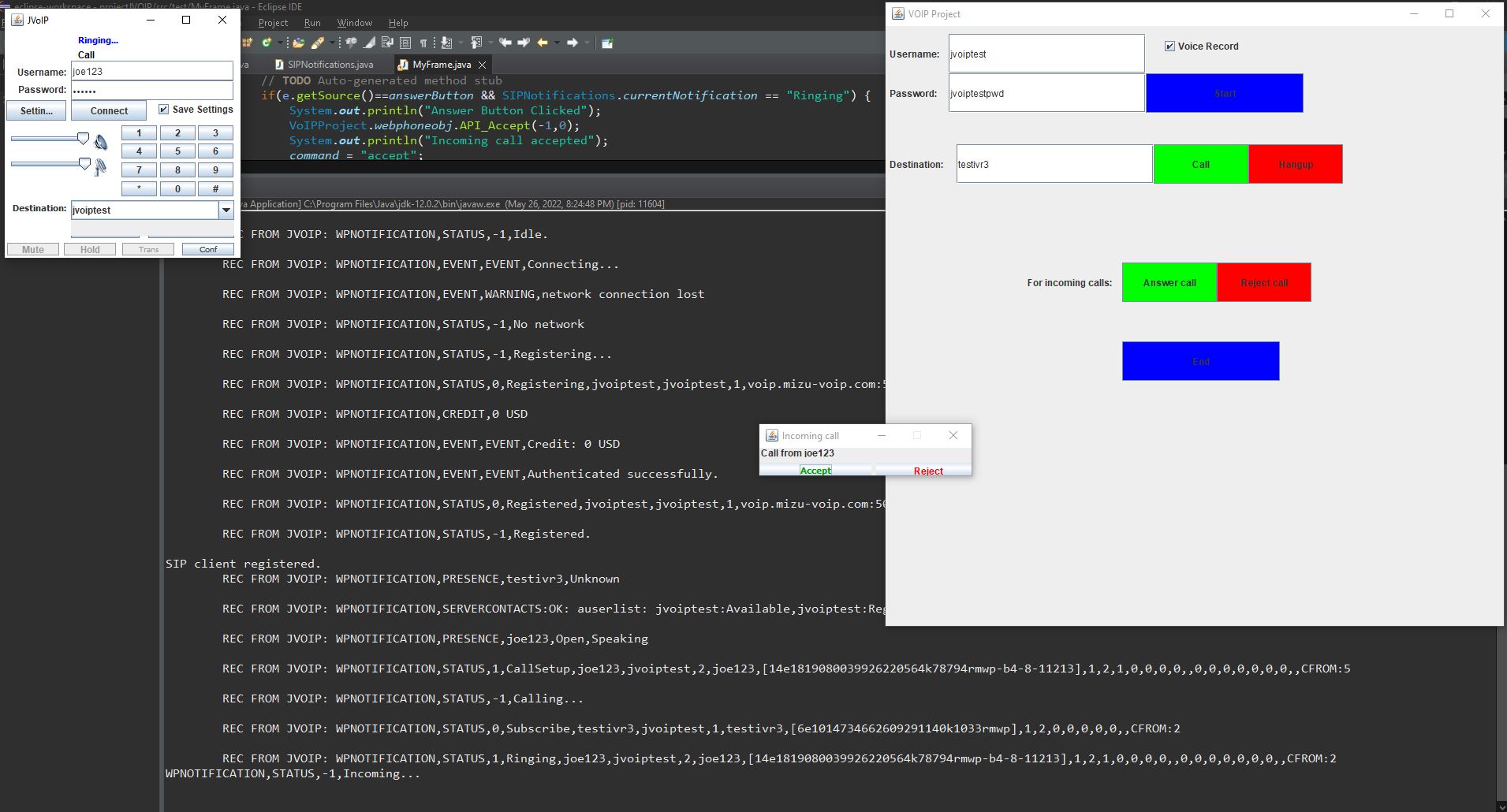


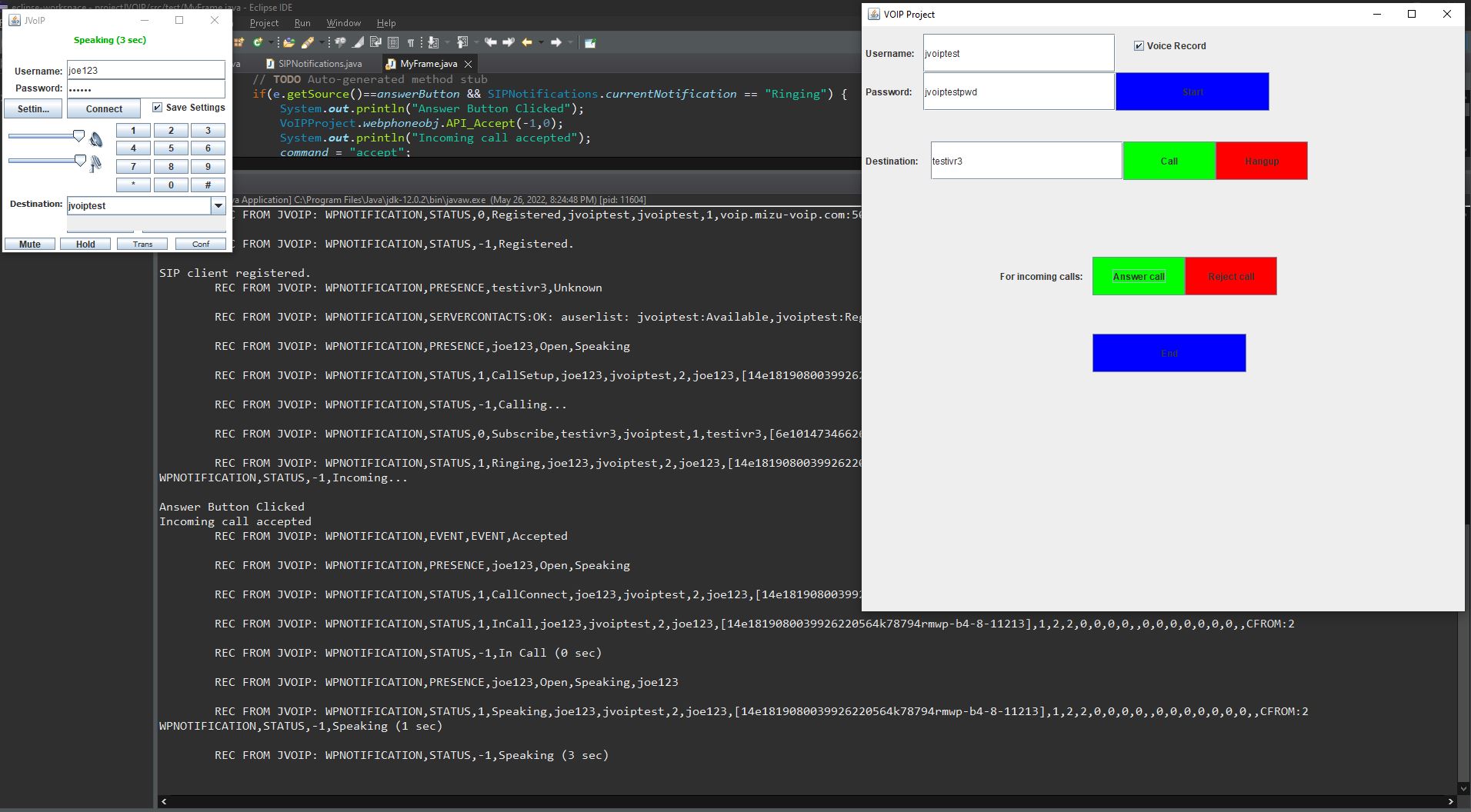


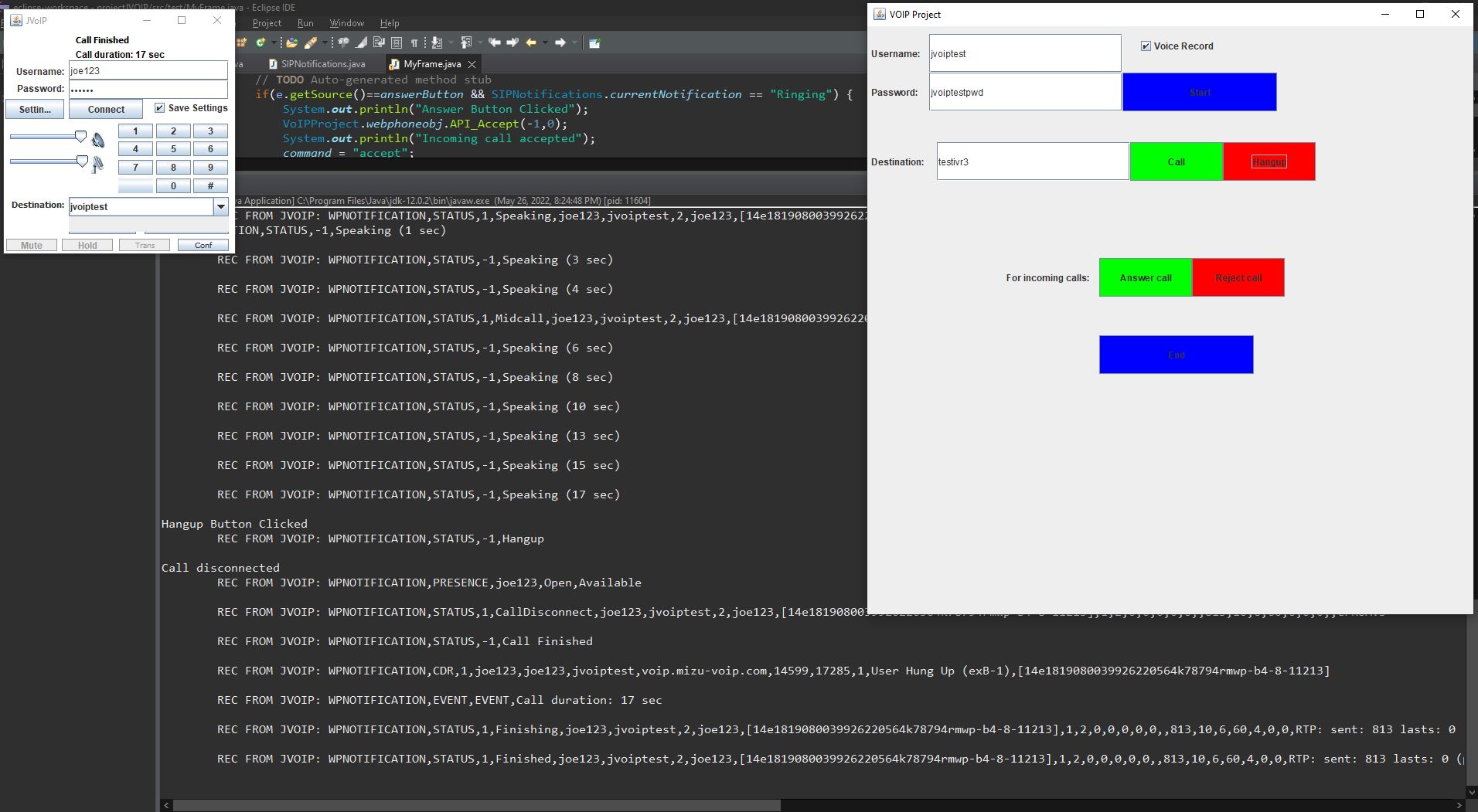


**Receiving and accepting a call from user “joe123”**

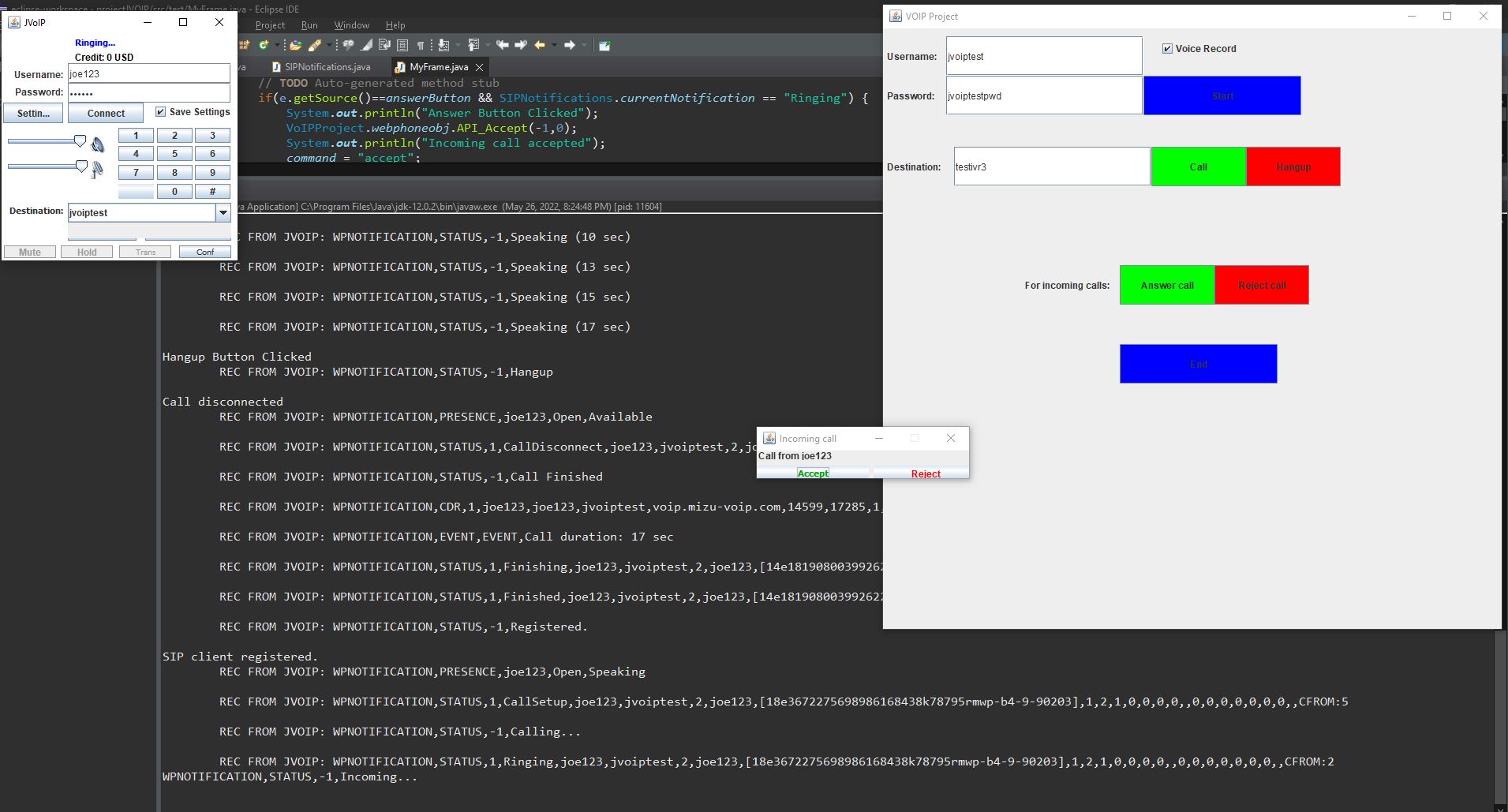
****

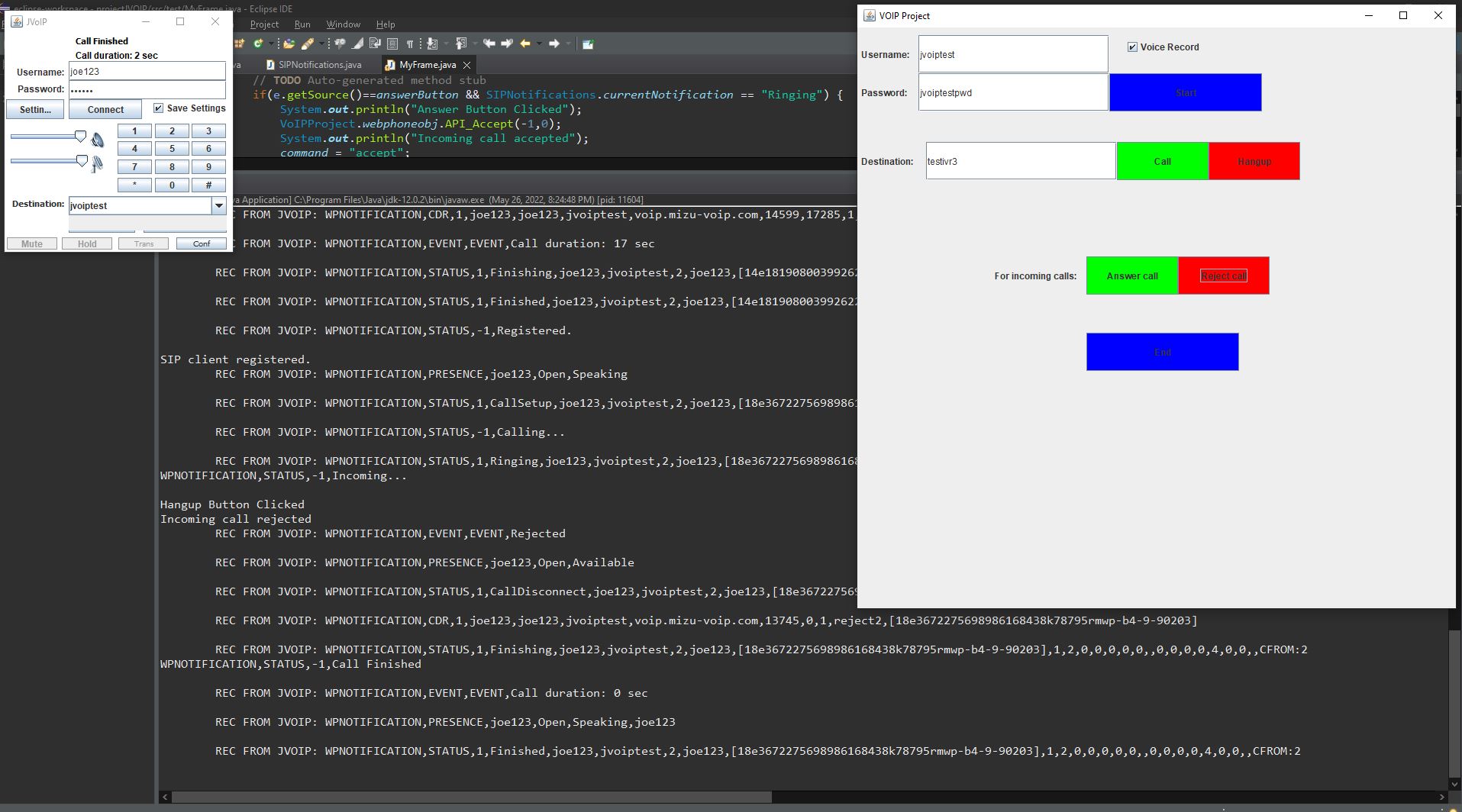
****

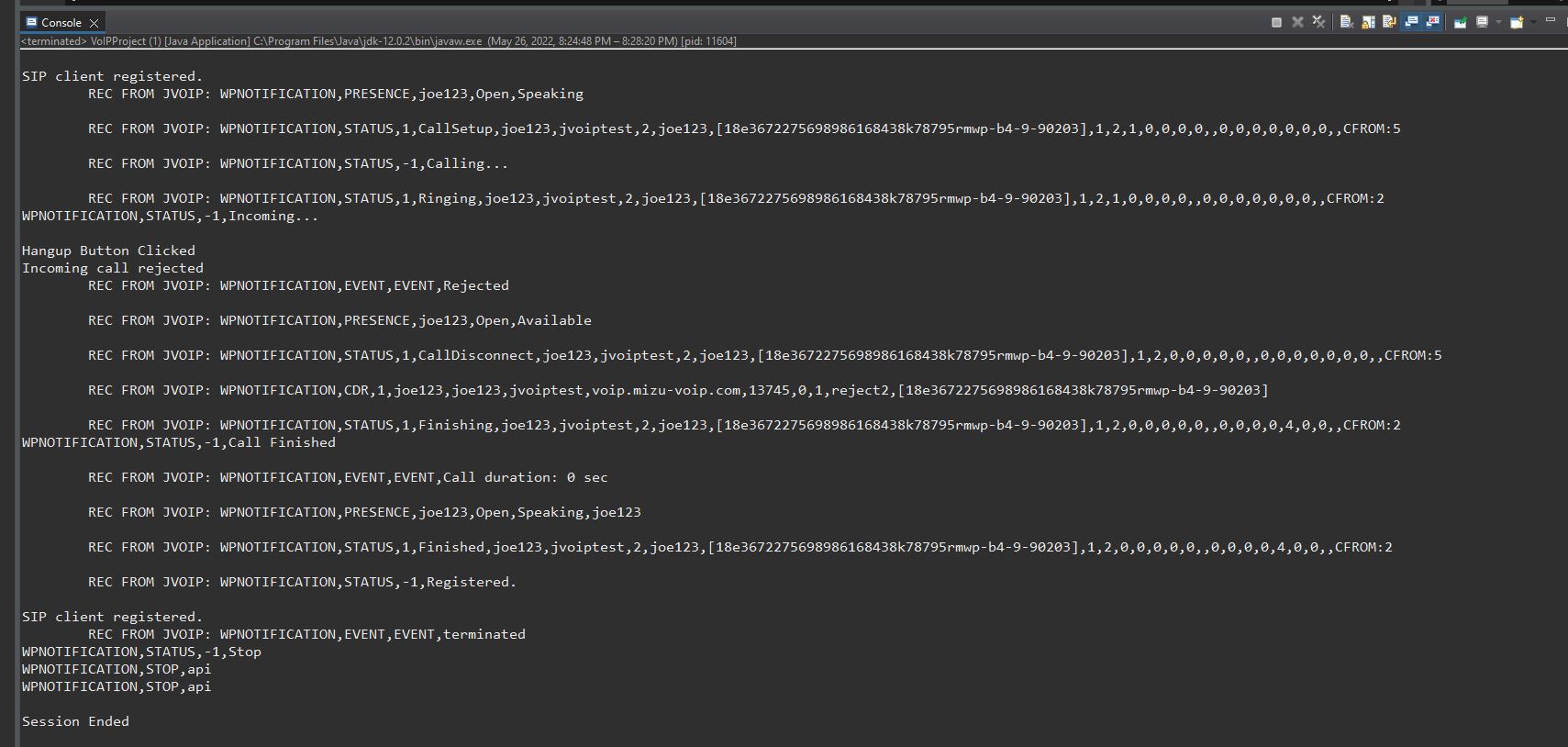
****

****

**Receiving and rejecting call from user “joe123”**

****

****

****

**References**

* <https://www.mizu-voip.com/Portals/0/Files/documentation/jvoip/index.html>
* <https://datatracker.ietf.org/doc/html/rfc3261>
* <https://en.wikipedia.org/wiki/Session_Initiation_Protocol>
* <https://docs.oracle.com/javase/7/docs/api/java/awt/package-summary.html>