JAVA Project

Wykonawca: Roman Uliashyn

Temat: „Call simulating”

Treść

1. Cel
2. Jak Działa
3. Środowisko programowe oraz ustawienia
4. Interfacy
5. Implementacja
6. Logika
7. Try

Cel:

Przetwarzanie rozmowy w trybie online. Nagrać zapisaną rozmowę

Jak dziala

*“Server słucha , Client mówi”*

Przy naciśnięciu **Call** , strona „serwer” usłyszy głos

Warunek konieczny to prawidłowy adres podłączenia i porty

1. Adres podłączenia – aktualny IPv4-adress
2. Port użytkownika domyślny - 8189
3. Port serwerowy ustawiony – 9786

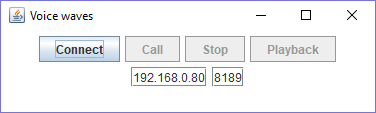
Śriodowisko programowe oraz ustawienia

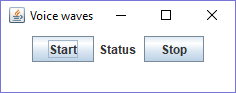
- Wykorzystane było środiowisko InteliJI IDEA 2019.1.2

- Project SDK 1.8 , java\_version 1.8.0\_152

- Utworzone pakiety Serwer a Client

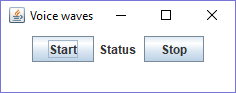
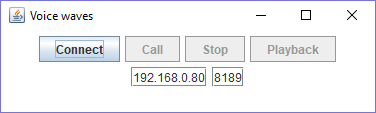
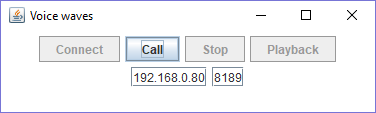
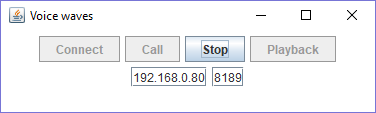
- Pliki koćnowe w formacie .jar (serwer.jar,client.jar)

Interfacy



Interface clienta Interface serwera

Implementacja

1. W pliku exportowanym Voicer z acriuma , przechodzimy szcieżką *Voicer\out\artifacts*
2. Otwieramy plik **loc-last**
3. Uruchamiamy najpierw serwer.jar
4. Klikamy Start , zmieni się w tym momencie status servera.
5. Teraz plik client.jar w tym samym katalogu **loc-last** – wpisujemy dane według informacji w punkcie **Jak działa**
6. Klikamy najpierw Connect, żeby zmienne port i IP były ustawione prawidłowo przed dzwonkiem
7. Potem klikamy Call - nasz program się kontaktuje z serwerem
8. Jeśli odpalił się przycisk Stop , a zamiast adresu IP nie pojawił się tekst „Adress Is Unreachable” – znaczy że podłączenie jest sukcesyjne , więc w tym momecie każdy dzwienk z naszego microfonu przetwarza się z głośnikach.
9. Klikając stop kończymy rozmowę i możemy zrobić to ponownie lub nagrać rozmowę która była chwile temu.
10. Okienko Serwer -> Stop

UWAGI:

1. Serwer w tym momencie działa , więc przed tym jak ponownie podłączyć się sprawdzamy **status servera**.

2. Jeśli **wyłączyć** serwer w momencie rozmowy, rozmowa dali istniejee i nagrywa się **dopóki nie klikniesz stop**

3. Jeśli **wyłączyłeś** serwer a **nie wyłączyłeś** rozmowe , możesz ponownie uruchomić server i rozmowa będzię nadal istnieć **dopóki nie klikniesz stop**.

4. Jeśli wyłączyłeś serwer , a probujesz połączyś się – program działa w trybie dyktofonu. Np. jak Radio w policji.

5. Jeśli wpiszesz nie prawidłowy adres otrzymasz po kliknięciu call tekst w TextBoxie adressu – „Adress is unreachable”

Logika

Uruchamiamy serwer klikając button „start”. Nasz status odpowednio zmienia się na status ON(włączony). Skoro uruchamiamy serwer , który tworzy socket , serwer stara się otzymać pakiet z dannymi ,czekając dopóki nie przyjdzie pakiet. W taki sposób:

serverSocket = new DatagramSocket(9786);  
serverSocket.setSoTimeout(200);  
VoiceReceiver\_Serwer.this.isRunning = true;  
byte[] receiveData = new byte[10000];  
System.*out*.println("Start"); //zrobione dla debugu  
while (VoiceReceiver\_Serwer.this.isRunning) {  
 DatagramPacket receivePacket = new DatagramPacket(receiveData, receiveData.length);  
 System.*out*.println("Inside"); //zrobione dla debugu  
 try {  
 serverSocket.receive(receivePacket);  
 } catch (Exception e) {  
 System.*out*.println("Cont");//zrobione dla debugu  
 continue;  
 }

Mamy zmienną globalną isRunning która reprezentuje status serwera.

Pętla działą dopóki nie przyjdzie pakiet , kontroluje to try/catch.

Kiedy w końcu przyleciał pakiet , consol pokazuje logi:

System.*out*.println("RECEIVED: " + receivePacket.getAddress().getHostAddress() + " " + receivePacket.getPort());

a jednocześnie przyjmujemy dane za pomocą:

sourceLine.open(adFormat);

Wiersz danych źródłowych(sourceLine)to ciąg danych, które mogą być zapisane dane.

SourceDataLine sourceLine;

Po openingu startujemy audio strumień

VoiceReceiver\_Serwer.this.sourceLine.start();

**Wyjaśnienie ważnych zmiennych**

adFormat to zmienna typu AudioFormat. AudioFormat prezentuje dokładne mejsca dannych w audio wątku. A właśnie za pomocą tego typu możemy interpretować bity w binary data.

adFormat = funkcji getAudioFormat(). Ta funkcja ustawia stan AudioFormatu:

private AudioFormat getAudioFormat() {  
 float sampleRate = 16000.0F; //jest liczba samplow ,ktore reprezentuje AUDIO  
 int sampleInbits = 16; //samples w bitach  
 int channels = 1; //1 to format mono , 2 format stereo  
 boolean signed = true; //status rejestracyjny  
 boolean bigEndian = false; //różnica między big a little endian , czy encoding dannych jest nie więcej niż jeden bajt  
 return new AudioFormat(sampleRate, sampleInbits, channels, signed, bigEndian);  
}

Zmienna targetDataLine jest to DataLine, z którego można odczytać dane audio

TargetDataLine targetDataLine;

Strumień wejściowy(InputStream) z określonym dźwiękowym formatem i długoścą

AudioInputStream InputStream;

Mapa Serwera

Wyłołanie main

public static void main(String[] args) {  
 (new serwer.VoiceReceiver\_Serwer()).Run\_Voice\_Waves();  
}

->

W metodzie Run\_Voice\_Waves opracowanie przycisku Start i Stop

->

Przycisk\_Start zajmuje się processem otrzymania danych a wątkami

Przycisk\_Stop wpływa na status servera

Mapa Clienta

Wyłołanie main , tworzy nowy VUClient

public static void main(String[] args) {  
 new VUClient();  
}

->

Konstruktor ustawia wszysktie ustawienia dla przycisków a tworzy okno.

W każdym przycisku organizujemy stan innych przycisków, żeby było intuicyjnie dla użytkownika.

Przycisk\_Connect

connect.addActionListener(new ActionListener() {  
 public void actionPerformed(ActionEvent e) {  
 VUClient.this.IP = VUClient.this.connect\_text.getText();//wpisany IP   
 port = Integer.*parseInt*(port\_text.getText()); //wpisany port  
  
 connect.setEnabled(false);  
 call.setEnabled(true);  
 stop.setEnabled(false);  
 play.setEnabled(false);  
 }  
});

- Przycisk\_Call

call.addActionListener(new ActionListener() {  
 public void actionPerformed(ActionEvent e) {  
 call.setEnabled(false);  
 stop.setEnabled(true);  
 play.setEnabled(false);  
 connect.setEnabled(false);  
 VUClient.this.captureAudio();// funckcja która łapie audio 107 wiersz  
 }  
});

- Przycisk\_Stop

stop.addActionListener(new ActionListener() {  
 public void actionPerformed(ActionEvent e) {  
 call.setEnabled(true);  
 stop.setEnabled(false);  
 play.setEnabled(true);  
 connect.setEnabled(true);  
 VUClient.this.stopaudioCapture = true; //wpływa to na 199 wiersz  
 VUClient.this.targetDataLine.close(); //zamyka strumien  
 }  
});

- Przycisk\_Play

play.addActionListener(new ActionListener() {  
 public void actionPerformed(ActionEvent e) {  
 VUClient.this.playAudio();//Nagrywa rozmowę 129 wiersz  
 }  
});

- Dodajemy za pomocą:

getContentPane().add(przycisk\_nazwa);

- Ustawiamy LayOut

this.getContentPane().setLayout(new FlowLayout());

-Dodaliśmy

this.setVisible(true);  
this.g = this.getGraphics();

W razie jakich kolwiek pytań proszę pisać na mail <romanuliashyn@gmail.com>

**Try**

Można sprobować połączyć się z innego komputera lokalnie.

Ścieżką *Voicer\out\*artifacts -> otwieramy **loc-try** ->

* uruchamiamy **inlocal.jar** .

Algorytm jest ten sam , zamiast call kliknąć capture. Po kliknięciu stop , zamknąć.