SOFE4790 Distributed Systems (Fall 2020 - Dr. Q. Mahmoud)

Assignment #2

Honour code: By submitting this assignment report, I (name and banner ID# below) confirm this is my own work, and I have not asked any of my fellow students or others for their source code or solutions to complete this assignment, and I have not offered my source code or solutions for this assignment to any of my fellow students.

Name: Umar Qureshi

Banner ID#: 100591742

1 Application idea

My Java RMI Application is a .wav File editor where the client chooses a .wav audio file of what needs to be edited in a GUI and the .wav is sent over to the client to be edited. The edited file is received by the client. This idea was inspired by Dr. Q. who stated in a lecture about a student who edited an image, I thought of doing video but that seemed like too large of a task, so I went one level down to audio. It was also inspired by VideoEditBot on twitter which is a versatile bot where all the client needs to do is tag the bot and issue a command and the bot on Server side will edit the videos speed, pitch, resolution, replace the audio with a song from YouTube and many other features. It is an excellent example of a Client-Server application and it showcases how my simple program can be of use, can be scaled up, more features and polish can be added.

2 Remote Method Calls

i) **Sending .wav file to server to be trimmed**

ii) **Retrieving/downloading trimmed file from server**

A user on the GUI types in the portion of the audio they would like to keep, they type in the start and end time.

iii) **Sending .wav file to server for spook effect to be added**

iv) **Retrieving/downloading spook edited file from server**

Clicking on the “spooky” radio button alters the sound of the .wav file, this was done by changing the Sample Rate of the audio file.

v) **Sending .wav file to server to be trimmed & for spook effect**

vi) **Retrieving/downloading trimmed & spook edited file from server**

3 Describe Novel Features

i **Simple GUI which doesn’t close and can send multiple requests**
The GUI only closes when the user exits so it can send one request after another. The GUI is simple to use

ii **Audio Player within the GUI.** The user does not need to leave the program to listen to the edited audio, they simple select the edited file or any .WAV file to listen to.

iii **Easy to use GUI file selector.** This file selector only allows .wav files to be selected allowing for no issues

4 Challenges and solutions

One of the main challenges I faced was converting my code to work in Linux as my Java RMI worked only on Linux. The first issue I faced when trying to run my first assignment on Linux was that open JDK 8 was not compatible with the stable version of JavaFX (GUI). I tried looking for an older version of JavaFX and tried downloading another JDKwhich both did not work. I tried downloading a JDK directly from Intellij and that worked with Java FX 11.

The next main challenge was taking the java RMI code that worked in Lab 2 and incorporating it with my assignment 1. The challenge was the original code worked with the command line and arguments being given, I had to make it work in the background and alongside my GUI. I faced multiple errors here and solved them step by step.

When I thought all errors were resolved I got a bothersome error where it would not allow me to run my code. This error occurred because I had the same class file in different folders, the FileInterface.java class. In my lab, for some reason, this was not an issue and ran but I could not get around this issue and be stuck on it trying various things and looking on the internet for a long time.

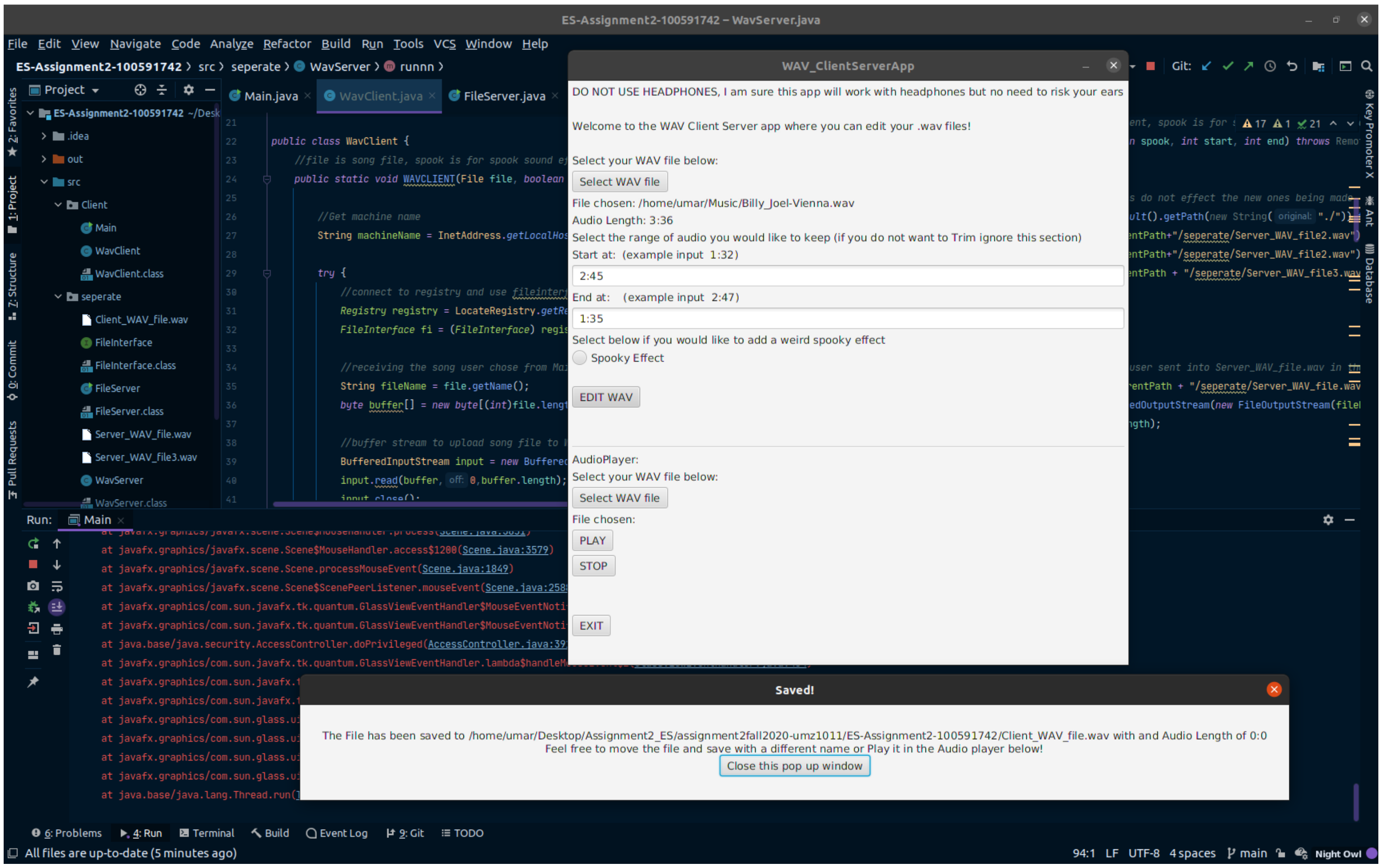
This was resolved after I messaged my friend to help me out, he stated it was best I had one File Interface class in the server and the client imports it. I had trouble doing that step as I was not using packages and even after implementing packages it would not compile with an error on the client side stating missing File Interface class. I showed my friend the issue and he stated that I needed to compile all classes at once in one javac command and that finally resolved my issue to make my code run.

I had minor issues with uploading the file which was resolved step by step. An issue that took longer than I thought was the client downloading/retrieving the edited file from the server. It seemed like the code could not find the file to download even though it was created on the server. After trying many different paths I decided to use code from my Assignment 1 that gains the parent path of where the file was located. This partially resolved my issue as the file was definitely retrieved from the server but it did not save inside the client folder but instead outside the src folder (still inside the project folder). I tried a multitude of path options with and without the parent path variable and no matter what it did not save in the Client folder, which is strange as in lab 2 it saved inside the Client folder. I am not sure if this is due to my GUI calling the Client in the background. Due to time, I decided to leave it where it saved and changed some code in my GUI to read the edited client file from outside the src folder.

What my next steps would have been to learn how to debug in java. My client runs in the background and I cannot see where it is printing. If i could see what value on Client side ParentPath returns I would be able to manipulate it to save the edited music file in the Client folder.

5 Testing

What happens if a user uses a bigger start then end:



```
public class WavClient {
    //file is song file, spook is for spook sound e
    public static void WAVCLIENT(File file, boolean

        //Get machine name
        String machineName = InetAddress.getLocalHos

        try {
            //connect to registry and use fileinter
            Registry registry = LocateRegistry.getRe
            FileInterface fi = (FileInterface) regis

            //receiving the song user chose from Ma

            String fileName = file.getName();
            byte buffer[] = new byte[(int)file.lengt

            //buffer stream to upload song file to 
            BufferedInputStream input = new Buffered
            input.read(buffer, off: 0,buffer.length);
            input.close();
```

```
ent, spook is for 
n spook, int start, int end) throws Remo

s do not effect the new ones being made
ult().getPath(new String( original: "./")
entPath+"/seperate/Server_WAV_file2.wav")
entPath+"/seperate/Server_WAV_file2.wav")
entPath + "/seperate/Server_WAV_file3.wav

user sent into Server_WAV_file.wav in th
rentPath + "/seperate/Server_WAV_file.wav
edOutputStream(new FileOutputStream(file
ngth);
```

```
at javafx.graphics/javafx.scene.Scene$MouseHandler.process(Scene.java:3851)
at javafx.graphics/javafx.scene.Scene$MouseHandler.access$1200(Scene.java:3579)
at javafx.graphics/javafx.scene.Scene.processMouseEvent(Scene.java:1849)
at javafx.graphics/javafx.scene.Scene$ScenePeerListener.mouseEvent(Scene.java:258
at javafx.graphics/com.sun.javafx.tk.quantum.GlassViewEventHandler$MouseEventNoti
at javafx.graphics/com.sun.javafx.tk.quantum.GlassViewEventHandler$MouseEventNoti
at java.base/java.security.AccessController.doPrivileged(AccessController.java:39
at javafx.graphics/com.sun.javafx.tk.quantum.GlassViewEventHandler.lambda$handleM
at javafx.graphics/com.sun.javafx.t
at javafx.graphics/com.sun.javafx.t
at javafx.graphics/com.sun.glass.ui
at javafx.graphics/com.sun.glass.ui
at javafx.graphics/com.sun.glass.ui
at javafx.graphics/com.sun.glass.ui
at java.base/java.lang.Thread.run(
```

WAV_ClientServerApp

DO NOT USE HEADPHONES, I am sure this app will work with headphones but no need to risk your ears

Welcome to the WAV Client Server app where you can edit your .wav files!

Select your WAV file below:
Select WAV file
File chosen: /home/umar/Music/Billy_Joel-Vienna.wav
Audio Length: 3:36
Select the range of audio you would like to keep (if you do not want to Trim ignore this section)
Start at: (example input 1:32)
2:45
End at: (example input 2:47)
1:35
Select below if you would like to add a weird spooky effect
Spooky Effect
EDIT WAV

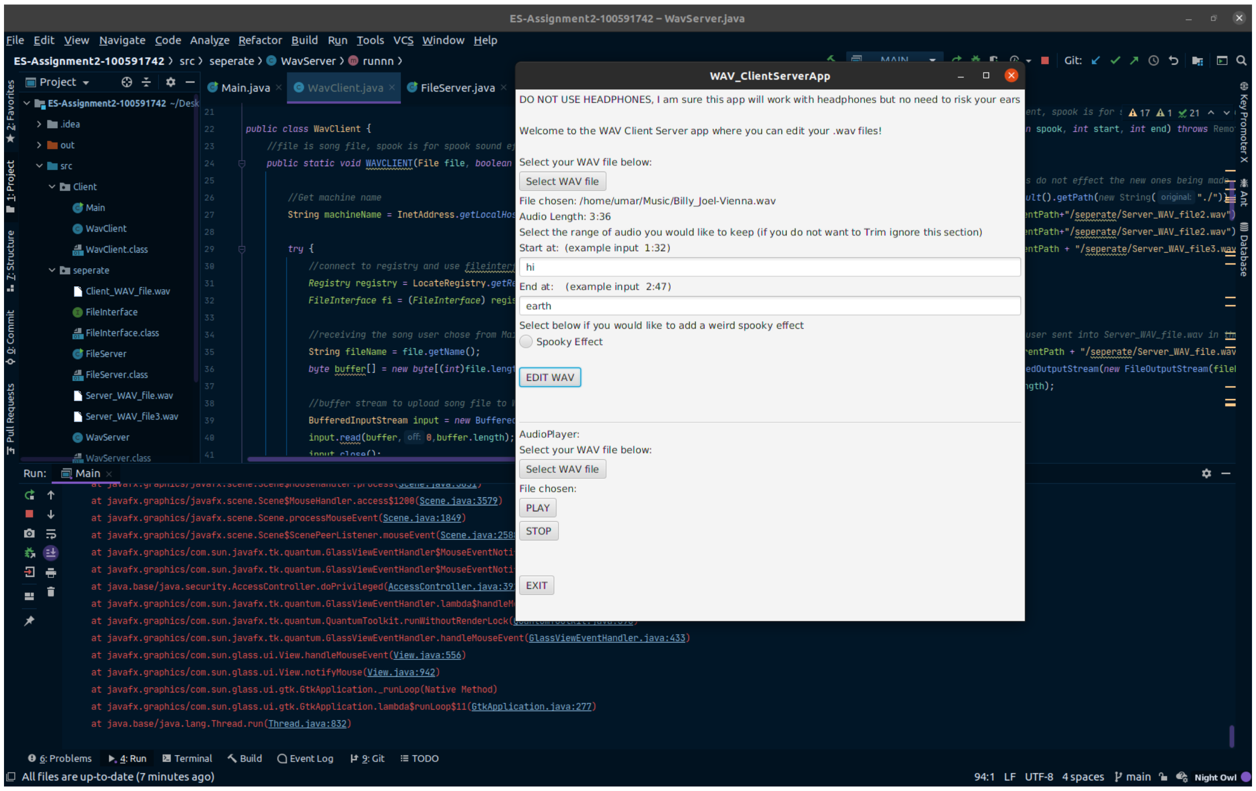
AudioPlayer:
Select your WAV file below:
Select WAV file
File chosen:
PLAY
STOP
EXIT

Saved!

The File has been saved to /home/umar/Desktop/Assignment2_ES/assignment2fall2020-umz1011/ES-Assignment2-100591742/Client_WAV_file.wav with and Audio Length of 0:0
Feel free to move the file and save with a different name or Play it in the Audio player below!
Close this pop up window

It saves but the music file has no length as seen by the red circles. Client crashes but it is not a big deal as the GUI or server has not closed and you can make another request, the user will not see this crash

What if the user enters a string:



```
ES-Assignment2-100591742 – WavServer.java

WAV_ClientServerApp
DO NOT USE HEADPHONES, I am sure this app will work with headphones but no need to risk your ears

Welcome to the WAV Client Server app where you can edit your .wav files!

Select your WAV file below:
Select WAV file
File chosen: /home/umar/Music/Billy_Joel-Vienna.wav
Audio Length: 3:36
Select the range of audio you would like to keep (if you do not want to Trim ignore this section)
Start at:  (example input  1:32)
hi
End at:   (example input  2:47)
earth
Select below if you would like to add a weird spooky effect
Spooky Effect

EDIT WAV

AudioPlayer:
Select your WAV file below:
Select WAV file
File chosen:
PLAY
STOP

EXIT
```

```
at javafx.graphics/javafx.scene.Scene$MouseHandler.access$1200(Scene.java:3579)
at javafx.graphics/javafx.scene.Scene.processMouseEvent(Scene.java:1849)
at javafx.graphics/javafx.scene.Scene$ScenePeerListener.mouseEvent(Scene.java:258
at javafx.graphics/com.sun.javafx.tk.quantum.GlassViewEventHandler$MouseEventNoti
at javafx.graphics/com.sun.javafx.tk.quantum.GlassViewEventHandler$MouseEventNoti
at java.base/java.security.AccessController.doPrivileged(AccessController.java:39
at javafx.graphics/com.sun.javafx.tk.quantum.GlassViewEventHandler.lambda$handleM
at javafx.graphics/com.sun.javafx.tk.quantum.QuantumToolkit.runWithoutRenderLock(
at javafx.graphics/com.sun.javafx.tk.quantum.GlassViewEventHandler.handleMouseEvent(GlassViewEventHandler.java:433)
at javafx.graphics/com.sun.glass.ui.View.handleMouseEvent(View.java:556)
at javafx.graphics/com.sun.glass.ui.View.notifyMouse(View.java:942)
at javafx.graphics/com.sun.glass.ui.gtk.GtkApplication._runLoop(Native Method)
at javafx.graphics/com.sun.glass.ui.gtk.GtkApplication.lambda$runLoop$11(GtkApplication.java:277)
at java.base/java.lang.Thread.run(Thread.java:832)
```

All files are up-to-date (7 minutes ago)

Crashes client and gives an error in run menu but does not crash the program/GUI and user can make another request.

These should cover most ways this program could crash but it is resistant and after any client crash user can still make new requests.

(If needed refer to Assignment 1 document for other tests, since they were mostly the same I only included the relevant ones)