**BILKENT UNIVERSITY**

**ENGINEERING FACULTY**

**DEPARTMENT OF COMPUTER ENGINEERING**

**CS 299**

# SUMMER TRAINING REPORT

## Deniz Yüksel

**21600880**

**Performed at**

## Cybersoft

**17.06.2019 – 12.07.2019**

# Table of Contents

# 1 Introduction

In the summer of 2019, I had an internship at Cybersoft for 20 workdays, between 17.06.2019 and 12.07.2019. Founded in 1995, Cybersoft is amongst the most known Turkish software companies. Cybersoft both have offices in Ankara and İstanbul, and I applied to the Ankara office in Cyberpark. Cyberpark mainly works for TOBB, Yapıkredi, Şekerbank and various other banks. However the main reason I applied to the company is because I would have an opportunity to work with Ferhat Savcı, who is the research and development leader in the Ankara office. Because of his prior work on sound processing and both of our interest in music and guitars, we agreed that I would work on a musical project by his supervision.

The project can be explained very briefly. It came out of an application which is already present and is useful: Shazam [1]. While Shazam finds the song that is playing by hearing, my application would not find the name of the song, but it would transcribe the notes to be playable on a guitar. The application processes the sound and translates the sound to the TAB notes. Unlike sheet music, TAB notes are used only by guitarists.

I built my project on a Xubuntu machine and on Netbeans IDE [2, 3]. The project was a desktop application that was coded with Java.

In the next sections of this report, I will first give more information about the company. I will also present more information about my supervisor and the work I did in detail. Then, I will mention which technologies I needed to self-learn to deliver my project. Finally, I will wrap up my report briefly.

# 2 Company Information

## 2.1 About the company

The company I spent my 20 workdays as an intern was Cybersoft and was located in Bilkent Cyberpark. As mentioned earlier, Cybersoft has offices in both Ankara and İstanbul. Most of the İstanbul offices are banks and the company works on the finance sector. The Ankara office develops projects for TOBB (The Union of Chambers and Commodity Exchanges of Turkey) the Ministry of Finance, and I have also seen some engineers working with the Istanbul office from afar [4, 5].

## 2.2 About your department

I was in the project development department but I did not work with the projects that were being done by the department. The projects were confidential and were being done for TOBB and the Ministry of Finance mostly. Although I did not work on the big projects, I was assigned a project that solely I worked by myself with the help of my supervisor.

## 2.3 About the hardware and software systems

There are various software programs that are being used by the company. The company uses frameworks that they had previously developed. The computers have Windows as their operating systems. Zamaneddin Safari, a software engineer and a team leader quoted that Ankara office uses Java for most of the project because of its robustness and usage in the finance sector. In the company, the majority of engineers use Windows 7 on desktop computers. I was also offered a desktop computer, however I preferred to bring my own laptop to work because my supervisor encouraged me to install Xubuntu. I did not use any of the software that the company used.

## 2.4 About your supervisor

My supervisor's name is Ferhat Savcı. He is an R&D Lead at Cybersoft. He graduated from Hacettepe University, computer science in 1991. Then he continued his masters study in Hacettepe University in computer science. He was a Phd. Student between 1993-1995 and he had research on speech recognition. Since 1996 he is a part of Cybersoft. He also has three licenses by Red Hat on being a delivery specialist.

# 3 Work Done

## 3.1 A Brief Explanation of GuitarTabGenerator

My first intention was to request company jobs and to do little tasks to help the company out but my supervisor told me that developing a project that will be one's own would be more satisfying. As mentioned earlier, I developed a project that recorded an instrument voice preferably, and transcribed the data to a TAB key, a note sheet that guitarists use. It also shows the notes in a sonogram form as they go lower and higher, on a black screen. When the user presses the start button, the program starts to record, and generate the image and the notes on the fly. This is the most important part as the system works as a real time application. This behavior was implemented with threads in Java. On the right hand side, there is a simple diagram that shows how the system works. Originally, I drew this on a blank paper to visualize how the system works, so that I can program the project more easily. The circles represent the classes that are mainly doing the job. There are other utility classes in the project that help these four classes to perform their work. The connections between the four bubbles are pipes that information flows. VoiceRecorder

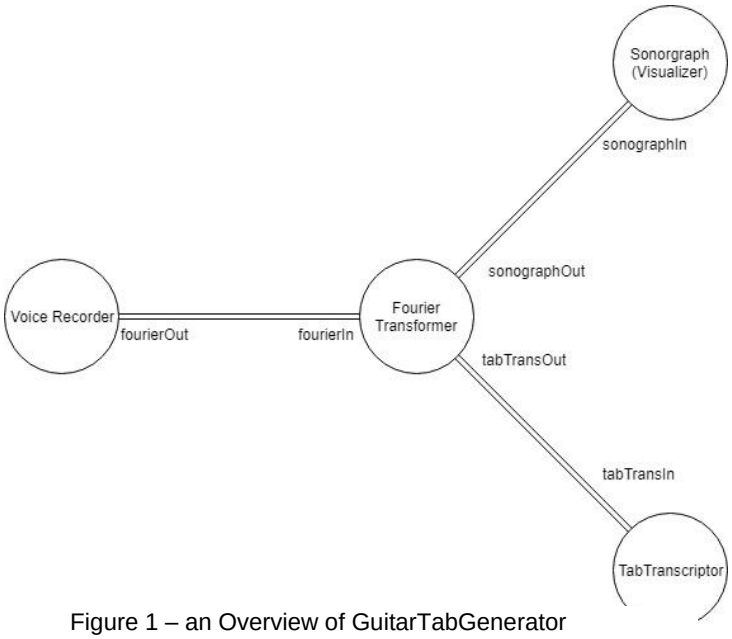


Figure 1 – an Overview of GuitarTabGenerator

thread turns on the microphone and records the voice, passes the information with a pipe, to FourierTransformer thread. Then, this class normalizes the wave and applies Fourier transform or Cepstrum analysis according to the preference of the user. The criteria for choosing Fourier transform or Cepstrum analysis will be explained more clearly in the next sections. The Fourier transform I used was implemented by Piotr Wendykier that was derived by a general purpose Fourier transform library in Kyoto University. The processes information is then passed to the Sonograph thread, a visualizer which basically a blank black screen that paints a white pixel on the screen according to the frequency of the voice. Simultaneously, the information is passed to TabTranscriptor thread that finds the note or the notes that are actually played. Then the notes on a TAB key is printed on the screen. Figure 2 represents the notes of C Major scale with a G clef, and a TAB notation [6]. The TAB notation shows the notes on 6 strings of a guitar which goes from low to high frequency as lines go from bottom to top, and the number of the fret is displayed accordingly.

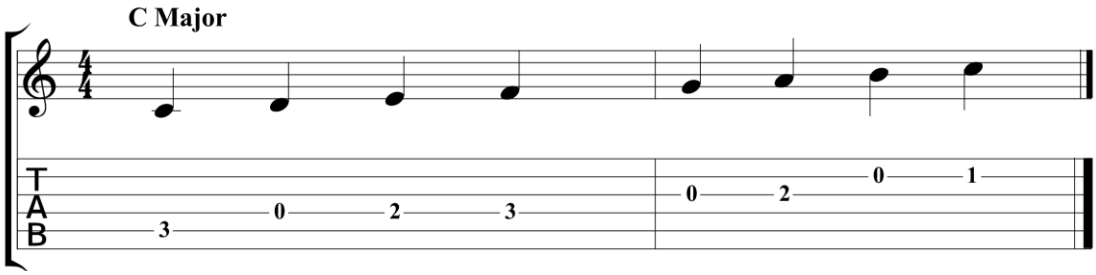


Figure 2 – C Major Scale depicted on G Clef and Tab Notation

## 3.2 Detailed Explanation of Major Classes in GuitarTabGenerator

There are four classes that are mainly responsible for the program to run properly. As mentioned earlier, those are VoiceRecorder, FourierTransformer, Sonograph, and TabTranscriptor. Each of the classes are threads and they operate safely in a way that each of the threads send and fetch the same amount of information in every pass.

### 3.2.1 VoiceRecorder

VoiceRecorder class is the starting thread. This class uses AudioFormat class to set the properties of the sound format that I will record. The sample rate, sample size in bits and number of channels are specified. More information can be found in Java Oracle Docs [7]. Then, the microphone is turned on and data is read to a byte array. The number of bytes that are going to be read depends on the sample rate which is fixed to 44100 Hz, and beats per minute that the user can select. In short, bpm refers to the tempo of the song or the notes that are being played. The reason I and my supervisor agreed on TAB key instead of regular note sheet was because tab key does not denote the length of the notes that are played. The sheet music is accurate and precise in terms of rhythm. Therefore, my supervisor advised me not to work on beats per minute detection but just use a bpm that is chosen by the user.

### 3.2.2 FourierTransformer

This class is another thread that is responsible for reading the data that was written to the write end of VoiceRecorder's pipe. The data is read to a byte array and then

converted to a float array by reading short values in each pass, because as sample rate is 44100 Hz for normal audio, the sample size in bits is 16 bits, and a short has 16 bits. In order to utilize JTransforms [8], the FFT (Fast Fourier Transform) library I had mentioned earlier, the wave needs to be normalized between 0 and 1 and the input of the constructor takes float values. The wave is processed with float Fourier Transformer and then absolute value of the array is taken. Afterwards, user can select if there needs to be one more transformation, namely Cepstrum [9,10,11]. If the user records pure sine soundwaves such as the sound of a tuning fork, whistle, Kalimba (finger piano), or a pure human voice, the transformation can stop. It is because Fourier Transform has the most accurate outcomes for pure sine waves as we found experimentally. However, if the user is recording an instrument voice that many waves are intertwined and sound is not very clean such as a guitar, we need to continue our transformation. For the next transformation, Cepstrum transformation is used. It is a derivation of Fourier Transform that is used for processing human and instrument voice. User can select beforehand, which transform method will be used manually.

After the values are processed, the array of values is ready to be transferred to the next stages. The data is written to two pipes simultaneously to be sent to Sonograph and TabTranscriptor threads.

### 3.2.3 Sonograph

The sonograph thread reads the bytes to an array from the sonographIn pipe end, and paints pixels black on a white screen. This classes' main purpose is to create the main data for the project. The height of a pixel painted shows which note it is, in essence. Therefore, after I coded the Sonograph class, I recorded all the notes from E2 (82 Hz) to C6 (1046 Hz) which range from the lowest pitched string to the highest pitched strings' 20$^{th}$ fret. This range covers the most of the guitar if we assume a guitar has 24 frets for each string. The reason I could not go further in measuring was because the doubling frequency of octaves. After I measured all the notes, I created a properties file to store my notes as key-value pairs.

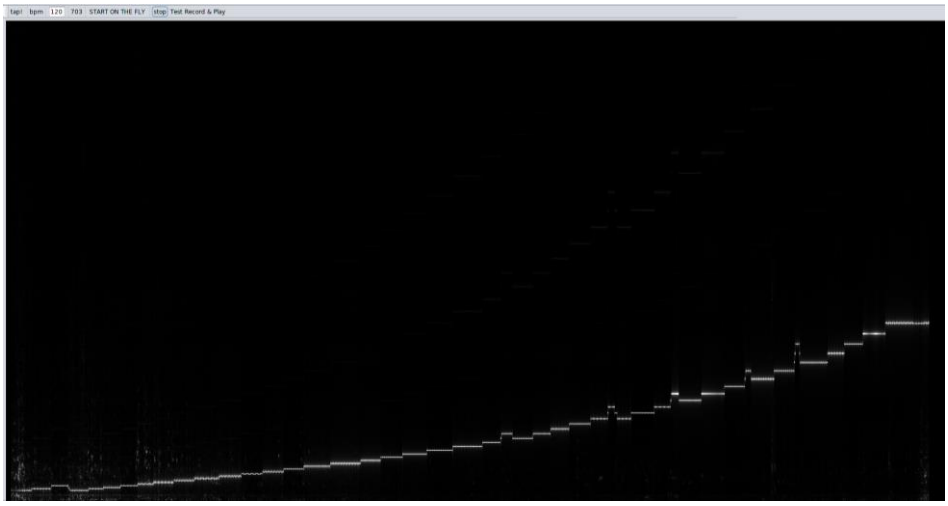

Figure 3 – A Screenshot of a Testrun of Sonograph Class in the Project

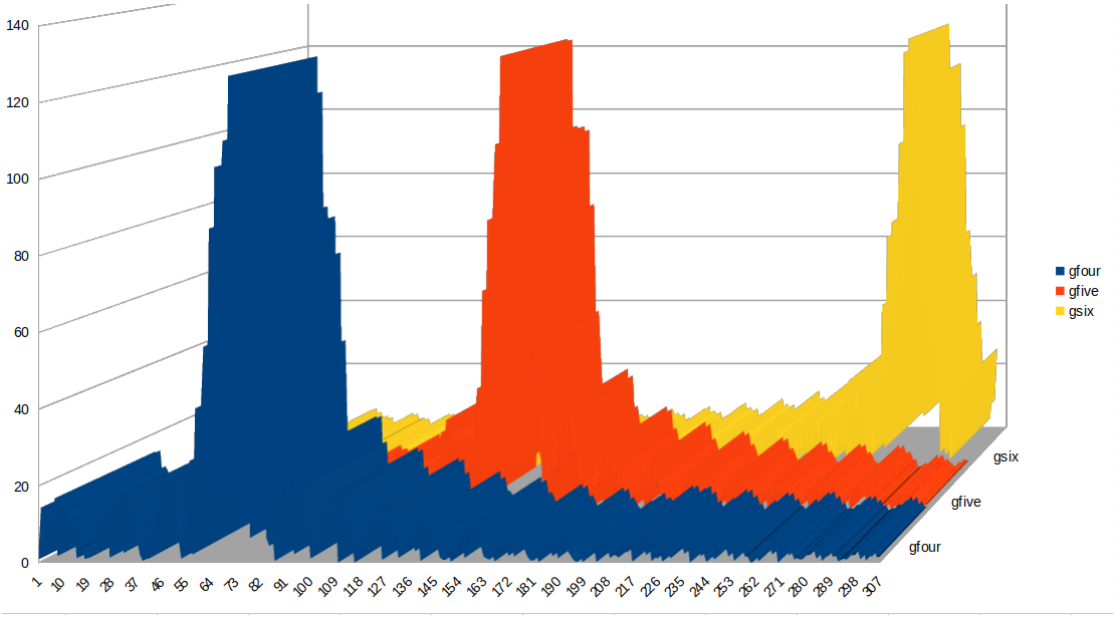


Figure 4 – Frequency Mapping of G4, G5 and G6

Figure 3 is a screenshot of a test run to measure musical notes from lower to higher frequencies. Figure 4 denotes the proof of doubling frequencies when octave increases. In this case G4, G5, G6 are represented with blue, red and yellow respectively. The distance between the peak points of red and yellow wave is double the length of the distance between blue and red.

### 3.2.4 TabTranscriptor

TabTranscriptor thread has two helper classes, a guitar model and a tab writer. The guitar model class is used to find the strings and the positions of a note on a guitar, and tab writer is another thread that prints out the TAB notes on the screen. TabTranscriptor's main purpose is to read the data from FourierTransformer, and find the correct note that is played at that moment, by comparing the position of the highest value in the array with the property sheet. The property sheet as mentioned earlier is a file that has key value pairs as note and position on the screen (height). This class also silences the low frequencies that can be viewed on the screen, but will not be reflected to the detected notes.

# 4 Performance and Outcomes

## 4.1 Applying Knowledge and Skills Learned at Bilkent

There were important skills that I had learned in Bilkent in order to overcome the obstacles while developing the project. Firstly, Bilkent teaches the students to stop and think for a moment, meaning that one does not simply sit in front of the computer and code the whole project. Instead, I drew diagrams and asked for opinions to others and had a careful planning of the project. I have observed myself in a way that building a model of my own before starting to code has become one of my essentials while I work. Honestly, I did not draw any UML diagrams because the project was small and did not needed any communication between other people.

For technical requirements, I used my Java skills that I learned in CS 102. Although I passed CS 101 and 102 with doubts that I did not manage to learn enough, I recalled many principles that I had learned from those courses. My project was in Java, and I found myself very comfortable with the language even after not writing Java code for 2 years.

I also consider myself lucky because I loved the course EEE 391 when I took it in the first semester. This project needed a significant amount of knowledge from the course, and I was prepared to some extent while I was doing the project.

## 4.2 Solving Engineering Problems

The problems I faced were occurring in the runtime for the most of the time. The project worked with threads and finding the root cause of the problems were challenging. Most of the time, the problems occurred because of tiny mathematical mistakes, like end of array and calculation mistakes with floating point numbers. I could not solve most of my problems by browsing from the internet, because the errors that compiler gave usually directed me to wrong solutions. For some instances, I had to close myself in a room for complete silence and tested the project over and over again with different scenarios to see if the code worked properly. Luckily, my supervisor had experience on the topic so I was able to find a very reliable helping hand.

## 4.3 Team Work

GuitarTabGenerator was a solo project that was assigned to me. In terms of team work, every day my supervisor gave me tasks and I tried to complete them. Day by day, the project was implemented. My supervisor also gave me research tasks (i.e. Cepstrum, Fourier Transform) for weekends so that I could adapt the knowledge to code easily and do not waste time. From time to time, I found myself not only asking to my supervisor, but to other developers as well.

## 4.4 Multi-Disciplinary Work

In my project, as I mentioned earlier I worked alone. However, as a social person, I had dialogues with the other software engineers about their projects and how crucially important communication is. I also managed to meet employees that are not computer engineers. The multi-disciplinary work was a concept that I did not apply, but instead observed.

## 4.5 Professional and Ethical Issues

During my internship I did not face with ethical issues because I was not assigned to a company project. Instead, I was assigned a project that I would develop on my own and ask to my supervisor when I struggled. Even though I demanded to work on a company project, I’ve been told that the projects were too large and complex for me to understand. However, I was able to observe some of the professional and ethical issues between the company and the clients. For instance, a client needed an information from the database of the company. But the information that was passed needed to be encrypted private. The data had many attributes but the client needed some of them as detached. This was a huge complication among the employees and resulted to communication mistakes between them. I observed that this problem continued for nearly a week.

### 4.6 Impact of Engineering Solutions

As I mentioned earlier, problems in my project did not trigger chain reactions that failed money transactions. Instead, I tried to observe how engineering solutions affected chain reactions in the finance sector. My project can be considered as an engineering solution for musicians to use. Although the system is not working 100% because I discovered the dominant ($5^{th}$) notes have tendency to dominate the root note, a guitarist who is having hardships transcribing notes of a song while he/she is listening to the song, can still benefit from my project.

### 4.7 Locating Sources and Self-Learning

My self-learning period was almost every day because the only information I had was of Java language. On the weekends my supervisor gave me articles on sound processing, Fourier Transformation, Cepstrum and design patterns. I learned about NetBeans' GUI builder which is rather simple.

In the company I did not watch any YouTube tutorials to learn because the website was banned for employees inside the company [12]. That was when I discovered that learning from documentations and reading is much more effective than watching a video and copying what the tutorial does.

### 4.8 Knowledge about Contemporary Issues

Throughout the internship I tried to gain knowledge about contemporary issues by listening to hall or lunch conversations between my colleagues. I tried to participate in them if I had even a trivial idea of the topic. I learned about the usage of Java in finance sector, how it works multi-threaded while NodeJS works single threaded [13]. I also learned about the importance of multiple layers in architecture by means of security. Also in my first day, I was taken to a cold server room where I was shown two huge server wardrobes. I had the opportunity to observe RAID 1+0 which is disk mirroring and striping which I had just learned in CS342 operating systems course. It was actually satisfying to see the information concretely in front of me.

### 4.9 Using New Tools and Technologies

While building this project, I used Maven while managing my dependencies [14]. It was new for me because I have never used a dependency manager before in any of my school projects. I realized that I tool like Maven is essential once there are greater amounts of dependencies to manage. I found NetBeans IDE very practical and intuitive to use unlike Eclipse IDE. It was my first time using NetBeans because I was used to use IntelliJ. I also learned more about using Linux and a special distribution of it which uses XFCE as desktop manager, called Xubuntu.

## 5 Conclusions

In my 1-month internship, the most significant concepts I learned were applying certain manipulations to a soundwave and creating a desktop application which works on the fly. Building an application which works real time was harder then it seemed. Therefore,

the application worked with threads. Between those threads, there needed to be enough time to sleep so read and write ends of the pipes would not overload.

The system in the project looks fairly simple but I had many struggles during my implementation. While developing the application, I had to change the range of the notes I inspected several times because the intervals between the note positions were not very precise. I experimented with channel size, audio sample rate, audio sample size until I started getting accurate results. The application still does not work optimal when two or more sounds collide. Here, I chose a tradeoff between accuracy and chord displaying. This means I preferred to give the output as single notes but more correctly rather than displaying a chord with many sounds that may be incorrect.

Overall, my internship experience in Cybersoft was a boost for my morale most importantly. It was my first official internship and I gained decent amount of knowledge about the sector for 1 month. I found opportunities to observe a company with many computer engineers and I was able to learn from them. Finally, I include a screenshot of the application that both the sonogram and transcribed notes can be seen.

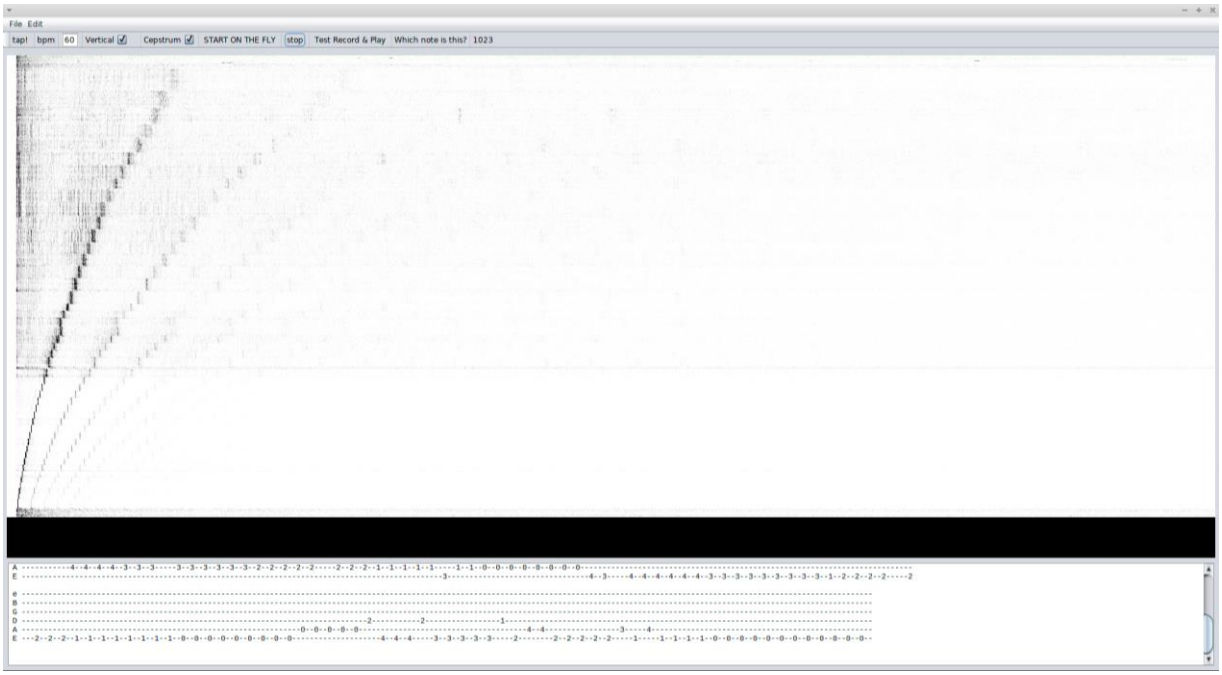


Figure 5 – a Screenshot from the Final Testing of the Project

## References


[1] Shazam. (2019). *Shazam*. [online] Available at: https://www.shazam.com/ [Accessed 28 Sep. 2019].

[2] community, T. (2019). *Xubuntu*. [online] Xubuntu.org. Available at: https://xubuntu.org/ [Accessed 28 Sep. 2019].

[3] Netbeans.org. (2019). *Welcome to NetBeans*. [online] Available at: https://netbeans.org/ [Accessed 28 Sep. 2019].

[4] Tobb.org.tr. (2019). [online] Available at: https://www.tobb.org.tr/Sayfalar/AnaSayfa.php [Accessed 28 Sep. 2019].

[5] En.hmb.gov.tr. (2019). *T.C. Hazine ve Maliye Bakanlığı*. [online] Available at: https://en.hmb.gov.tr/ [Accessed 28 Sep. 2019].

[6] Graehmefloyd.com. (2019). [online] Available at: https://www.graehmefloyd.com/wp-content/uploads/2018/12/c-major-tab.png [Accessed 28 Sep. 2019].

[7] Docs.oracle.com. (2019). *Java Platform SE 7*. [online] Available at: https://docs.oracle.com/javase/7/docs/api/ [Accessed 28 Sep. 2019].

[8] GitHub. (2019). *wendykierp/JTransforms*. [online] Available at: https://github.com/wendykierp/JTransforms [Accessed 28 Sep. 2019].

[9] Research.cs.tamu.edu. (2019). [online] Available at: http://research.cs.tamu.edu/prism/lectures/sp/l9.pdf [Accessed 28 Sep. 2019].

[10] Johndcook.com. (2019). *Cepstrum, quefrency, & pitch detection | cepstral analysis*. [online] Available at: https://www.johndcook.com/blog/2016/05/18/cepstrum-quefrency-and-pitch/ [Accessed 28 Sep. 2019].

[11] Mathworks.com. (2019). *Cepstrum Analysis- MATLAB & Simulink*. [online] Available at: https://www.mathworks.com/help/signal/ug/cepstrum-analysis.html [Accessed 28 Sep. 2019].

[12] Youtube.com. (2019). *Your browser is deprecated, please upgrade. - YouTube*. [online] Available at: https://www.youtube.com/ [Accessed 28 Sep. 2019].

[13] Foundation, N. (2019). *Node.js*. [online] Node.js. Available at: https://nodejs.org/en/ [Accessed 28 Sep. 2019].

[14] Porter, B., Zyl, J. and Lamy, O. (2019). *Maven – Welcome to Apache Maven*. [online] Maven.apache.org. Available at: http://maven.apache.org/ [Accessed 28 Sep. 2019].

## Appendices

In this appendix section, the codes for section 3.2.2 that is the Fourier and Cepstrum transformer, without imports is included. The test classes are not included. The guitar model class that maps guitar notes on TAB key is also included.

Below is a code piece for receiving the data from VoiceRecorder and processing the array accordingly. This is the code for section 3.2.2 which is the Fourier Transformer.

```
/**
        *
        * @author yukseldeniz
        */
        public class FourierTransformer implements Runnable {


        private PipedInputStream fourierIn;
        private PipedOutputStream tabTransOut;
        private PipedOutputStream sonographOut;
        public static final int START_FOURIER_POS = 4;
        public static final int CEPSTRUM_LENGTH = 4096;


        public FourierTransformer(PipedInputStream fourierIn, PipedOutputStream
        sonographOut, PipedOutputStream tabTransOut) {
        this.fourierIn = fourierIn;
        this.tabTransOut = tabTransOut;
        this.sonographOut = sonographOut;
        }


        @Override
        public void run() {
        System.out.println("fourier started");
        int bufferSize = VoiceRecorder.calculateBufferSize();
        int bytesInSec = VoiceRecorder.calculateBytesInSec();
        int remaining = bytesInSec;
        ByteArrayOutputStream bout = new ByteArrayOutputStream();
        while (MainFrame.longTrainRunning) {
        Map timeMap = new HashMap();
        try {
        long start = System.currentTimeMillis();
        long loopstart = start;
        int n;
```

```
do {
byte[] fourierPiece = new byte[remaining];
n = fourierIn.read(fourierPiece);
//                    System.out.println("n: " + n);
if (n > 0) {
bout.write(fourierPiece, 0, n);
remaining -= n;
}
} while (n >= 0 && remaining > 0);


//System.out.println("Read buffer size: " + bout.size());
byte[] readVoice = bout.toByteArray();


//                for( int i = 0; i < readVoice.length; i++)
//                    System.out.println("Read voice: " + readVoice[i]);
//
byte[] audioData = new byte[bufferSize];


//                for (int i = 0; i < bufferSize; i = i +
readVoice.length) {
//
//                    if (i + readVoice.length > bufferSize) {
//                        System.arraycopy(readVoice, 0, audioData, i,
bufferSize - i);
//                    } else {
//                        System.arraycopy(readVoice, 0, audioData, i,
readVoice.length);
//                    }
//                }


if( bufferSize > readVoice.length * 2){
System.arraycopy(readVoice, 0, audioData, audioData.length / 2,
readVoice.length);
int halfAudioLength = audioData.length / 2;
for( int i = 0; i < readVoice.length; i++){
audioData[halfAudioLength - readVoice.length + i] = audioData[
halfAudioLength + i];
}
}
else{
System.arraycopy(readVoice, 0, audioData, 0, readVoice.length);
```

```
}

ByteBuffer byteBuf = ByteBuffer.wrap(audioData);

long end = System.currentTimeMillis();
timeMap.put("Read buffer", (end - start));
start = end;

float[] floatArray = new float[audioData.length / 2];

short min = 0;
short max = 0;

for (int i = 0; i < audioData.length / 2; i++) {
short s = byteBuf.getShort();

floatArray[i] = s;
if (s < min) {
min = s;
}

if (s >= max) {
max = s;
}
}

// normalize the wave.
/*
for (int i = 0; i < audioData.length / 2; i++) {
floatArray[i] += -min;
floatArray[i] /= (max - min);
if (floatArray[i] < 0.1f) {
floatArray[i] = 0.0f;
}
}
*/
```

```
end = System.currentTimeMillis();
timeMap.put("Normalizaiton: ", (end - start));
start = end;


//fouirer...
FloatFFT_1D fft = new FloatFFT_1D(VoiceRecorder.PRECISION);
fft.realForward(floatArray);


end = System.currentTimeMillis();
timeMap.put("Fourier: ", (end - start));
start = end;


int maxView = MainFrame.VIEW_LENGTH;


if (maxView > VoiceRecorder.PRECISION / 2) {
maxView = VoiceRecorder.PRECISION / 2;
}


if (MainFrame.useCepstrum) {
maxView = FourierTransformer.CEPSTRUM_LENGTH;
}


//take absolute value...
float[] absFourier = new float[floatArray.length];


float maxFourier = 0f;
int maxIndex = 0;
for (int i = 0; i < absFourier.length / 2; i++) {


absFourier[i] = (floatArray[2 * i] * floatArray[2 * i] + floatArray[2 * i
+ 1] * floatArray[2 * i + 1]);


if (MainFrame.useCepstrum) {
absFourier[i] = (float) Math.log10(absFourier[i]);
} else {
absFourier[i] = (float) Math.sqrt(absFourier[i]);
}
```

```
if (i > START_FOURIER_POS - 1 && absFourier[i] > maxFourier) {
maxFourier = absFourier[i];
}
}

if (MainFrame.useCepstrum) {
maxFourier = 0f;

FloatFFT_1D fftCepstrum = new FloatFFT_1D(CEPSTRUM_LENGTH * 2);

float[] cepstrumData = new float[CEPSTRUM_LENGTH * 4];

for (int i = 0; i < cepstrumData.length; i = i + absFourier.length) {
if (i + absFourier.length > cepstrumData.length) {
System.arraycopy(absFourier, 0, cepstrumData, i, cepstrumData.length -
i);
} else {
System.arraycopy(absFourier, 0, cepstrumData, i, absFourier.length);
}
}

fftCepstrum.realForward(cepstrumData);

float[] absCepstrum = new float[CEPSTRUM_LENGTH];

for (int i = 0; i < absCepstrum.length; i++) {
absCepstrum[i] = (cepstrumData[2 * i] * cepstrumData[2 * i] +
cepstrumData[2 * i + 1] * cepstrumData[2 * i + 1]);
//absCepstrum[i] = (float) Math.sqrt(absCepstrum[i]);

}
absFourier = new float[absCepstrum.length];
for (int i = 0; i < absCepstrum.length; i++) {
absFourier[i] = absCepstrum[i];

if (i < CEPSTRUM_LENGTH / 2 && i > 14 && absCepstrum[i] > maxFourier) {
maxFourier = absCepstrum[i];
```

```
maxIndex = i;
}
}
}

System.out.println("Max fourirei " + maxFourier + " Max index: " +
maxIndex);
//normalize fourier or cepstrum transformed wave.
float factor = 127.0f / maxFourier;

int byteArrayLength = (maxView - START_FOURIER_POS) * 4;

byte[] sendWave = new byte[byteArrayLength];
ByteBuffer buf = ByteBuffer.wrap(sendWave);

for (int i = START_FOURIER_POS; i < maxView; i++) {
buf.putFloat(absFourier[i] * factor);
//System.out.println("Absfourier[i] = " + absFourier[i] + " i: " + i);
}

end = System.currentTimeMillis();
timeMap.put("Normalize Fourier", (end - start));
start = end;

sonographOut.write(sendWave);
end = System.currentTimeMillis();
timeMap.put("sonograph write: ", (end - start));
start = end;

tabTransOut.write(sendWave);
end = System.currentTimeMillis();
timeMap.put("tab write", (end - start));
start = end;

bout = new ByteArrayOutputStream();
remaining = bytesInSec / 2;
bout.write(audioData, remaining, remaining);
end = System.currentTimeMillis();
```

```
timeMap.put("byte array output: ", (end - start));
start = end;


timeMap.put("loop time : ", (end - loopstart));


System.out.println("Fourier time: " + System.currentTimeMillis() + "
durations: " + timeMap.toString());


} catch (IOException ex) {
Logger.getLogger(FourierTransformer.class.getName()).log(Level.SEVERE,
null, ex);
}
}
}


}
```

The GuitarModel class is a helper class to map a note on the guitar. The code is included below.

```
/**
 *
 * @author yukseldeniz
 */
public class GuitarModel {


    private static Map<String, LinkedListNote> guitarMap;
    private final String E_LOW_START = "E4";
    private final String B_START = "B3";
    private final String G_START = "G3";
    private final String D_START = "D3";
    private final String A_START = "A2";
    private final String E_HIGH_START = "E2";
    private static final int MAX_FRET = 21;
    private static final int STRINGS = 6;
    private static final String[] STRING_HEADS = {"E4", "B3", "G3", "D3",
"A2", "E2"};
    private static final String[] solfege = {"C", "Cs", "D", "Ds", "E",
"F", "Fs", "G", "Gs", "A", "As", "B"};
```

```
    public GuitarModel() {

        guitarMap = initMap();
        fillMap();

    }

    private Map initMap() {

        Properties noteMap = TabTranscriptor.getNoteMap();
        int size = noteMap.size();

        guitarMap = new LinkedHashMap<String, LinkedListNote>();
        String[] noteKeys = TabTranscriptor.getNoteKeys();

        for (int i = 0; i < noteKeys.length; i++) {
            guitarMap.put(noteKeys[i], new LinkedListNote());
        }
        return guitarMap;

    }

    public void printGuitarFretBoard() {

        //List<String>              keyList              =              new
ArrayList<String>(guitarMap.keySet());
        //Collections.sort(keyList);
        String[] keyList = guitarMap.keySet().toArray(new String[0]);
        for (int i = 0; i < guitarMap.size(); i++) {
            System.out.println(keyList[i]);
            guitarMap.get(keyList[i]).printList();
            System.out.println();
        }
```

```
    }


    public Map getGuitarMap() {
        return guitarMap;
    }


    public int getStringCount() {
        return STRINGS;
    }

    public String[] getSolfege(){
        return solfege;
    }


    public void fillMap() {
        int strCount = 0;
        int fretCount = 0;
        String curStringStart;


        while (strCount < STRINGS) {


            curStringStart = STRING_HEADS[strCount];
            System.out.println("TEL: " + curStringStart);
            //int keyPosition = keyList.indexOf(curStringStart);
            //int destination = keyPosition + MAX_FRET;


            int solfejPos = -1;
            int octave = Integer.valueOf(curStringStart.substring(1));
            String   noteWithoutOctave   =   curStringStart.substring(0,
curStringStart.length() - 1);
            for (int i = 0; i < solfege.length; i++) {
                if (noteWithoutOctave.equals(solfege[i])) {
                    solfejPos = i;
                }
            }


            if (solfejPos < 0) {
                System.out.println("UYARI");
            } else {
```

```
                for (int fret = 0; fret < MAX_FRET; fret++) {
                    String note = solfege[solfejPos] + octave;
                    if (solfejPos == solfege.length - 1) {
                        solfejPos = 0;
                        octave++;
                    } else {
                        solfejPos++;
                    }
                    //System.out.println("Note: " + note);


                    LinkedListNote listPositions = guitarMap.get(note);
                    listPositions.add(strCount, fret);


                }
            }
            strCount++;
        }
    }


   /*
    public static void main(String[] args) {
        GuitarModel model = new GuitarModel();
        model.printGuitarFretBoard();
    }
    */

}
```

## Self-Checklist for Your Report

*Please check the items here before submitting your report. This signed checklist should be the final page of your report.*

- ☐ Did you provide detailed information about the work you did?
- ☐ Is supervisor information included?
- ☐ Did you use the Report Template to prepare your report, so that it has a cover page, the 8 major sections and 13 subsections specified in the Table of Contents, and uses the required section names?
- ☐ Did you follow the style guidelines?
- ☐ Does you report look professionally written?
- ☐ Does your report include all necessary References, and proper citations to them in the body?
- ☐ Did you remove all explanations from the Report Template, which are marked with yellow color? Did you modify all text marked with green according to your case?

Signature: _____