COS 126

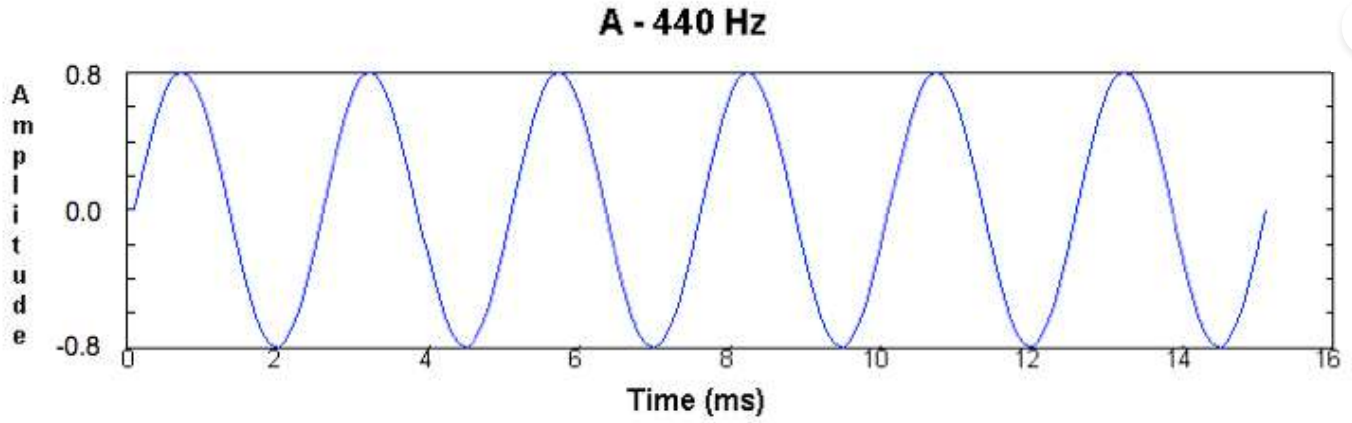
Programming Assignment

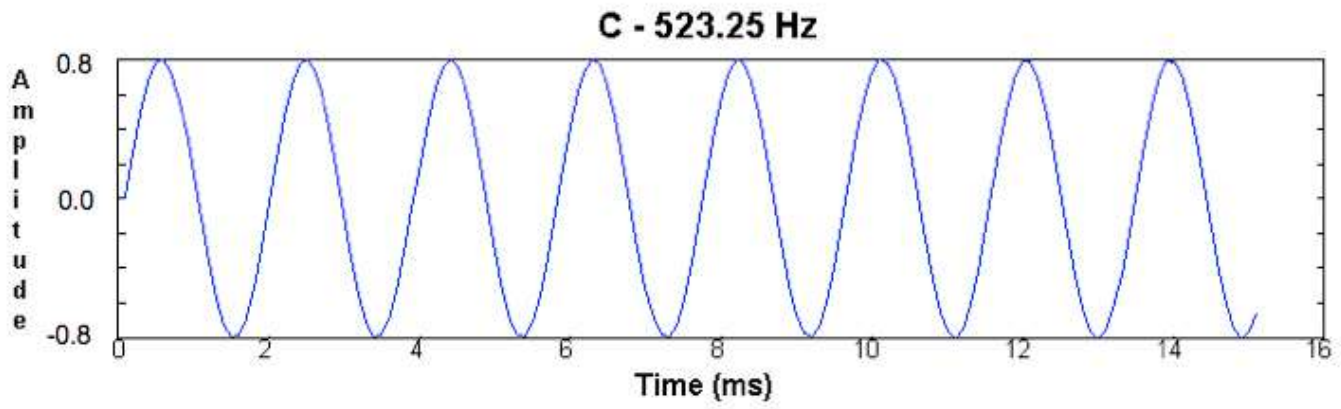
Digital Signal Processing

Due: Monday

Write a program to generate sound waves, apply an echo filter to an MP3 file, and plot the waves.

A crash course in sound. A music note can be characterized by its *frequency* of oscillation. For example, the music note *concert A* is a sine wave repeated 440 times per second (440 Hz); the note C is a sine wave repeated approximately 523.25 times per second. We amplify the signal to an audible level by multiplying it by a constant, say 0.8. Below are pictures of an A and a C with duration 15 milliseconds and amplitude 0.8. The picture for A consists of 0.015 × 440 = 6.6 sine waves.

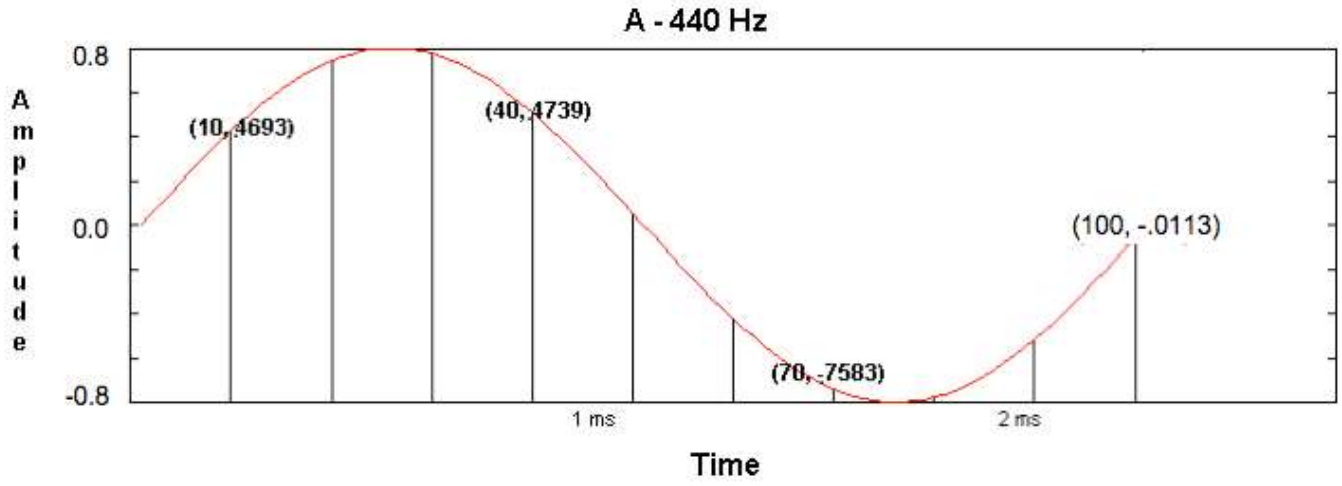




Below is a table containing the frequencies in Hertz of all twelve musical notes in the fifth octave. The ratio between the frequency of successive notes is the 12th root of two (1.05946). An *octave* is a doubling or halving of the frequency. For reference, the normal range of human hearing is between 20 and 20,000 Hz.

A	A#	B	C	C#	D	D#	E	F	F#	G	G#	A
440.00	466.16	493.88	523.25	554.37	587.33	622.25	659.26	698.46	739.99	783.99	830.61	880.00

Digital audio. Digital audio is produced by *sampling* the instantaneous amplitude of the continuous sound wave many times a second. Each sample is a real number between -1 and +1 that represents the amplitude of the sound wave at a particular instant in time. The *sampling rate* is the number of samples taken per second. Audio from CDs typically uses a sampling rate of 44,100 and two channels (left and right). Below is a picture of an A using a sampling rate of 44,100 and a duration of approximately 1/440th of a second. The figure below displays every 10th sample.



The following arrays contain the values representing the height of the wave above, at a sampling rate of 44,100:

```
left  = { 0, .0501, .1000, .1495, .1985, .2466, .2938, .3399, .3846, .4277, .4693, ..., -.0113 }
right = { 0, .0501, .1000, .1495, .1985, .2466, .2938, .3399, .3846, .4277, .4693, ..., -.0113 }
```

Sample i is given by .8 sin(2 π * 440 * i / 44,100), and we've rounded the numbers towards zero.

Sound synthesis. Your first task is to create a data type `Wave` to store and manipulate sound wave samples so that the program A.java below plays concert A for 2 seconds with maximum amplitude 0.8. Similarly, FurElise.java plays the first nine notes of *Fur Elise*.

```
public class A {
    public static void main(String[] args) {
        StdPlayer.open();
        Wave A = new Wave(440.0, 2.0, .8);
        A.play();
        StdPlayer.close();
        System.exit(0);
    }
}
```

For this part of the assignment, you need to write a data type `Wave` to initialize and access the data. Implement the samples for the left and right channels using two arrays of type `double[]`. First, implement the constructor

```
public Wave(double Hz, double seconds, double amplitude)
```

It should create a new `Wave` object that represents a sine wave of the specified number of Hz that is sampled 44,100 times per second over the specified number of seconds. Use `Math.sin()` and `Math.PI` to initialize the left and right channels. Initialize both arrays with the same values so that the note will play in both speakers. Next, implement a method

```
public void play()
```

that sends the `Wave` data to the sound card. Use the static library method `StdPlayer.playWave(left, right)`, which takes as input two `double` arrays representing the left and the right channel.

Compilation and execution. In order to play music on your computer, you will need to use a specialized library. Download the file stdplayer.jar and put in your working directory. A .jar file is an archive like a .zip file, but there is no need to unzip it or peek inside this one unless you are extremely curious. This library is a modified version of the JavaLayer 1.0 MP3 Player. (As per the GPL license, the jar file contains the original JavaLayer library and our modified source code.) To test the library on your system, use the JavaLayer Player that comes with `stdplayer.jar` by typing the following command:

```
java -cp stdplayer.jar javazoom.jl.player.jlp felten.mp3
```

You can replace felten.mp3 with the name of another MP3 file in the working directory. In order to use the library in your Java program, you must put the following import statement at the beginning:

```
import javazoom.jl.player.StdPlayer;
```

You also need to tell Java where to find the library by compiling and executing with the following commands:

	Compile	Execute
OS X / Linux	`javac -cp .:stdplayer.jar A.java`	`java -cp .:stdplayer.jar A`
Windows	`javac -cp .;stdplayer.jar A.java`	`java -cp .;stdplayer.jar A`

Creating a tune. Now that you can play single notes, your next challenge is to play several notes at once, for example, to play a chord. To accomplish this, first add a new constructor

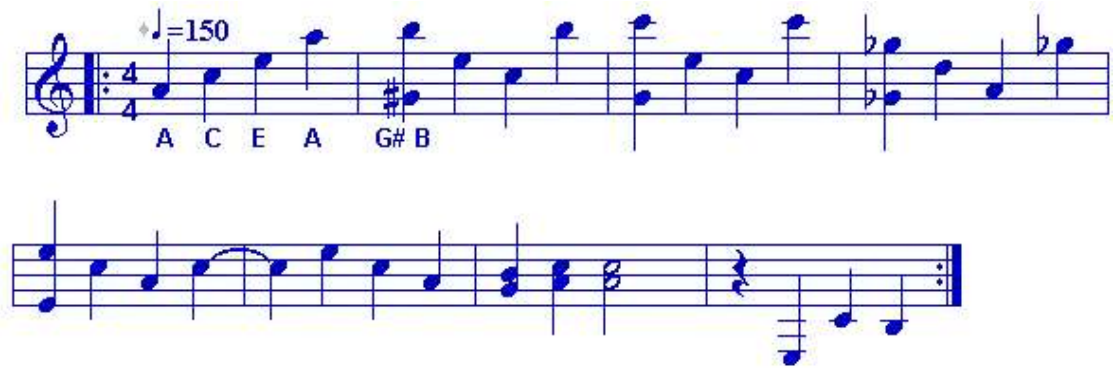
```
public Wave (double[] left, double[] right)
```

to `Wave.java` that takes two `double` arrays as arguments and initializes a new `Wave` object with the given data. Now, to create a chord, you can create individual notes and combine them together by writing a method:

```
public Wave plus(Wave b)
```

so that `a.plus(b)` returns the sum of `a` and `b`. To add two Waves, add the corresponding array entries for the left and right channels, element-by-element. Test your data type by using the program StairwayToHeaven.java, which is the beginning of the famous Led Zeppelin tune. Note the fifth wave played is created by the following sequence of statements:

```
Wave B6  = new Wave(493.88 * 2, .4, .4);
Wave Gs4 = new Wave(830.61 / 2, .4, .4);
Wave GsB = B6.plus(Gs4);
```



Playing an MP3 file. The program MP3Player.java decodes the MP3 file specified by the command line input and plays it. Assuming that you implemented the `Wave` API above, there is no need to write any code for this part (but you should test it and enjoy). Compile and execute the program with the following commands.

```
javac -cp .;stdplayer.jar MP3Player.java
java  -cp .;stdplayer.jar MP3Player felten.mp3
```

The key part of the code is the following loop. It will be useful for the remaining parts of the assignment.

```
StdPlayer.open(args[0]);
while (!StdPlayer.isEmpty()) {
    double[] left  = StdPlayer.getLeftChannel();
    double[] right = StdPlayer.getRightChannel();
    Wave w = new Wave(left, right);
```

```
        w.play();
    }
```

This program uses three new methods from the `StdPlayer` library to decode data from an MP3 file. The `String` argument to the function `StdPlayer.open()` specifies which MP3 file to use. The function `StdPlayer.getLeftChannel()` returns an array of 1,152 real numbers between -1 and +1 that are the samples of the music intended for the left speaker. The function `StdPlayer.getRightChannel()` is analogous.

Echo filter. Now that you can decode and play MP3 files, you're ready to modify the data and change the characteristics of the sound waves. An *analog filter* accomplishes this by manipulating the electrical signals that represent the sound wave; a *digital filter* does this by manipulating the digital data that represents that `Wave`. Your task is to write a program `EchoFilter.java` that implements a digital echo filter. An *echo filter* of delay 10 is a filter that adds an echo to the sound by adding the sound wave at time *t - 10* to the one at time *t*. To create this effect, maintain an array of the past 10 `Wave` objects and add the `Wave` that was originally read 10 waves ago to the current `Wave`. The echo filter is a client program and should be written entirely in `EchoFilter.java`. To test the filter, you can use pearlharbor.mp3 which contains the speech President Roosevelt delivered after the attack on Pearl Harbor.

Plotting the waves. The final part of the assignment is to write a program `MP3Viewer.java` that takes the name of an MP3 file as a command line argument and animates both stereo channels. This program is not supposed to play the MP3 file, only to animate the sound waves. Add the following method to `Wave.java` to plot the left channel on the top half of the screen, and the right channel on the bottom half:

```
public void draw()
```

Deliverables. Submit `Wave.java`, `EchoFilter.java`, and `MP3Viewer.java` along with a `readme.txt` file.

Challenge for the bored. Write a program `MP3Visualizer.java` that plays the MP3 file and simultaneously displays a cool effect based on the raw data. To keep the music and animation smooth, you may need to tweak a few parameters. For example, adjust the delay in the method `StdDraw.show()` to keep the wave from scrolling by too fast. Also, try plotting every other wave if your computer is too slow.

This assignment was created by Bradley Zankel and Kevin Wayne.