CSCE 145: Homework 7
Music Processing: A Low-Pass Sound Filter and an Echo

Objectives
In this assignment you will learn about
- Waveforms
- Characteristics of sound
- Filtering

Program Specification
The purpose of this assignment is for you to learn about sound waveforms and how to process them, which is known as digital signal processing (DSP). Your program will read a sound waveform from a file, encoded in the .wav format, process it, and write the result to a new file. You will write two DSP methods for low-pass filtering and echoing.

1. Apply a low-pass filtering algorithm to the waveform. You should try various amounts of filtering (20, 40, 80, and 160 samples) and listen to the results. Save one of the results to an output .wav file.
2. Add an echo to the sound waveform. Try delays of 500, 1000, and 2000 samples. Save one of the results to an output .wav file.

Details
Create a project in Eclipse for this assignment. Download this zipped file http://www.cse.sc.edu/~carrollh/CSCE145/files/SoundStuff.zip and unzip its eight (8) java source files into the src folder that Eclipse created for your project. Unzip its four (4) .gif image files into the bin folder that Eclipse created. (In eclipse, you will see all of these files inside the folder "(default package)"). Next, download the class http://www.cse.sc.edu/~carrollh/CSCE145/lib/Example.java and also store it in the src folder of your project. Look at the code and try to understand how it works, so that you will be able to modify it for your own purposes.

A low-pass filtered version of the sound can be created by replacing each sample with the average of its neighboring samples. I suggest you try 20 samples first. Specifically, have your program create a new Sound object having the same length as the original one and set each sample sound2.setSampleAt(n, avgValue) to be (sound1.getSample(n) + sound1.getSample(n+1) + ... + sound1.getSample(n+19)
)/20. Save the processed Sound object 'sound2' by the line of code: sound2.write("FilteredSound.wav");

By using an average, the waveform is smoothed out, which removes the highest frequencies. The lowest frequencies are unchanged, i.e., the filter passes the low frequencies and removes the high ones. You shold be able to hear the difference. Here is a bit of a song by the German singer Nena on which you can test your algorithm "Tokyo" by Nena (http://www.cse.sc.edu/~carrollh/CSCE145/files/TokyoNena_1.wav). Here is a stereo version (http://www.cse.sc.edu/~carrollh/CSCE145/files/TokyoNena.wav) that sounds better, but only one channel will be affected by the code in Example.java and it is difficult to hear the changes. To use these music files without having to include folder information in your code, save them in the folder having the name of your project (not in its src subfolder).

To create an echo effect, add a delayed version of the sound to itself. Try a delay of 500 samples first. Assume the echo is half as loud as the original. Specifically, create a new Sound object with length that is 500 samples shorter than the original Sound object. The value of the nth sample of the new Sound object is (sound1.getSample(n)/2.0 + sound1.getSample(n+500))/1.5

Upload your source code and your three .wav files (original, low-pass filtered, and echoed) to the dropbox at https://dropbox.cse.sc.edu.