# COSC 1P02 – Lab 07

## Exercise 1 - Let's Make Some Noise

*Estimated Time: 30 min*

Noise is the bane of recording engineers and audiophiles. What is noise? Noise is an unwanted signal (waveform) that distorts the desired signal (e.g. the music) (see: http://en.wikipedia.org/wiki/Noise). Noise occurs from many different sources. Background noise is extraneous sounds occurring in the background such as conversations at other tables in a restaurant. Electronic noise or hum is generated by electronic circuits and high voltage lines. Hiss (or white noise) is noise that is produced by random effects such as cosmic rays, random fluctuations in current etc.

When two sounds occur in the same space, their waveforms combine just like the ripples from two stones thrown into a pond. The waveform that is produced is just the result of adding the amplitudes of corresponding samples in the two sounds.

Let's write a program to distort a sound by adding hiss. Hiss is just random signal. We could generate a hiss waveform by simply selecting random values for the samples in turn. To distort a sound with hiss, we can simply add random values (the samples of the hypothetical hiss sound) to the amplitudes of the samples in the sound.

To generate the random amplitude values, we can again use the `random` method from the `java.lang.Math` library. Remember we can produce random integers in a desired range using the expression:

```
(int)(n*random())+s
```

where *`n`* is the number of distinct values we want and *`s`* is the smallest value we want.

For example, to generate integers from `1` to `10` we could use (`10` values starting at `1`):

```
(int)(10*random())+1
```

To generate integers between `-v` and `v`, we could use:

```
(int)((2*v+1)*random())-v
```

since there are `2v+1` values between `-v` and `v` inclusive and `-v` is the smallest value we want.

Write a method:

```
private void makeNoisy ( Sound aSound, int nAmp ) {
```

that adds hiss with maximum amplitude `nAmp` to the sound `aSound`.
Write a program to use this method to add hiss with amplitude 1000 to a sound loaded from disk.

## Exercise 2 - Let The Light Shine In

*Estimated Time: 30 min*

In lecture, we looked at *normalizing* a Sound clip. Specifically, we amplified the magnitudes of all a Sound's samples to the maximum extent possible without *clipping*.

We did this by first completing an 'initial pass' of all the values, to find out what the largest magnitude in the whole Sound was. We then calculated the multiplier needed to stretch that largest magnitude into the maximum possible value (32,767),and used that same multiplier across all the samples. Since it was, by definition, 'safe' for the largest magnitude, it was also safe for the remaining ones.

So… why not do the same for brightness? Well… for this exercise you'll give it a try!

Consider this crop of a larger image:

It's included as `monocrop.jpg` for the lab. You will see that it doesn't make a large difference. Try `monocrop2.jpg` and compare the difference.

Write a program to perform a similar normalization to this image. It's barely even new:

- Instead of trying to find the 'highest magnitude', you're looking for the *highest channel value* (without concern for whether it's a red, a green, or a blue)
- The maximum possible value is now 255, instead of 32,767
- Instead of multiplying each sample's amplitude, you're multiplying the reds, greens, *and* blues
- Instead of calling it something like `amplify`, `brighten` makes more sense

So give it a shot! It shouldn't take *too* long (so long as you attended the lecture).

## Exercise 3 -Hitting the right note

***Estimated Time: 20 min***

We've generated noise (hiss and hum), but can we generate a pure note (a sine wave). Remember, a sine wave is generated by the function `sin` (in the Math library) and has a period (cycle) of `2π`. If we want to generate a sine wave with a specific frequency (i.e. a note such as A above middle C with frequency 440Hz), we need to have 440 cycles per second (i.e. the values for the parameter (coordinate) for the `sin` function have to move through `0-2π 440` times each second. Remember that the `sin` function itself is cyclic, (i.e. the value of `sin(0)` is the same as the value of `sin(2π)` which is the same as the value of `sin(4π)`), so all we have to do is keep incrementing the parameter (coordinate) by an amount such that it goes from `0-2π 440` (frequency) in one

second. Since the sampling rate tells us the number of samples in one second, we can work that out.

Write a method:

```
private Sound makeSineWave ( int duration, int freq, int amp )
```

that creates a new sound (with default attributes including sampling rate) which is a sine wave with the specified duration (in samples), frequency (cycles per second) and amplitude.

To get a specific amplitude, we need a sound whose highest sample is the given amplitude. Since the values produced by `sin` are between `-1` and `1`, we simply have to multiply the result from the function by `amp` when generating the wave.

Try generating some waves of specific frequencies (e.g. 440Hz, 880Hz). Keep the durations relatively small and the amplitude in the range 10,000-20,000.

## Exercise 4 - "I Write the Songs"[1] (Based on Labs 06 and 07)

***Estimated Time: 30 min***

We now have all the parts to be able to write our own songs. The method `makeSineWave` from Exercise 3 can be used to create a note of a composition. The method `splice` from Exercise 4 allows us to put notes together in sequence.

Write a program that uses an `ASCIIPrompter` to ask the user how many notes are in the composition and then, repeatedly for each note, prompts for a duration, frequency and amplitude for the note, creates a sine wave for the note and then splices it on the end of the song being written.

What about the first note? There is no song to join it to. We can consider a song of zero length (i.e. no notes, no samples, sample length zero) as the song to which we splice the first note. With such a starting point, the first note can be treated just as the others.

Write the program and use it to compose your own song.

[1] Bruce Johnston (1975) as recorded by Barry Manilow