CSC 110 Assignment 4 :

Create your own digital sound mix!

Bring a pair of headphones for use in the lab so that you can hear what your code does!!

Motivation:

The music industry today heavily relies on the use of software technology in order to edit, mix, remaster, and synthesize digital audio recordings. In addition, there is a growing trend among musicians to make use of computers and technology for both music composition and performance. This assignment will have you develop your own set of software tools for manipulating digital audio recordings. You will then use these tools in order to compose your own *creative* sound recording!

When computers store sound, what they store are a number of pieces of audio, called **samples**, that are used to represent the sound at a particular (or *discrete*) point in time. **Each one of these samples is simply a floating-point number between -1 and +1 !!** When you listen to an MP3 through a set of laptop or PC speakers, your computer's hardware is taking the stream of numbers that make up the digital sound and is converting them into an electric current which your speakers then convert into sound (pressure) waves !!

Learning Outcomes:

- How to create a 1 dimensional array, and use it to locate table elements based on an index value.
- How to read from and write to array elements, using both explicit and computed index values.

Pair Programming :

This assignment can be completed as part of a pair programming team. If you choose to do so:

- read the description of pair programming provided in the Resources section of the course web page
- find a suitable pair programming partner.
- (both members of the pair programming team must) send an email to the course email (csc110@csc.uvic.ca) informing the instructional staff of the pairing.
- together with your partner, create a schedule for assignment completion

Before You Begin:

For this assignment, you will be using several Java classes that are provided to you so that you can quickly and easily access and manipulate digital audio files. The .class files for these Java classes have been packaged into a single JAR (Java ARchive) file. In order to use the provided classes:

1. Download the JAR file (media.jar) and save it into the directory where your source code will be,
   - **DO NOT work on this assignment on your H: drive.** Instead, keep your source code, audio, and this JAR file in a directory on the desktop. After you have finished a work session, copy this directory onto your H: drive so that your files will be saved for future access.
2. Use the following *to compile* your source code from the command line of a PC: javac -classpath media.jar**;**. MyJavaApp.java
3. Use the following *to execute* your application from the command line of a PC: java -classpath media.jar**;**. MyJavaApp
   - Note: If you are working on a Mac running OS X, the semi-colon (**;**) should be replaced with a colon (**:**)

This will allow you to be able to access the classes contained in the JAR file *without* having to use any import statements in your source code.

Audio File Types: For this assignment, you will need to work with WAV (.wav) or AU (.au) files ONLY !

In addition, the files should be relatively short in duration (< 1 min), and be comprised of a single channel of audio (mono). Choose your digital audio files from those provided through the Assignment4Sounds folder under the Resources link of your course web page in order to ensure this criteria is met.

Details:

This assignment contains two parts:

- In the first part you will develop the tools you need to create your own digital sound mix.
- In the second part, you will use the tools you have developed to create a unique and creative digital sound mix from digital audio files that you choose.
  Both parts are due immediately before your lab during the week of November 15 through 19. For this assignment only, either of Parts 1 will be weighted at 3% and the other at 1%.

PART 1: Sound editing in Java

In this assignment, you need to use of a Java class, called Sound, in order to access and manipulate digital audio recordings. That class is made available to you in a file called *media.jar* file (that you downloaded earlier.) The methods of that class are demonstrated in the file SoundCheck.java that is available on under the Resources link on the course web site. Note that the possible values that **each sound sample can hold** are in the range from **-1 to +1**.

For part 1 of this assignment, you will create several static Java methods enabling you to manipulate digital audio recordings.

1.1 Implement EACH of the 5 methods described below. In your main() method provide, for each method, one or more test call(s) that demonstrates the functionality of your method.

A) `public static void adjustVolume( double [] samples, int startIndex, int endIndex, double factor )`

- A method for controlling sound volume is to scale (i.e. multiply) each of the values by some factor.
- **samples** is an array containing the sound samples to be scaled.
- The samples to be scaled are located from indices **startIndex** (inclusive, i.e. including) to **endIndex** (exclusive, i.e. not including).
- **factor** represents a scalar that each SoundSample will be multiplied by.
- Place a comment in your method's code explaining what happens to the volume if factor > 1 and if factor < 1?
- WARNING: What should happen if the user specifies **endIndex** to have a value greater than the length of the **samples** array?
- Try this with your working method, make a decision on how to best handle this situation then alter your method!

B) `public static void add( double [] samples1, double [] samples2 )`

- A method that mixes two sounds together, the sample values at corresponding indices of the two sounds are simply added together!
- The result of adding the two sets of samples together should be placed into the array **samples1**, i.e. the **samples1** array will be modified by this method, but the **samples2** method will *not* be modified
- WARNING: What should happen if **samples1** contains more samples than **samples2**, or vice versa?
- Try this with your working method, make a decision on how to best handle this situation then alter your method!
- METHOD USAGE WARNING: If invoking this method (e.g. from **main( )**) results in the addition of two samples to give a magnitude larger than 1, "clipping" will occur which will create a noisy effect in your recording. If this happens, you may try invoking **adjustVolume(...)** on the **samples1** array *after* calling the **add(...)** method in order to *decrease* the magnitude of the sample values.

C) `public static void reverse( double [] samples )`

- Reverse the samples so that if, for example, the original sound contained the sample values { 0.1 0.2 0.3 0.4 0.5 }, the result of invoking this method would be an array with samples { 0.5 0.4 0.3 0.2 0.1 }.

D) `public static void forceToExtremes( double [] samples )`

- For each sound sample, if the sample has a value >= 0, this method will replace it with +1, or if the sample has a value < 0, will replace it with -1.
- The noise you hear introduced is the result of a phenomenon called "clipping". What is interesting is that despite the drastic change made to the sound recording, you can still make out words and instruments!

E) `public static void splice( double [] srcSamples, int srcStart, int srcStop, double [] destSamples, int destStart )`

- This method will copy a segment of the source array, **srcSamples**, from sample index **srcStart** (inclusive) to **srcStop** (exclusive), to a destination array, **destSamples**, beginning at sample index **destStart**
- WARNING: What should happen if **srcStop** has a value that indexes beyond the last element in the array **srcSamples**, OR if the **destSamples** array does not contain sufficient samples to store all the samples copied from **srcStart** (inclusive) to **srcStop** (exclusive) ? Try this with your working method, make a decision on how to best handle this situation then alter your method!

F) `public static void adjustFrequency( double [] samples, double factor )`

- The following pseudocode describes how you will implement this method.

```
SET original  = duplicate array of samples
SET origIndex = 0

FOR EACH index FROM 0 to samples.length-1

    IF origIndex >= original.length
        samples [index ] = 0
    ELSE
        samples [index ] = original [origIndex ]
    ENDIF
```

```
            origIndex = origIndex + factor

        ENDFOR
```

- What is **factor**? What if **factor** has a value > 1? or < 1 ?
    - Place a comment in your method's code explaining what happens to the volume if factor > 1 and if factor < 1?

PART 2: Using your tools creatively

You have now created your own software that will allow you to channel your creativity into an exciting digital audio mixing project!

Using **between 3 and 5** digital audio recordings selected from those provided (see Resources link of the course web site), develop a creative audio collage using the methods you developed in PART 1 of this assignment. Your audio collage should be **25 to 35 seconds** in duration.

Create a **flow chart** that illustrates how the independent source recordings were manipulated and combined to produce the final masterpiece.

What to Hand In:

Submit the following:

- **Source code** and **documentation** from PART 1 and PART 2
- **Flow chart** created in PART 2
- A **short reflective summary** (1/4 to 1/2 page single-spaced) that may include topics such as describing any inspiration you may have had for the mix you developed, the process you went through to create it, how you feel the software you created may have enabled/inhibited you during the process, any changes you would like to make to your original audio mix, or ideas for future mixes, etc., etc., etc.

More to Discover:

Some musicians create music solely through interaction with computers and technology. For example, PLOrk, the Princeton University Laptop Orchestra, is composed of computer musicians who use laptops and computer programming to produce music. Other musicians, such as those at the Music Technology Group in Barcelona, Spain, have been able to create innovating technology-driven instruments such as the Reactable, a sound-synthesizing multi-touch-sensitive table.

In this assignment, you have use programming to create tools for audio manipulation. Some computer musicians create music by sampling pre-recorded tracks in this way. Other computer musicians synthesize digital (discrete) sound waves from scratch, which can then be converted into analog (continuous) sound and played through a set of speakers. If you would like to learn more about computer music, check out the following resources : [2], [3]. In addition, you may be interested in exploring one of several free software tools for computer music composition and performance, such as CSound [4, 5], jMusic [6], and Pure Data (PD) [7].

Reference:

[1] Mark Guzdial , Barbara Ericson, *Introduction to Computing and Programming with Java: A Multimedia Approach*, Prentice-Hall, Inc., Upper Saddle River, NJ, 2006.

[2] Charles Dodge, Thomas A. Jerse, *Computer music: Synthesis, composition, and performance*,

Schirmer Books: London, 1997.

[3] Curtis Roads, *The computer music tutorial*, MIT Press, Cambridge, MA, 1996.

[4] Richard Boulanger (ed), *The csound book: Perspectives in synthesis, sound design, signal processing, and programming*, MIT Press, Cambridge, MA, 2000.

[5] CSound : http://www.csounds.com/

[6] jMusic : http://jmusic.ci.qut.edu.au

[7] Pure Data : http://puredata.info