PHAS3459
Scientific Programming Using Object-Oriented Languages

Exam 2017

18th January, 14:00 to 17:00

[Dr Simon Jolly and Dr Ben Waugh]

Please read the exam guidelines, rules, instructions and marking criteria at `https://moodle.ucl.ac.uk/mod/wiki/view.php?pageid=15228` (linked from the Examinations and coursework page). This exam is worth 50% of your final mark for the course. The duration of the exam is 3 hours. The Java source code of your solution to the programming exercise should be uploaded using Moodle under the section headed Exam II. Each class should be uploaded as a separate file. Your classes should be created in a package called `exam2`. You must upload all your classes used in your solution, including any you have copied or imported from earlier coursework modules. The code you upload must be self-contained: the marker must be able to compile and run it using only the classes uploaded and the Java API. If you use your own classes from earlier modules, make sure you copy them into the `exam2` package and upload them along with any new classes you create during the exam. You are advised to read the entire exam paper before starting work.

In this exam you will process some sound recordings. The input data files are provided in the web directory

`http://www.hep.ucl.ac.uk/undergrad/3459/exam-data/2016-17/`

and their contents are described below.

- The file `index.txt` contains details of the audio files to be processed. Each line contains two fields, separated by white space:
  - the filename;
  - the type of musical instrument used to create the recorded sound.
- Each of the other files in the directory is an audio file in text format. The first line of each file contains three numbers separated by white space:
  - The sampling frequency f_s in Hertz;
  - The number of samples N;
  - The maximum possible amplitude of the signal $a_{\max}$.

  Each of the remaining N lines contains one integer, representing the amplitude of the audio signal at the corresponding time. So for example a recording of duration 1.5 s made at a sampling frequency of 10 kHz, using 24 bits per sample, would contain $N = 15000$ samples, have a maximum amplitude of $a_{\max} = 2^{23} = 8388608$, and would start something like this:

```
10000 15000 8388608
0
2
3
1
-3
-7
-11
-10
-5
1
```

Part 1

(20 marks)

Write a program, with a main class called `ExamPart1`, and any other classes you choose, to do the following:

- Read the data from the files detailed above, and store them in one or more appropriate data structures.
- For each file, calculate (where necessary) and print the following information:
  - the filename;
  - the duration ($T = N/f_s$) of the recording;
  - the amplitude of the signal in dBFS, defined as

$$A = 20 \log_{10} \frac{a_{\mathrm{RMS}}}{a_{\mathrm{max}}}$$

    where a_{RMS} is the root-mean-square (RMS) amplitude of the signal;
  - the instrument used in making the recording.

Part 2

(20 marks)

A researcher is investigating different algorithms for classifying audio samples, in order to identify musical instruments or animals from the sounds they make. You have been asked to create a Java framework to test these algorithms.

- Create an interface to represent an algorithm for classifying sounds: it should define a function that takes appropriate input and returns a string representing the classification of the sound.
- Create an implementation that simply classifies sounds as "long" or "short", depending on whether their duration is greater than 1 s or not.
- Create another implementation that classifies sounds as "loud" or "quiet", depending on whether their amplitude is greater than -20 dBFS or not.
- Write a program, with a main class called `ExamPart2`, that demonstrates the results of your implementations on the files that were used in part 1.

You may make use of your classes from part 1 as well as creating any others you choose.

Part 3

(10 marks)

The *spectral density* of a signal at frequency f is given by

$$S_{xx}(f) = \frac{(\Delta t)^2}{T} \left| \sum_{n=0}^{N-1} x_n e^{-2\pi i f T n/N} \right|^2$$

where T is the duration of the audio sample, N is the number of samples, $\Delta t = T/N$, and the values x_n are the individual samples, with n taking values from 0 to $N-1$ inclusive. You may use the following Java implementation:

```
private double spectralDensity(long[] samples, double t, double f) {
    int bigN = samples.length;
    double z = 2 * Math.PI * f * t / bigN;
    double sumCos = 0;
    double sumSin = 0;
    for (int n = 0; n < bigN; ++n) {
        sumCos += samples[n]* Math.cos(z*n);
        sumSin += samples[n]* Math.sin(z*n);
    }
    double norm = t / (bigN*bigN);
    return norm * (sumCos*sumCos + sumSin*sumSin);
}
```

- Create a class that implements the interface from part 2 and classifies sounds as "low", "medium" or "high" using the following simple algorithm:
  - Calculate the spectral density at each of the following frequencies: 100 Hz, 400 Hz, and 1 kHz.
  - The string returned depends on which of the spectral densities is greatest:
    - 100 Hz: "low"
    - 400 Hz: "medium"
    - 1 kHz: "high"
- Write a program, with a main class called `ExamPart3`, that reports the results of this classifier on each of the audio files provided.

END OF PAPER