CS 1316 - Homework 6 – Sound and Morse Code

Due before 6pm on Friday, March 5th.

THIS IS AN INDIVIDUAL ASSIGNMENT!

You must work individually on this assignment. You may collaborate with other students in this class. Collaboration means talking through problems, assisting with debugging, explaining a concept, etc. You should not exchange code or write code for others. For individual assignments, each student must turn in a unique program. Your submission must not be substantially similar to another student's submission. Collaboration at a reasonable level will not result in substantially similar code.

For this assignment, you will be creating a subclass of Sound called MorseCode which adds some functionality. MorseCode will have two static methods which are used to produce Morse code.

Note: Sound does not have a default constructor, in order to instantiate MorseCode you need to create a constructor using a parameter that is understood in the superclass. (Java will not let you have an object without a constructor of some type.) However, as you are creating two static methods, you do not *need* to instantiate an object of type MorseCode.

Sine Wave

In order to produce normal sounding tones, we need a static method to make sinusoidal waves. This method should take two parameters: time in seconds (double) and samples per wavelength (int). An example syntax:

```
MorseCode dot = MorseCode.sin(0.3,84);
```

Remember that Frequency is measured in Hz, or wavelengths per second. With CD quality sound, each second of sound has 44,100 samples. The SimpleSound class (which is sub-classed by the Sound class) uses a default sampling rate of ½ of this, or 22,050 samples per second.

Conversion [String → Morse Code]

Morse code is a method of communication used when distance made voice indecipherable over analog radio. It consists of a series of dots and dashes

(also known as dits and dahs) that represents letters. You will need to have a basic understanding of Morse code before we attempt to code it.

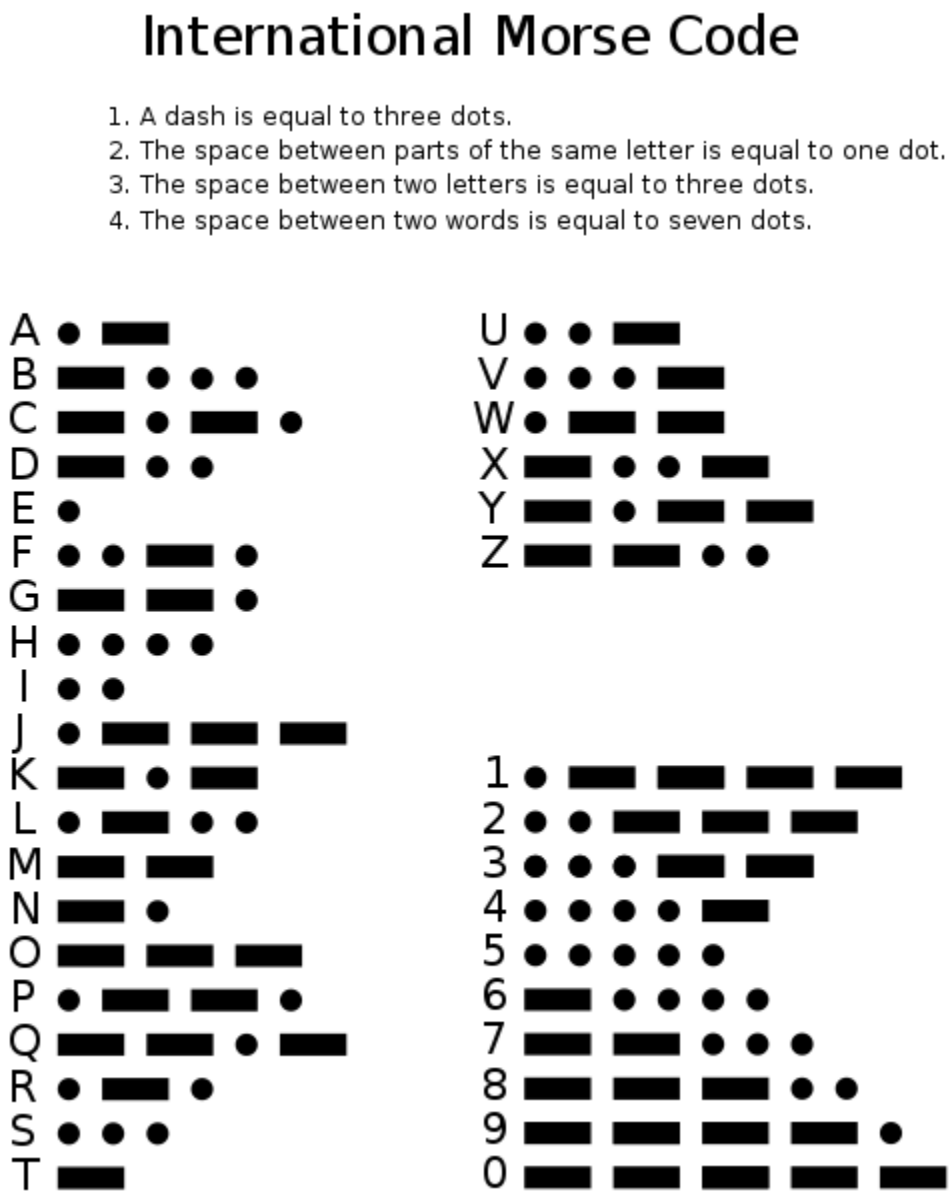


Image from Wikipedia.

The dot is the basic unit of measurement; whatever length is used to create the dot, the rest of the Morse code will be based off of multiples of the same length. Using our sin wave method, we will define what a dot is. For this assignment, a dot is defined to be 0.3 seconds long and takes 84 samples per wavelength. See above for syntax. Since the dot is 0.3 seconds long, a

dash will be 0.9 seconds long with the same number of samples per wavelength, and so on. Spaces between dots and dashes within a word are equal in length to dots, but silent.

You will create a static method called convert that takes in an input of a string and returns a `Sound`. An example syntax:

```
Sound converted = MorseCode.convert("abc");
```

To make things easy for you, convert only needs to work with strings that have the first 5 lowercase letters (a,b,c,d,e), and it will have no spaces.

Hints:

sin

`Math.sin` accepts radians, not degrees! Look at `Math.toRadians()` for help!

convert

Strings are made up of individual chars. (`char` is another data type)

Notice that a `char` is set off by single quotes. `' '`

`String.charAt(i)` returns a `char` located at index `i` inside the string

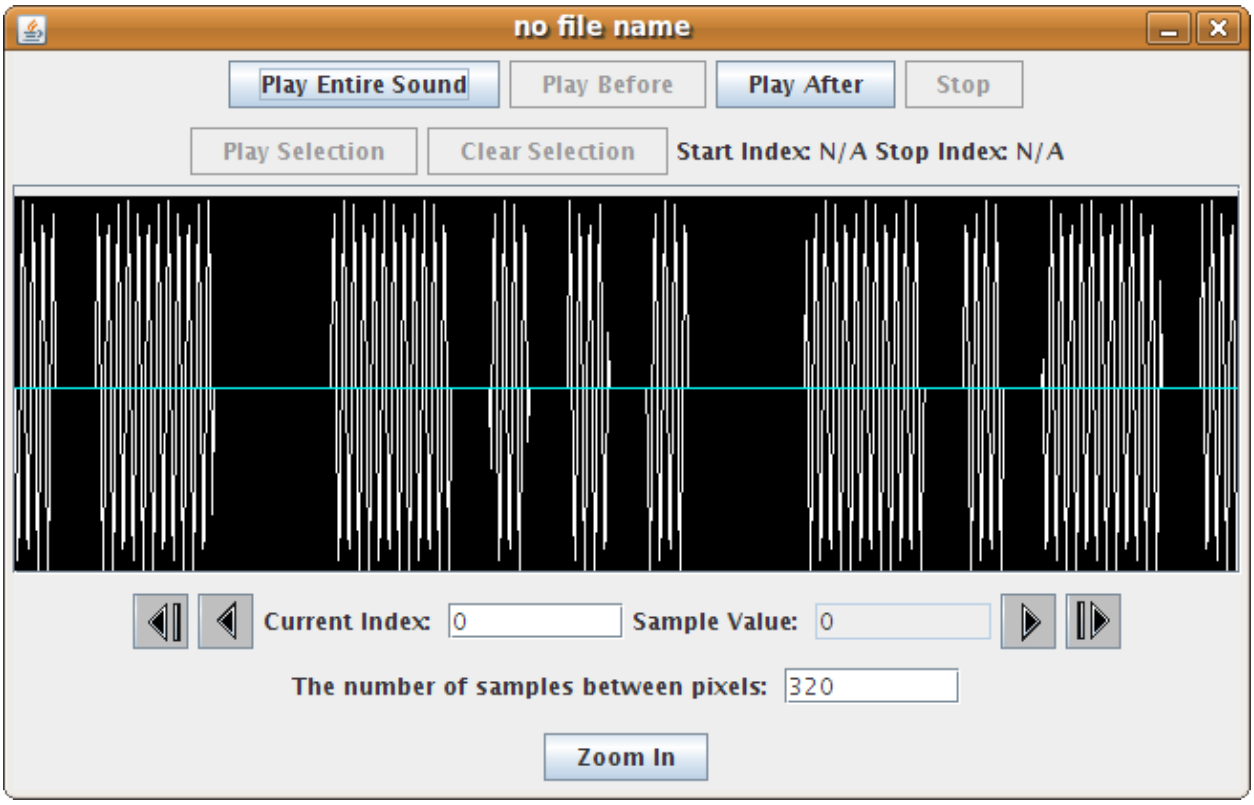
Look at the following code, which was typed in the interactions pane, for a better understanding:

```
> String str = "abc"
> char letter = str.charAt(1)
> letter
'b'
>  str.charAt(1) == 'b'
true
>
```

Think about how using a `Sound []` could be useful.

Don't forget about the extra space between each letters!

After you have built your sound, be sure to explore it to make sure it is correct! For example, here is a correct version of "abc":



Resources

This homework may call for a bit more prior-knowledge on physics and sound, so here are some links with relevant information.

- http://en.wikipedia.org/wiki/Sine_wave
- http://en.wikipedia.org/wiki/Frequency
- http://en.wikipedia.org/wiki/Hertz
- http://en.wikipedia.org/wiki/Morse_code

Extra Credit

- Account for all 26 letters in the alphabet
- Account for multiword strings
- Build an even more rugged interface for convert() with more parameters for other effects (such as speed and volume)
- Other interesting functionalities will be considered, just ask Dr. Summet or a TA.

Grading Criteria

- sin
    - Header correct 5pt
    - Produces a sine wave 10pt
    - Sine wave is the correct length and frequency 15pt
- convert
    - Header correct 5pt
    - Parses string input correct 20pt
    - The code for each letter is correct (5pts) 25pt total
    - Correctly “builds” the final sound 15pt
    - Final sound output is correct 5pt