Learning Goals

• Introduce Sound manipulation as a way to learn about
  – Arrays
  – Declaring variables
  – Sending objects messages
  – Iteration (Loops)
  – Conditionals
  – Writing methods
Acoustics, the physics of sound

• Sounds are waves of air pressure
  – Sound comes in cycles
  – The *frequency* of a wave is the number of cycles per second (cps), or *Hertz*
    • (Complex sounds have more than one frequency in them.)
  – The amplitude is the maximum height of the wave
Volume and Pitch

• Our perception of volume is related (logarithmically) to changes in amplitude
  – If the amplitude doubles, it’s about a 3 decibel (dB) change.
  – A *decibel* is a ratio between two intensities: \(10 \times \log_{10}(I_1/I_2)\)
  – As an absolute measure, it’s in comparison to threshold of audibility
    • 0 dB can’t be heard.
    • Normal speech is 60 dB.
    • A shout is about 80 dB

• Our perception of pitch is related (logarithmically) to changes in frequency
  – Higher frequencies are perceived as higher pitches
  – We can hear between 5 Hz and 20,000 Hz (20 kHz)
  – A above middle C is 440 Hz
Digitizing Sound

• In calculus you learn to estimate a curve by creating rectangles

• We can do the same to estimate the sound curve
  – Analog-to-digital conversion (ADC) will give us the amplitude at an instant as a number: a sample
  – How many samples do we need?
Nyquist Theorem

• We need twice as many samples as the maximum frequency in order to represent (and recreate, later) the original sound.
• The number of samples recorded per second is the *sampling rate*
  – If we capture 8000 samples per second, the highest frequency we can capture is 4000 Hz
    • That’s how phones work
  – If we capture more than 44,000 samples per second, we capture everything that we can hear (max 22,000 Hz)
    • CD quality is 44,100 samples per second
Try It

• Call a friend on a phone and play some music over the phone
  – How does it sound?
  – Phones only transmit 8,000 samples per second
  – The highest frequency you can transmit by phone is 4000 Hz
  – This is fine for voice but what does it do to music?
Digitizing Sound in the Computer

- Each sample is stored as a number (two bytes)
- What’s the range of available combinations?
  - 16 bits, $2^{16} = 65,536$
  - But we want both positive and negative values
    - To indicate compressions and rarefactions.
  - What if we use one bit to indicate positive (0) or negative (1)?
  - That leaves us with 15 bits
  - 15 bits, $2^{15} = 32,768$
  - One of those combinations will stand for zero
    - We’ll use a “positive” one, so that’s one less pattern for positives so the range is from -32,768 to 32,767
### Sound Basics

- **new Sound(fileName)**
  - Will create a new Sound object from the data in the file with the passed file name

- **soundObj.play()**
  - Will start the sound playing

- **soundObj.explore();**
  - Will open a sound explorer on the object

- **soundObj.blockingPlay()**
  - Will play the sound and wait to return until the sound is finished

- **soundObj.write(String fileName)**
  - Will write out the sound to the file
How to Create a Sound

• Ask the class Sound to make a new object
  – And pass arguments to the constructor to initialize the object from data in a .wav file
  – new Sound(fileName);

• How do we get the file name?
  – FileChooser.pickAFile();
    • To pick using a file chooser
  – FileChooser.getMediaPath(“croak.wav”);
    • If you know the base name
    • And the media path directory is set
Creating a Sound

• Create a sound in the interactions pane
  new Sound(FileChooser.getMediaPath("croak.wav"));
• What is the problem with just doing this?
  – Can we refer to the Sound object again?
• We need to have a way to refer to the object again
  – Sound sound1 = new
    Sound(FileChooser.getMediaPath("croak.wav"));
  – Sound sound2 = new
    Sound(FileChooser.getMediaPath("c4.wav"));
Play and Explore a Sound

Sound Explorer

```java
> sound1.play();
> sound1.explore();
> 
```

Type here
The Sound Explorer

• Not all of the sound is shown when you explore a sound
  – Skips values to fit in the window

• You can zoom in
  – To see all sample values

• You can zoom out
  – To fit the sound in the window again
Getting the Sound Sample Values

• A Sound has many values in it
  – Numbers that represent the sound at that time in the sample

• You can get an array of SoundSample objects
  – SoundSample[] sampleArray = sound1.getSamples();
Explore the Sound Sample Values

- Zoom in to see all the sound values

**Click here to pick an index**

**See the value**

**Type in an index**

**Click here to go to the next index**
Print the Sound Sample Value

• You can get the SoundSample object from the array at an index
  
  • and then get the value from that
    – SoundSample sample = sampleArray[0];
    – System.out.println(sample.getValue());

• What are the first 10 values of the Sound created from the file croak.wav?
Changing the Value of a Sound Sample

- You can set the value of a SoundSample
  ```java
  sample.setValue(value);
  ```
  This will change the value in the Sound object as well
- So how would you change the value to the original value * 2?

```java
SoundSample sample = sampleArray[0];
sample.setValue(sample.getValue() * 2);
```
Doubling all the Sound Values

• You could change each SoundSample by hand
  – There are 8808 SoundSamples in croak.wav
  – Do you really want to do that?
    • How long would it take you?

• Let’s let the computer do it in a loop
  – while
  – for
While Loop

- Loops while a boolean (true or false) test is true
  - When the test is false execution continues with the first statement after the loop
    ```
    while(test) {
        // statements to be done while the test is true
    }
    ```
  - To use a while loop you may need to declare variables before the loop and change them in the loop
While Loop to Process Sound Samples

```java
int index = 0; // starting index
SoundSample sample = null; // current sample obj
int value = 0; // value at sample
while (index < sampleArray.length)
{
    sample = sampleArray[index]; // get current obj
    value = sample.getValue(); // get the value
    sample.setValue(value * 2); // set the value
    index++; // increment index
}
```
Make the Sound Louder

• The way to make the sound louder is to increase the value of each sample
  – Can’t go past the maximum value: 32,767

• Get the array of SoundSample objects and loop through them and change each value to 2x the original value
public void increaseVolume()
{
    SoundSample[] sampleArray = this.getSamples(); // get array
    int index = 0; // starting index
    SoundSample sample = null; // current sample obj
    int value = 0; // value at sample

    // loop through SoundSample objects
    while (index < sampleArray.length)
    {
        sample = sampleArray[index]; // get current obj
        value = sample.getValue(); // get the value
        sample.setValue(value * 2); // set the value
        index++; // increment index
    }
}
Testing `increaseVolume`

- String `file = FileChooser.getMediaPath("gettysburg10.wav");`
- Sound `soundObj = new Sound(file);`
- `soundObj.play();`
- `soundObj.explore();`
- `soundObj.increaseVolume();`
- `soundObj.play();`
- `soundObj.explore();`
Tracing Execution

- The index is set to 0
- The value is set to the value in the array at that index (59)
- The sample value at the current index is set to 2 * value
- The index changes to the next index (1)
- We check if the index is less than the length of the array and
  - If so do the loop again
  - Else jump to the first statement after the loop
Memory versus Disk

• When we read from a file we read from disk into memory
  – Computers only do calculations on memory
• We change the values in memory
• The file on the disk hasn’t changed
• To save our new sound we need to write a file to the disk
  – soundObj.write(fileName);
Decrease Volume Exercise

• Write a method to decrease the volume of the sound
  – decreaseVolume()
  – Multiply each value by 0.5
• What parts need to change from the last method?
  – Only the calculation of the new value
• Try it:
  Sound s = new Sound(
      FileChooser.getMediaPath(" gettysburg10.wav");
  s.explore();
  s.decreaseVolume();
  s.explore();
While Loop versus For Loop

• It is easy to make mistakes when you use a while loop for looping a set number of times
  – Forget to declare variables before the loop
  – Forget to increment the variables in the loop before the next test

• Programmers use a For loop when the number of times to loop is known
  – And a while loop when you don’t know
For Loop

• A for loop allows you to declare and initialize variables, specify the test, and specify the way the variables change
  – All in one place
  – But, they still happen in the usual place
  ```java
  for(int index = 0; index < sampleArray.length; index++) {
    //
  }
  ```
public void increaseVolume()
{
    SoundSample[] sampleArray = this.getSamples();
    SoundSample sample = null;
    int value = 0;

    // loop through all the samples in the array
    for (int index = 0; index < sampleArray.length; index++)
    {
        sample = sampleArray[index];
        value = sample.getValue();
        sample.setValue(value * 2);
    }
}
Modify decreaseVolume Exercise

• Change decreaseVolume to use a for loop
  – Comment out declaration of the index
  – Comment out the increment of the index at the end of the loop
  – Comment out the while and put in a for loop
• Test it to make sure it still works
  – String file = FileChooser.getMediaPath("gettysburg10.wav");
  – Sound soundObj = new Sound(file);
  – soundObj.explore();
  – soundObj.decreaseVolume();
  – soundObj.explore();
General Change Volume Method

• The methods `increaseVolume` and `decreaseVolume` are very similar
  – They multiply the current sound values by a given amount
    • To change this you would need to modify the method and compile
  – The methods would be more reusable if we pass in the amount to multiply the current sound values by
    • As a parameter to the method
public void changeVolume(double factor) {
    SoundSample[] sampleArray = this.getSamples();
    SoundSample sample = null;
    int value = 0;

    // loop through all the samples in the array
    for (int i = 0; i < sampleArray.length; i++) {
        sample = sampleArray[i];
        value = sample.getValue();
        sample.setValue((int) (value * factor));
    }
}
Methods calling Methods

• One method can call another
  – Change decreaseVolume and increaseVolume to call changeVolume

• You can use
  – this.changeVolume
    • To invoke the method on the current object

• Or you can use
  – changeVolume
    • The this is implicit and will be added by the compiler
Normalize Sounds

• Make the whole sound as loud as possible
  – How loud can it be?
    • The max positive value is 32767.0
    • The max negative value is -32768.0
  – First we need to find the largest value (positive or negative) in the current sound
    • Create a variable to hold the max
      – And set it to the first value in the array
    • And loop through the array and if the absolute value of the current value is greater
      – Store that one instead
Normalize Sound - Continued

• After we find the maximum value
  – Determine the factor that we can multiply all values by
    • And not go over the maximum allowed value
      double multiplier = 32767.0 / largest;
  – Call the method changeVolume with the multiplier

• Test with
  String file =
    FileChooser.getMediaPath(“double preamble.wav”);
  Sound soundObj = new Sound(file);
  soundObj.explore();
  soundObj.normalize();
  soundObj.explore();
public void normalize()
{
    int largest = 0;
    int maxIndex = 0;
    SoundSample[] sampleArray = this.getSamples();
    SoundSample sample = null;
    int value = 0;

    // loop comparing values
    // to the current largest
    for (int i = 0; i < sampleArray.length; i++)
    {
        sample = sampleArray[i];
        value = Math.abs(sample.getValue());
        if (value > largest)
        {
            largest = value;
            maxIndex = i;
        }
    }

    System.out.println("largest "+
        largest + " at index "+
        maxIndex);

    // calculate the multiplier
    double multiplier = 32767.0 / largest;

    // change the volume
    this.changeVolume(multiplier);
}
Conditionals

• Allow you to only execute statements if an expression is true
  – Or (optionally) only if it is false
• If you clean your room
  – You can go out
• Else
  – You must stay home
Setting Values to Extremes Exercise

• Write the method `forceToExtremes()`
  – Change all positive values (including 0) to the max positive value 32767
  – and change all the negative values to -32768.
  – Using a conditional
    • if and else

• Test with:
  String file =
    FileChooser.getMediaPath("preamble.wav");
Sound soundObj = new Sound(file);
soundObj.explore();
soundObj.forceToExtremes();
soundObj.explore();
Summary

• Sounds are made by changes in air pressure
  – Compression followed by refraction
• The pitch is based on the frequency
  – Distance between cycles
• The volume is based on the amplitude (height of the wave)
  – Multiplying the value by > 1 increases the volume
• We can create Sound objects, play them, write them back out to a file
• We can get the array of SoundSample objects from a Sound
• We can get and set the value of a SoundSample