

Media Computation

Lecture 15.1, December 1, 2008

Steve Harrison

Playing a Sound

- We can create a Sound object just as we created a Picture object
 - Get a file name and save a reference to it
 - `String fileName = FileChooser.pickAFile();`
 - Pick a file that ends in .wav
 - Create the sound object by asking the class to create a new Sound object and initialize it by reading data from the given file name
 - `Sound sound1 = new Sound(fileName);`
 - Play the Sound
 - `sound1.play();`

Play Sound Exercise

- Try creating a Sound object and playing it by
 - Specifying it in steps
 - Specifying it all at once
- How would you play the same sound twice?

The screenshot shows the DrJava IDE interface. The top panel has three tabs: 'Interactions', 'Console', and 'Compiler Output'. The 'Interactions' tab is active and shows the following text:

```
Welcome to DrJava.  
> new Sound(FileChooser.pickAFile()).play();  
  
>
```

Below the interactions panel is a code editor window titled '(untitled)'. It shows a list of line numbers from 1 to 14. The bottom panel has three tabs: 'Interactions', 'Console', and 'Compiler Output'. The 'Interactions' tab is active and shows the following text:

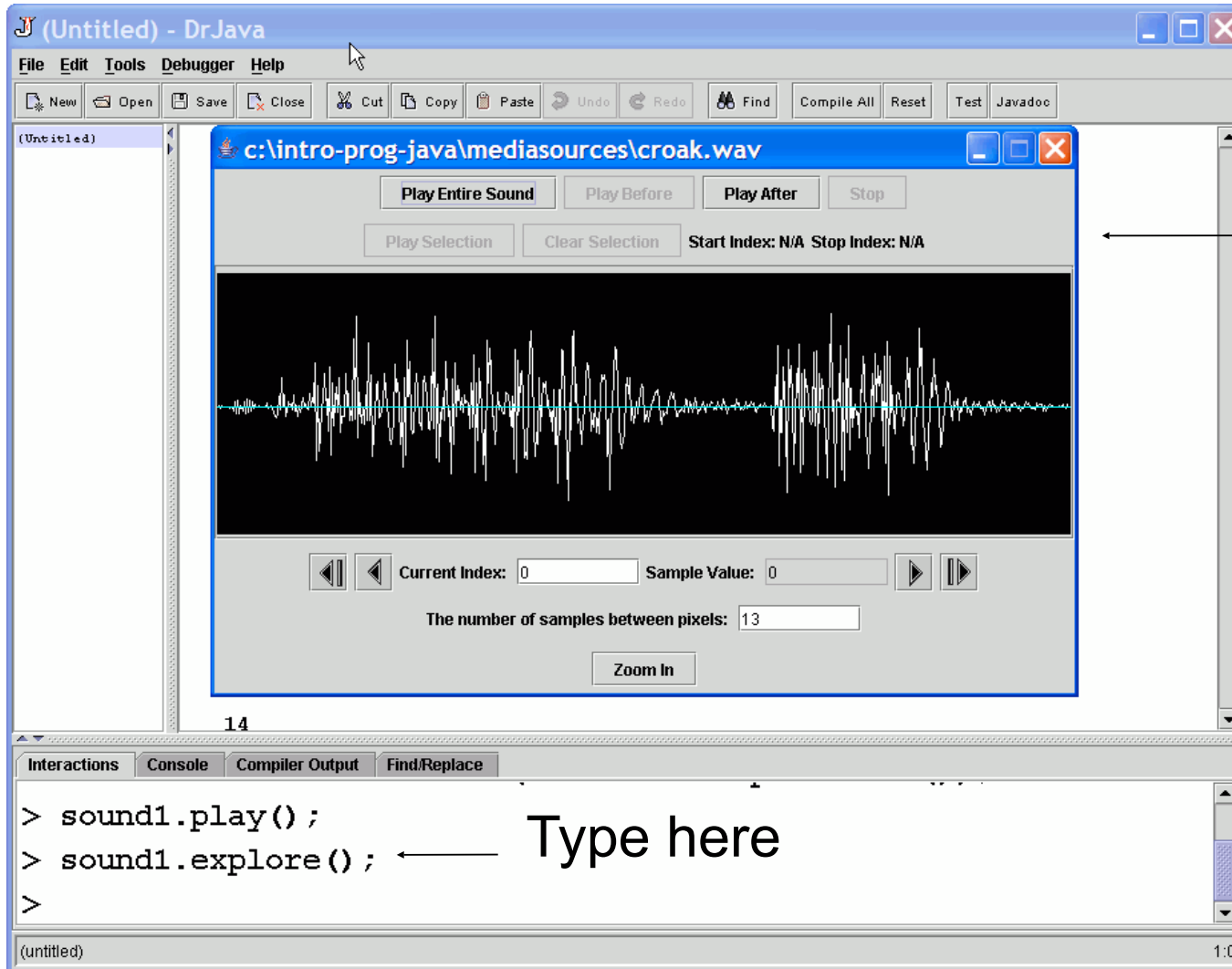
```
> String fileName = FileChooser.pickAFile();  
> Sound sound1 = new Sound(fileName);  
> sound1.play();
```

The Windows taskbar at the bottom shows the Start button, several open applications including 'Intro-MediaComp...', '(Untitled) - DrJava', and 'mediasources', along with system icons and the time '7:13 PM'.

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

Play and Explore a Sound



The screenshot shows the DrJava IDE interface. The main window is titled "(Untitled) - DrJava" and contains a menu bar (File, Edit, Tools, Debugger, Help) and a toolbar with icons for New, Open, Save, Close, Cut, Copy, Paste, Undo, Redo, Find, Compile All, Reset, Test, and Javadoc. A file explorer on the left shows "(Untitled)". A sound player window is open, titled "c:\intro-prog-java\mediasources\croak.wav". It features a waveform display and several control buttons: "Play Entire Sound", "Play Before", "Play After", "Stop", "Play Selection", and "Clear Selection". Below the waveform, there are input fields for "Current Index: 0" and "Sample Value: 0", and a "Zoom In" button. The console window at the bottom shows the following code:

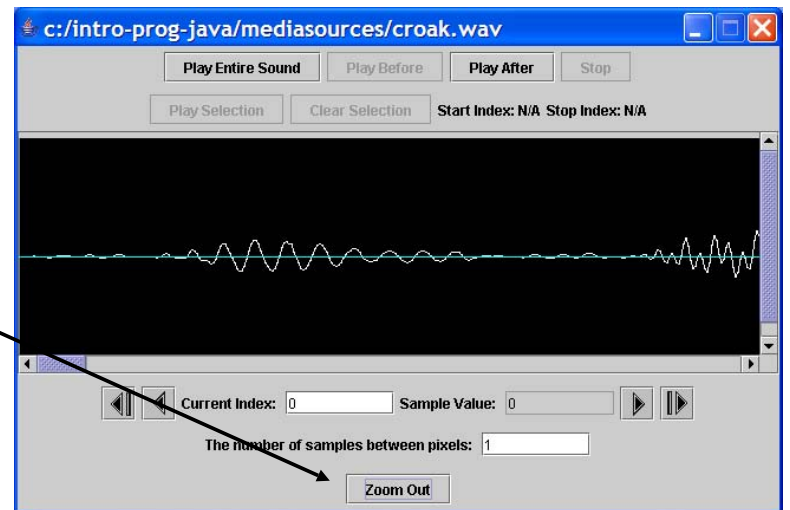
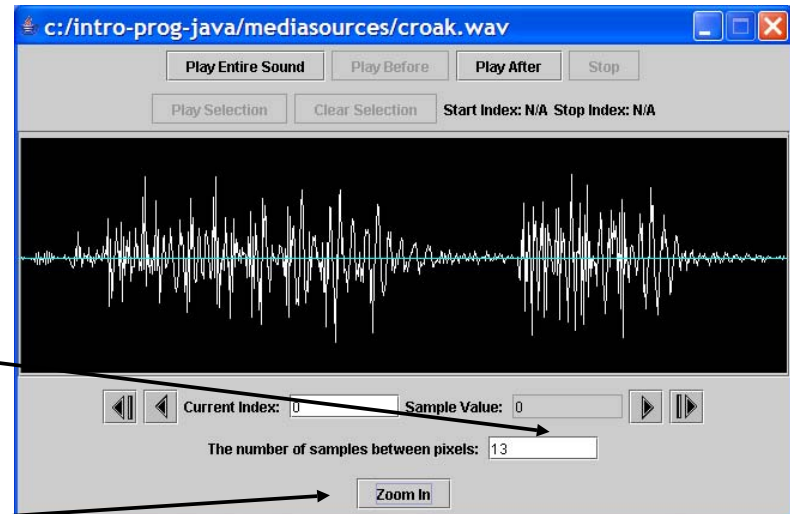
```
> sound1.play();  
> sound1.explore();  
>
```

The text "Type here" is written next to the second line of code, with an arrow pointing to the end of the line.

Sound Explorer

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

Current Index: 3 Sample Value: 0

The number of samples between pixels: 1

Zoom Out

Print the Sound Sample Value

- You can get the SoundSample object from the array at an index
 - `SoundSample sample = sampleArray[0];`
- And then get the value from that
 - `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
 - `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?

```
SoundSample sample = sampleArray[0];  
sample.setValue(sample.getValue() * 2);
```

For-Each Loop (Java 5.0)

- For each of the elements in a collection of objects do the body of the loop
 - Each time through the loop the `variableName` will refer to a different object in the collection

```
for (type variableName : collection)
{
    // statement to repeat
}
```

For-Each Loop to Process Sound Samples

```
SoundSample[] sampleArray = this.getSamples();  
int value = 0;  
  
for (SoundSample sample : sampleArray)  
{  
    value = sample.getValue();    // get the value  
    sample.setValue(value * 2);  // set the value  
}
```

Increase Volume with For-Each Loop

```
public void increaseVolume()
{
    SoundSample[] sampleArray = this.getSamples();
    int value = 0;           // value at sample

    // loop through SoundSample objects
    for (SoundSample sample : sampleArray)
    {
        value = sample.getValue(); // get the value
        sample.setValue(value * 2); // set the value
    }
}
```

Testing increaseVolume

String file =

```
FileChooser.getMediaPath("gettysburg10.wav");
```

```
Sound soundObj = new Sound(file);
```

```
soundObj.play();
```

```
soundObj.explore();
```

```
soundObj.increaseVolume();
```

```
soundObj.play();
```

```
soundObj.explore();
```

While Loop to Process Sound Samples

```
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
}
```

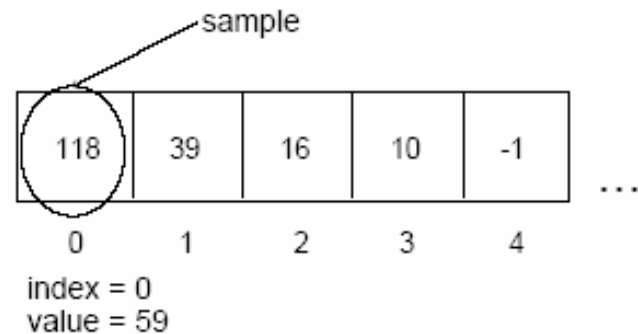
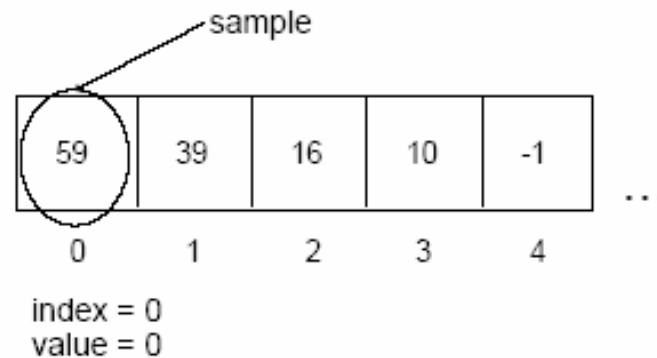
Increase Volume with While Loop

```
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
    }
}
```


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 $2 * \text{value}$
- 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);`

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

```
for (int index = 0;  
     index < sampleArray.length;  
     index++)  
{  
}
```

Increase Volume with a For Loop

```
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);
    }
}
```

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

General changeVolume 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));
    }
}
```

Find the Largest Value in an Array

- Start with the first value in the array
 - The one at index 0
- Loop through the rest of the items in the array
 - Compare the absolute value of the value at current index in the array to the absolute value of the largest

```
int max = valueArray[0];
```

- If it is bigger store it in max

```
for (int i = 1; i < valueArray.length; i++)  
{  
    if (Math.abs(valueArray[i]) > Math.abs(max))  
        max = valueArray[i];  
}
```


Find the Largest Value in an Array

2000 1000 -2344 100 3300

```
int max = first element
max = valueArray[0] = 2000;
for (int i = 1; i < valueArray.length; i++)
{
    if (Math.abs(valueArray[i]) >
        Math.abs(max))
        max = valueArray[i];
}
```

Find the Largest Value in an Array

2000 | 1000 | -2344 | 100 | 3300

```
int max = first element
max = valueArray[0] = 2000;
for (int i = 1; i < valueArray.length; i++)
{
    if (Math.abs(valueArray[i]) >
        Math.abs(max))
    {
        max = valueArray[i];
    }
}
```

Find the Largest Value in an Array

- What is the value of max each time through the loop?

0	1	2	3	4
2000	1000	-2344	100	3300

i	value	max
		2000
1	1000	2000
2	-2344	-2344
3	100	-2344
4	3300	3300

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 / max;
 - 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();
```

Normalize Method

```
public void normalize()
{
    SoundSample[] sampleArray =
        this.getSamples();
    SoundSample sample =
        sampleArray[0];
    int value = 0;
    int max = soundSample.getValue();

    // loop comparing values
    // to the current largest
    for (int i = 1; i <
        sampleArray.length; i++)
    {
        sample = sampleArray[i];
        value = sample.getValue();

        if (Math.abs(value) >
            Math.abs(max))
        {
            max = value;}
    }

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

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

Testing Normalize

- How do we know if it worked?
 - We can play the sound but it may not sound all that different
 - We can use the explorer to view the sound wave before and after we normalize the sound
 - And see if the values changed
 - Check more than one index
 - We can use `System.out.println` to print out the largest value and the index of it
 - We need to save the index of the maximum value
 - And use the explorer to check the value at that index

Normalize Method – Revised

```
public void normalize()
{
    SoundSample[] sampleArray =
        this.getSamples();
    SoundSample sample =
        sampleArray[0];
    int value = 0;
    int max = soundSample.getValue();
    int maxIndex = 0;

    // loop comparing values
    // to the current largest
    for (int i = 1; i <
        sampleArray.length; i++)
    {
        sample = sampleArray[i];
        value = sample.getValue();

        if (Math.abs(value) >
            Math.abs(max))
        {
            max = value;
            maxIndex = i;
        }
    }

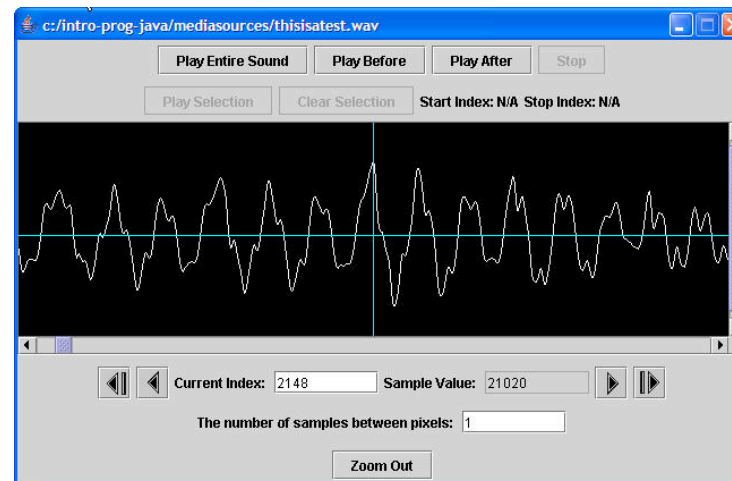
    System.out.println("largest " +
        max + " at index " +
        maxIndex);

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

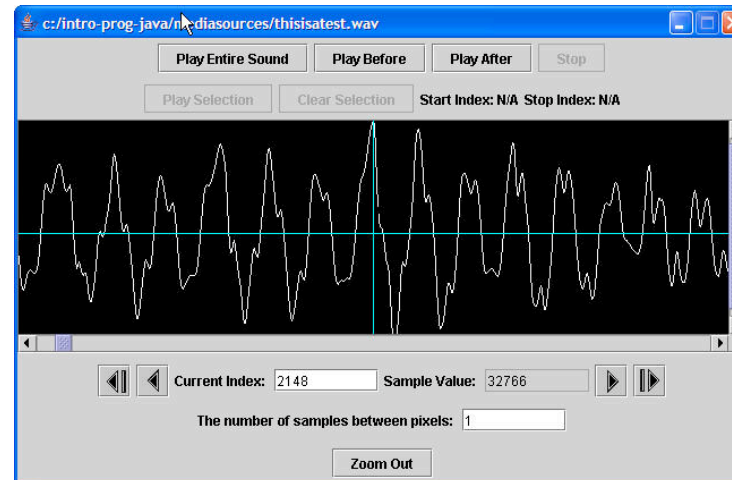
    // change the volume
    this.changeVolume(multiplier);
}
```

Testing Normalize

- Before Normalize



- After Normalize



Exercise - modify to handle special cond

PYTHON: Normalizing

- A few ways to think about “normalizing”:
 - **use the whole enchilada (don’t waste any bits...)**
 - **make everything use the same scale (0 to 100%)**
 - **need 2 loops -- one to find largest and one to reset**

```
def normalize( sound ) :
```

```
    largest = 0
```

```
    for sample in getSamples(sound):
```

```
        largest = max( largest, getSample(sample) )
```

```
    multiplier = 32767.0 / largest
```

```
    print “Largest”, largest, “multiplier is”, multiplier
```

```
    for sample in getSamples(sound):
```

```
        setSample(sample, getSample(sample) * multiplier)
```

PYTHON: Normalizing (modified)

```
def normalize( sound ) :  
    largest = 0  
    for sample in getSamples(sound):  
        largest = max( largest, abs( getSample(sample) ) )  
        if largest > 32766 :  
            return sound  
    multiplier = 32768.0 / largest  
    print "Largest", largest, "multiplier is", multiplier  
    for sample in getSamples(sound):  
        setSample(sample, getSample(sample) * multiplier)  
    return sound
```

Exercise - modify to handle special cond

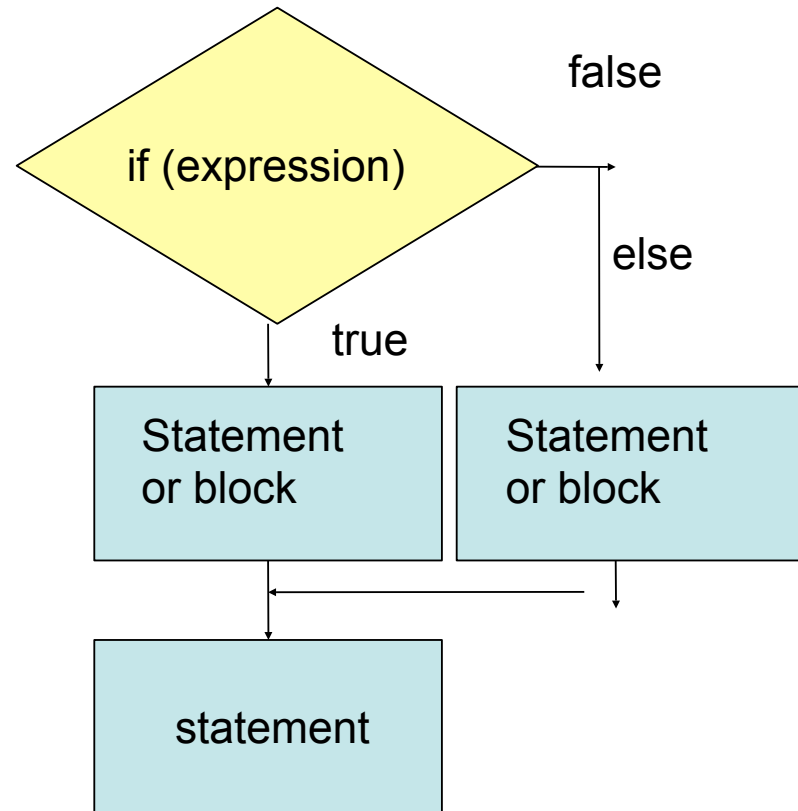
- Don't do reset values if already normalized
- Consider problem of `Math.abs(-32768) > 32767`
- Also consider that Python versions did a returned value, but this is a “class method” which operates on the object directly. *We called that “side effect” in functional programming.

Force to Extremes

- What if we want to make all values in a sound the maximum positive or negative value?
 - If the value is positive make it 32,767
 - If the value is negative make it -32,768
- We need a way to execute code based on if a test is true
 - We can use a conditional (if and else)

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();
```

To Create a Sound Clip

- Create a new Sound object
 - Of the appropriate size
 - Ending value – starting value + 1
- Loop from start to end (inclusive)
 - for (int x = start; x <= end; x++)
- Use `getSampleValueAt(index)`
 - And `setSampleValueAt(index,value);`
- Return the new sound object

Clip Method

```
public Sound clip(int start,
                  int end)
{
    // calc the num samples
    int numSamples =
        end - start + 1;
    Sound target =
        new Sound(numSamples);
    int value = 0;
    int targetIndex = 0;

    // copy from start to end
    for (int i = start; i <= end;
         i++, targetIndex++)
    {
        value = this.getSampleValueAt(i);
        target.setSampleValueAt(targetIndex,
                                value);
    }
    return target;
}
```

Based on slides by Barb Ericson,
Georgia Institute of Technology

Testing the Clip Method

```
String file = FileChooser.getMediaPath(
    "thisisatest.wav");
Sound s = new Sound(file);
Sound s2 = s.clip(0,8500);
s2.write(
    FileChooser.getMediaPath("this.wav"));
s2.play();
```

Challenge

- Create a clip of “is” from thisisatest.wav
- Determine where to start and end the clip
- Create the clip
- Write it to a file

Returning a Value from a Method

- To return a value from a method
 - Include a return statement in the body of the method
 - The type of the thing being returned must match the declared return type
 - The clip method declared that it returned a Sound object
 - The return statement returned the target Sound object
 - If the types don't match you will get a compile error

Splicing Sounds Together

- Originally meant cutting the sound tape into segments and then assembling them in the right order
- Easy to do digitally
- Copy more than one sound into a target sound
 - Track the source index and target index

Splice Method

```
public void splice()
{
    Sound sound1 = new Sound(FileChooser.getMediaPath("guzdial.wav"));
    Sound sound2 = new Sound(FileChooser.getMediaPath("is.wav"));
    int targetIndex = 0; // the starting place on the target
    int value = 0;

    // copy all of sound 1 into the current sound (target)
    for (int i = 0;
        i < sound1.getLength();
        i++, targetIndex++)
    {
        value = sound1.getSampleValueAt(i);
        this.setSampleValueAt(targetIndex, value);
    }
}
```

Based on slides by Barb Ericson,
Georgia Institute of Technology

Splice Method - Continued

```
// create silence between words by setting values to 0
for (int i = 0; Notice that targetIndex is not initialized ! I starts from its last value.
    i < (int) (this.getSamplingRate() * 0.1);
    i++, targetIndex++) {
    this.setSampleValueAt(targetIndex,0);
} Also notice that i is declared and initialized each time. Why? (Because it is
  "scoped" for just that block and forgotten for the next for loop block.)
// copy all of sound 2 into the current sound (target)
for (int i = 0;
    i < sound2.getLength();
    i++, targetIndex++) {
    value = sound2.getSampleValueAt(i);
    this.setSampleValueAt(targetIndex,value);
}
}
```

Based on slides by Barb Ericson,
Georgia Institute of Technology

Testing the Splice Method

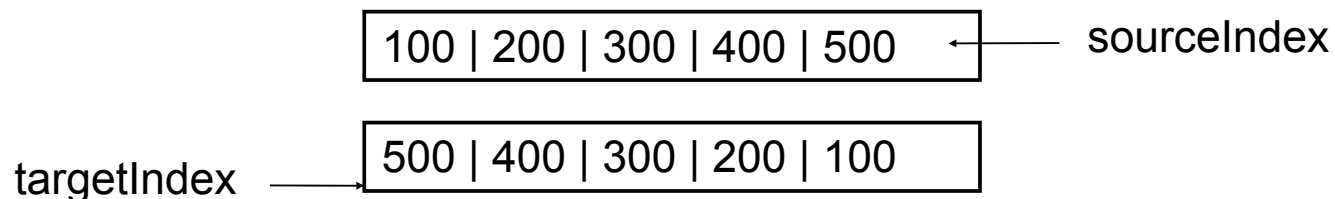
```
String silence = FileChooser.getMediaPath(  
    "sec3silence.wav");  
Sound target = new Sound(silence);  
target.explore();  
target.splice();  
target.explore();
```


Create an Audio Sentence Exercise

- Create a method that splices 3 sounds together to finish the sentence, “Guzdial is”.
 - Make sure that you don’t copy past the end of the current sound
 - Be sure to include silence between words
 - Can you make this method more general?
- How about a method to splice a sound into the middle of another sound?
 - Take starting point in target for splice

Reversing a Sound

- To reverse a sound
 - Create a copy of the original sound
 - `Sound orig = new Sound(this.getFileName());`
 - Then loop starting the `sourceIndex` at the last index in the source and the `targetIndex` at the first index in the target
 - Decrement the `sourceIndex` each time
 - Increment the `targetIndex` each time



based on slides by Barb Ericson,
Georgia Institute of Technology

Reversing Method

```
public void reverse()
{
    Sound orig = new Sound(this.getFileName());
    int length = this.getLength();

    // loop through the samples
    for (int targetIndex = 0, sourceIndex = length - 1;
        targetIndex < length && sourceIndex >= 0;
        targetIndex++, sourceIndex--)
        this.setSampleValueAt(targetIndex,
                               orig.getSampleValueAt(sourceIndex));
}
```

based on slides by Barb Ericson,
Georgia Institute of Technology

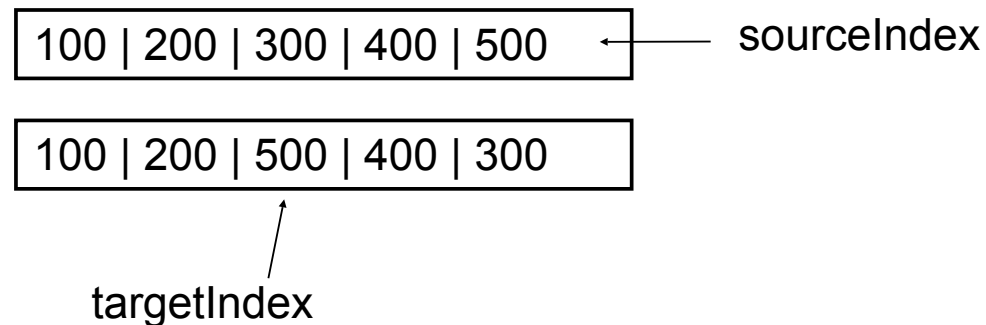
Testing the Reverse Method

```
String file = FileChooser.getMediaPath(  
    "thisisatest.wav");  
Sound s = new Sound(file);  
s.explore();  
s.reverse();  
s.explore();
```

based on slides by Barb Ericson,
Georgia Institute of Technology

Reverse Part of a Sound Exercise

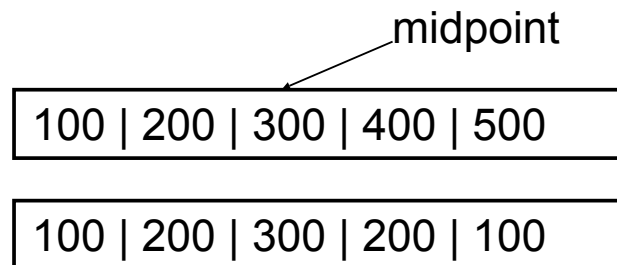
- Reverse just the second half of a sound
 - Start the `targetIndex` at the `length / 2`
 - Start the `sourceIndex` at the `length - 1`
 - Loop while the `targetIndex < length`



based on slides by Barb Ericson,
Georgia Institute of Technology

Mirror a Sound

- Copy the first half of the sound to the second half
 - And reverse the sounds in the second half
 - This is very similar to mirroring a picture
 - Calculate the midpoint ($\text{length} / 2$)
 - Start the source index at 0 and copy from index to $\text{length} - \text{index} - 1$
 - While $\text{index} < \text{midpoint}$



based on slides by Barb Ericson,
Georgia Institute of Technology

Mirror Sound Method

```
public void mirrorFrontToBack()
{
    int length = this.getLength(); // save the length
    int mirrorPoint = length / 2; // mirror around this
    int value = 0; // hold the current value

    // loop from 1 to mirrorPoint
    for (int i = 0; i < mirrorPoint; i++)
    {
        value = this.getSampleValueAt(i);
        this.setSampleValueAt(length - i - 1, value);
    }
}
```

based on slides by Barb Ericson,
Georgia Institute of Technology

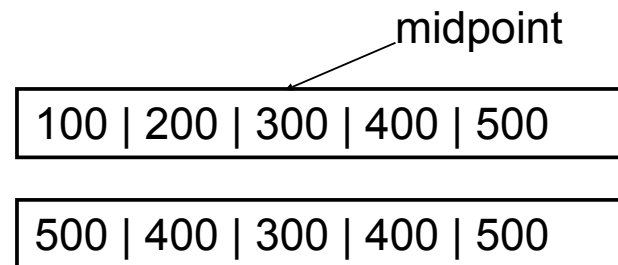
Testing Mirror Method

```
Sound s = new  
    Sound(FileChooser.getMediaPath(  
        "croak.wav"));  
s.explore();  
s.mirrorFrontToBack();  
s.explore();
```

based on slides by Barb Ericson,
Georgia Institute of Technology

Mirror Back to Front Exercise

- Write a method to mirror from the back to the front
 - Copy the back half of the sound reversed to the front



based on slides by Barb Ericson,
Georgia Institute of Technology

Blend Sounds

- Like blending pictures we can blend two sounds:
 - Copy the first 20,000 values of sound1
 - Copy from both by adding $.5 * \text{sound1 value}$ and $.5 * \text{sound2 value}$
 - Copy the next 20,000 values of sound 2

Blend Sounds Method

```
public void blendSounds()
{
    Sound sound1 =
        new Sound(FileChooser.getMediaPath("aah.wav"));
    Sound sound2 =
        new Sound(FileChooser.getMediaPath("bassoon-
c4.wav"));
    int value = 0;

    // copy the first 20,000 samples from sound1 into
    target
    for (int index=0; index < 20000; index++)
        this.setSampleValueAt(index,
            sound1.getSampleValueAt(index));
```

based on slides by Barb Ericson,
Georgia Institute of Technology

Blend Sounds - Continued

```
// copy the next 20,000 samples from sound1 and blend that
// with the first 20,000 samples from sound2
for (int index = 0; index < 20000; index++)
{
    value = (int) ((sound1.getSampleValueAt(index + 20000) *
        0.5) +
        (sound2.getSampleValueAt(index) * 0.5));
    this.setSampleValueAt(index + 20000,value);
}

// copy the next 20,000 samples from sound2 into the target
for (int index=20000; index < 40000; index++)
    this.setSampleValueAt(index + 20000,
        sound2.getSampleValueAt(index));
}
```

based on slides by Barb Ericson,
Georgia Institute of Technology

Testing Blend Sounds

```
String fileName =  
    FileChooser.getMediaPath(  
        "sec3silence.wav");  
Sound target = new Sound(fileName);  
target.explore();  
target.blendSounds()  
target.explore();
```

based on slides by Barb Ericson,
Georgia Institute of Technology

Modify Blend Sounds Exercise

- Create another blendSounds method
 - That takes the file name of the sounds to blend
 - And a value to start the blend at and another to stop the blend at
 - Modify the original blendSounds method to call this one

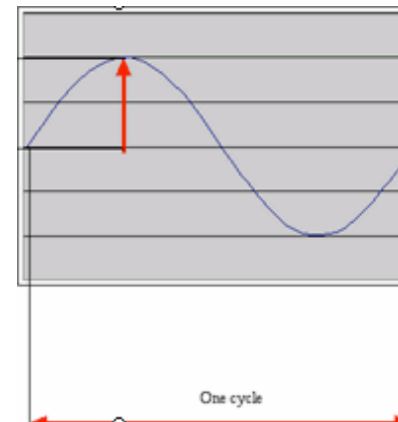
Overloading Methods

- You can have several methods with the same name
 - As long as the parameter list is different
 - In number of parameters
 - And/or types of parameters
 - blendSounds()
 - blendSound(String name1, String name2, int startBlend, int endBlend)

NOTE: This is different from Python where the name of the function or method is on the part before the “(“.

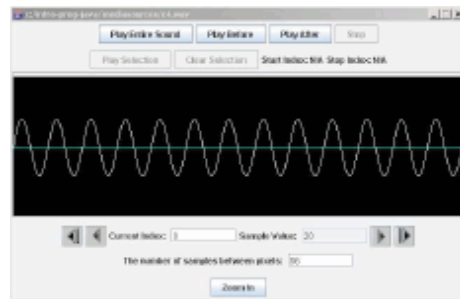
Changing the Sound Frequency

- The *frequency* of a wave is the number of cycles per second (cps), or *Hertz (Hz)*
 - (Complex sounds have more than one frequency in them.)
- 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

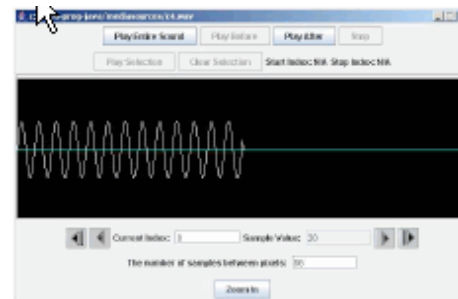


Double the Frequency

- If we take every other sample we double the frequency of the sound
 - Completes two cycles instead of one in the same time
 - It will sound higher



100 | 200 | 300 | 400 | 500



100 | 300 | 500 | 0 | 0

Double Frequency Method

```
public void doubleFreq()
{
    // make a copy of the original sound
    Sound s = new Sound(this.getFileName());

    /* loop and increment target index
     * by one but source index by 2,
     * and set target value
     * to the copy of the original sound
     */
}
```

Double Frequency - Continued

```
for (int sourceIndex=0, targetIndex = 0;
    sourceIndex < this.getLength();
    sourceIndex=sourceIndex+2, targetIndex++)
    this.setSampleValueAt(targetIndex,
                          s.getSampleValueAt(sourceIndex));

// clear out the rest of this sound to silence (0)
for (int i = this.getLength() / 2;
    i < this.getLength();
    i++)
    this.setSampleValueAt(i,0);
}
```

Test Double Frequency

```
Sound s = new  
    Sound(FileChooser.getMediaPath(  
        "c4.wav"));  
s.explore();  
s.doubleFreq();  
s.explore();
```

Challenge

- Create a method that will take every third sample
 - Will this sound higher or lower?
- Can you make this method more general?
 - By passing in the amount to add to the source index each time?

Halving the Frequency

- We can copy each source value twice to half the frequency
 - Only get through half a cycle in the same time we used to get through a full cycle
 - It will sound lower
- This is the same algorithm that we used to scale up a picture

100 | 200 | 300 | 400 | 500

100 | 100 | 200 | 200 | 300

Halve Frequency Method

```
public void halveFreq()
{
    // make a copy of the original sound
    Sound s = new Sound(this.getFileName());

    /* loop through the sound and increment target index
    * by one but source index by 0.5 and set target value
    * to the copy of the original sound
    */
    for (double sourceIndex=0, targetIndex = 0;
        targetIndex < this.getLength();
        sourceIndex=sourceIndex+0.5, targetIndex++)
        this.setSampleValueAt((int) targetIndex,
            s.getSampleValueAt((int) sourceIndex));
}
```

Notice that “for” loops do NOT require “++” !

Testing Halve Frequency

```
Sound s = new  
    Sound(FileChooser.getMediaPath(  
        "c4.wav"));  
s.explore();  
s.halveFreq();  
s.explore();
```


Change Frequency Exercise

- Write a method that will copy each sound value 4 times to the target
 - Will the new sound be higher or lower?
- Can you make this more general?
 - By passing in the number of times to copy the source value
 - Try it with 3 times and check the index values to make sure that you are doing it right

Last Homework Project - “Ohce”

- Due Thursday December 11 @ 10 am
- Echo backward - the echo gets progressively louder until the actual sound is heard. (Or, just like an echo only before instead of after.)

Coming Attractions

- For Friday:
 - final code for group projects due 4:20 PM
 - upload

	1	2	3	4	5	6	7	8
A	Burton	Talley	Davies	Bowers	Demase	Burke	Currin	Thayer
	Highman	D'Augustine	Taylor	Knowles	Ho	Maier	Malhotra	Heitzer
B	Regione	Zhang	Howell	Ha	Tran	Roithmayr	Pham	Walsh
	Messick	Rhyner	Nassery	Dahiya	Duffy	Merrow	Slack	Hughes