• Reading for next class: 11.1-11.8
  – Class construction steps and concepts
• Today’s topics:
  – More Sounds!
• Start PSA 8!
1. The **splice method from the book** is provided below:

```java
/*Method to copy part of the passed sound into this sound at given start index*/
public void splice (Sound, source, int sourceStart, int sourceStop, int targetStart)
{
    //loop copying from source to target
    for(int sourceIndex = sourceStart, targetIndex = targetStart;
        sourceIndex < sourceStop && sourceIndex < this.getLength();
        sourceIndex++, targetIndex++)
    {
        this.setSampleValueAt(targetIndex,
            source.getSampleValueAt(sourceIndex));
    }
}
```

There seems like one parameter might be missing. Why is there no parameter: targetStop -- the index you want to stop copying into in the target sound?

A. We always copy up to the end of the target (calling object) sound, so we don't need a targetStop
B. This is an error, there should be a targetStop parameter
C. The targetStop index is implicitly defined based on the targetStart and the (sourceStop-sourceStart)
D. targetStop is the same as sourceStop, so we don't have to pass it separately
public void mirrorFrontToBack()
{
    int length = this.getLength();
    int mirrorPoint = length / 2;
    int value = 0;

    for(int i = 0; i < mirrorPoint; i++)
    {
        value = this.getSampleValueAt(i);
        this.setSampleValueAt(____________, value);
    }
}

What should go in the blank to complete this method?

A. length
B. length - 1
C. length - i
D. length - 1 - i

D. length - 1 - i
3. How does sound mirroring process differ from picture mirroring process?

A. It doesn’t, they both “copy” values from one side of the mirror point to the other side

B. When mirroring a sound you don’t use a mirror point

C. When mirroring an image you don’t use a mirror point
4. What is the goal of the research described in the video?

A. Design and build computer programs to measure the quality of a song.

B. Design and build computer programs that automatically classify music files to for example build better recommendation systems.

C. Design and build computer programs to balance the volume of all the songs automatically.

D. Design and build computer programs to measure the duration of all our songs.
Exam 3. Problem 1

Part of a method in the Picture class

```java
for ( int x = 0; x < this.getWidth(); x++ ) {
    for ( int y = 0; y < this.getHeight(); y++ ) {
        if ( y < 2 || y >= 8 ) {
            this.getPixel(x, y).setColor( new Color(255, 255, 255) );
        }
        else if ( x < 2 || x >= this.getWidth()-2 ) {
            this.getPixel(x, y).setColor( new Color(0, 0, 0) );
        }
    } // end of y loop
} // end of x loop
```

What is wrong with the if-condition above (the red text)?

A. It will not work for pictures with height != 10
B. The || should be &&
C. The >= should be >
D. You need to make sure y < this.getHeight()
E. Nothing is wrong
Part of a method in the Picture class

```java
for ( int x = 0; x < this.getWidth(); x++ ) {
    for ( int y = 0; y < this.getHeight(); y++ ) {
        if ( y < 2 || y >= this.getHeight()-2 ) {
            this.getPixel(x, y).setColor( new Color(255, 255, 255) );
        } else if ( (x < 2 || x >= this.getWidth()-2) && (y>=2 && y<this.getHeight()-2) ) {
            this.getPixel(x, y).setColor( new Color(0, 0, 0) );
        }
    } // end of y loop
} // end of x loop
```

Which is true about the line in red above?
A. It must be removed in order for the code to work
B. It is fine to include, but removing it will also work
C. It must be included for the code to work
D. It causes a compile error
Picture source = new Picture("tritons.jpg"); //source image
Picture target; //new image
target=source.zoomIn_TopRight(); //target gets assigned the returned Picture

source: 

target (after call to zoomIn_TopRight):

In the source picture, highlight (shade) the Pixel that should be copied to the location (2, 3) in the target in order to create the stretched picture correctly and write the coordinates of that source pixel in the blank below.

Pixel with coordinates (2, 3) in target
Coordinates of pixel from source that will be copied to position (2, 3) in target: __(4,1)______________
public Picture zoomIn_TopRight()
{
    Picture zoomedPic = new Picture(this.getWidth(), this.getHeight()); // or this
    Pixel sourcePix, targetPix;
    int xTarget, yTarget;
    double xSource, ySource;

    for ( xSource = (this.getWidth()/2), xTarget = 0;
         xTarget<this.getWidth(); xSource = xSource+0.5, xTarget = xTarget+1 ) {
        for ( ySource = 0, yTarget = 0;
             yTarget<this.getHeight(); ySource = ySource+0.5, yTarget = yTarget+1 ){
            //Access the "source" and "target" pixels:
            sourcePix = this.getPixel( (int) xSource, (int) ySource);
            // OTHER OPTION
            // sourcePix = this.getPixel(this.getWidth()/2+xTarget/2, yTarget/2);
            targetPix = zoomedPic.getPixel(xTarget,yTarget);

            // Assign source pixel color to target pixel
            targetPix.setColor(sourcePix.getColor());
        }
    }

    return zoomedPic;
}
The Sample Rate that the Sound class ASSUMES is 22 KHz:
How long is a SoundSample[] in a Sound object of 1 second?

A. 22 elements
B. 11,000 elements
C. 22,000 elements
D. 44,000 elements
E. We can’t tell from the data provided
Frequency and Pitch of a Sound

http://www.youtube.com/watch?v=ngk-ECb8ccQ
What's printed by this code?
(assume calling object as shown)

A. 0,9
B. 60,0
C. 90,5
D. 100,4
E. None of the above
How would you describe what this code is doing?

```java
int a=0, b=0, foo=0;
for (int i=0; i<noiseArray.length; i++)
{
    SoundSample sample = noiseArray[i];
    foo = sample.getValue();
    if (foo > a)
    {
        a = foo;
        b = i;
    }
}
```
public void normalize()
{
    SoundSample[] noiseArray = this.getSamples();
    int maxVal, maxIndex = 0;
    for (int i=0; i<noiseArray.length; i++)
    {
        SoundSample sample = noiseArray[i];
        int val = sample.getValue();
        if (val > maxVal)
        {
            maxVal = val;
            maxIndex = i;
        }
    }
    double factor = 32767.0 / maxVal;
    for (int i = 0; i < noiseArray.length; i++)
    {
        SoundSample sample = noiseArray[i];
        sample.setValue((int) (sample.getValue() * factor));
    }
}
public void normalize()
{
    SoundSample[] noiseArray = this.getSamples();
    int maxVal, maxIndex = 0;
    for (int i=0; i<noiseArray.length; i++)
    {
        SoundSample sample = noiseArray[i];
        int val = sample.getValue();
        if (val > maxVal)
        {
            maxVal = val;
            maxIndex = i;
        }
    }
    double factor = 32767.0 / maxVal;
    for (int i = 0; i < noiseArray.length; i++)
    {
        SoundSample sample = noiseArray[i];
        sample.setValue((int) (sample.getValue() * factor));
    }
}
public int findMax(SoundSample[] noiseArray) {
    int maxVal;
    for (int i=0; i<noiseArray.length; i++) {
        SoundSample sample = noiseArray[i];
        int val = sample.getValue();
        if (val > maxVal) {
            maxVal = val;
        }
    }
    return maxVal;
}

public void normalize() {  
    SoundSample[] noiseArray = this.getSamples();
    int maxVal = findMax(noiseArray);
    double factor = 32767.0 / maxVal;
    for (int i = 0; i < noiseArray.length; i++) {
        SoundSample sample = noiseArray[i];
        sample.setValue((int) (sample.getValue() * factor));
    }
}
public void normalize() {
    SoundSample[] noiseArray = this.getSamples();
    int maxVal = findMax(noiseArray);
    double factor = 32767.0 / maxVal;
    for (int i = 0; i < noiseArray.length; i++)
    {
        SoundSample sample = noiseArray[i];
        sample.setValue((int)(sample.getValue() * factor));
    }
}
public void normalize() {

    final double maxAllowedAmplitude = 32767.0;

    SoundSample[] noiseArray = this.getSamples();
    int maxVal = findMax( noiseArray );
    double factor = maxAllowedAmplitude / maxVal;
    for (int i = 0; i < noiseArray.length; i++)
    {
        SoundSample sample = noiseArray[i];
        sample.setValue((int) (sample.getValue() * factor));
    }
}

“final” means this is a “variable” whose value can never be changed. You must declare and assign final variables a value on one line.
Changing Pitch of a Sound (test at home)

• Play a recording of someone reading a sentence

• Now play it so that it sounds “high pitched”
  – How long does it take to play the sound high pitched, compared to how long the original takes?

• Now play it so that it sounds “low pitched”
  – How long does it take to play the sound low pitched, compared to how long the original takes?
Raise the pitch of a Sound

- Take only every nth sample from the original sound.
- The length of the sound sample is $1/n$ of the original, and all frequencies in the sound have been increased by a factor of n.
- Example, with $n=2$:
Options to raisePitch

• Create new Sound
  – V1) Of exact length needed for higher pitched sound
  – V2) Of same length as original with “silence” at end
Write a method as part of the Sound class that returns a new Sound object whose pitch is double the calling object and whose length is half as long. Create the new sound by taking every other sample from the calling object.

What is the method header for this method?

A. public void raisePitch(Sound s)
B. public void raisePitch()
C. public Sound raisePitch()
D. public Sound raisePitch(Sound s)
public Sound raisePitch()
{
    SoundSample[] original = this.getSamples();
    Sound highP = new Sound( original.length / 2 );
    SoundSample[] higher = highP.getSamples();
    int newPlace = 0;
    for (int origI = 0; origI < original.length; origI+=2) {
        // What object will this method return?
        // A. this
        // B. highP
        // C. original
        // D. higher
        // E. void
    }
}
public Sound raisePitch()
{
    SoundSample[] original = this.getSamples();
    Sound highP = new Sound( original.length / 2 );
    SoundSample[] higher = highP.getSamples();
    int newPlace = 0;
    for (int origI = 0; origI < original.length; origI+=2)
    {
        higher[newPlace].setValue( original[origI].getValue() )
        newPlace++;
    }

    return highP;
}
public Sound raiseP() {
    Sound highP = new Sound(this);
    SoundSample[] original = this.getSamples();
    SoundSample[] higher = highP.getSamples();
    int newPlace = 0;
    for (int origI = 0; origI < original.length; origI+=2) {

How would the code below change the SoundSample array?

// In the sound class
public void mystery() {
    SoundSample[] original = this.getSamples();
    for (int index = 0; index < original.length; index++) {
        original[index].setValue( original[index/2].getValue() );
    }
}

It causes an error
Lowering the Pitch of the Sound

// In the sound class
public void lowerPitch() {
    SoundSample[] original = this.getSamples();
    for ( int index = 0; index < original.length; index++ )
    {
        original[index].setValue( original[index/2].getValue() );
    }
}

Problem: We are overwriting the values we need to use before we have used them!
Possible solutions?

before  100  150  200  300  140  10  -40  -100  -250  -150

after   100  100  150  150  200  200  300  300  140  140
// In the sound class
public void lowerPitch() {
    SoundSample[] original = this.getSamples();

    for (int index = 0; index < original.length; index++) {
        original[index].setValue(original[index/2].getValue());
    }
}
Another name for this method...

// In the sound class
public void stretchInPlace() {
    SoundSample[] original = this.getSamples();

    for (int index = original.length - 1; index >= 0; index--)
    {
        original[index].setValue( original[index/2].getValue() );
    }
}
...that could apply to Pictures too!

Complete the code below to stretch the calling object Picture both horizontally and vertically.

// In the sound class
public void stretchInPlace() {
    SoundSample[] original = this.getSamples();
    for (int index = original.length - 1; index >= 0; index--) {
        original[index].setValue(original[index/2].getValue());
    }
}

// In the Picture class
public void stretchInPlace() {

    for (int x =
    {
        for (int y =
        {
            Pixel source = this.getPixel();
            Pixel target = this.getPixel();
            target.setColor(source.getColor());
        }
    }
}
Concept Summary

• When you want to create a “new” object…
  – Call a “constructor” with new.
  – Look in the file of that class to find out what constructors are available
    • What parameters you can send

• Don’t forget to return the object you created with a return statement!

• When working with 2 (multiple) arrays
  – Sometimes you will want 2 index variables (to index into them) moving independently
  – If you are indexing “in synchrony” then use one index variable— it’s easier to understand!
TO-DO

• Reading for next class: 11.1-11.8 (how to build a class!)
• Get started ASAP with next PSA!