Quick Exercise

At the end of this sequence,
how many different colors can the pixel be?

```
if ( p.getRed() > 127 )
   red = 255;
else
   red = 0;

if ( p.getGreen() > 127 )
   green = 255;
else
   green = 0;

if ( p.getBlue() > 127 )
   blue = 255;
else
   blue = 0;
```

Alternative 1

How about this one?

```
if ( p.getRed() > 127 || p.getGreen() > 127 || p.getBlue() > 127 )
{
   red   = 255;
   green = 255;
   blue  = 255;
}
else
{
   red = 0;
   green = 0;
   blue = 0;
}
```

... what if we change the **`||`** to **`&&`**?

Alternative 2

How about *this* one?

```
if ( p.getAverage() > 127 )
{
   red   = 255;
   green = 255;
   blue  = 255;
}
else
{
   red = 0;
   green = 0;
   blue = 0;
}
```

One Short Cut

A type in the problem left you with a hint...

```
int red   = 0;
int green = 0;
int blue  = 0;

if ( p.getRed() > 127 )
   red = 255;

if ( p.getGreen() > 127 )
   green = 255;

if ( p.getBlue() > 127 )
   blue = 255;
```

A Programming Pattern

When you make a design decision:

```
red + green < blue
```

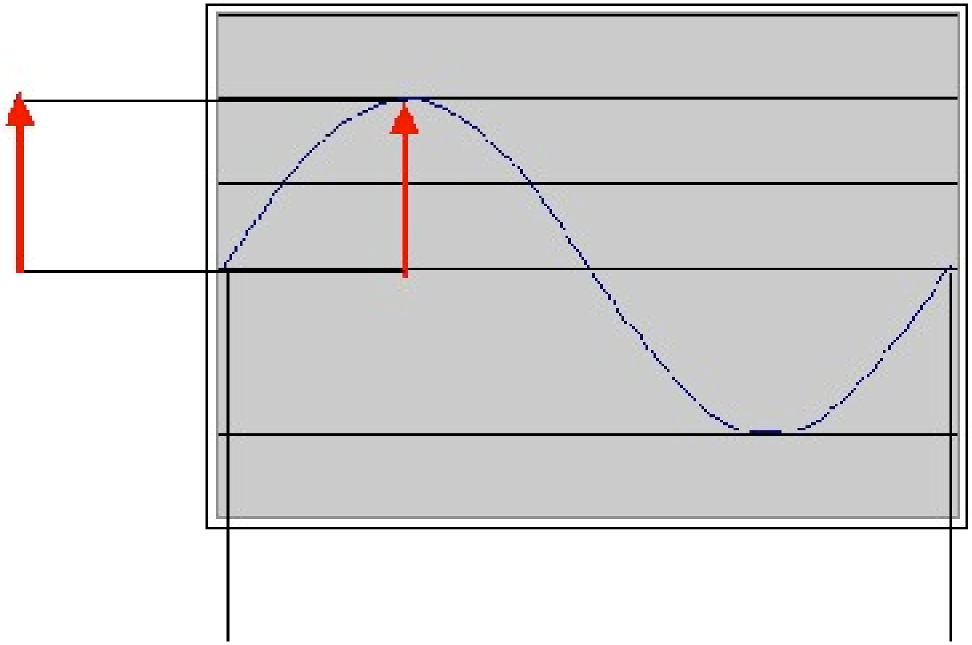
is a good way to determine if a pixel is blue

... write a method to encode the decision:

```
public void isBlue( Pixel p )
```

The Physics of Sound

amplitude



<---- one cycle --->

The Psychology of Sound

We perceive *volume* as **change in amplitude**.

If amplitude doubles, that is change of about 3 decibels (dB).

We perceive *pitch* as **change in frequency**.

We can hear between 5 Hz and 20000 Hz (20 kHz).

Logarithm Scale

Human hearing works with **ratios**, not differences.

For pitch, this means ...

200 -> 400 Hz ~ 500 -> 1000 Hz
300 -> 600 Hz ~ 1500 -> 3000 Hz

Volume on Log Scale

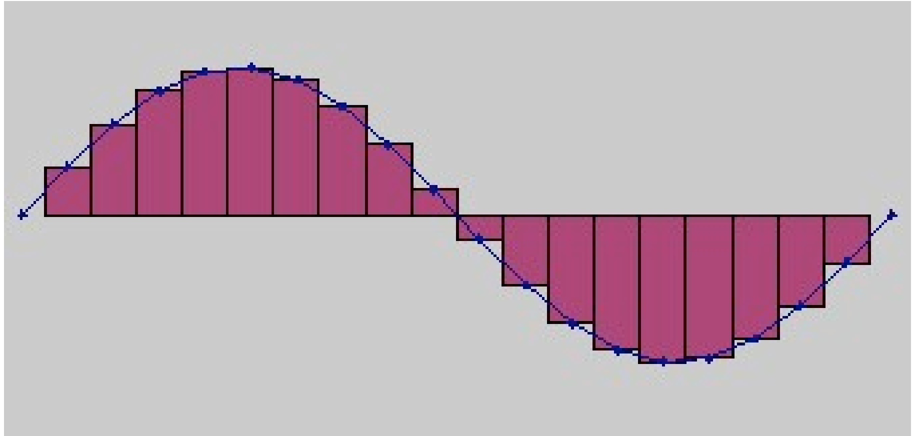
A decibels is based on the ratio between two volumes:

```
10 * log( V1 / V2 )
```

The absolute measure is in comparison to the threshold of our hearing:

0 dB cannot be heard.
60 dB is normal speech
80 dB is considered shouting

Digitizing Sound



We can estimate the area under a curve
using a sampling of rectangles.

To encode a sound,
we record the amplitude at a point in time —
the height of an implicit rectangle.

How Many Samples?

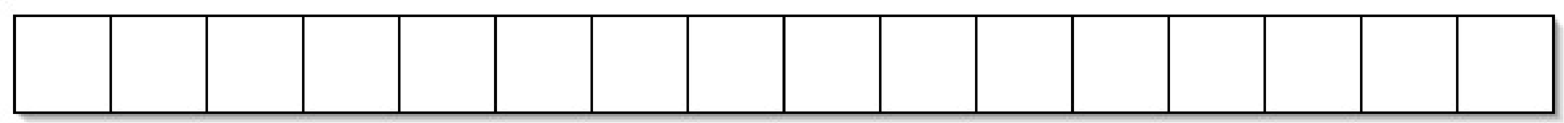
Nyquist's Theorem
To represent sounds
with a maximum frequency of n,
we need 2n samples.

Human voices max out at ~ 4 KHz.
So phones work with 8000 samples per second.

Human hearing maxes out at ~ 22 KHz.
So most digital audio works with 44,000 samples/second.

Encoding a Sample of Sound

Each sample = 2 bytes, or 16 bits.



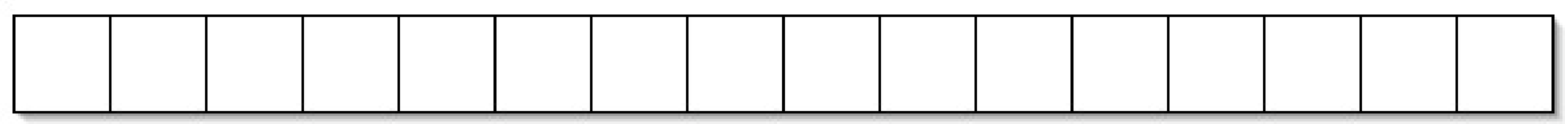
^ <------------ the rest stores the value ------------>

used
to indicate
sign

```
2^16 = 65,536    -2^8   = -32,768
                  2^8-1 =  32,767
```

Encoding a Sound

Each sound is an array of samples.



<------------ one slot for each sample ------------>

44,100 samples = 1 second of sound

Working with Sound

```
new Sound( ... )

getSamples()
getLength()
getSamplingRate()

getSampleValueAt( int slot )
setSampleValueAt( int slot, int newValue )
```