

Exam Question 2

Write a **Sound** method named **maximizeAt(int cap)** that modifies the sound so that its largest sample value is **cap**. All other samples are changed proportionally.

```
public void maximizeAt( int cap )
{
    double maximumValue = this.maximumAbsoluteValue();
    double multiplier    = cap / maximumValue;

    for ( SoundSample sample : this.getSamples() )
        sample.setValue( (int) (sample.getValue()
                                * multiplier) );
}
```

Exam Question 4

Write a **Sound** method named ***reverseRange(int start, int end)***, which reverses seconds ***start*** through ***end*** of the sound.

```
double samplingRate = this.getSamplingRate();
int    startSample  = (int) (start * samplingRate);
int    endSample    = (int) (end   * samplingRate);

int    rangeLength  = endSample - startSample;
int    midPoint     = startSample + rangeLength/2;

for ( int i = startSample; i < midPoint; i++ )
{
    int sampleFront = this.getSampleValueAt( i );
    int sampleBack  = this.getSampleValueAt( endSample - i );

    this.setSampleValueAt( i, sampleBack );
    this.setSampleValueAt( endSample - i, sampleFront );
}
```

Exam Question 5c

*Under what conditions, if any,
can our **DiffSound** compression be lossless?*

```
public class DiffSound
{
    private int firstSample;
    private byte[] differences;
    ...
}
```

As long as the differences fit in a byte!

Exam Question 6

Write a **Sound** method named **playBlended(Sound trailer)** that plays the first half of the sound, then the second half of the sound added to the beginning of **trailer**, and finally the rest of **trailer**.

```
int midPoint = this.getLength()/2;
for ( int i = 0; i < midPoint; i++ )
{
    result.setSampleValueAt( i, this.getSampleValueAt(i) );
}

for ( int i = 0; i < midPoint; i++ )
{
    int sum = this.getSampleValueAt(midPoint+i) +
              trailer.getSampleValueAt(i);
    result.setSampleValueAt( midPoint+i, sum );
}

for ( int i = 0; i < trailer.getLength()-midPoint; i++ )
{
    result.setSampleValueAt(
        this.getLength()+i,
        trailer.getSampleValueAt(midPoint+i) );
}
```

Exam Question 8c

*What does the keyword **static** mean when we declare a field? A method?*

In Java, `static` does not mean "unchanging".

It means "no instance required".

A `static` field belongs to the class.

A `static` method is executed via the class.

Exam Question 9

... Write a piece of Java code that determines how popular "Eugene" was that year.

```
String[] rankedNames = allNames.split( " " );
for (int i = 0; i < rankedNames.length; i++)
{
    if ( name.equals( rankedNames[i] ) )
        return i + 1;
}
return -1;
```


