Sampled Sounds

Homework 5

Due: Friday Feb 26th, 2010 before 6pm

This is a pair programming problem! You are expected to work with the person you have been pared with in class, and you are both responsible for submitting the exact same code to T-Square.

You may collaborate with other students in this class. Collaboration means talking through problems, assisting with debugging, explaining a concept, etc. You should not exchange code or write code for others. For pair programming assignments, you and your partner should turn in identical assignments. Collaboration at a reasonable level will not result in substantially similar code to another pair.

Homework 5 – Sounds

For this homework, you will be adding one static method and two object methods to the Sound class (You will need to edit the Sound.java file in java-sources, and when you are done, you will need to submit the Sound.java file. Make sure you add a comment with your name and collaboration statement to the top of the file). The static method will produce a triangle wave while the object methods will carry out sound manipulations.

Triangle Wave

Your triangle method should return a Sound object and take in a double for the time (in seconds) and an integer for the wavelength (in number of samples). Here is a syntax example:

```
Sound saw = Sound.triangle(1.5, 400);
```

In this example, you should create a sound that is 1.5 seconds long and peaks every 400 samples. Note that the Sound class uses 22,050 samples per second. This means that the sound would start at an amplitude of -32000, go up to 32000 over the first 200 samples, and then drop back down to -32000 over the next 200 samples. The triangle wave should go from -32000 to 32000 and back in a linear manner, using uniform steps.

For further details see: http://en.wikipedia.org/wiki/Triangle_wave

Inverse

When called on a Sound object, your inverse method should return a new sound with an inverted phase. Here is a syntax example (note snd is an existing Sound object):

```
Sound inv = snd.inverse();
```

The inverse should behave such that silence is produced when equally mixing a sound and its inverse. Example:

```
Sound mix = snd.mix(inv,0.5);
```

For further details see: http://en.wikipedia.org/wiki/Sound_cancellation

BlankOut

When called on a Sound object, your blankOut method should return a new sound with silence beginning after some duration (passed in as seconds in the first argument) and continuing for as many seconds as the second input (both doubles). This should overwrite whatever information was present at this point. Here is a syntax example (note: snd is an existing Sound object):

```
Sound ext = snd.blankOut(1.0, 0.5);
```

In this example, ext would sound exactly like snd except after the first second of play, half a second of silence should occur. Your code must ensure that the original sound object is long enough to accommodate the first input (i.e. the sound must already exist when the silence begins, but your silence may carry over beyond the original sound's duration). If the original sound is not long enough for the silence to carry over, you must extend the length of your new sound. If the original sound is not long enough to accommodate the first input, return null instead.

Grading Criteria

Triangle

- Header correct 10pt
- Produces a sound of the correct length 10pt
- Sound is a triangle wave 20pt

Inverse

- Header correct 5pt
- Produces a sound of the correct length 5pt
- Sound is the inverse of the source 10pt

BlankOut

- Header correct 5pt
- Silence begins after the correct duration 15pt
- Produces silence of the correct length 20pt

Total: 100pts

Late Penalty (-10pt)

No Collaboration Statement (-100pt)