reduceVolume ( )

Takes an array of integers representing a sound and returns a new array of integers representing the same sound, just quieter by a factor of 10.

```
class SoundExamples {

  static int[] reduceVolume (int[] sound) {

    int[] soundQuiet = new int[                ];

    for(int i =            ;                ; i += 1) {

      soundQuiet[i] =

    }
    return soundQuiet;
  }

  static int[] firstHalf(int[] sound) {

    int[] halfSound = new int[                ];

    for(int i =            ;                ; i += 1) {

      halfSound[i] =

    }
    return halfSound;
  }

  static int[] faster(int[] sound) {

    int[] fasterSound = new int[                ];

    for(int i =            ;                ;              ) {

      fasterSound[i] =

    }
    return fasterSound;
  }


  public static void main(String[] args) {
    int[] beforeQuiet = {220, -1000, 40};
    int[] expect = {22, -100, 4};
    int[] quieted = soundQuiet(startSound);
    System.out.println("Expected:\t " + Arrays.toString(expect));
    System.out.println("Actual:\t " + Arrays.toString(quieted));

    int[] beforeHalf = {4, 5, -3, 2, 1, 6};
    int[] halved = firstHalf(beforeHalf);
    int[] exHalf = {4, 5, -3};
    System.out.println("Expected:\t " + Arrays.toString(exHalf));
    System.out.println("Actual:\t " + Arrays.toString(halved));

    int[] beforeFast = {4, 5, -3, 2, 1, 6};
    int[] fasted = faster(beforeFast);
    int[] exFast = {4, -3, 1};
    System.out.println("Expected:\t " + Arrays.toString(exFast));
    System.out.println("Actual:\t " + Arrays.toString(fasted));




  }

}
```

SoundExamples.java

```
$ javac -cp lib/*:. SoundExamples.java
$ java –cp lib/*:. SoundExamples
```

int[]	**CSE8ALib.readSound**(**String** path) Takes a path to a file expected to be in .wav format and produces an int array representing the sound recorded in that file.
boolean	**CSE8ALib.play**(int[] sound) Takes an int array representing a sound and plays it (using the computer's speakers / headphones), returns true if the operation was performed successfully and false otherwise.
boolean	**CSE8ALib.explore**(int[] sound) Takes an int array representing a sound and opens a window that displays the sound waveform along with sampled values, returns true if the operation was performed successfully and false otherwise.

`<array>[<index>] = <value>`

Array update or *Array assignment*:
Updates the array on the heap **referenced by** the <array> expression to have the given <value> at <index>.

`new <type>[<size>]`

Creates a new array on the heap of the given size, and returns a reference to it. The size can be any expression that evaluates to an int.

`<type>[] <name> = { <e1>, <e2>, ... };`

Creates an **array** and stores it in the variable <name>.

All elements e1, e2, must have the given type.

`<array>[<index>]`

Array lookup:

Arrays can be indexed as in Python. <index> should evaluate to an int, and <array> to an array value. Indices start at 0.

Array Creation Template

You can get much more creative than this, but this template describes a lot of methods that create and return new arrays based on their argument values.

```
int[] <methodName> (...) {
  int size = <decide on size>;
  int[] newArray = new int[size];
  for(int i = 0; i < size; i += 1) {
    newArray[i] = <expression to compute value based on params and i>;
  }
  return newArray;
}
```