Homework Turnin

Name:	Xuqing Wu
Account:	xw88 (xw88@uw.edu)
Student ID:	1933202
Section:	AD
Course:	CSE 143 20wi
Assignment:	a2
Receipt ID:	f54b342cf2a55ab127c2e0ec628b496f

Turnin script completed with output:

Note: support/StdAudio.java uses or overrides a deprecated API. Note: Recompile with -Xlint:deprecation for details.

Turnin Successful!

The following file(s) were received:

Guitar37.java (2149 bytes, sha256: ba10d123538e716ff4684af52392a5e3)

```java
// Xuqing Wu
// 1/22/2020
// CSE143
// TA: Eric Fan
// Assignment #2
//
// Class Guitar37 keeps track of a musical instrument
// with 37 strings. It implements guitar class to
// process pitch and frequency and play the tune.

public class Guitar37 implements Guitar {
   public static final String KEYBOARD =
      "q2we4r5ty7u8i9op-[=zxdcfvgbnjmk,.;/' ";  // keyboard layout
   public static final int TOTAL = 37;
   private GuitarString[] stringAll;   //construct the array to store guitar string
   private int num;   //number of times tic has been called

   //post: create 37 guitar strings with different frequency
   public Guitar37() {
      stringAll = new GuitarString[TOTAL];
      for(int i = 0; i < TOTAL; i++) {
         stringAll[i] = new GuitarString(440.0 * Math.pow(2, (i - 24) / 12.0));
      }
   }

   //post: specify which note to play by passing a pitch
   //ignore pitch if it cant be played
   public void playNote(int pitch) {
      int i = pitch + 24;
      if(i >= 0 && i < TOTAL) {
         stringAll[i].pluck();
      }
   }

   //post: return true if the character passed has a
   //corresponding string
   public boolean hasString(char string) {
      return KEYBOARD.indexOf(string) != -1;
   }

   //pre: the char given is contained in the string KEYBOARD
   //post: indicates which note to play by processing the character passed
   public void pluck(char string) {
      if(! hasString(string)) {
         throw new IllegalArgumentException();
      }
      for(int i = 0; i < TOTAL; i++) {
         if(string == KEYBOARD.charAt(i)) {
            stringAll[i].pluck();
         }
      }
   }

   //post: return the sum of all samples from the strings
   public double sample() {
      double all = 0.0;
      for(int i = 0; i < TOTAL; i++) {
         all += stringAll[i].sample();
      }
      return all;
   }

   //post: advance the time forward one tic
   public void tic() {
      for(int i = 0; i < TOTAL; i++) {
         stringAll[i].tic();
      }
      num++;
   }

   //post: returns the number of times tic has been called
   public int time() {
      return num;
   }
}
```

GuitarString.java (2137 bytes, sha256: ee419004605514acc9a0f8e3f0ec104c)

```java
// Xuqing Wu
// 1/22/2020
// CSE143
// TA: Eric Fan
// Assignment #2
//
// Class GuitarString is used to models a vibrating guitar string
// of a given frequency by keeping track of a ring buffer

import java.util.*;
public class GuitarString {
   public static final double DECAY_FACTOR = 0.996;
   private Queue<Double> q;   //ring buffer
   private int N;   //capacity of ring buffer

   //pre: right frequency is passed and the size is appropriate
   //(throw IllegalArgumentException if not)
   //post: Constructs a ring buffer of the given frequency
   public GuitarString(double frequency) {
      N = (int)Math.round(StdAudio.SAMPLE_RATE / frequency);
      if(frequency <= 0 || N < 2) {
         throw new IllegalArgumentException();
      }
      q = new LinkedList<>();
      for(int i = 0; i < N; i++) {
         q.add(0.0);
      }
   }

   //pre: array passed has more than one element
   //(throw IllegalArgumentException if not)
   //post: Constructs a ring buffer and put the values
   //in the given array into the ring buffer
   public GuitarString(double[] init) {
      if(init.length < 2) {
         throw new IllegalArgumentException();
      }
      q = new LinkedList<>();
      for(int i = 0; i < init.length; i++) {
         q.add(init[i]);
      }
   }

   //post: fill the ring buffer with random values
   //between -0.5 inclusive and +0.5 exclusive
   public void pluck() {
      Random r = new Random();
      double element = 0.0;
      int time = 0;
      while(time < N) {
         element =  r.nextDouble() - 0.5;
         q.remove();
         q.add(element);
         time++;
      }
   }

   //post: delete the sample at the front of the ring buffer
   //and add value to the end of the ring buffer, the value
   //added is calculated through a list of calculations
   public void tic() {
      double removed = q.remove();
      double next = q.peek();
      double added = (removed + next) / 2 * DECAY_FACTOR;
      q.add(added);
   }

   //post: return the value at the front of the ring buffer
   public double sample() {
      return q.peek();
   }
}
```