



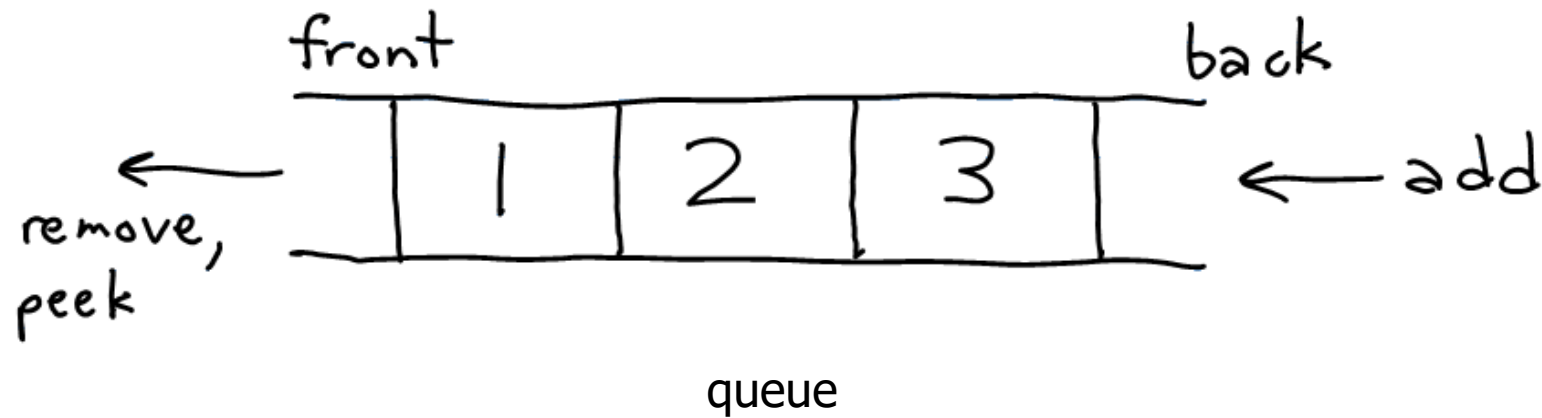
# Guitar Hero

# Goal

- Simulate the vibration of a string and produce actual sound wave
- Simulate a guitar you can play with keyboard
- Enhance your understand of queues, interfaces, objects and arrays of objects
- Learn about efficient data structures that are crucial for application performance



# Queues



- FIFO (First In First Out)
- Elements are stored in order of insertion but don't have indexes.

# Programming with Queues

<code>add (value)</code>	places given value at back of queue
<code>remove ()</code>	removes value from front of queue and returns it; throws a <code>NoSuchElementException</code> if queue is empty
<code>peek ()</code>	returns front value from queue without removing it; returns <code>null</code> if queue is empty
<code>size ()</code>	returns number of elements in queue
<code>isEmpty ()</code>	returns <code>true</code> if queue has no elements

```
Queue<Integer> q = new LinkedList<Integer>();  
q.add(42);  
q.add(-3);  
q.add(17);           // front [42, -3, 17] back  
System.out.println(q.remove());    // 42
```

- **IMPORTANT:** When constructing a queue you must use a new `LinkedList` object instead of a new `Queue` object.
  - There is no Queue Object, Queue is an interface.

# Possible Progress Steps

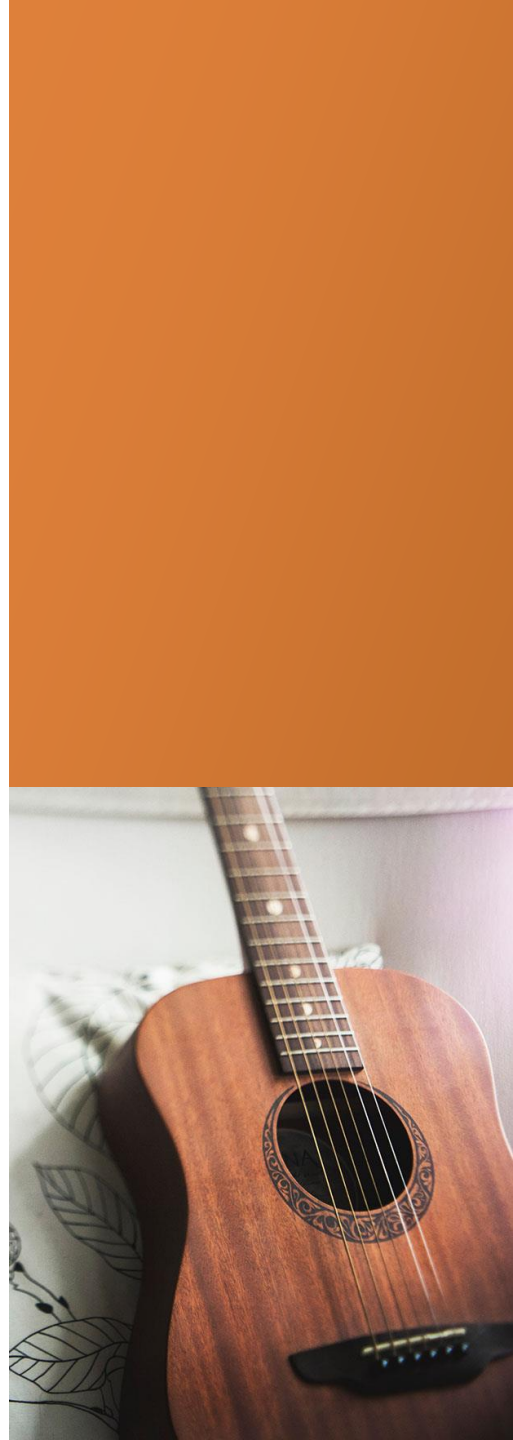
- GuitarString.java

- Constructors : initialize all data members

```
/**  
 * - create a ring buffer with capacity N (sampling rate / frequency, rounded to  
the      * nearest int)  
 * - initialize with N zeros (enqueue)  
 * - sampling rate = StdAudio.SAMPLE_RATE  
 * if frequency <= 0 or ringbuffer size <2, throw IllegalArgumentException  
 */
```

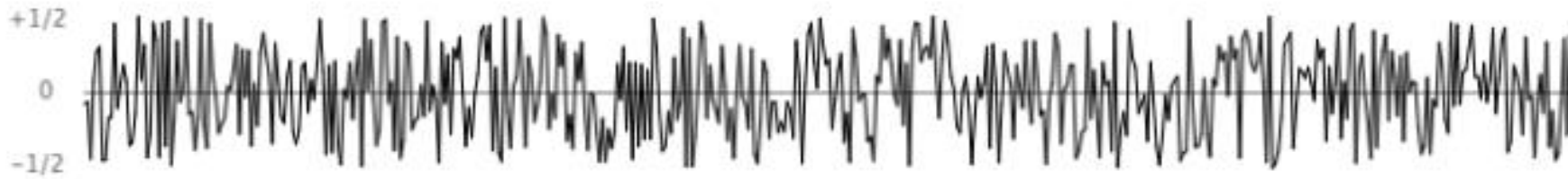
```
public GuitarString (double frequency)
```

```
public GuitarString (double[] init) // for testing and debugging
```

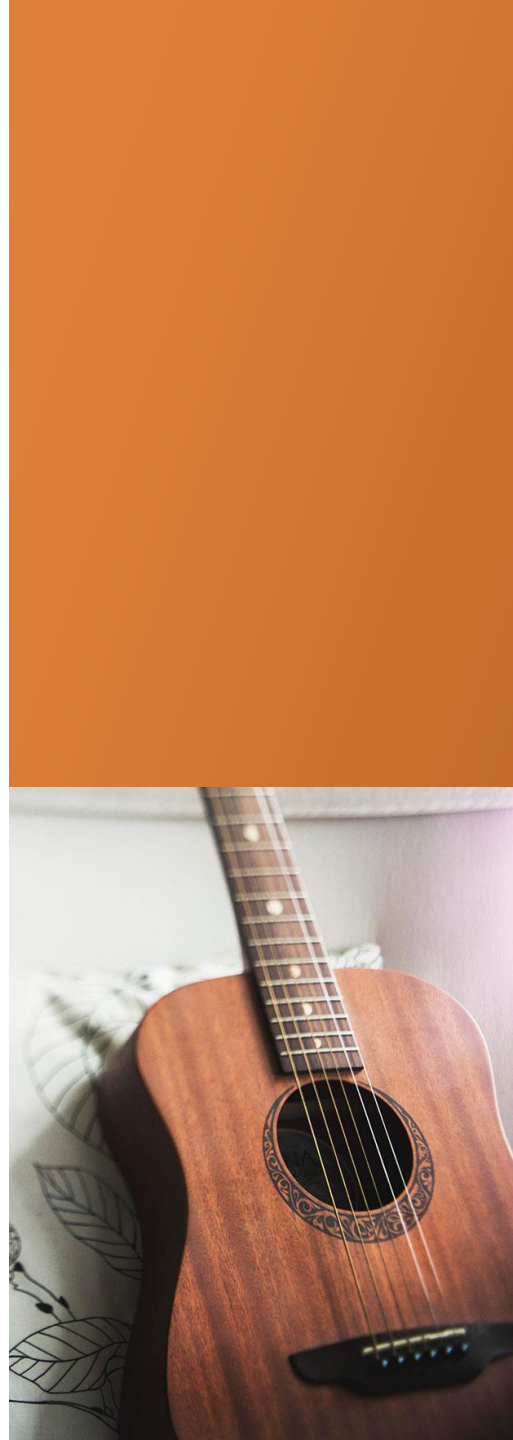




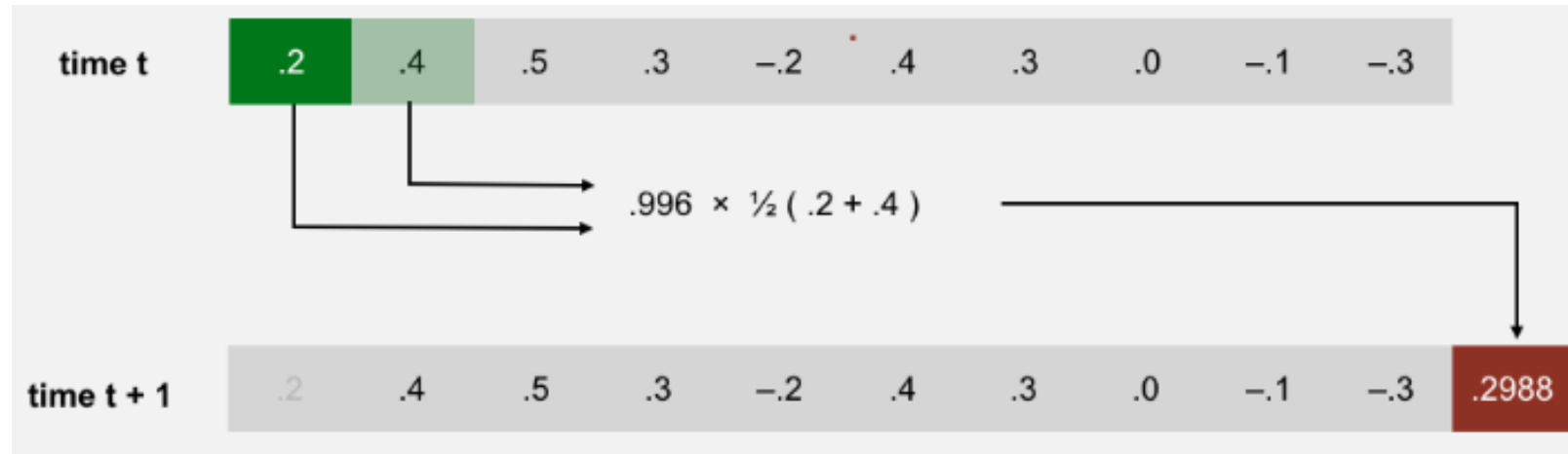
# public void pluck()



- When a string is plucked, it vibrates and creates sound.
- Simulating the excitation of the string
  - Replace the ringBuffer with white noise. How?
  - Random real numbers between  $-1/2$  and  $+1/2$
  - Use Random class with a uniform random method

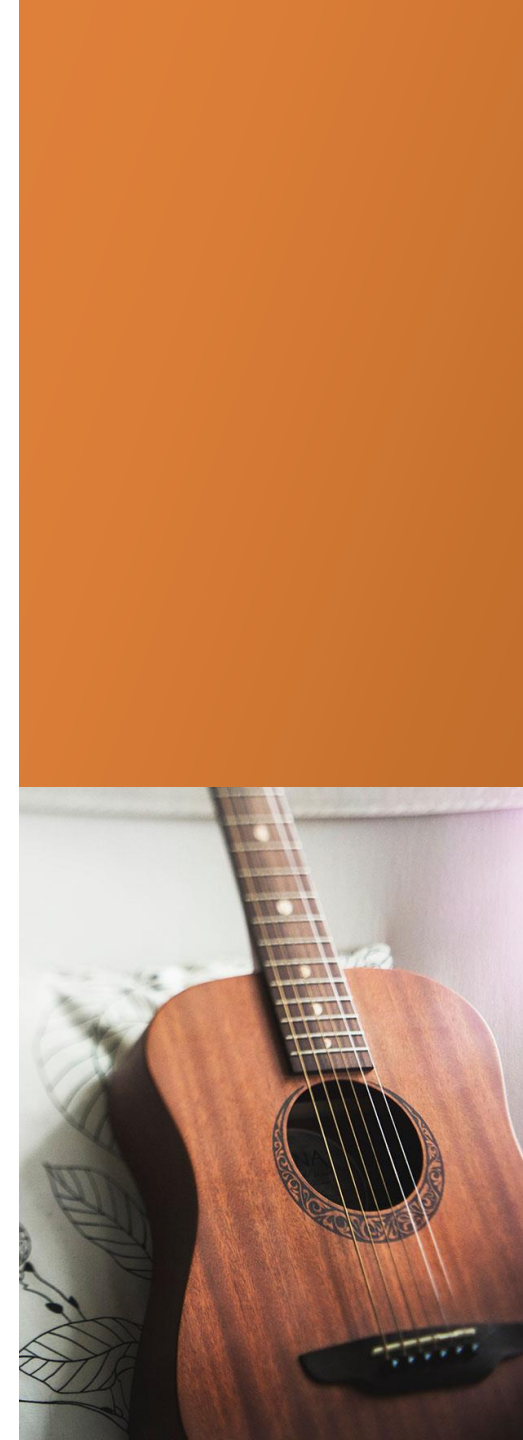


# public void tic()



- `tic()` simulates another time step of the sound wave we are calculating by:
  1. Calculate the avg of the front two elements
  2. Multiply by `DECAY_FACTOR` (0.996)
  3. Remove the first
  4. Add the result

Q: How can you look at the second (.4) without removing it from RingBuffer?



# Possible Progress Steps

- Testing GuitarString.java with TestString.java
  - TestString is provided.
  - TestString will test your GuitarString class
  - The file uses string.txt to test





# Possible Progress Steps

- Writing Guitar37 class
  - Use GuitarLite as an example
    - implements the Guitar interface
    - has two strings, A and C
    - Poorly documented



# From 2 to 37

```
// create two guitar strings, for concert A and C
```

```
public GuitarLite() {  
    double concertA = 440.0;  
    double concertC = concertA * Math.pow(2, 3.0/12.0);  
    stringA = new GuitarString(concertA);  
    stringC = new GuitarString(concertC);  
}
```

Create an array of GuitarString  
Use for-loop to initialize with frequency

```
public void playNote(int pitch) {  
    if (pitch == 0) {  
        stringA.pluck();  
    } else if (pitch == 3) {  
        stringC.pluck();  
    }  
}
```

Index = pitch + 24, do NOT use 37-way if  
statement, Significant points will be deducted

```
public boolean hasString(char string) {  
    return (string == 'a' || string == 'c');  
}
```

Use indexOf(string)  
Returns the index of the string, -1 otherwise

```
public void pluck(char string) {  
    if (string == 'a') {  
        stringA.pluck();  
    }  
}
```

# From 2 to 37

```
public void pluck(char string) {  
    if (string == 'a') {  
        stringA.pluck();  
    } else if (string == 'c') {  
        stringC.pluck();  
    }  
}
```

doNOT use 37 way if statement, How to map string to KEYBOARD chars; consider using charAt

```
public double sample() {  
    return stringA.sample() + stringC.sample();  
}
```

```
public void tic() {  
    stringA.tic();  
    stringC.tic();  
}
```

```
public int time() {  
    return -1; // not implemented  
}
```

Return the number of times the tic() has been called

# Possible Progress Steps

- Testing Guitar37 in test37 folder
  - Copy GuitarString.java, Guitar37.java, GuitarHelo.java

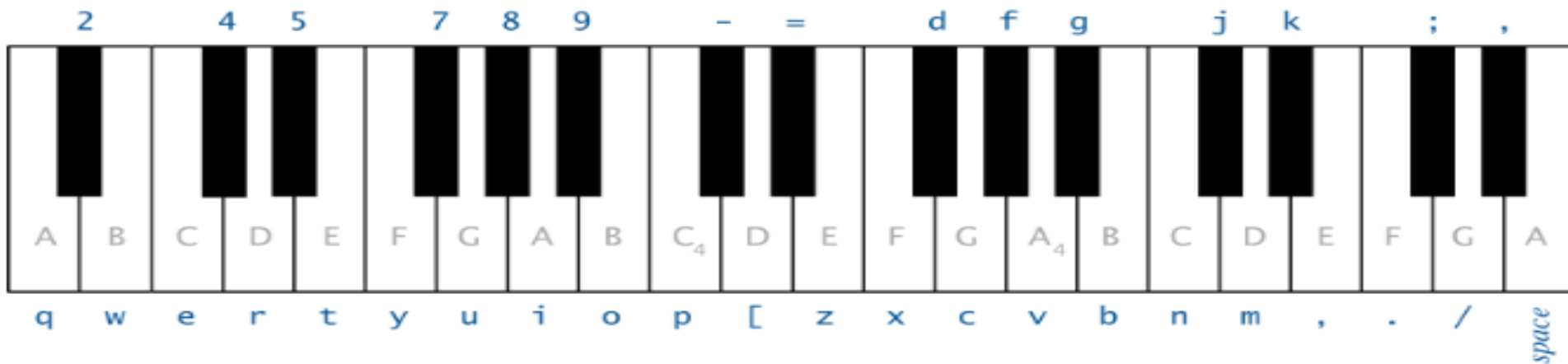
```
public class GuitarHero {  
    public static void main(String[] args) {  
        Guitar g = new GuitarLite();  
        // this is an infinite loop--user must quit the application  
        for (;;) {  
            // check if the user has typed a key; if so, process it  
            if (StdDraw.hasNextKeyTyped()) {  
                char key = Character.toLowerCase(StdDraw.nextKeyTyped());  
                if (g.hasString(key)) {  
                    g.pluck(key);  
                } else {  
                    // ...  
                }  
            }  
        }  
    }  
}
```

Interfaces are powerful!!  
Simply Change GuitarLite -> Guitar37



# Possible Progress Steps

- Play music and have fun!







# Questions?