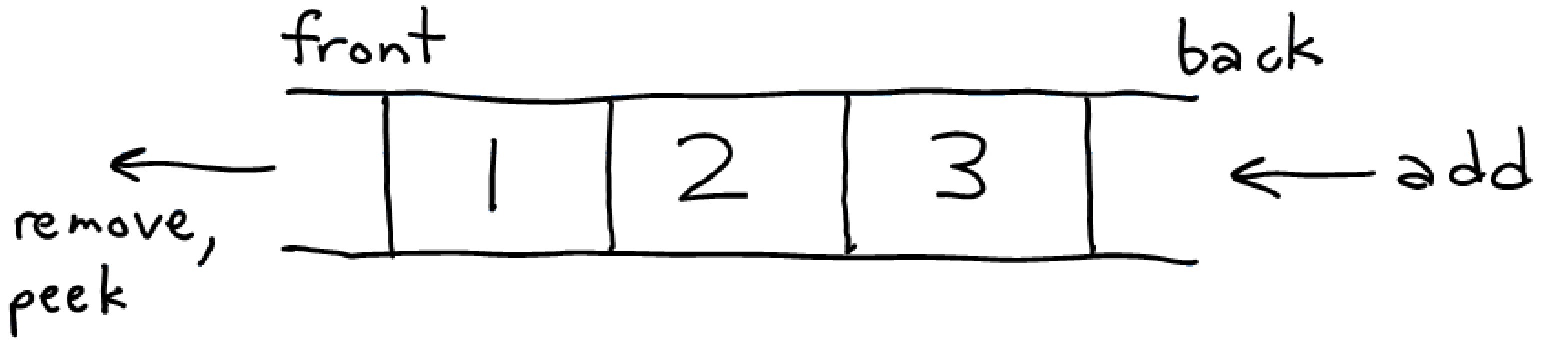
Guitar Hero

Goal

- Simulate the vibration of a string and produce actual sound wave **GuitarString.java**
- Simulate a guitar you can play with keyboard **Guitar37.java**
- Enhance your understand of queues, interfaces, objects and arrays of objects
- Learn about efficient data structures that are crucial for application performance

Queues



queue

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

Programming with Queues

`add(`**`value`**`)`	places given value at back of queue
`remove()`	removes value from front of queue and returns it; throws a `NoSuchElementException` if queue is empty
`peek()`	returns front value from queue without removing it; returns `null` if queue is empty
`size()`	returns number of elements in queue
`isEmpty()`	returns `true` 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
```

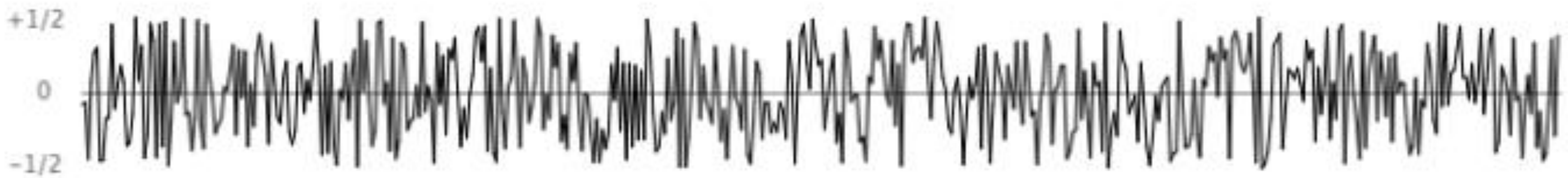
- **IMPORTANT**: When constructing a queue you must use a new `LinkedList` object instead of a new `Queue` object.
  - There is no Queue Object, Queue is an interface.

Possible Progress Steps

- GuitarString.java
  - Constructors : initialize all data members

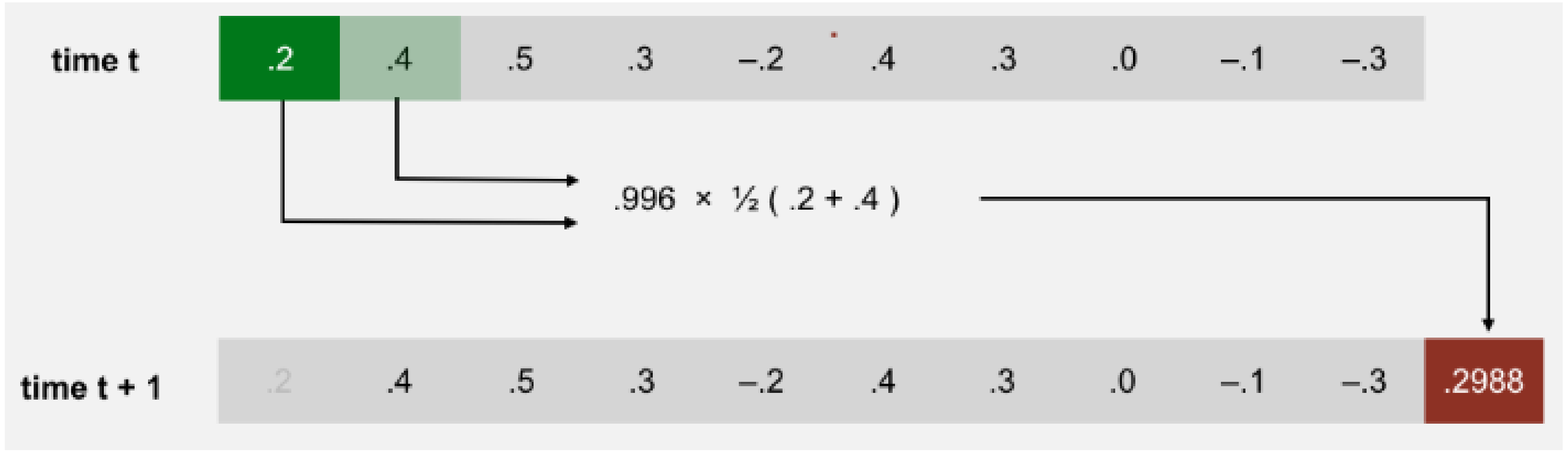
```
/**
 * - create a ring buffer with capacity N (sampling rate / frequency, rounded to
the        * nearest int)
 * - initialize with N zeros (enqueue)
 * - sampling rate = StdAudio.SAMPLE_RATE
 * if frequency <= 0 or ringbuffer size <2, throw IllegalArgumentException
 */
public GuitarString (double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

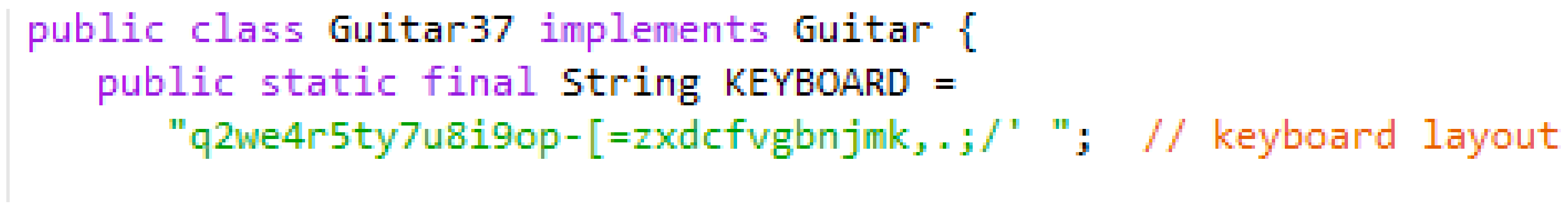
The Guitar Interface

- There can be different types of Guitars
  - A 2-string guitar (A and C) -> GuitarLite
    - This is already completed and included for your reference
  - A 37-string guitar -> Guitar37
    - You need to create and implement this file.

The Guitar Interface

```
public interface Guitar {
    public void playNote(int pitch);
    public boolean hasString(char key);
    public void pluck(char key);
    public double sample();
    public void tic();
    public int time();
}
```

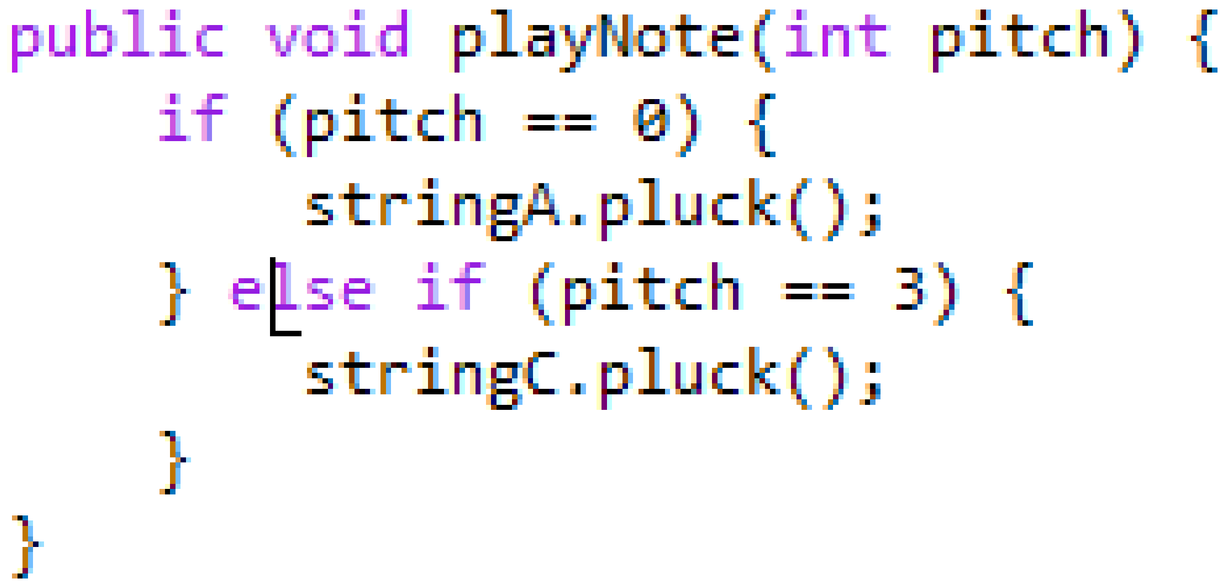
Keyboard



```
public class Guitar37 implements Guitar {
   public static final String KEYBOARD =
      "q2we4r5ty7u8i9op-[=zxdcfvgbnjmk,.;/' ";  // keyboard layout
```

- Note keyboard ‘a’ does work anymore
- KEYBOARD.charAt (0) is q
- KEYBOARD.charAt(1) is 2
- KEYBOARD.indexOf (“w”) will give 2

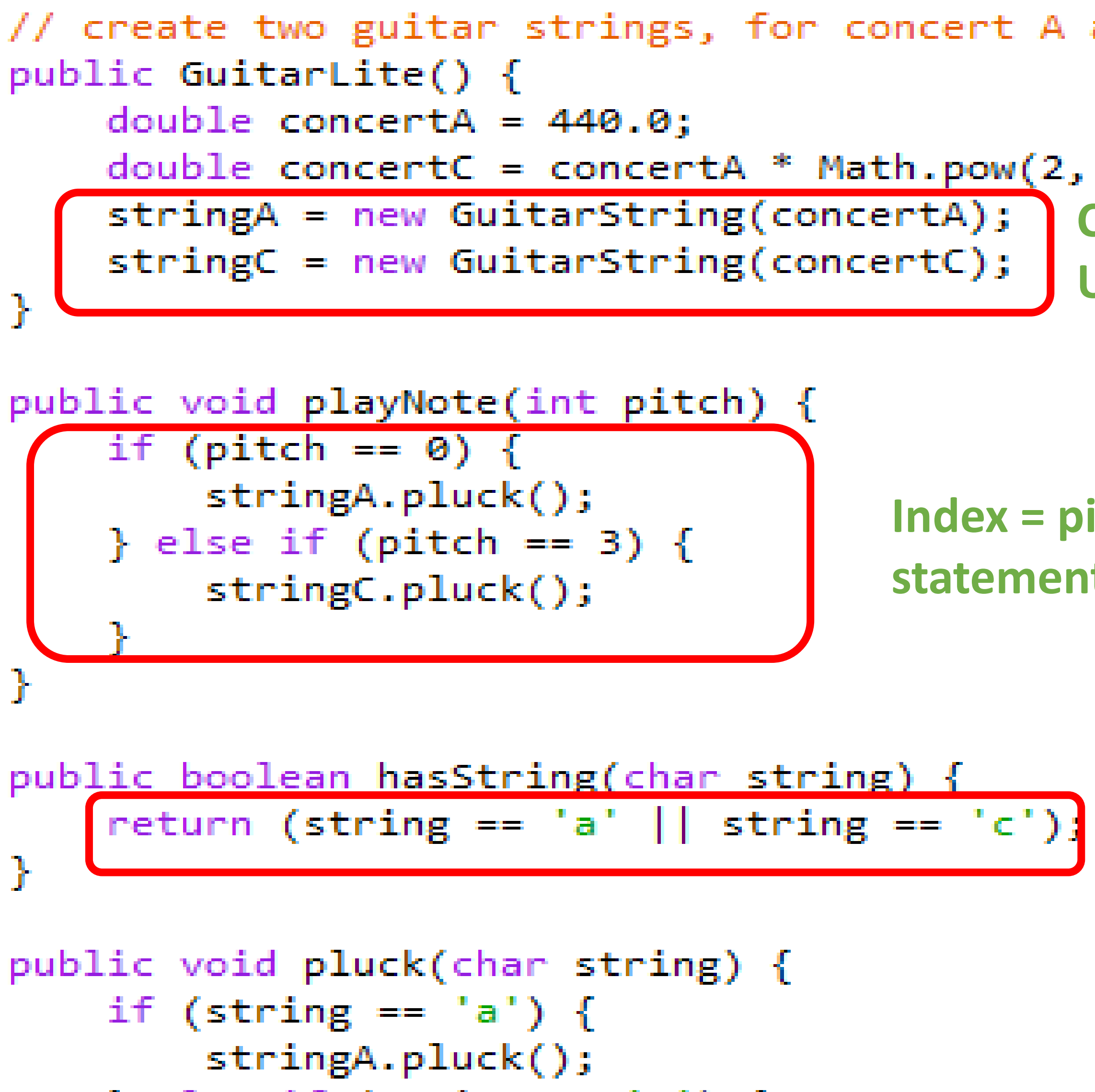
playNote



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

- Pitch of 0 → Concert A

From 2 to 37



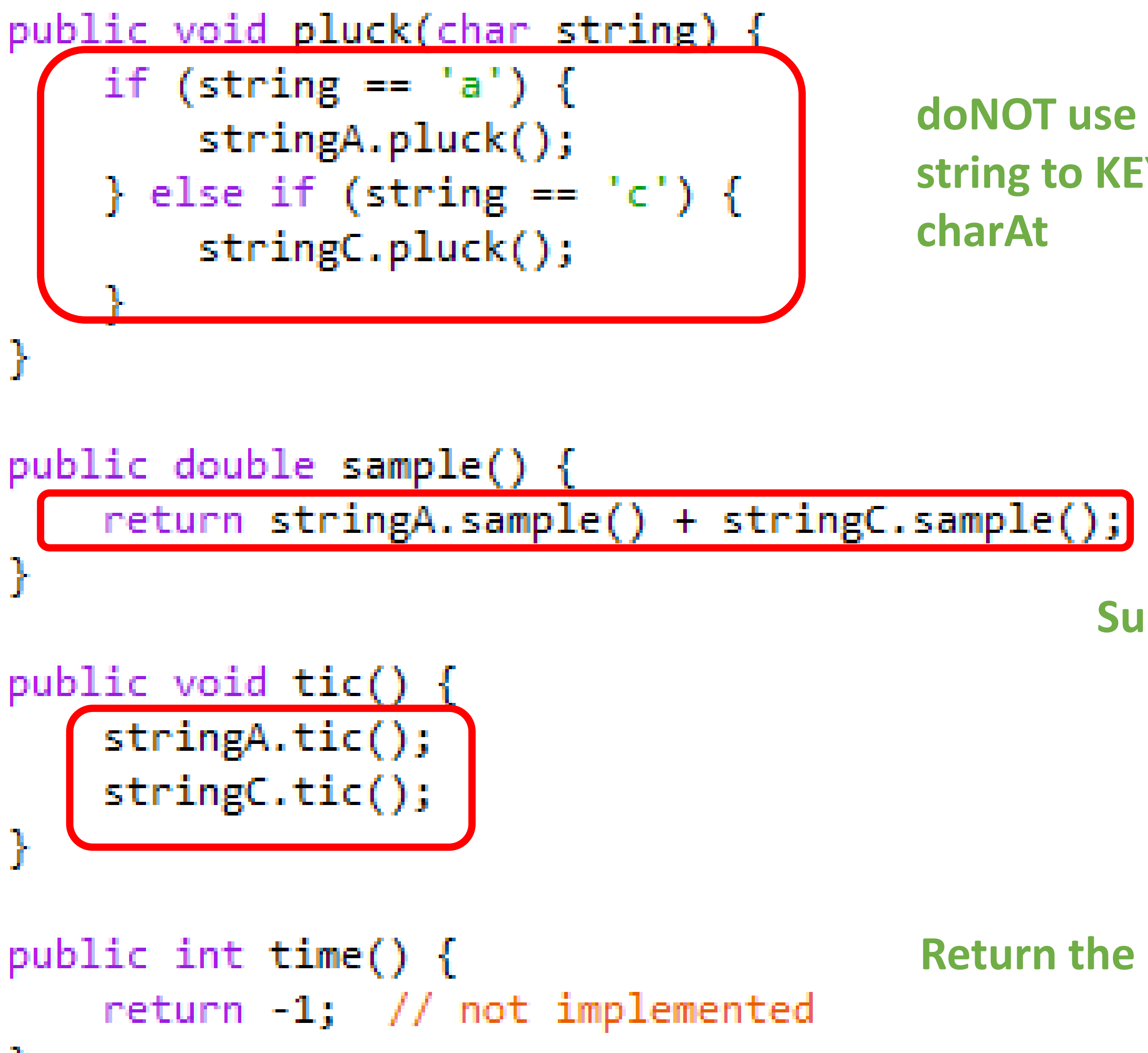
```
// create two guitar strings, for concert A and C
public GuitarLite() {
    double concertA = 440.0;
    double concertC = concertA * Math.pow(2, 3.0/12.0);
    stringA = new GuitarString(concertA);
    stringC = new GuitarString(concertC);
}

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

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

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

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

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

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

From 2 to 37



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

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

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

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

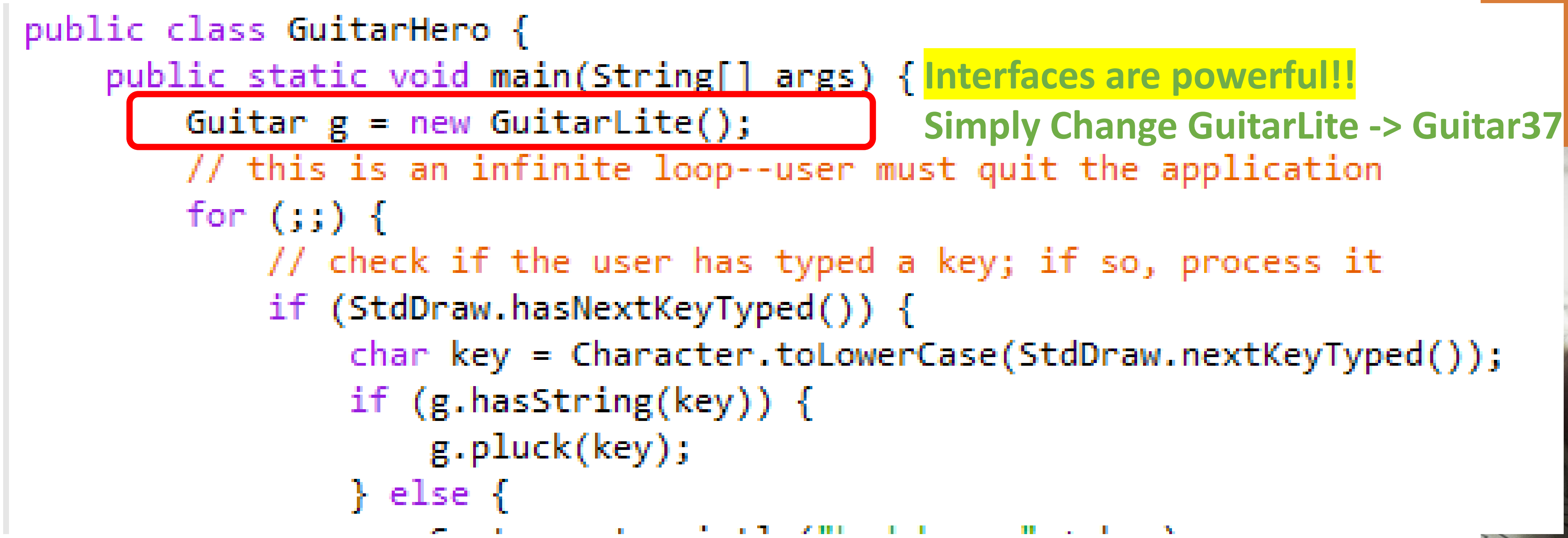
doNOT use 37 way if statement, How to map string to KEYBOARD chars; consider using charAt

Sum of all 37 samples

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

Possible Progress Steps

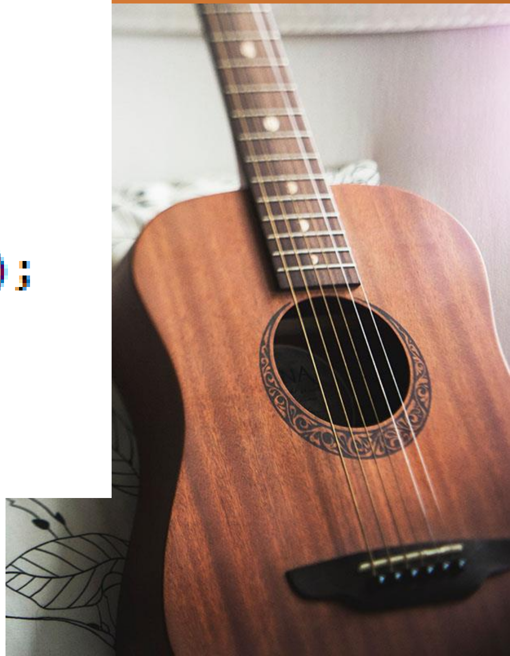
- Testing Guitar37 in test37 folder
  - Copy GuitarString.java, Guitar37.java, GuitarHelo.java



```
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 {
```

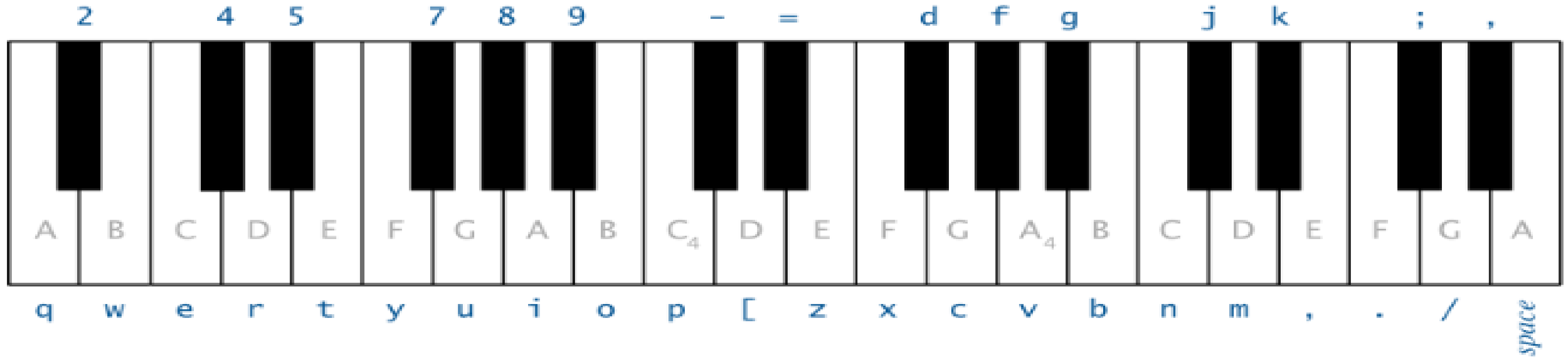
Interfaces are powerful!!

Simply Change GuitarLite -> Guitar37



Possible Progress Steps

- Play music and have fun!



2 4 5 7 8 9 - = d f g j k ; '
A B C D E F G A B C4 D E F G A4 B C D E F G A
q w e r t y u i o p [z x c v b n m , . / space

Questions?