Announcements

Critique of Version 1

- So where’s the music?
  - We don’t think about notes as just one long strand.
- Where are the phrases/riffs/elements?
  - We just have one long line of notes.
- How do we explore patterns like this?
  - `insertAfter` and `delete` are just as hard here as in sampled sounds!
Version 2

- Let’s re-think **NoteArray** as a collection of *elements* that we can shuffle around as we’d like.
- We can make any element follow any other element.
- Call each element a *node*.

What’s in each element?

**NoteNode**

**Data**
- it’s Note
- what comes next
  - another NoteNode

**Actions**
- getPhrase()
  - get the phrase beginning at this Note and continuing until the end

Introducing the **Linked List**

- A linked list is information broken into smaller pieces, where each piece knows the next piece, but none other.
public class NoteNode
{
    //---//-- Fields //---//--//
    Note note;
    NoteNode next;

    //---//-- Constructors //---//--//
    public NoteNode(Note note)
    {
        this.note = note;
    }

    public NoteNode getNext()
    {
        return next;
    }

    public void setNext(NoteNode next)
    {
        this.next = next;
    }

    public static void main(String [] args)
    {
        NoteNode n1 = new NoteNode(new Note(JMC.D2, JMC.ENT));
        NoteNode n2 = new NoteNode(new Note(JMC.FS2, JMC.ENT));
        NoteNode n3 = new NoteNode(new Note(JMC.A2, JMC.ENT));

        n1.next = n2;
        n2.next = n3;
    }
}

//---//-- Methods //---//--//--
public Note getNote()
{
    return note;
}

public void setNote(Note note)
{
    this.note = note;
}
```java
public static void main(String[] args) {
    NoteNode n1 = new NoteNode(new Note(JMC.D2, JMC.ENT));
    NoteNode n2 = new NoteNode(new Note(JMC.FS2, JMC.ENT));
    NoteNode n3 = new NoteNode(new Note(JMC.A2, JMC.ENT));
    n1.setNext(n2);
    n2.setNext(n3);
}
```

```java
public int size() {
    NoteNode current = this;
    int size = 0;
    while (current != null) {
        size++;
        current = current.next;
    }
    return size;
}
```

```java
public Phrase getPhrase() {
    NoteNode current = this;
    Phrase phr = new Phrase();
    while (current != null) {
        phr.addNote(current.note);
        current = current.next;
    }
    return phr;
}
```

```java
public static void main(String[] args) {
    NoteNode n1 = new NoteNode(new Note(JMC.D2, JMC.ENT));
    NoteNode n2 = new NoteNode(new Note(JMC.FS2, JMC.ENT));
    NoteNode n3 = new NoteNode(new Note(JMC.A2, JMC.ENT));
    n1.next = n2;
    n2.next = n3;
    Phrase phr1 = n1.getPhrase();
    Part part1 = new Part("Bass", JMC.PBASS, phr1);
    Score score = new Score("Buster", 100.0, part1);
    View.notate(score);
}
```