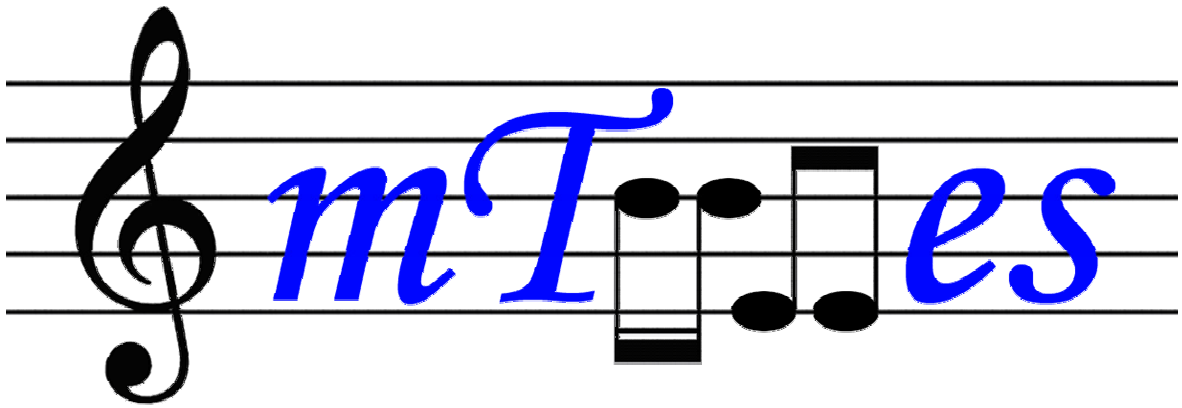


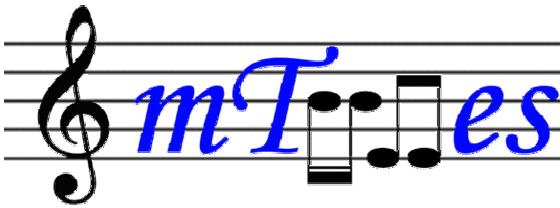
midi composition language

HeeWook Lee, Sharon Price, Hideki Sano, Huitao Sheng

pre-existing solutions

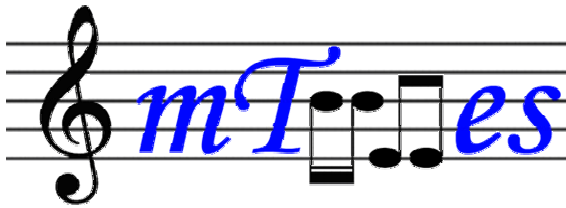
- Many midi composition software tools already available – Midisoft Studio, Digital Performer, Cakewalk
- Professional software comes with a professional price tag
- Most are based around composition through a gui, one note at a time
- Few can be programmed to generate music

- javax.sound.midi package allows for midi composition in Java
  - difficult to use beyond playing single notes
- Open-source packages such as jFugue and jMusic
  - based on javax.sound.midi
  - still require knowledge of Java
  - extremely verbose to create simple songs



a better solution

- Eliminate need to learn Java-syntax and rules
- Simpler syntax that does not require any superfluous code
- Ability to generate notes in functions instead of one at a time
- Generate the same .midi files that can be played (and edited) using regular players

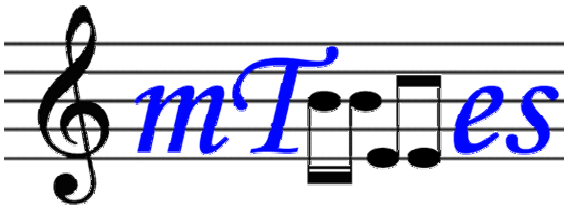


example .mt file

```
save “poke.midi”
I1 = GUITAR
Sequence s1 = C D E F
Sequence s2 = G A B C
add I1 s1
add I1 s2
I1 = PIANO
add I1 s1
add I1 s2
playSong
```

quick overview

- There are three main data structures – Notes, Sequences, and Instruments
- The fourth data structure, Files, is only used once to extract the data from a file to the three main data structures
- Functions and Procedures can be written to manipulate Notes and Sequences, as well as add them to an Instrument's queue
- if/else statements, for loops, and foreach loops allow for control flow



Notes

Notes are the most basic data structure in mTunes. They contain everything that a musical note is expected to – pitch (including octave), duration, and volume.

Note initialization, part one

```
Note note1
```

Note initialization, part two

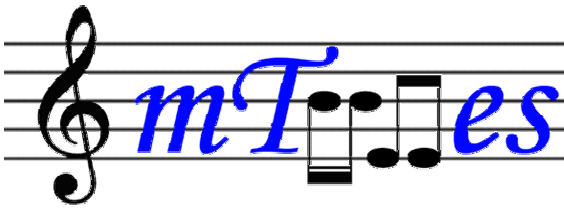
```
Note note1 = C
```

Note initialization, part three

```
Note note1 = C#4_0.25
```

Access of Note elements

```
note1.pitch = note1.pitch + 1

note1.length = 2 * note1.length

note1.octave = 5

note1.volume = 0.5 * note1.volume
```



Sequences

Sequences are groups of Notes. The Notes are added to Sequences in the order that they are supposed to be played. Like a Note, a Sequence can be initialized without assigning it a value.

Sequence initialization

```
Sequence seq1
```

Sequence initialization, with a value

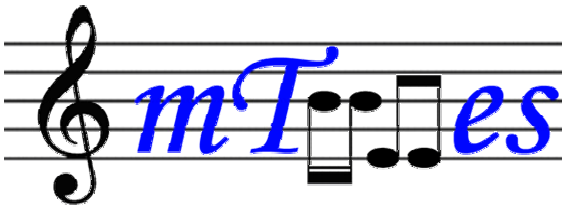
```
Sequence seq1 = C D E n1
```

Append one Sequence to another

```
seq1 = seq1 + seq2
```

Append a Note to a Sequence

```
seq1 = note1 + seq1
```



Instruments

Instruments are the mTunes version of tracks. There are 16 available tracks, denoted I1, I2, …, I16. Each track can be set to play Notes using any of the standard midi instruments. The type of instrument that an Instrument is set to only applies to Notes and Sequences added to the Instrument after the change. Every Instrument has a queue consisting of all the Notes and Sequences added to it, in the order that they were added.

Instrument initialization

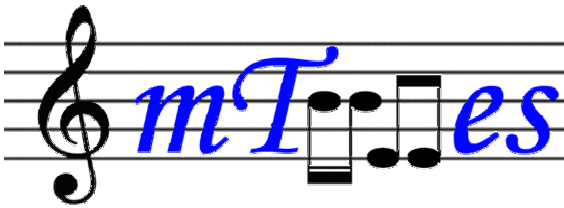
```
I4 = PIANO
```

Instrument initialization, part two

```
I7 = STEEL_DRUMS
```

Append to queue

```
add I7 note1

add 17 seq1
```



midi instruments

PIANO	ELECTRIC_JAZZ_GUITAR	ORCHESTRA_HIT	LEAD_CHIFF	SOUNDTRACK	GUITAR_FRET_NOISE	CRASH_CYMBOL_2
ACOUSTIC_GRAND	ELECTRIC_CLEAN_GUITAR	TRUMPET	CHIFF	FX_CRYSTAL	BREATH_NOISE	VIBRASLAP
BRIGHT_ACOUSTIC	ELECTRIC_MUTED_GUITAR	TROMBONE	LEAD_CHARANG	CRYSTAL	SEASHORE	RIDE_CYMBAL_2
ELECTRIC_GRAND	OVERDRIVEN_GUITAR	TUBA	CHARANG	FX_ATMOSPHERE	BIRD_TWEET	HI_BONGO
HONKEY_TONK	DISTORTION_GUITAR	MUTED_TRUMPET	LEAD_VOICE	ATMOSPHERE	TELEPHONE_RING	LOW_BONGO
ELECTRIC_PIANO	GUITAR_HARMONICS	FRENCH_HORN	VOICE	FX_BRIGHTNESS	HELICOPTER	MUTE_HI_CONGA
ELECTRIC_PIANO_1	ACOUSTIC_BASS	BRASS_SECTION	LEAD_FIFTHS	BRIGHTNESS	APPLAUSE	OPEN_HI_CONGA
ELECTRIC_PIANO_2	ELECTRIC_BASS_FINGER	SYNTHBRASS_1	FIFTHS	FX_GOBLINS	GUNSHOT	LOW_CONGA
HARPISCHORD	ELECTRIC_BASS_PICK	SYNTHBRASS_2	LEAD_BASSLEAD	GOBLINS	ACOUSTIC_BASS_DRUM	HIGH_TIMBALE
CLAVINET	FRETLESS_BASS	SOPRANO_SAX	BASSLEAD	FX_ECHOES	BASS_DRUM	LOW_TIMBALE
CELESTA	SLAP_BASS_1	ALTO_SAX	PAD_NEW_AGE	ECHOES	SIDE_STICK	HIGH_AGOGO
GLOCKENSPIEL	SLAP_BASS_2	OBOE	NEW_AGE	FX_SCI-FI	COUSTIC_SNARE	LOW_AGOGO
MUSIC_BOX	SYNTH_BASS_1	ENGLISH_HORN	PAD_WARM	SCI-FI	HAND_CLAP	CABASA
VIBRAPHONE	SYNTH_BASS_2	BASSON	WARM	SITAR	ELECTRIC_SNARE	MARACAS
MARIMBA	VIOLIN	CLARINET	PAD_POLYSYNTH	BANJO	LOW_FLOOR_TOM	SHORT_WHISTLE
XYLOPHONE	VIOLA	PICCOLO	POLYSYNTH	SAMISEN	CLOSED_HI_HAT	LONG_WHISTLE
TUBULAR_BELLS	CELLO	FLUTE	PAD_CHOIR	KOTO	HIGH_FLOOR_TOM	SHORT_GUIRO
DULCIMER	CONTRABASS	RECORDER	CHOIR	KALIMBA	PEDAL_HI_HAT	LONG_GUIRO
DRAWBAR_ORGAN	TREMOLO_STRINGS	PAN_FLUTE	PAD_BOWED	BAGPIEP	LOW_TOM	CLAVES
PERCUSSIVE_ORGAN	PIZZICATO_STRINGS	BLOWN_BOTTLE	BOWED	FIDDLE	OPEN_HI_HAT	HI_WOOD_BLOCK
ROCK_ORGAN	ORCHESTRAL_STRINGS	SKAKUHACHI	PAD_METALLIC	SHANAI	LOW_MID_TOM	LOW_WOOD_BLOCK
CHURCH_ORGAN	TIMPANI	WHISTLE	METALLIC	TINKLE_BELL	CRASH_CYMBAL_1	MUTE_CUICA
REED_ORGAN	STRING_ENSEMBLE_1	OCARINA	PAD_HALO	AGOGO	HIGH_TOM	OPEN_CUICA
ACCORDIAN	STRING_ENSESMBLE_2	LEAD_SQUARE	HALO	STEEL_DRUMS	RIDE_CYMBAL_1	MUTE_TRIANGLE
HARMONICA	SYNTHSTRINGS_1	SQUARE	PAD_SWEEP	WOODBLOCK	CHINESE_CYMBAL	OPEN_TRIANGLE
TANGO_ACCORDIAN	SYNTHSTRINGS_2	LEAD-SAWTOOTH	SWEEP	TAIKO_DRUM	RIDE_BELL	
GUITAR	CHOIR_AAHS	SAWTOOTH	FX_RAIN	MELODIC_TOM	TAMBOURINE	
NYLON_STRING_GUITAR	VOICE_OOHS	LEAD_CALLIOPE	RAIN	SYNTH_DRUM	SPLASH_CYMBAL	
STEEL_STRING_GUITAR	SYNTH_VOICE	CALLIOPE	FX_SOUNDTRACK	REVERSE_CYMBAL	COWBELL	

mTunes basic statements

```
save "poke.midi"
I1 = GUITAR
Sequence s1 = C D E F
Sequence s2 = G A B
Note n1 = C
s2 = s2 + n1
add I1 s1
add I1 s2
I1 = PIANO
add I1 s1
add I1 s2
playSong
```

save statement indicates what to name midi file

instrument assignment statement sets instrument I1

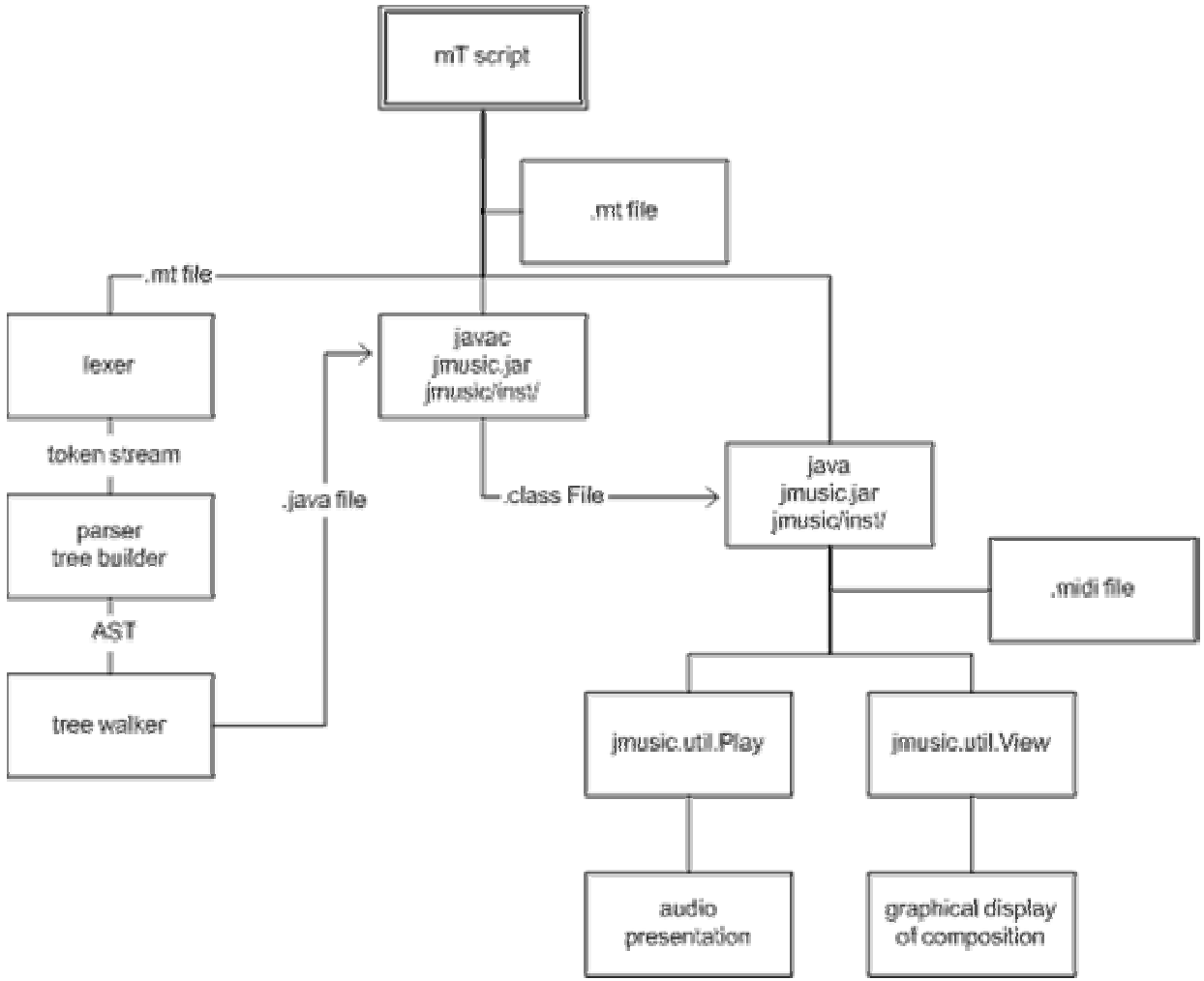
variable creation statements

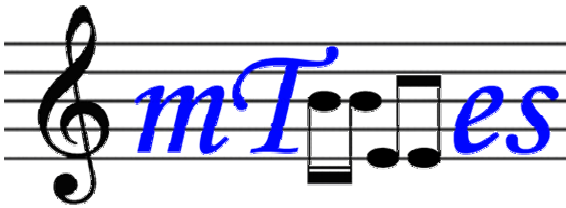
variable assignment statement appends n1 to the end of s2

add statement appends the sequences to the queue for I1

play statement tells mTunes to play resulting song

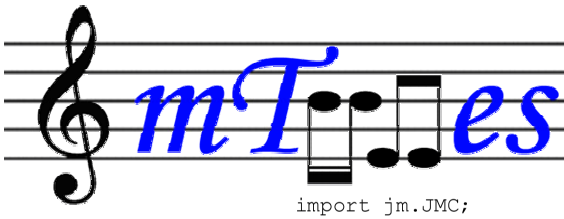
mTunes inside the compiler





Jingle Bells .mt

```
save jinglebell.midi
I1 = PIANO
I2 = STEEL_DRUMS
Sequence soprano
Sequence bass
setOctave 4
Sequence temp = A4_0.5 A4_0.5 A4_1
temp = repeat(2,temp)
Sequence temp2 = A4_0.5 C5_0.5 F4_0.75 G4_0.25 A4_2 B$4_0.5 B$4_0.5 B$4_0.75 B$4_0.25 B$4_0.5
temp = temp + temp2
Sequence temp3 = A4_0.5 A4_0.5 A4_0.25 A4_0.25
temp = temp + temp3
Sequence temp4 = A4_0.5 G4_0.5 G4_0.5 A4_0.5 G4_1 C5_1

Sequence firstpart = temp + temp4
Sequence temp5 = C5_0.5 C5_0.5 B$4_0.5 G4_0.5 F4_2
Sequence secondpart = temp + temp5
soprano = firstpart + secondpart

function repeat(int times, sequence block){
                Sequence result
                for(1,times,count){
                                result = result + block
                }
                return result
}
```



Jingle Bells jMusic

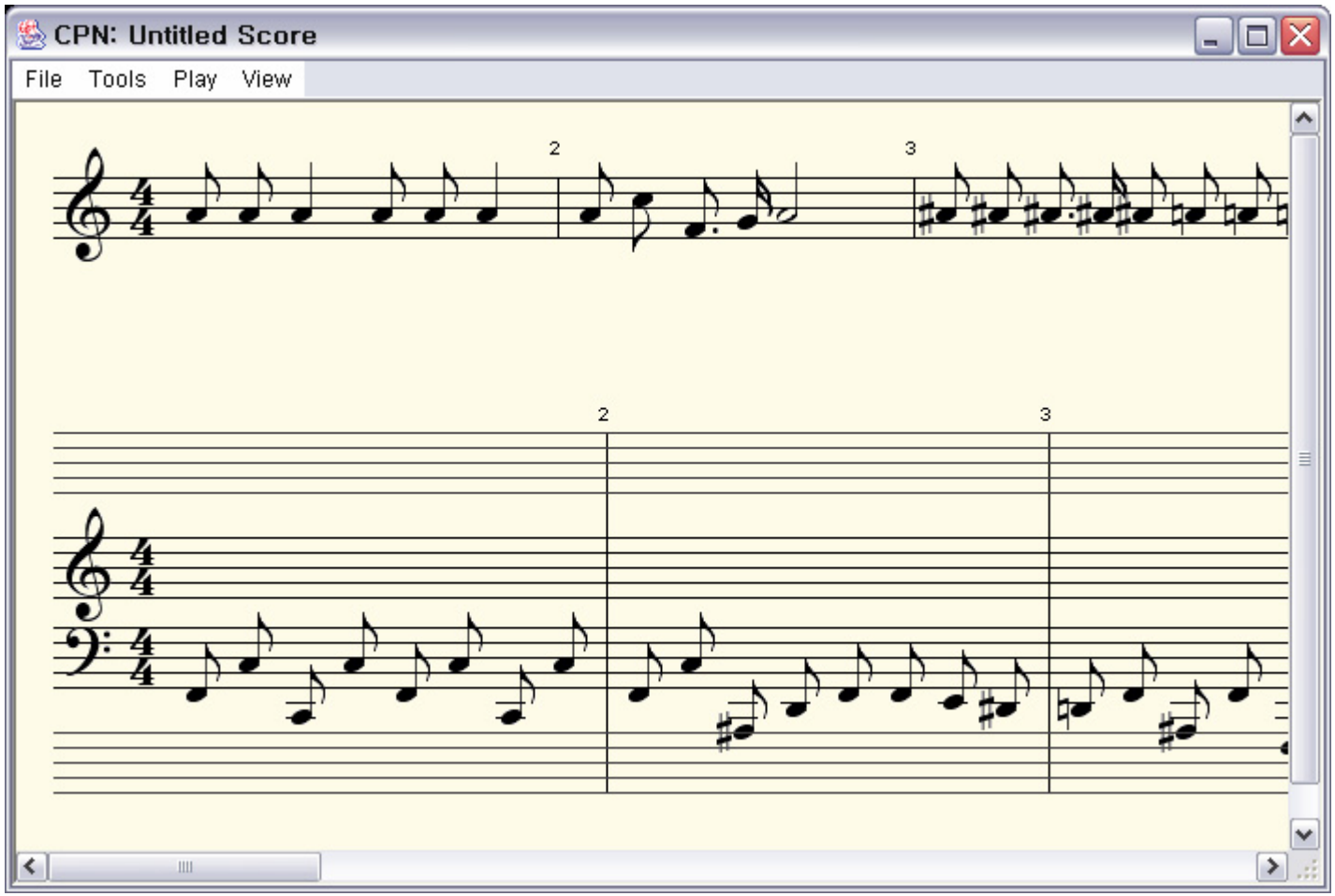
```
import jm.JMC;
import jm.util.*;
import jm.audio.Instrument;
import jm.audio.io.*;
import jm.audio.*;
import jm.audio.synth.*;
import jm.music.data.*;;

public class Demo implements JMC {

                public static void main(String[] args) {
                    new Demo();
                }

                public Demo() {
                                Phrase sopPhr = new Phrase();
                                Phrase bassPhr = new Phrase();
                                double[] sopList1 = {A4,0.5,A4,0.5,A4,1};
                                sopPhr.addNoteList(sopList1);
                                sopPhr.addNoteList(sopList1);
                                double[] sopList2 = {A4,0.5,C5,0.5,F4,0.75,G4,0.25,A4,2};
                                sopPhr.addNoteList(sopList2);
                                Note bbf4 = new Note(BF4, 0.5);
                                sopPhr.addNote(bbf4);
                                sopPhr.addNote(bbf4);
                                sopPhr.addNote(new Note(BF4,0.75));
                                Note sbbf4 = new Note(BF4, 0.25);
                                sopPhr.addNote(sbbf4);
                                sopPhr.addNote(bbf4);
```

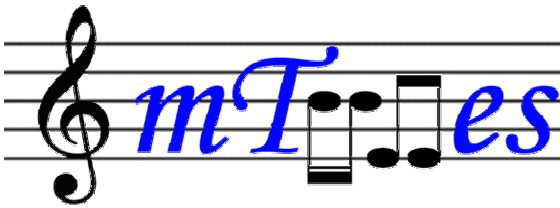
```
double[] sopList3 = {A4,0.5,A4,0.5,A4,0.25,A4,0.25};
double[] sopList4 = {A4,0.5,G4,0.5,G4,0.5,A4,0.5,G4,1,C5,1};
sopPhr.addNoteList(sopList3);
sopPhr.addNoteList(sopList4);
/*repeat*/
sopPhr.addNoteList(sopList1);
sopPhr.addNoteList(sopList1);
sopPhr.addNoteList(sopList2);
sopPhr.addNote(bbf4);
sopPhr.addNote(bbf4);
sopPhr.addNote(new Note(BF4,0.75));
sopPhr.addNote(sbbf4);
sopPhr.addNote(bbf4);
sopPhr.addNoteList(sopList3);

double[]sopList5 = {C5,0.5,C5,0.5,BF4,0.5,G4,0.5,F4,2};
sopPhr.addNoteList(sopList5);

double[]bassList1 = {F2,0.5,C3,0.5,C2,0.5,C3,0.5};
bassPhr.addNoteList(bassList1);
bassPhr.addNoteList(bassList1);
double[]bassList2 = {F2,0.5,C3,0.5,BF1,0.5,D2,0.5,F2,0.5,F2,0.5,E2,0.5,EF2,0.5};
bassPhr.addNoteList(bassList2);
double[]bassList3 = {D2,0.5,F2,0.5,BF1,0.5,F2,0.5,F1,0.5,F2,0.5,C3,0.5,F2,0.5};
bassPhr.addNoteList(bassList3);
double[]bassList4 = {C2,0.5,G2,0.5,G1,0.5,G2,0.5,C2,0.5,C2,0.5,D2,0.5,E2,0.5};
bassPhr.addNoteList(bassList4);

bassPhr.addNoteList(bassList1);
```

```
                bassPhr.addNoteList(bassList1);
                bassPhr.addNoteList(bassList2);
                bassPhr.addNoteList(bassList3);
                double[]bassList5 = {C2,0.5,C2,0.5,D2,0.5,E2,0.5,F2,0.5,C2,0.5,F2,1};
                bassPhr.addNoteList(bassList5);

                Part sPart = new Part(sopPhr, "Soprano", PIANO, 0);
                Part bPart = new Part(bassPhr, "Bass", STEEL_DRUMS, 1);
                sPart.setTempo(88);
                bPart.setTempo(88);
                Part[] parts = {sPart,bPart};
                Score score = new Score(parts);
                score.setTempo(88);
                Write.midi(score, "jinglebells.mid");
                View.notate(score);
                Play.midi(score);

        }
}
```

mTunes

Jingle Bells .midi





lessons learned

- Despite the fact that cvs is really annoying, using it wouldn't have been *that* bad an idea
- Don't assume that the source code from last year's projects is really good and can be used as a model (they *did* do a different language….)
- Sleep. At least occasionally.
- Even if you think something in the language is obvious, it probably won't be to anyone outside the group
- Having a member of the target audience test out the language earlier, rather than later, would have been more useful
- No matter what, it's impossible to stick to a timeline
- Don't forget to sleep.