COMSW4115: Programming Languages and Translators
The DJ Language Final Report

William Falk-Wallace (wgf2104), Hila Gutfreund (hg2287),
Emily Lemonier (eql2001), Thomas Elling (tee2103)

December 21, 2013

Contents

1 Introduction

We propose a procedural scripting language, DJ, which provides a programming paradigm for algorithmic music production. Through its utilization of themes and motifs, music is naturally repetitive and often dynamic. DJ provides control-flow mechanisms, including `for` and `loop` functions, which simplify the development of structured iterative music. The DJ Language also makes use of conditional logic and supports standard MIDI sound banks to facilitate the production of deeply textured musical compositions. Our goal in the specification of The DJ Language is to abstract away the intricacies and limitations of the MIDI specification, including channeling, patch-maps and instrumentation, while retaining conventions familiar to programmers of Java as well as MIDI, allowing the artist to focus on her or his work: composing songs.

2 The DJ Language

See Appendix A for Language Outline

3 Language Tutorial

3.1 Using the Compiler

Inside wdjc Master/, type `make`. This creates the WDJC compiler, `wdjc`. When executed, the compiler takes one of four command line arguments (please note for the $-c$ argument, a file name must also be specified). These arguments and their outputs are listed below:

Argument	Output
$-a$	Pretty prints AST to screen
$-s$	Pretty prints SAST to screen
$-j < filename >$	Pretty prints Java translation to screen. $filename$ is the name of the .mid file written out in the `song` function. Default $filename$ is song.
$-c < filename >$	Writes java program to file named filenamedj.java. From this point, run `./compile filenamedj` to produce $filename.mid$ in the Master/. Default $filename$ is song.

We present an example of how to compile the program using the file test/helloWorld.dj. Please note- the source code is presented in a later section.

- Pretty print the AST to screen:
  `> ./wdjc -a < tests/helloWorld.dj`
- Pretty print the SAST to screen:
  `> ./wdjc -s < tests/helloWorld.dj`
- Pretty print Java to screen:
  `> ./wdjc -j < tests/helloWorld.dj`
  `> ./wdjc -j hello < tests/helloWorld.dj`
- Compile and Produce midi file:
  - Produce hellodj.java file:
    `> ./wdjc -c hello < tests/helloWorld.dj` (note: our compiler appends a 'dj' to the end of the filename; this allows for tests named 'for', 'while', etc. and prevents java compilation errors)
  - Produce hello.mid:
    `> ./compile hellodj` (this file is located in your current directory, presumably Master/)

3.2 Program Structure

Every DJ program is required to contain a song function which returns a score: `song score(){...}`. This function takes no arguments and is analogous to the Java `main` function. Global variables and functions may be defined elsewhere in the program. These global functions may return any of the basic data types DJ supports.

3.3 Basic Types

There are five basic types defined by the DJ language. Type identifiers always begin with either a upper-case or lower-case letter followed by a sequence of one or more legal identifier characters. These built-in types include:

- `double`: We define a double to be any combination of digits with an optional decimal point. For example: 1.0, .008, 4
- `note`: A structure representing character attributes of a music note, such as: pitch, volume, and duration (in beats)
- `rest`: An atomic unit of a composition that doesn't have a pitch, or volume, but does maintain a duration
- `chord`: A collection of notes which begin on the same beat
- `track`: A series of chords which are played sequentially by the same instrument
- `score`: A collection of tracks which begin on the same beat

3.3.1 Example with Doubles - Source Code

Please note that in this example, a declaration always comes before an initialization and an inline initialization may be used. Please note that an empty score is written to the midi file. This is required as this is the `song` function

```
song score ()
{
        /* declaration must be before initialization */

        double var1;
        var1 = 1;
        double var2 = 2;
        double var3 = var1 + var2;
        print( var3 );

        score s = score();
        return s;
}
```

3.3.2 Example with Doubles - Output

```
3

/* empty score output */
------------------------------ Writing MIDI File ------------------------------
Converting to SMF data structure...
MIDI file 'controlFlow.mid' written from score 'Untitled Score' in 0.019 seconds.
-------------------------------------------------------------------------------
```

3.4 Control Flow

DJ's Control flow statements closely resemble those supported in Java such as for, while, and if-else statements. For example, the condition must evaluate to a binary value (ie 1/0). Furthermore, there are parentheses around the header and curly braces around the body. DJ also supports an additional control-flow mechanism,`loop`. The goal of the `loop` is to simplify the development of structured iterative music as music composition often involves repetition.

3.4.1 Control Flow Example - Source

This example demonstrates the various control flow mechanisms DJ supports.

```
song score ()
{
        double var1 = 0;
        double i;

        /* for loop */
        for(i = 0; i < 5; i = i+1)
        {
                print( var1 );
                var1++;
        }

        /* while loop  */
        while( var1 > 0)
        {
                print (var1 );
                var1--;
        }

        /* loop */
        loop(10){

                print( var1 );
                var1++;

        }

        /* if-else */
        if(var1 >= 0){
                print ( 1000 );
        }
        else{
                print ( -1000 );
        }

        score s = score();
        return s;
}
```

3.4.2 Control Flow Example - Output

```
/* for loop output */
0.0
1.0
2.0
3.0
4.0

/* while loop output */
5.0
4.0
3.0
2.0
1.0

/* loop function output */
0.0
1.0
2.0
3.0
4.0
5.0
6.0
7.0
8.0
9.0

/* if-else output */
1000

/* empty score output */
------------------------------ Writing MIDI File ------------------------------
Converting to SMF data structure...
MIDI file 'controlFlow.mid' written from score 'Untitled Score' in 0.019 seconds.
-------------------------------------------------------------------------------
```

4 Language Reference Manual

See Appendix B for Language Reference Manual

5 Project Plan

5.1 Project Processes

5.1.1 Planning Process

At the start of the semester we initially set deadlines for main project goals and milestone deadlines for building the WDJC compiler. By speaking with other groups and our TA, Julian Rosenblum, we determined these deadlines. Simultaneously, we designed short-term goals which contributed to our milestone goals.

5.1.2 Specification Process

We initially outlined the specifications of our languages features in the Language Reference Manual. From the beginning, we planned for DJ to syntactically similar to Java, with a set of features designed to facilitate music composition. Our first concrete specifications were the lexical and syntax specifications, which we implemented in the scanner and parser.

5.1.3 Development Process

Our development process closely followed the stages of the compiler architecture. We began our compiler by outlining our scanner, singling out keywords and solidifying DJ's modifiers, operators, functions, expressions and statements. After completing the scanner, we completed the parser, and the AST. The semantic checker (and the SAST) and then the code generator were developed in parallel, adding features sequentially. Most of the early development process focused around group-design and group coding sessions where team members would solve problems and write code collaboratively.

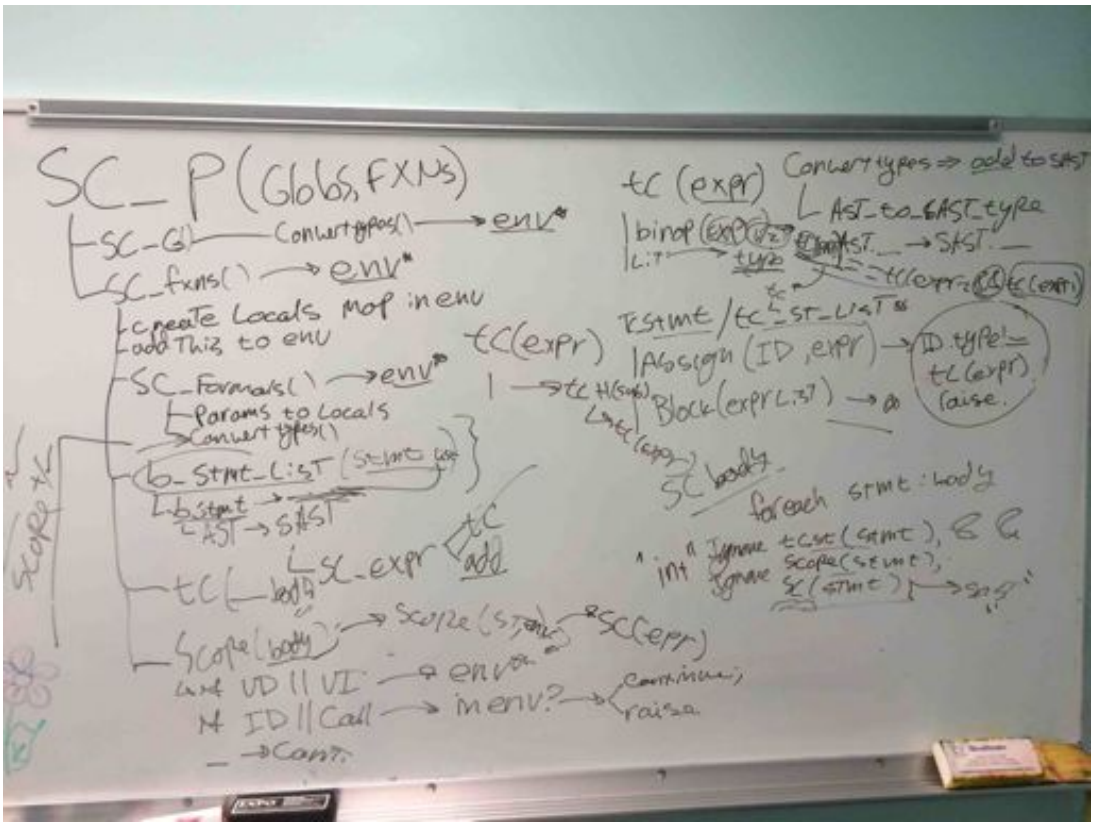

Figure 1: Whiteboarding during project development session

5.1.4 Testing Process

Initially, we developed a test suite before constructing our compiler. In doing this we hoped to guide our development process by identifying the DJ language specifications. We utilized this test suite at every stage in our development process. Occasionally, we modified our test suite as the implementation of some of our features changed. Once we completed our compiler, we added more tests to display certain features of our program.

5.2 Style Guide

5.2.1 Code

- type checking functions take prefix 'type_'

- Sast building functions take prefix 'build_'

- Transition entry functions/print functions should take the form 'string_of_program'

- try to avoid OCaml's '= function', use 'match' explicitly instead; named arguments are clearer to follow

- with 'if' statements or string type-checking, check the negation first

- pull request and assign if you are unsure or it is a major feature.

- have fun

- 'junk' is a valid placeholder for content that will not be used

- try to avoid having to use 'junk'

5.2.2 Git

As far as git/github goes,

1. Use imperative present tense (e.g. fix, add, change) or descriptive present (e.g. fixes, adds, changes)

2. Don't end lines with a period.

3. If you're fixing an issue add "`fixes #xxx`" where xxx is the issue number.

4. If you're referencing an issue add "`#xxx`" where xxx is the issue number.

5. Read SVN best practices document in the *documents* repo, most of it applies (use branch-when-necessary, at the end)

6. setup gitignore so we dont get `.DS_store`, binaries, or other junk

7. COMMENT EVERYTHING

8. Commit often/atomically, but only push working/functioning changesets (see above, 5)

5.2.3 Fun

Geotagging: Set it up if you can. See the [readme][https://github.com/WHET-PLT/fun/blob/master/geotagging.md] in the *fun* repo.

5.2.4 Extras

Some extra Git resources: [Git Concepts Simplified][http://gitolite.com/gcs/index.html]

5.3 Project Timeline

September 9	Group formation
September 25	Language Proposal submitted
October 28	LRM submitted
December 11	Scanner, Parser, AST mostly finalized
December 16	SAST, Semcheck, and Java Gen working
December 19	Presentation given
December 20	Final Report submitted

5.4 Roles

Will	Semantic analysis and type checking, SAST construction, Java generation features, test suite, AST and parser design
Emily	Java generation, semantic analysis, SAST construction, AST and parser design
Tom	Parser and AST design and construction, test suite, semantic and type checking design and features
Hila	Java library research, SAST deconstruction for Java generation, documentation

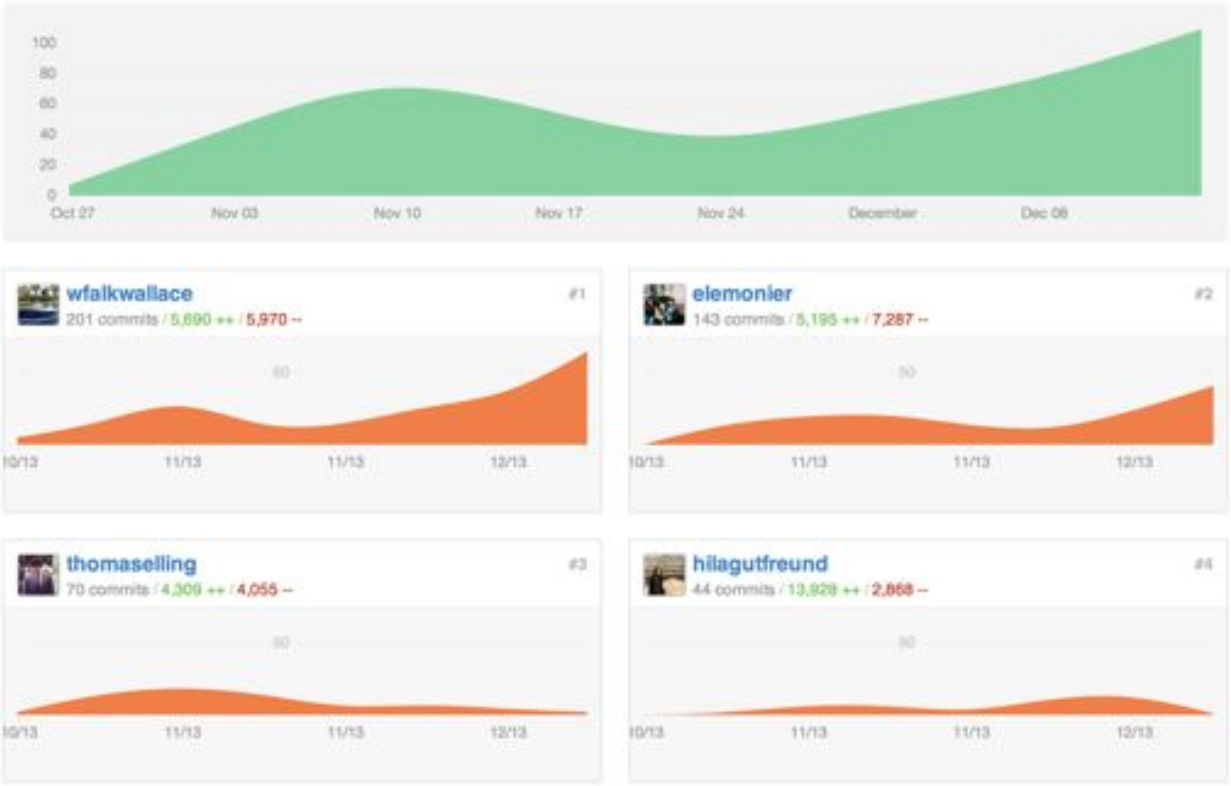

Figure 2: Team Member Commits throughout project

5.5 Tools and Languages

· JMusic: Java MIDI Library

· writelatex.com: real-time co-editing LaTeX documents

· Git/Github: code and documents storage and version control

· Dropbox: references files, binaries, and static/large file transfer and storage

· Google drive: coediting, versioning and storage for management documents and notes

· Google hangouts: remote meetings, work sessions, and discussions

· Sublime Text 2: code editor with OCaml, Java, and Make extensions

· Eclipse: Java IDE for java output code practice

· Ocaml: INRIA OCaml Binaries for translator source

· Java 6: secondary compilation and MIDI production

· Make: automated source compilation

· BASH: autoated tests and Java compilation utility

5.6 Project Log

See Appendix D for a full Git log

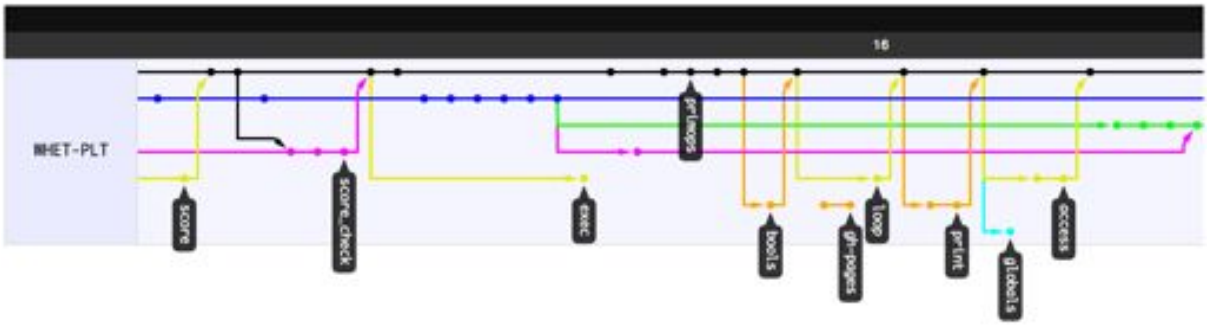

Figure 3: Branch Network Graph Sample

Figure 4: Team Member Commit Locations throughout project

6 Architectural Design

6.1 Language Design Theory

DJ is designed to be a simple yet rich language to create music, which abstracts the difficulty of programming MIDI directly. The language features multiple different data types and control flow statements to support algorithmic manipulation to music that make creating programmatically sophisticated programs simple.

6.2 Structure

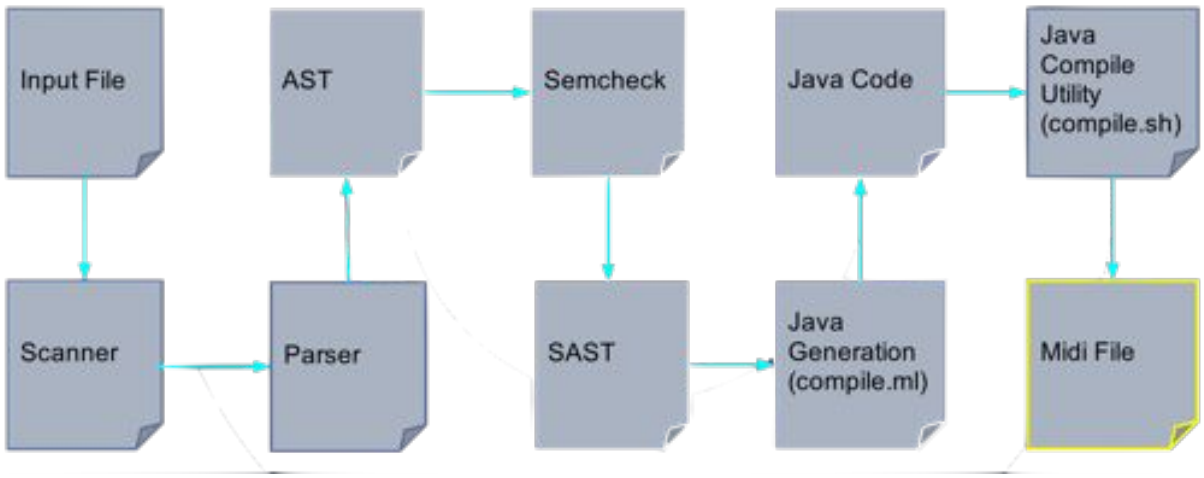

Figure 5: Block Diagram Representing Compiler Structure/Operation Flow

6.3 Interfaces Between Components

6.3.1 Scanner

The scanner takes in a stream of characters and converts them into a token stream for the parser. The tokens are all defined in the scanner.ml and include simple digits, reserved words, control flow statements, operators, and any other statements and tokens that are needed to create the DJ program language.

6.3.2 Parser

The parser takes the stream of tokens from the scanner and matches them against the grammar rules for the language. This is where rules for binary addition operations, constructors, etc. are defined. The parser hands off this set of tokenized strings to the AST.

6.3.3 AST

The AST builds the token string from the parser into an AST data type. This is a syntactically checked data type according to rules in the AST.ml file. The AST is then passed to the SAST.

6.3.4 SAST

The SAST defines the structure of a semantically checked AST.

6.3.5 Semantic checking

Semcheck.ml converts the data from the AST to a SAST where the type and scope of expressions and statements are semantically checked.

6.3.6 Compile

Compile.ml performs Java code generation, converting and SAST into a valid Java source file representing the original DJ program.

6.3.7 Compile Utility

The compile BASH script carries out java compilation and runtime, to carry the java intermediate source file and output MIDI sound files.

6.3.8 wdjc

Includes compile options. Responsible for compilation.

6.4 Implementation Roles

Will	Semantic and type checking design and infrastructure, connecting SAST and semantic checking, AST to SAST conversion
Emily	Global, local, formals, and function semantic checking, SAST name passing for Java consistency, semantic analysis infrastructure
Tom	Statement and Expression semantic checking, parser implementation, semantic analysis infrastructure
Hila	SAST deconstruction into Java

7 Testing

We created a few examples in DJ that we began to implement to see if we could compile the code we had. As we got further into coding development, we used those examples in order to see if we could get through to the AST, SAST, semcheck, and finally java generation through different compilation flags (-A, -S, -J, etc).

7.1 Sample Programs

Several representative DJ language programs, paired with their Java language generated source. The MIDI output of the samples can be found in the source tarball as well as online in the Github repository.[1]

helloWorld.dj

```
song score ()
{
        /*simple note test */

        double pitchA;
        double volume;
        double duration;

        pitchA = 440;
        volume=50;
        duration=4;

        note n = note (pitchA, volume, duration);

        chord c = chord(n);

        track t = track( 0 );

        t = t . c;
```

[1] "DJ Language MIDI test samples," github.com, 20 Dec 2013. <https://github.com/WHET-PLT/wdjc/tree/master/mid>.

```
20
21          score s = score(t);
22
23          return s;
24  }
```

helloWorld.dj - Java Intermediary Representation

```
1  import java.util.*;
2  import jm.JMC;
3  import jm.music.data.*;
4  import jm.util.*;
5
6  public class hellodj implements JMC {
7  public static void main(String[] args){
8           Note [] notes_array;
9
10                  double pitchA;
11                  double volume;
12                  double duration;
13                  pitchA = 440;
14                  volume = 50;
15                  duration = 4;
16                  Note n = new Note((double)pitchA, duration, (int) volume);
17                  CPhrase c =  new CPhrase();
18                  c.setAppend(true);
19                  notes_array = new Note [] {n};
20                  c.addChord(notes_array);;
21                  Part t = new Part( (int) 0);
22                  t =     t;
23  t.addCPhrase(c);
24                  Score s = new Score();
25                  s.addPart(t);
26                  Write.midi(s, "hello.mid");
27
28                  }
29
30  }
```

zelda.dj

```
1  song score () {
2
3          /* higher notes */
4          double C3 = 130.81;
5          double CSharp3 = 138.59;
6          double D3 = 146.83;
7          double DSharp3 = 155.56;
8          double E3 = 164.81;
9          double F3 = 174.61;
10         double FSharp3 = 185;
11         double G3 = 196;
12         double GSharp3 = 207.65;
13         double ASharp3 = 233.08;
14         double C4 = 261.63;
15         double CSharp4 = 277.18;
16         double D4 = 293.66;
17         double DSharp4 = 311.13;
```

```
18          double E4 = 329.63;
19          double F4 = 349.23;
20          double FSharp4 = 369.99;
21          double G4 = 392;
22          double GSharp4 = 415.3;
23          double ASharp4 = 466.16;
24          double C5 = 523.25;
25          double CSharp5 = 554.37;
26          double D5 = 587.33;
27          double DSharp5 = 622.25;
28          double E5 = 659.26;
29          double F5 = 698.46;
30          double FSharp5 = 739.99;
31          double G5 = 783.99;
32
33          /* lower notes */
34          double E2 = 82.41;
35          double F2 = 87.31;
36          double FSharp2 = 92.5;
37          double G2 = 98;
38          double GSharp2 = 103.83;
39          double A2 = 110;
40          double ASharp2 = 116.54;
41          double B2 = 123.47;
42          double A3 = 220;
43          double B3 = 246.94;
44
45
46          double whole = 16/8.0;
47          double half = 8/8.0;
48          double quarter = 4/8.0;
49          double eighth = 2/8.0;
50          double sixteenth = 1/8.0;
51
52
53          double pipe_organ = 19;
54          double piano = 0;
55
56
57          /* GETTIN INTO TREBLE */
58          track treble_clef = track ( pipe_organ );
59
60          /* THE KEY PLAYERS */
61          note as4_10 = note (ASharp4, 100, half + eighth);
62          note f4_2 = note (F4, 100, eighth);
63          note as4_2 = note (ASharp4, 100, eighth);
64          note gs4_1 = note (GSharp4, 100, sixteenth);
65          note fs4_1 = note (FSharp4, 100, sixteenth);
66          note gs4_14 = note (GSharp4, 100, quarter + eighth + half);
67
68          rest wr = rest ( whole );
69
70          note as4_4 = note (ASharp4, 100, quarter);
71          note f4_6 = note (F4, 100, quarter + eighth);
72          note as4_1 = note (ASharp4, 100, sixteenth);
73          note c5_1 = note (C5, 100, sixteenth);
74          note d5_1 = note (D5, 100, sixteenth);
```

```
75          note ds5_1 = note (DSharp5, 100, sixteenth);
76
77          note f5_2 = note (F4, 100, eighth);
78          note f5_8 = note (F4, 100, half);
79
80
81
82          /* CHORDING THEM OUT */
83          chord c1 = chord (as4_10);
84          chord c2 = chord (f4_2);
85          chord c2b = chord (f4_2);
86          chord c3 = chord (as4_2);
87          chord c4 = chord (gs4_1);
88          chord c5 = chord (fs4_1);
89          chord c6 = chord (gs4_14);
90
91          chord c7 = chord (wr);
92          chord c8 = chord (wr);
93          chord c9 = chord (as4_4);
94          chord c10 = chord (f4_6);
95          chord c11 = chord (as4_2);
96          chord c12 = chord (as4_1);
97          chord c13 = chord (c5_1);
98          chord c14 = chord (d5_1);
99          chord c15 = chord (ds5_1);
100
101         chord c16 = chord (f5_2);
102         chord c17 = chord (as4_2);
103         chord c18 = chord (as4_1);
104         chord c19 = chord (c5_1);
105         chord c20 = chord (d5_1);
106         chord c21 = chord (ds5_1);
107         chord c22 = chord (f5_8);
108
109
110         loop (2) {
111                 treble_clef = treble_clef . c1;
112                 treble_clef = treble_clef . c2;
113                 treble_clef = treble_clef . c2b;
114                 treble_clef = treble_clef . c3;
115                 treble_clef = treble_clef . c4;
116                 treble_clef = treble_clef . c5;
117                 treble_clef = treble_clef . c6;
118         }
119
120         treble_clef = treble_clef . c7;
121         treble_clef = treble_clef . c8;
122         treble_clef = treble_clef . c9;
123         treble_clef = treble_clef . c10;
124         treble_clef = treble_clef . c11;
125         treble_clef = treble_clef . c12;
126         treble_clef = treble_clef . c13;
127         treble_clef = treble_clef . c14;
128         treble_clef = treble_clef . c15;
129         treble_clef = treble_clef . c16;
130         treble_clef = treble_clef . c17;
131         treble_clef = treble_clef . c18;
```

```
132         treble_clef = treble_clef . c19;
133         treble_clef = treble_clef . c20;
134         treble_clef = treble_clef . c21;
135         treble_clef = treble_clef . c22;
136
137
138
139
140         /* BASS */
141         track bass_clef = track ( pipe_organ );
142
143         note as2_4 = note (ASharp2, 100, quarter);
144         note f3_4 = note (DSharp3, 100, quarter);
145         note b3_8 = note (ASharp3, 100, half);
146         note gs2_4 = note (GSharp2, 100, quarter);
147         note ds3_4 = note (DSharp3, 100, quarter);
148         note a3_8 = note (ASharp3, 100, half);
149         note fs2_4 = note (FSharp2, 100, quarter);
150         note cs3_4 = note (CSharp3, 100, quarter);
151         note g3_8 = note (ASharp3, 100, half);
152         note f2_4 = note (F2, 100, quarter);
153         note c3_4 = note (C3, 100, quarter);
154         note f3_8 = note (F3, 100, half);
155
156         note f3_2 = note (F3, 100, eighth);
157         note cs3_2 = note (CSharp3, 100, eighth);
158         note as2_2 = note (ASharp2, 100, eighth);
159         note f3_1 = note (F3, 100, sixteenth);
160         note as2_1 = note (ASharp2, 100, sixteenth);
161
162
163         chord b0 = chord (as2_4);
164         chord b1 = chord (f3_4);
165         chord b2 = chord (b3_8);
166         chord b3 = chord (gs2_4);
167         chord b4 = chord (ds3_4);
168         chord b5 = chord (a3_8);
169         chord b6 = chord (fs2_4);
170         chord b7 = chord (cs3_4);
171         chord b8 = chord (g3_8);
172         chord b9 = chord (f2_4);
173         chord b10 = chord (c3_4);
174         chord b11 = chord (f3_8);
175
176         chord b12 = chord (f3_2 , cs3_2);
177         b12 = b12 : as2_2;
178         chord b13 = chord (f3_1 , as2_1);
179         chord b14 = chord (as2_1);
180         chord b15 = chord (f3_2 , as2_2);
181         chord b16 = chord (f3_1 , as2_1);
182         chord b17 = chord (as2_1);
183         chord b18 = chord (f3_2 , as2_2);
184         chord b19 = chord (f3_1 , as2_1);
185         chord b20 = chord (as2_1);
186         chord b21 = chord (f3_1 , as2_1);
187         chord b22 = chord (as2_1);
188         chord b23 = chord (f3_1 , as2_1);
```

```
        chord b24 = chord (as2_1);

        bass_clef = bass_clef . b0;
        bass_clef = bass_clef . b1;
        bass_clef = bass_clef . b2;
        bass_clef = bass_clef . b3;
        bass_clef = bass_clef . b4;
        bass_clef = bass_clef . b5;
        bass_clef = bass_clef . b6;
        bass_clef = bass_clef . b7;
        bass_clef = bass_clef . b8;
        bass_clef = bass_clef . b9;
        bass_clef = bass_clef . b10;
        bass_clef = bass_clef . b11;

        bass_clef = bass_clef . b12;
        bass_clef = bass_clef . b13;
        bass_clef = bass_clef . b14;
        bass_clef = bass_clef . b15;
        bass_clef = bass_clef . b16;
        bass_clef = bass_clef . b17;
        bass_clef = bass_clef . b18;
        bass_clef = bass_clef . b19;
        bass_clef = bass_clef . b20;
        bass_clef = bass_clef . b21;
        bass_clef = bass_clef . b22;
        bass_clef = bass_clef . b23;
        bass_clef = bass_clef . b24;

        score s = score ( treble_clef, bass_clef );
        return s;
}
```

zelda.dj - Java Intermediary Representation

```
import java.util.*;
import jm.JMC;
import jm.music.data.*;
import jm.util.*;

public class zeldadj implements JMC {
public static void main(String[] args){
        Note [] notes_array;

                                double C3 = 130.81;
                                double CSharp3 = 138.59;
                                double D3 = 146.83;
                                double DSharp3 = 155.56;
                                double E3 = 164.81;
                                double F3 = 174.61;
                                double FSharp3 = 185;
                                double G3 = 196;
                                double GSharp3 = 207.65;
                                double ASharp3 = 233.08;
                                double C4 = 261.63;
                                double CSharp4 = 277.18;
                                double D4 = 293.66;
                                double DSharp4 = 311.13;
```

```
                double E4 = 329.63;
                double F4 = 349.23;
                double FSharp4 = 369.99;
                double G4 = 392;
                double GSharp4 = 415.3;
                double ASharp4 = 466.16;
                double C5 = 523.25;
                double CSharp5 = 554.37;
                double D5 = 587.33;
                double DSharp5 = 622.25;
                double E5 = 659.26;
                double F5 = 698.46;
                double FSharp5 = 739.99;
                double G5 = 783.99;
                double E2 = 82.41;
                double F2 = 87.31;
                double FSharp2 = 92.5;
                double G2 = 98;
                double GSharp2 = 103.83;
                double A2 = 110;
                double ASharp2 = 116.54;
                double B2 = 123.47;
                double A3 = 220;
                double B3 = 246.94;
                double whole = 16 / 8.0;
                double half = 8 / 8.0;
                double quarter = 4 / 8.0;
                double eighth = 2 / 8.0;
                double sixteenth = 1 / 8.0;
                double pipe_organ = 19;
                double piano = 0;
            Part treble_clef = new Part( (int) pipe_organ);
            Note as4_10 = new Note((double)ASharp4, half + eighth, (int) 100);
            Note f4_2 = new Note((double)F4, eighth, (int) 100);
            Note as4_2 = new Note((double)ASharp4, eighth, (int) 100);
            Note gs4_1 = new Note((double)GSharp4, sixteenth, (int) 100);
            Note fs4_1 = new Note((double)FSharp4, sixteenth, (int) 100);
            Note gs4_14 = new Note((double)GSharp4, quarter + eighth + half,
               (int) 100);
            Note wr = new Note( REST, whole);
            Note as4_4 = new Note((double)ASharp4, quarter, (int) 100);
            Note f4_6 = new Note((double)F4, quarter + eighth, (int) 100);
            Note as4_1 = new Note((double)ASharp4, sixteenth, (int) 100);
            Note c5_1 = new Note((double)C5, sixteenth, (int) 100);
            Note d5_1 = new Note((double)D5, sixteenth, (int) 100);
            Note ds5_1 = new Note((double)DSharp5, sixteenth, (int) 100);
            Note f5_2 = new Note((double)F4, eighth, (int) 100);
            Note f5_8 = new Note((double)F4, half, (int) 100);
            CPhrase c1 =  new CPhrase();
            c1.setAppend(true);
            notes_array = new Note [] {as4_10};
            c1.addChord(notes_array);;
            CPhrase c2 =  new CPhrase();
            c2.setAppend(true);
            notes_array = new Note [] {f4_2};
            c2.addChord(notes_array);;
            CPhrase c2b =  new CPhrase();
```

```
80                  c2b.setAppend(true);
81                  notes_array = new Note [] {f4_2};
82                  c2b.addChord(notes_array);;
83                  CPhrase c3 =  new CPhrase();
84                  c3.setAppend(true);
85                  notes_array = new Note [] {as4_2};
86                  c3.addChord(notes_array);;
87                  CPhrase c4 =  new CPhrase();
88                  c4.setAppend(true);
89                  notes_array = new Note [] {gs4_1};
90                  c4.addChord(notes_array);;
91                  CPhrase c5 =  new CPhrase();
92                  c5.setAppend(true);
93                  notes_array = new Note [] {fs4_1};
94                  c5.addChord(notes_array);;
95                  CPhrase c6 =  new CPhrase();
96                  c6.setAppend(true);
97                  notes_array = new Note [] {gs4_14};
98                  c6.addChord(notes_array);;
99                  CPhrase c7 =  new CPhrase();
100                 c7.setAppend(true);
101                 notes_array = new Note [] {wr};
102                 c7.addChord(notes_array);;
103                 CPhrase c8 =  new CPhrase();
104                 c8.setAppend(true);
105                 notes_array = new Note [] {wr};
106                 c8.addChord(notes_array);;
107                 CPhrase c9 =  new CPhrase();
108                 c9.setAppend(true);
109                 notes_array = new Note [] {as4_4};
110                 c9.addChord(notes_array);;
111                 CPhrase c10 =  new CPhrase();
112                 c10.setAppend(true);
113                 notes_array = new Note [] {f4_6};
114                 c10.addChord(notes_array);;
115                 CPhrase c11 =  new CPhrase();
116                 c11.setAppend(true);
117                 notes_array = new Note [] {as4_2};
118                 c11.addChord(notes_array);;
119                 CPhrase c12 =  new CPhrase();
120                 c12.setAppend(true);
121                 notes_array = new Note [] {as4_1};
122                 c12.addChord(notes_array);;
123                 CPhrase c13 =  new CPhrase();
124                 c13.setAppend(true);
125                 notes_array = new Note [] {c5_1};
126                 c13.addChord(notes_array);;
127                 CPhrase c14 =  new CPhrase();
128                 c14.setAppend(true);
129                 notes_array = new Note [] {d5_1};
130                 c14.addChord(notes_array);;
131                 CPhrase c15 =  new CPhrase();
132                 c15.setAppend(true);
133                 notes_array = new Note [] {ds5_1};
134                 c15.addChord(notes_array);;
135                 CPhrase c16 =  new CPhrase();
136                 c16.setAppend(true);
```

```
137                 notes_array = new Note [] {f5_2};
138                 c16.addChord(notes_array);;
139                 CPhrase c17 =  new CPhrase();
140                 c17.setAppend(true);
141                 notes_array = new Note [] {as4_2};
142                 c17.addChord(notes_array);;
143                 CPhrase c18 =  new CPhrase();
144                 c18.setAppend(true);
145                 notes_array = new Note [] {as4_1};
146                 c18.addChord(notes_array);;
147                 CPhrase c19 =  new CPhrase();
148                 c19.setAppend(true);
149                 notes_array = new Note [] {c5_1};
150                 c19.addChord(notes_array);;
151                 CPhrase c20 =  new CPhrase();
152                 c20.setAppend(true);
153                 notes_array = new Note [] {d5_1};
154                 c20.addChord(notes_array);;
155                 CPhrase c21 =  new CPhrase();
156                 c21.setAppend(true);
157                 notes_array = new Note [] {ds5_1};
158                 c21.addChord(notes_array);;
159                 CPhrase c22 =  new CPhrase();
160                 c22.setAppend(true);
161                 notes_array = new Note [] {f5_8};
162                 c22.addChord(notes_array);;
163                 for (int w = 0; w < 2; w ++)            {
164                 treble_clef =   treble_clef;
165 treble_clef.addCPhrase(c1);
166                 treble_clef =   treble_clef;
167 treble_clef.addCPhrase(c2);
168                 treble_clef =   treble_clef;
169 treble_clef.addCPhrase(c2b);
170                 treble_clef =   treble_clef;
171 treble_clef.addCPhrase(c3);
172                 treble_clef =   treble_clef;
173 treble_clef.addCPhrase(c4);
174                 treble_clef =   treble_clef;
175 treble_clef.addCPhrase(c5);
176                 treble_clef =   treble_clef;
177 treble_clef.addCPhrase(c6);
178 }
179                 treble_clef =   treble_clef;
180 treble_clef.addCPhrase(c7);
181                 treble_clef =   treble_clef;
182 treble_clef.addCPhrase(c8);
183                 treble_clef =   treble_clef;
184 treble_clef.addCPhrase(c9);
185                 treble_clef =   treble_clef;
186 treble_clef.addCPhrase(c10);
187                 treble_clef =   treble_clef;
188 treble_clef.addCPhrase(c11);
189                 treble_clef =   treble_clef;
190 treble_clef.addCPhrase(c12);
191                 treble_clef =   treble_clef;
192 treble_clef.addCPhrase(c13);
193                 treble_clef =   treble_clef;
```

```
194 treble_clef.addCPhrase(c14);
195                 treble_clef =   treble_clef;
196 treble_clef.addCPhrase(c15);
197                 treble_clef =   treble_clef;
198 treble_clef.addCPhrase(c16);
199                 treble_clef =   treble_clef;
200 treble_clef.addCPhrase(c17);
201                 treble_clef =   treble_clef;
202 treble_clef.addCPhrase(c18);
203                 treble_clef =   treble_clef;
204 treble_clef.addCPhrase(c19);
205                 treble_clef =   treble_clef;
206 treble_clef.addCPhrase(c20);
207                 treble_clef =   treble_clef;
208 treble_clef.addCPhrase(c21);
209                 treble_clef =   treble_clef;
210 treble_clef.addCPhrase(c22);
211                 Part bass_clef = new Part( (int) pipe_organ);
212                 Note as2_4 = new Note((double)ASharp2, quarter, (int) 100);
213                 Note f3_4 = new Note((double)DSharp3, quarter, (int) 100);
214                 Note b3_8 = new Note((double)ASharp3, half, (int) 100);
215                 Note gs2_4 = new Note((double)GSharp2, quarter, (int) 100);
216                 Note ds3_4 = new Note((double)DSharp3, quarter, (int) 100);
217                 Note a3_8 = new Note((double)ASharp3, half, (int) 100);
218                 Note fs2_4 = new Note((double)FSharp2, quarter, (int) 100);
219                 Note cs3_4 = new Note((double)CSharp3, quarter, (int) 100);
220                 Note g3_8 = new Note((double)ASharp3, half, (int) 100);
221                 Note f2_4 = new Note((double)F2, quarter, (int) 100);
222                 Note c3_4 = new Note((double)C3, quarter, (int) 100);
223                 Note f3_8 = new Note((double)F3, half, (int) 100);
224                 Note f3_2 = new Note((double)F3, eighth, (int) 100);
225                 Note cs3_2 = new Note((double)CSharp3, eighth, (int) 100);
226                 Note as2_2 = new Note((double)ASharp2, eighth, (int) 100);
227                 Note f3_1 = new Note((double)F3, sixteenth, (int) 100);
228                 Note as2_1 = new Note((double)ASharp2, sixteenth, (int) 100);
229                 CPhrase b0 =  new CPhrase();
230                 b0.setAppend(true);
231                 notes_array = new Note [] {as2_4};
232                 b0.addChord(notes_array);;
233                 CPhrase b1 =  new CPhrase();
234                 b1.setAppend(true);
235                 notes_array = new Note [] {f3_4};
236                 b1.addChord(notes_array);;
237                 CPhrase b2 =  new CPhrase();
238                 b2.setAppend(true);
239                 notes_array = new Note [] {b3_8};
240                 b2.addChord(notes_array);;
241                 CPhrase b3 =  new CPhrase();
242                 b3.setAppend(true);
243                 notes_array = new Note [] {gs2_4};
244                 b3.addChord(notes_array);;
245                 CPhrase b4 =  new CPhrase();
246                 b4.setAppend(true);
247                 notes_array = new Note [] {ds3_4};
248                 b4.addChord(notes_array);;
249                 CPhrase b5 =  new CPhrase();
250                 b5.setAppend(true);
```

```
251                 notes_array = new Note [] {a3_8};
252                 b5.addChord(notes_array);;
253                 CPhrase b6 =  new CPhrase();
254                 b6.setAppend(true);
255                 notes_array = new Note [] {fs2_4};
256                 b6.addChord(notes_array);;
257                 CPhrase b7 =  new CPhrase();
258                 b7.setAppend(true);
259                 notes_array = new Note [] {cs3_4};
260                 b7.addChord(notes_array);;
261                 CPhrase b8 =  new CPhrase();
262                 b8.setAppend(true);
263                 notes_array = new Note [] {g3_8};
264                 b8.addChord(notes_array);;
265                 CPhrase b9 =  new CPhrase();
266                 b9.setAppend(true);
267                 notes_array = new Note [] {f2_4};
268                 b9.addChord(notes_array);;
269                 CPhrase b10 =  new CPhrase();
270                 b10.setAppend(true);
271                 notes_array = new Note [] {c3_4};
272                 b10.addChord(notes_array);;
273                 CPhrase b11 =  new CPhrase();
274                 b11.setAppend(true);
275                 notes_array = new Note [] {f3_8};
276                 b11.addChord(notes_array);;
277                 CPhrase b12 =  new CPhrase();
278                 b12.setAppend(true);
279                 notes_array = new Note [] {f3_2, cs3_2};
280                 b12.addChord(notes_array);;
281                 b12 =   b12;
282         notes_array = new Note [] {as2_2};
283 b12.addChord( notes_array );
284                 CPhrase b13 =  new CPhrase();
285                 b13.setAppend(true);
286                 notes_array = new Note [] {f3_1, as2_1};
287                 b13.addChord(notes_array);;
288                 CPhrase b14 =  new CPhrase();
289                 b14.setAppend(true);
290                 notes_array = new Note [] {as2_1};
291                 b14.addChord(notes_array);;
292                 CPhrase b15 =  new CPhrase();
293                 b15.setAppend(true);
294                 notes_array = new Note [] {f3_2, as2_2};
295                 b15.addChord(notes_array);;
296                 CPhrase b16 =  new CPhrase();
297                 b16.setAppend(true);
298                 notes_array = new Note [] {f3_1, as2_1};
299                 b16.addChord(notes_array);;
300                 CPhrase b17 =  new CPhrase();
301                 b17.setAppend(true);
302                 notes_array = new Note [] {as2_1};
303                 b17.addChord(notes_array);;
304                 CPhrase b18 =  new CPhrase();
305                 b18.setAppend(true);
306                 notes_array = new Note [] {f3_2, as2_2};
307                 b18.addChord(notes_array);;
```

```
                CPhrase b19 =  new CPhrase();
                b19.setAppend(true);
                notes_array = new Note [] {f3_1, as2_1};
                b19.addChord(notes_array);;
                CPhrase b20 =  new CPhrase();
                b20.setAppend(true);
                notes_array = new Note [] {as2_1};
                b20.addChord(notes_array);;
                CPhrase b21 =  new CPhrase();
                b21.setAppend(true);
                notes_array = new Note [] {f3_1, as2_1};
                b21.addChord(notes_array);;
                CPhrase b22 =  new CPhrase();
                b22.setAppend(true);
                notes_array = new Note [] {as2_1};
                b22.addChord(notes_array);;
                CPhrase b23 =  new CPhrase();
                b23.setAppend(true);
                notes_array = new Note [] {f3_1, as2_1};
                b23.addChord(notes_array);;
                CPhrase b24 =  new CPhrase();
                b24.setAppend(true);
                notes_array = new Note [] {as2_1};
                b24.addChord(notes_array);;
                bass_clef =     bass_clef;
bass_clef.addCPhrase(b0);
                bass_clef =     bass_clef;
bass_clef.addCPhrase(b1);
                bass_clef =     bass_clef;
bass_clef.addCPhrase(b2);
                bass_clef =     bass_clef;
bass_clef.addCPhrase(b3);
                bass_clef =     bass_clef;
bass_clef.addCPhrase(b4);
                bass_clef =     bass_clef;
bass_clef.addCPhrase(b5);
                bass_clef =     bass_clef;
bass_clef.addCPhrase(b6);
                bass_clef =     bass_clef;
bass_clef.addCPhrase(b7);
                bass_clef =     bass_clef;
bass_clef.addCPhrase(b8);
                bass_clef =     bass_clef;
bass_clef.addCPhrase(b9);
                bass_clef =     bass_clef;
bass_clef.addCPhrase(b10);
                bass_clef =     bass_clef;
bass_clef.addCPhrase(b11);
                bass_clef =     bass_clef;
bass_clef.addCPhrase(b12);
                bass_clef =     bass_clef;
bass_clef.addCPhrase(b13);
                bass_clef =     bass_clef;
bass_clef.addCPhrase(b14);
                bass_clef =     bass_clef;
bass_clef.addCPhrase(b15);
                bass_clef =     bass_clef;
```

```
bass_clef.addCPhrase(b16);
                bass_clef =     bass_clef;
bass_clef.addCPhrase(b17);
                bass_clef =     bass_clef;
bass_clef.addCPhrase(b18);
                bass_clef =     bass_clef;
bass_clef.addCPhrase(b19);
                bass_clef =     bass_clef;
bass_clef.addCPhrase(b20);
                bass_clef =     bass_clef;
bass_clef.addCPhrase(b21);
                bass_clef =     bass_clef;
bass_clef.addCPhrase(b22);
                bass_clef =     bass_clef;
bass_clef.addCPhrase(b23);
                bass_clef =     bass_clef;
bass_clef.addCPhrase(b24);
                Score s = new Score();
                s.addPart(treble_clef);
                s.addPart(bass_clef);
                Write.midi(s, "zelda.mid");

                }

}
```

accessor.dj

```
createOtherNote note (note n){
        /* creates + returns new note*/
        double p = n -> pitch + 40;
        double d = n -> dur + 5;
        double v = n -> vol + 10;

        return note(p, v, d);

}

song score () {
        note n1 = note(440, 100, 5);
        note n2 = createOtherNote(n1);
        note n3 = createOtherNote(n2);
        chord c = chord(n1);
        c = c:n2; /* :n3;*/
        c = c:n3;
        /* c = c.n3;*/
        track t = track(26);
        t = t.c;
        score s = score(t);
        return s;
}
```

accessor.dj - Java Intermediary Representation

```
import java.util.*;
import jm.JMC;
import jm.music.data.*;
import jm.util.*;
```

```
5
6  public class onotedj implements JMC {
7  private static Note createOtherNote( Note n )
8  {
9          Note [] notes_array;
10                                 double p = n.getFrequency() + 40;
11                                 double d = n.getDuration() + 5;
12                                 double v = n.getDynamic() + 10;
13                 return new Note((double)p, d, (int) v);
14
15                 }
16 public static void main(String[] args){
17         Note [] notes_array;
18
19                 Note n1 = new Note((double)440, 5, (int) 100);
20                 Note n2 = createOtherNote(n1);
21                 Note n3 = createOtherNote(n2);
22                 CPhrase c =  new CPhrase();
23                 c.setAppend(true);
24                 notes_array = new Note [] {n1};
25                 c.addChord(notes_array);;
26                 c =     c;
27         notes_array = new Note [] {n2};
28 c.addChord( notes_array );
29                 c =     c;
30         notes_array = new Note [] {n3};
31 c.addChord( notes_array );
32                 Part t = new Part( (int) 26);
33                 t =     t;
34 t.addCPhrase(c);
35                 Score s = new Score();
36                 s.addPart(t);
37                 Write.midi(s, "onote.mid");
38
39                 }
40
41 }
```

7.2 Test Suite

7.2.1 DJ Language Test Corpus

for.dj

```
1  song score ( ) {
2
3  /* Simple for loop test */
4
5    double i;
6    for (i = 0 ; i < 5 ; i = i + 1) {
7
8    }
9    score s = score();
10
11   return s;
12 }
```

rest.dj

```
song score ()
{
        /* rest of duration 5  */
        /* make a chord which is a rest + a note */
        rest r;
        chord d;
        double i = 3;
        double a;
        double b;
        double c;


        r = rest (5);
        r = rest (i);


        note n = note (a, b, c);
        d = d : n;

        score s = score();

        return s;


}
```

globs.dj

```
double glob

song score ( ) {

/*
Simple arithmetic test.
Can test comments too.
 */
glob = 5;
print(1+1);

score s = score();

return s;

}
```

score.dj

```
name score()
{
        /*simple note test */

        double pitchA;
        double volume;
        double duration;

        double instr = 40;
```

```
10
11          pitchA = 440;
12          volume=2;
13          duration=4;
14          note n = note (pitchA, volume, duration);
15          chord c = chord(n);
16          track t = track(instr);
17          score s = score(t);
18
19          return s;
20  }
21
22  song score ()
23  {
24          return name();
25  }
```

helloWorld.dj

```
1   song score ()
2   {
3           /*simple note test */
4
5           double pitchA;
6           double volume;
7           double duration;
8
9           pitchA = 440;
10          volume=50;
11          duration=4;
12
13          note n = note (pitchA, volume, duration);
14
15          chord c = chord(n);
16
17          track t = track( 0 );
18
19          t = t . c;
20
21          score s = score(t);
22
23          return s;
24  }
```

serial.dj

```
1   createOtherNote note (note n){
2           /* creates + returns new note*/
3           double p = n -> pitch + 40;
4           double d = n -> dur + 5;
5           double v = n -> vol + 10;
6
7           return note(p, d, v);
8
9   }
10
11  song score () {
12          note n1 = note(440, 5, 100);
```

```
        note n2 = createOtherNote(n1);
        note n3 = createOtherNote(n2);
        chord c = chord(n1);
        c = c:n2; /* :n3;*/
        c = c:n3;
        /* c = c.n3;*/
        track t = track(26);
        t = t.c;
        score s = score(t);
        return s;
}
```

ZELDA.dj

```
song score () {

        /* higher notes */
        double C3 = 130.81;
        double CSharp3 = 138.59;
        double D3 = 146.83;
        double DSharp3 = 155.56;
        double E3 = 164.81;
        double F3 = 174.61;
        double FSharp3 = 185;
        double G3 = 196;
        double GSharp3 = 207.65;
        double ASharp3 = 233.08;
        double C4 = 261.63;
        double CSharp4 = 277.18;
        double D4 = 293.66;
        double DSharp4 = 311.13;
        double E4 = 329.63;
        double F4 = 349.23;
        double FSharp4 = 369.99;
        double G4 = 392;
        double GSharp4 = 415.3;
        double ASharp4 = 466.16;
        double C5 = 523.25;
        double CSharp5 = 554.37;
        double D5 = 587.33;
        double DSharp5 = 622.25;
        double E5 = 659.26;
        double F5 = 698.46;
        double FSharp5 = 739.99;
        double G5 = 783.99;

        /* lower notes */
        double E2 = 82.41;
        double F2 = 87.31;
        double FSharp2 = 92.5;
        double G2 = 98;
        double GSharp2 = 103.83;
        double A2 = 110;
        double ASharp2 = 116.54;
        double B2 = 123.47;
        double A3 = 220;
        double B3 = 246.94;

```

```
45
46          double whole = 16/8.0;
47          double half = 8/8.0;
48          double quarter = 4/8.0;
49          double eighth = 2/8.0;
50          double sixteenth = 1/8.0;
51
52
53          double pipe_organ = 19;
54          double piano = 0;
55
56
57          /* GETTIN INTO TREBLE */
58          track treble_clef = track ( pipe_organ );
59
60          /* THE KEY PLAYERS */
61          note as4_10 = note (ASharp4, 100, half + eighth);
62          note f4_2 = note (F4, 100, eighth);
63          note as4_2 = note (ASharp4, 100, eighth);
64          note gs4_1 = note (GSharp4, 100, sixteenth);
65          note fs4_1 = note (FSharp4, 100, sixteenth);
66          note gs4_14 = note (GSharp4, 100, quarter + eighth + half);
67
68          rest wr = rest ( whole );
69
70          note as4_4 = note (ASharp4, 100, quarter);
71          note f4_6 = note (F4, 100, quarter + eighth);
72          note as4_1 = note (ASharp4, 100, sixteenth);
73          note c5_1 = note (C5, 100, sixteenth);
74          note d5_1 = note (D5, 100, sixteenth);
75          note ds5_1 = note (DSharp5, 100, sixteenth);
76
77          note f5_2 = note (F4, 100, eighth);
78          note f5_8 = note (F4, 100, half);
79
80
81
82          /* CHORDING THEM OUT */
83          chord c1 = chord (as4_10);
84          chord c2 = chord (f4_2);
85          chord c2b = chord (f4_2);
86          chord c3 = chord (as4_2);
87          chord c4 = chord (gs4_1);
88          chord c5 = chord (fs4_1);
89          chord c6 = chord (gs4_14);
90
91          chord c7 = chord (wr);
92          chord c8 = chord (wr);
93          chord c9 = chord (as4_4);
94          chord c10 = chord (f4_6);
95          chord c11 = chord (as4_2);
96          chord c12 = chord (as4_1);
97          chord c13 = chord (c5_1);
98          chord c14 = chord (d5_1);
99          chord c15 = chord (ds5_1);
100
101         chord c16 = chord (f5_2);
```

```
102         chord c17 = chord (as4_2);
103         chord c18 = chord (as4_1);
104         chord c19 = chord (c5_1);
105         chord c20 = chord (d5_1);
106         chord c21 = chord (ds5_1);
107         chord c22 = chord (f5_8);
108
109
110         loop (2) {
111                 treble_clef = treble_clef . c1;
112                 treble_clef = treble_clef . c2;
113                 treble_clef = treble_clef . c2b;
114                 treble_clef = treble_clef . c3;
115                 treble_clef = treble_clef . c4;
116                 treble_clef = treble_clef . c5;
117                 treble_clef = treble_clef . c6;
118         }
119
120         treble_clef = treble_clef . c7;
121         treble_clef = treble_clef . c8;
122         treble_clef = treble_clef . c9;
123         treble_clef = treble_clef . c10;
124         treble_clef = treble_clef . c11;
125         treble_clef = treble_clef . c12;
126         treble_clef = treble_clef . c13;
127         treble_clef = treble_clef . c14;
128         treble_clef = treble_clef . c15;
129         treble_clef = treble_clef . c16;
130         treble_clef = treble_clef . c17;
131         treble_clef = treble_clef . c18;
132         treble_clef = treble_clef . c19;
133         treble_clef = treble_clef . c20;
134         treble_clef = treble_clef . c21;
135         treble_clef = treble_clef . c22;
136
137
138
139
140         /* BASS */
141         track bass_clef = track ( pipe_organ );
142
143         note as2_4 = note (ASharp2, 100, quarter);
144         note f3_4 = note (DSharp3, 100, quarter);
145         note b3_8 = note (ASharp3, 100, half);
146         note gs2_4 = note (GSharp2, 100, quarter);
147         note ds3_4 = note (DSharp3, 100, quarter);
148         note a3_8 = note (ASharp3, 100, half);
149         note fs2_4 = note (FSharp2, 100, quarter);
150         note cs3_4 = note (CSharp3, 100, quarter);
151         note g3_8 = note (ASharp3, 100, half);
152         note f2_4 = note (F2, 100, quarter);
153         note c3_4 = note (C3, 100, quarter);
154         note f3_8 = note (F3, 100, half);
155
156         note f3_2 = note (F3, 100, eighth);
157         note cs3_2 = note (CSharp3, 100, eighth);
158         note as2_2 = note (ASharp2, 100, eighth);
```

```
159         note f3_1 = note (F3, 100, sixteenth);
160         note as2_1 = note (ASharp2, 100, sixteenth);
161
162
163         chord b0 = chord (as2_4);
164         chord b1 = chord (f3_4);
165         chord b2 = chord (b3_8);
166         chord b3 = chord (gs2_4);
167         chord b4 = chord (ds3_4);
168         chord b5 = chord (a3_8);
169         chord b6 = chord (fs2_4);
170         chord b7 = chord (cs3_4);
171         chord b8 = chord (g3_8);
172         chord b9 = chord (f2_4);
173         chord b10 = chord (c3_4);
174         chord b11 = chord (f3_8);
175
176         chord b12 = chord (f3_2, cs3_2);
177         b12 = b12 : as2_2;
178         chord b13 = chord (f3_1, as2_1);
179         chord b14 = chord (as2_1);
180         chord b15 = chord (f3_2, as2_2);
181         chord b16 = chord (f3_1, as2_1);
182         chord b17 = chord (as2_1);
183         chord b18 = chord (f3_2, as2_2);
184         chord b19 = chord (f3_1, as2_1);
185         chord b20 = chord (as2_1);
186         chord b21 = chord (f3_1, as2_1);
187         chord b22 = chord (as2_1);
188         chord b23 = chord (f3_1, as2_1);
189         chord b24 = chord (as2_1);
190
191         bass_clef = bass_clef . b0;
192         bass_clef = bass_clef . b1;
193         bass_clef = bass_clef . b2;
194         bass_clef = bass_clef . b3;
195         bass_clef = bass_clef . b4;
196         bass_clef = bass_clef . b5;
197         bass_clef = bass_clef . b6;
198         bass_clef = bass_clef . b7;
199         bass_clef = bass_clef . b8;
200         bass_clef = bass_clef . b9;
201         bass_clef = bass_clef . b10;
202         bass_clef = bass_clef . b11;
203
204         bass_clef = bass_clef . b12;
205         bass_clef = bass_clef . b13;
206         bass_clef = bass_clef . b14;
207         bass_clef = bass_clef . b15;
208         bass_clef = bass_clef . b16;
209         bass_clef = bass_clef . b17;
210         bass_clef = bass_clef . b18;
211         bass_clef = bass_clef . b19;
212         bass_clef = bass_clef . b20;
213         bass_clef = bass_clef . b21;
214         bass_clef = bass_clef . b22;
215         bass_clef = bass_clef . b23;
```

```
        bass_clef = bass_clef . b24;

        score s = score ( treble_clef, bass_clef );
        return s;
}
```

if.dj

```
/* Test for if statement */

song score ()
{

        double i;
        double j;

        i = 0;
        j = 1;

        /* ID < LIT */
        if( i < 1)
        {
                i = i + 1;
        }

        /* ID > LIT */
        if( 1 > i)
        {
                i = i + 1;
        }

        /* ID == ID */
        if( 0 == 1 )
        {
                i = i + 1;
        }

        /* LIT != LIT */
        if( i != j )
        {
                i = i + 1;
        }

        score s = score();

        return s;

}
```

simple_arith.dj

```
song score ( ) {

/*
Simple arithmetic test.
Can test comments too.
 */
double i = (1+1);
```

```
note n;

score s = score();

return s;

}
```

accessor.dj

```
createOtherNote note (note n){
        /* creates + returns new note*/
        double p = n -> pitch + 40;
        double d = n -> dur + 5;
        double v = n -> vol + 10;

        return note(p, v, d);

}

song score () {
        note n1 = note(440, 100, 5);
        note n2 = createOtherNote(n1);
        note n3 = createOtherNote(n2);
        chord c = chord(n1);
        c = c:n2; /* :n3;*/
        c = c:n3;
        /* c = c.n3;*/
        track t = track(26);
        t = t.c;
        score s = score(t);
        return s;
}
```

incrdecr.dj

```
song score ( ) {

/*
Incr and Decr
 */

 double i;
 i ++;


 score s = score();

 return s;


}
```

simple_arith_print.dj

```
song score ( ) {

/*
```

```
Simple arithmetic test.
Can test comments too.
 */
print(1+1);

score s = score();

return s;

}
```

addressor.dj

```
song score () {

        double index = 2;
        double volume = 1;
        track t = track(0);

        loop(5)
        {

                t = t . chord(  note (241, volume, 3),
                                                note (257, volume, 3),
                                                note (312, volume, 3) );

                volume++;
        }
        print(volume);

        chord c = t[index];

        score s = score( t );

        return s;

}
```

initialize.dj

```
hello track ( ) {
/* Simple note text. */

double pitchA = 60;

return track(5);

}

song score () {


track t = track (13);
score s = score(t);


        return s;
}
```

track.dj

```
song score ()
{
        track t;
        chord c;
        chord d;
        t = t.c;

        score s = score(t);
        return s;

}
```

assign.dj

```

song score ( ) {
/* Simple note text. */

/* individual declaration + assignment */


double pitchA;
double pitchB;
/* pitchA = 60; */


/* declaration + initialization */
pitchB = 900;

score s = score();

return s;

}
```

modifier.dj

```
song score ( ) {

/*
modifiers
 */

note n;

n^;
n~;

 score s = score();
 return s;

}
```

while.dj

```
/* Test for if statement */
```

```
2
3  song score ()
4  {
5
6          double i;
7          double j;
8
9          i = 0;
10         j = 1;
11
12         /* ID < LIT */
13         while( i < 1)
14         {
15                 i = i + 1;
16         }
17
18         /* ID > LIT */
19         while( 1 > i)
20         {
21                 i = i + 1;
22         }
23
24         i = 1;
25         j = 1;
26
27         /* ID == ID */
28         while( i == j )
29         {
30                 i = i + 1;
31         }
32
33         /* LIT != LIT */
34         while( i != j )
35         {
36                 i = j;
37         }
38         /*track t = new track(0);
39         double k = 1;
40         score s = new score(t); */
41         score s = score();
42         return s;
43
44 }
```

chord.dj

```
1  /*
2  Tests chord
3  Note- right now, all declarations must come before
4  ALL inititializations even if they are for different
5  variables.
6
7  Recall: chord must be enclosed by parenthesis
8  */
9
10 song score ()
11         {
12                 note n1;
```

```
                note n2;
                note n3;
                chord p;
                double a;
                double b;
                double c;

                n2 = note ( a, b, c);
                p = chord ( n1 , n2 , n3 );

                score s = score();

                return s;

        }
```

note.dj

```
song score ( ) {

/*
Simple note text.
*/


double pitchA;
double volume;
/* double volume; */
double duration;
note n;
note n1;
note n2;


pitchA = 1;
volume = 2;
duration = 4;


n = note (pitchA, volume, duration);
n1 = note (1.33, 2, 3);
n2 = note (n1 -> pitch, volume, 4);
track t = track (5);
score s = score( t );

return s;

}
```

declare.dj

```
song score ( ) {
/* Simple note text. */

double pitchA;

note mynote;

```

```
score s = score();

return s;

}
```

7.2.2 Java Test Corpus

Java Makefile

```
# the main class name
MAIN=main

# Location of trees.
SOURCE_DIR=src
OUTPUT_DIR=class

# Java tools
JAVA=java
JFLAGS=-classpath $(CLASSPATH)

JAVAC=javac
JCFLAGS=-sourcepath $(SOURCE_DIR) -d $(OUTPUT_DIR) -classpath $(CLASSPATH)

JAVADOC=javadoc
JDFLAGS=-sourcepath $(SOURCE_DIR) -d $(DOC_DIR)

# jMusic Jars
JM_JAR=jMusic/jMusic1.6.4.jar
JM_INSTR=jMusic/inst/

# Set the CLASSPATH
CLASSPATH=$(OUTPUT_DIR):$(JM_JAR):$(JM_INSTR):.

# List the sourcefiles
FILES=$(SOURCE_DIR)/$(MAIN).java

# compile and run default
default: compile run

# Compile the source
.PHONY: compile
compile:
        mkdir -p $(OUTPUT_DIR)
        $(JAVAC) $(JCFLAGS) $(FILES)

# Run the java main
.PHONY: run
run:
        $(JAVA) $(JFLAGS) $(MAIN)

.PHONY: clean
clean:
        rm -rf $(OUTPUT_DIR)/* $(DOC_DIR)/*

# all - Perform all tasks for a complete build
.PHONY: all
```

```
all: default javadoc

.PHONY: cp
cp:
	@echo CLASSPATH='$(CLASSPATH)'
```

Arpeggio1.java

```
import jm.JMC;
import jm.music.data.*;
import jm.music.tools.*;
import jm.util.*;

/**
* This class turns a series of pitches into a repeating arpeggio
* @author Andrew Brown
*/

public class Arpeggio1 implements JMC {

        public static void main(String[] args) {
                new Arpeggio1();
    }

    public Arpeggio1() {
        int[] pitches = {C4, F4, BF4};
        // turn pitches into a phrase
        Phrase arpPhrase = new Phrase();
        for(int i = 0; i < pitches.length; i++) {
            Note n = new Note(pitches[i], SEMIQUAVER);
            arpPhrase.addNote(n);
        }

        // repeat the arpeggio a few times
        Mod.repeat(arpPhrase, 3);
        Mod.repeat(arpPhrase,2);

        // save it as a file
        Write.midi(arpPhrase, "midi/Arpeggio1.mid");

        }
}
```

Chords.java

```
import jm.JMC;

import jm.util.*;
import jm.music.data.*;
import jm.util.*;

/**
 * This class uses the jMusic CPhrase (Chord Phrase)
 * The class generates a chord progression
 * around the cycle of 5ths
 * It uses static methods in the one file.
 * @author Andrew Brown and edited by Hila Gutfreund
 */
```

```
14 public final class Chords implements JMC{
15
16   //private static Score s = new Score("CPhrase class example");
17   private static Part p = new Part("Piano", 0, 0);
18
19   public static void main(String[] args){
20     //Let us know things have started
21     System.out.println("Creating chord progression ... .");
22
23 //     //choose rootPitch notes around the cycle of fifths
24     int rootPitch = 60; //set start note to middle C
25     for (int i = 0; i < 6; i++) {
26       secondInversion(rootPitch);
27       rootPitch += 7;
28       rootPosition(rootPitch);
29
30     }
31
32     // write the score to a MIDIfile
33     Write.midi(p, "midi/Chords.mid");
34   }
35
36   private static void rootPosition(int rootPitch) {
37     // build the chord from the rootPitch
38     int[] pitchArray = new int[4];
39     pitchArray[0] = rootPitch;
40     pitchArray[1] = rootPitch + 4;
41     pitchArray[2] = rootPitch + 7;
42     pitchArray[3] = rootPitch + 10;
43
44     //add chord to the part
45     CPhrase chord = new CPhrase();
46     chord.addChord(pitchArray, C);
47     p.addCPhrase(chord);
48   }
49
50   private static void secondInversion(int rootPitch) {
51     // build the chord from the rootPitch
52     int[] pitchArray = new int[4];
53     pitchArray[0] = rootPitch;
54     pitchArray[1] = rootPitch + 4;
55     pitchArray[2] = rootPitch - 2;
56     pitchArray[3] = rootPitch - 5;
57     //add chord to the part
58     CPhrase chord = new CPhrase();
59     chord.addChord(pitchArray, C);
60     p.addCPhrase(chord);
61   }
62 }
```

CreateChord.java

```
1 import java.util.ArrayList;
2 import jm.JMC;
3 import jm.music.data.*;
4 import jm.music.tools.*;
5 import jm.util.*;
6
```

```
/**
* This class turns a series of integers into notes.
* @author Hila Gutfreund
*/

public class CreateChord implements JMC {

  public static void main(String[] args) {
  ArrayList<Integer> notes = new ArrayList<Integer>();
   notes.add(440.0);
   notes.add(650.0);
   notes.add(69.0);

   new CreateChord(notes);


   public CreateChord(ArrayList<Notes> jnotes){
    CPhrase chordPhrase = new CPhrase();
    Part p = new Part();
    for(note: jnotes){
      Note n = new Note (note, 0.5, 0.5);
      chordPhrase.addNote(n);
    }


    p.addPhrase(notePhrase);

    Write.midi(p, "midi/creatChord.mid");
  }
}
```

createNotes.java

```

import jm.JMC;
import jm.music.data.*;
import jm.music.tools.*;
import jm.util.*;

/**
* This class turns a series of integers into notes.
* @author Hila Gutfreund
*/

public class createNotes implements JMC {

  public static void main(String[] args) {
   int[] notes = {30, 250, 54};
   new createNotes(notes);
    }

  public createNotes(int[] notes){
    Phrase notePhrase = new Phrase();
    for(int note:notes){
      if((note >= 0) && (note <= 127)){
        Note n = new Note(note,1.0);
        notePhrase.addNote(n);

```

```
      }else if(note > 127){
        Note n = new Note ((double)note, 0.5);
        notePhrase.addNote(n);
      }else{
        Note n = new Note ((double)note, 0.5);
        notePhrase.addNote(n);
      }
    }
    Write.midi(notePhrase, "midi/createNotes.mid");
  }
}
```

CreateNotesFromFreq.java

```
import java.util.ArrayList;
import jm.JMC;
import jm.music.data.*;
import jm.music.tools.*;
import jm.util.*;

/**
* This class turns a series of integers into notes.
* @author Hila Gutfreund
*/

public class CreateNotesFromFreq implements JMC {

  public static void main(String[] args) {
  // ArrayList<Integer> notes = new ArrayList<Integer>();
  //  notes.add(440);

   new CreateNotesFromFreq();
   //  }

   public CreateNotesFromFreq(){
    Phrase notePhrase = new Phrase();
    Part p = new Part();
    Note n = new Note ((440*1.0), 0.5);
    notePhrase.addNote(n);

    p.addPhrase(notePhrase);

    Write.midi(p, "midi/createNotesFreq.mid");
  }
}
```

CreateScore.java

```
import java.util.ArrayList;
import jm.JMC;
import jm.music.data.*;
import jm.music.tools.*;
import jm.util.*;

/**
* This class turns a series of integers into notes.
* @author Hila Gutfreund
*/
```

```
11
12 public class CreateChord implements JMC {
13
14   public static void main(String[] args) {
15   ArrayList<Integer> notes = new ArrayList<Integer>();
16    notes.add(440.0);
17    notes.add(650.0);
18    notes.add(69.0);
19
20    ArrayList<Integer> notes2 = new ArrayList<Integer>();
21   notes.add(440.0);
22    notes.add(250.0);
23    notes.add(69.5);
24
25
26    new CreateChord(notes);
27
28
29    public CreateChord(ArrayList<Notes> jnotes){
30     Score theScore = new Score();
31     CPhrase chordPhrase1 = new CPhrase();
32     Part p1 = new Part("piano", PIANO, 0);
33     CPhrase chordPhrase2 = new CPhrase();
34     Part p2 = new Part("piano", PIANO, 1);
35     for(note: jnotes){
36       Note n = new Note (note, 0.5, 0.5);
37       chordPhrase1.addNote(n);
38     }
39     p.addPhrase(chordPhrase1);
40
41     for(note: notes2){
42       Note n = new Note (note, 0.5, 0.5);
43       chordPhrase2.addNote(n);
44     }
45     p2.addPhrase(chordPhrase2);
46
47     score.add(p1);
48     score.add(p2);
49
50     Write.midi(theScore, "midi/PartCreate.mid");
51   }
52 }
```

DJ.java

```
1 import java.util.*;
2 import jm.JMC;
3 import jm.music.data.*;
4 import jm.util.*;
5
6 public class DJ implements JMC{
7
8 public static void main(String[] args){
9 double pitchA;
10 double volume;
11 double duration;
12 pitchA = 440;
13 volume = 100;
```

```
duration = 4;
Note n = new Note((double)pitchA, duration, (int) volume);
CPhrase c =  new CPhrase();
Note [] notes_array = {n};
c.addChord(notes_array);
Part t =  new Part();
t.addCPhrase(c);

Score s =  new Score();
s.addPart(t);
Write.midi(s, "createNotes.mid");
}
}
```

midiPLTTest.java

```
import javax.sound.midi.*;

public class midiPLTTest {

public static void main(String[] args) {
midiPLTTest mini = new midiPLTTest();
if (args.length < 2) {
System.out.println("Don't forget the instrument and note args");
} else {

//sound-synthesis algorithm with certain parameter settings usually emulate specific real world instruments.
  //stored in collection (soundbanks)
  //must first be loaded onto synthesizer and then it must be selected for use on one more channels
int instrument = Integer.parseInt(args[0]);


int note = Integer.parseInt(args[1]);
mini.play(instrument,note);
}
} // close main

//method that plays the note
public void play(int instrument, int note) {
try {

//sequencer is a hardware or software device that plays back a midi
//sequence.

Sequencer player = MidiSystem.getSequencer();

//
player.open();

//sequence is a data structure containing musical info (song or composition) that
//can be played back by a sequencer. it contains timing info and one or more tracks.
//PPQ == the tempo based timing tpe for which the resolution is expressed in pulses (ticks) per quarter note
Sequence seq = new Sequence(Sequence.PPQ, 4);

//an independent stream of midi events that can be stored along with other tracks
```

```
// with other tracks in a midi file. a midii file can contina any number of trakcs.
Track track = seq.createTrack();

//events contain a midi message and a corresponding time-stamp expressed in time ticks and can be represented the midi event info
//stored in a midi file or a sequence object. the duration of a tick is specified by a timing info contained in the midi file or seq obj
MidiEvent event = null;


ShortMessage first = new ShortMessage();
first.setMessage(192, 1, instrument, 0);
MidiEvent changeInstrument = new MidiEvent(first, 1);
track.add(changeInstrument);

//shortmessage baiscally allows you to put put in midi data bytes
ShortMessage a = new ShortMessage();
a.setMessage(144, 1, note, 100); //sets the parameters for message: takes up to two bytes?
//command for note on message
MidiEvent noteOn = new MidiEvent(a, 1);
track.add(noteOn);

ShortMessage b = new ShortMessage();
b.setMessage(128, 1, note, 100);
//command for note off message
MidiEvent noteOff = new MidiEvent(b, 16);
track.add(noteOff);

player.setSequence(seq);
player.start();
} catch (Exception ex) {ex.printStackTrace();}
} // close play

} // close class
```

RowYourBoat.java

```
import jm.JMC;
import jm.music.data.*;
import jm.util.*;
import jm.music.tools.*;
/**
 * Plays a melody as a round in three parts
 * @author Andrew Sorensen and Andrew Brown with comments for understanding by Hila
 *
 *
 * Took this so that we could understand looping.
 */

public final class RowYourBoat implements JMC{

  public static void main(String[] args){
    //Create the data objects we want to use
    Score score = new Score("Row Your Boat");
    //Parts can have a name, instrument, and channel.
    Part flute = new Part("Flute", FLUTE, 0);
    Part trumpet = new Part("Trumpet", TRUMPET, 1);
```

```
21      Part clarinet = new Part("Clarinet", CLARINET, 2);
22
23      //Lets write the music in a convenient way.
24
25      //these are the actual notes. This is not how we want to write ours
26      //but for the purpose of learning loops its ok.
27      int[] pitchArray = {C4,C4,C4,D4,E4,E4,D4,E4,F4,G4,C5,C5,C5,G4,G4,G4,E4,E4,E4,
28              C4,C4,C4,G4,F4,E4,D4,C4};
29
30      //this is rythm. this is quarter notes QT, whole notes C, etc.
31      double[] rhythmArray = {C, C,CT,QT,C,CT,QT,CT, QT, M, QT, QT, QT, QT, QT,
32              QT, QT, QT, QT, QT, QT, QT, CT, QT, CT, QT, M};
33      //add the notes to a phrase
34      Phrase phrase1 = new Phrase(0.0);
35      phrase1.addNoteList(pitchArray, rhythmArray);
36
37      //Make two new phrases and change start times to make a round
38      Phrase phrase2 = phrase1.copy();
39      phrase2.setStartTime(4.0);
40      Phrase phrase3 = phrase1.copy();
41      phrase3.setStartTime(8.0);
42
43      //Play different parts in different octaves
44      // mod == A utility class that handles the modification of the basic jMusic
            types.
45      Mod.transpose(phrase1, 12);
46      Mod.transpose(phrase3, -12);
47
48      //loop phrases once
49      //Makes the CPhrase n times as long by repeating.
50      Mod.repeat(phrase1, 1);
51      Mod.repeat(phrase2, 1);
52      Mod.repeat(phrase3, 1);
53
54      //add phrases to the parts
55      flute.addPhrase(phrase1);
56      trumpet.addPhrase(phrase2);
57      clarinet.addPhrase(phrase3);
58
59      //add parts to the score
60      score.addPart(flute);
61      score.addPart(trumpet);
62      score.addPart(clarinet);
63
64      //OK now we do a SMF write
65      Write.midi(score, "midi/rowboat.mid");
66    }
67  }
```

TwoParts.java

```
1  import jm.JMC;
2  import jm.util.*;
3  import jm.music.data.*;
4  import jm.util.*;
5
6  /**
7   * This class uses the jMusic CPhrase (Chord Phrase)
```

```
 * @author Hila Gutfreund
 */
public final class TwoParts implements JMC{

  private  Score s = new Score("CPhrase class example");
  private  Part piano = new Part("Piano", 0, 0);
  private Part bassPart = new Part("left hand", 0, 1);
  //private double[] rhythms = new double[] {0.25, 0.5, 1.0, 2.0, 4.0};
  //find out what rythms are!


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

  public TwoParts() {
    int rootPitch = 60; //set start note to middle C
    for (int i = 0; i < 6; i++) {
      firstPart(rootPitch);
      rootPitch -= 7;
      secondPart(rootPitch);
    }

      //pack the part into a score
      s.addPart(piano);
      s.addPart(bassPart);

      // write the score to a MIDIfile
      Write.midi(s, "midi/TwoParts.mid");
    }


    private void firstPart(int rootPitch){
      // build the chord from the rootPitch
      int[] pitchArray = new int[4];
      pitchArray[0] = rootPitch;
      pitchArray[1] = rootPitch + 4;
      pitchArray[2] = rootPitch + 7;
      pitchArray[3] = rootPitch + 10;
      CPhrase chord = new CPhrase();
      chord.addChord(pitchArray, C);
      piano.addCPhrase(chord);
    }

    private void secondPart(int rootPitch){
      // build the chord from the rootPitch
      int[] pitchArray = new int[4];
      pitchArray[0] = rootPitch;
      pitchArray[1] = rootPitch + 4;
      pitchArray[2] = rootPitch + 7;
      pitchArray[3] = rootPitch + 10;
      CPhrase chord = new CPhrase();
      chord.addChord(pitchArray, C);
      bassPart.addCPhrase(chord);
    }

  }
```

7.3 Test Suite Explanation

Test cases were chosen for two purposes: either to define minimum working examples for unit testing specific features, or to extend and mix several features to test the limits of their capabilities.

7.4 Testing Automation

The following testing script was used to perform automated build and run tests for each level of functionality of the DJ Language translator.

test (.sh)

```
#!/bin/bash
WDJC="./wdjc"

# make the executables
echo ">> Making WDJC into" $(pwd) "<<"
echo
make
echo

# Set time limit for all operations
ulimit -t 30

#check for command line args (flags) can be -a, -s, or -j for now; default to a
#if flag is a - AST
if [[ $1 =~ '-a' ]]; then
        echo ">> Compiling AST <<"
        echo
        for file in ./tests/*.dj
        do
                echo $file
                echo ----------------------------
                $WDJC -a < $file
                echo
        done
        echo ">> Done Compiling AST <<"
        echo
#if flag is s - SAST
elif [[ $1 =~ '-s' ]]; then
        echo ">> Compiling SAST <<"
        echo
        for file in ./tests/*.dj
        do
                echo $file
                echo ----------------------------
                $WDJC -s < $file
                echo
        done
        echo ">> Done Compiling SAST <<"
        echo
#if flag is j - JAVA
elif [[ $1 =~ '-j' ]]; then
        echo ">> Compiling JAVA <<"
        echo
        for file in ./tests/*.dj
        do
                name=$(basename $file .dj)
                echo $file
```

```
48                  echo ----------------------------
49                  $WDJC -j $name < $file
50                  echo
51          done
52          echo ">> Done Compiling JAVA <<"
53          echo
54  #if flag is j - JAVA
55  elif [[ $1 =~ '-c' ]]; then
56          echo ">> Compiling <<"
57          echo
58          for file in ./tests/*.dj
59          do
60                  name=$(basename $file .dj)
61                  echo $file
62                  echo ----------------------------
63                  $WDJC -c $name < $file
64                  echo
65          done
66          echo ">> Done Compiling <<"
67          echo
68
69  #if no flag, default to ast
70  else
71          echo ">> Compiling AST <<"
72          echo
73          for file in ./tests/*.dj
74          do
75                  echo $file
76                  echo ----------------------------
77                  $WDJC -a < $file
78                  echo
79          done
80          echo ">> Done Compiling AST <<"
81          echo
82  fi
83
84  # cleanup
85  echo ">> Cleaning Up <<"
86  make clean
```

7.5 Roles

Will	Team Leader, AST, SAST, Java tests, Testing Suite
Emily	AST, SAST, Java tests
Tom	AST, SAST tests, Testing Suite
Hila	Jmusic example tests, Java tests

8 Lessons Learned and Advice to Future Students

8.1 Will

The most important lesson is the fact that to build a language once, you have to leave time to build it twice. The project is daunting and your language will not be well designed: it is tough to design a language without knowing the limitations of a compiler. It's only at the end of the process that you will realize how the compiler should have actually been built. You understand the errors you made along the way and you know how the language and should

have been designed. As a corollary, dont pick a language because it looks easy and dont shy from a language that seems complicated. Once you understand how to build the compiler, you may see the metrics you originally thought important reversed.

8.2 Hila

The earlier you start the better. It is also good to spend a good deal of time designing the language and understanding the limitations that you may have. The worst part is when you realize that you cannot do something that was an essential part of the language. Know what language you are generating to and understand the libraries you are using well to make sure that this does not happen. However, I think the most important thing is to understand the field you are developing the language very well. I understand some music, but I definitely had to learn and read a lot to get to a firm enough understanding to proceed with language creation and basic design decisions. Finally, sometimes coding together is hard, but it is definitely worth it. Some issues are really hard to resolve on your own and its good when you have someone to talk through issues with.

8.3 Emily

It is essential for every member to be, at least, tangentially involved with every aspect of the project. This makes it easier to distribute tasks across the group since all members of the groups understand the program as a whole. For instance, originally Tom and I were working on different subsections separately from each other and the group as a whole. Three weeks later, we completely redesigned our semcheck infrastructure completely with the help of Will and Hila. After the redesign, all group members understood the semcheck code, so debugging and small redesigns went much quicker. Eventually, we all had a decent grasp of the entire code so that it was easier to distribute tasks throughout the group, because we werent so compartmentalized. The more the group understands of the entire project, even if each member is working on something else, the better. In addition, be proactive and know your way around the entire project from the get go. The earlier everyone is on board the better – I wish I had involved my team team members in my sub-projects earlier.

8.4 Thomas

Start early! Every group will say this in their final report and it is good advice. Just as important as that is spend sufficient time in designing the language all the way through. This means detailing not only the basic functions of the language but also the features of the language. This will help you not only understand the language you will have to eventually code for but also resolve ambiguities of how something should work when you are coding. Also, meet in person if possible. This creates wonderful discussions (arguments) about a certain feature or how something should be done. This creates a robust and more interesting language.

9 Appendix A: The DJ Language Proposal

COMSW4115: Programming Languages and Translators
The DJ Language: MIDI Synthesizer Language Proposal

William Falk-Wallace (wgf2104), Hila Gutfreund (hg2287),
Emily Lemonier (eql2001), Thomas Elling (tee2103)

September 25, 2013

Contents

1 Purpose

The goal of our project is to create a programmatic control interface for the Musical Instrument Digital Interface Specification (MIDI). MIDI is a technology standard that allows a wide variety of electronic musical instruments, computers, and other related devices to connect and communicate with one another.[1] Through the specification of this programming language, called The DJ Language (extension `.dj`), we are able to bring synthesized electronic music production as well as musical score design capabilities directly to an artist's computer.

2 Overview

We propose a procedural scripting language, DJ, which provides a programming paradigm for algorithmic music production. Through its utilization of themes and motifs, music is naturally repetitive and often dynamic. DJ provides control-flow mechanisms, including `for` and `loop` functions, which simplify the development of structured iterative music. The DJ Language also makes use of conditional logic and offers built-in effects (including pitch bend, tremolo and vibrato). Moreover, it supports extensible sound banks to facilitate the production of deeply textured musical compositions. Our goal in the specification of The DJ Language is to abstract away the intricacies and limitations of the MIDI specification, including channeling, patch-maps and instrumentation, allowing the artist to focus on her or his work: composing songs.

3 Features

- Note, Chord, and Track are defined as primitives and are hierarchical. The hierarchy is as follows: Tracks are composed of Chords, which are composed of Notes and Rests.
- Notes are represented by ordered seven-tuples defining characteristic attributes, including pitch, instrumentation, volume, duration (in beats), the presence of effects including tremolo, vibrato, and pitch bend. The primitive Rest object allows for a pause in a Track.
- Tracks, Chords, and Notes may be added in series or parallel. A new Track is produced by adding Tracks in series or parallel. Chords produce Tracks when added in series. Notes added produce Chords when added in parallel.
- Several mutative operators exist for manipulating Note attributes at the Note, Chord, and Track level.
- All programs consist of a single main function, called `SONG`, that returns an array of tracks, intended to start simultaneously and be played in parallel. Each array element can be considered as a polyphonic MIDI channel. This array of tracks is compiled into a bytecode file containing the complete set of MIDI-messages required to produce the programmed song. A third party bytecode-to-MIDI interpreter will be used to produce the final sound file.
- Song-wide properties are specified to the compiler. Attributes such as tempo/beats per minute and channel looping are available as compiler options.
- This structure, as well as the use of the MIDI specification and interface, allows for a fairly extensible language and production capability. For example, through the manipulation or linking of sound banks, new sounds and samples are able to be incorporated to produce rich and interesting programmatic music.

[1] "MIDI Overview" MIDI.org, 21 Sep 2013. Web. 24 Sep 2013. <http://www.midi.org/aboutmidi/tut_midimusicsynth.php>.

4 Syntax

The following subsections and tables represent the primitives, operators, and functions defined in the DJ Language specification.

4.1 Primitives

Integer	Used for addressing and specifying Note/Chord/Track attributes.
Array	Fixed-length collection of elements (int, Note, Chord, Track), each identified by at least one array index.
Note	Ordered tuple containing pitch (pitch), instrument (instr), volume (vol), duration (dur), tremolo (trem), vibrato (vib), pitch bend (pb) (n.b. pitch number is sequentially numbered in tonal half-step increments; tremolo and vibrato attributes are boolean).
Rest	A durational note with no volume and no pitch and which is not responsive to pitch, volume, or effect operations.
Chord	Vector of Notes (size ≥ 1).
Track	Vector of Chords (size ≥ 1).

4.2 Operators

$>, <$	Pitchbend: changes the pitch bend of a Note, the Notes of a Chord, or all Notes of a Track. (binary)
$+, -$	Increase/Decrease pitch of an individual note, all Notes in a Chord, or all Notes in a Track, respectively, by a specified amount. (binary)
$++, --$	Increase/Decrease respective pitch of Notes, either atomically or in a Chord or Track by a single integer increment (tonal half-step). (unary)
[<int>]	Address Array, Chord, or Track element at given index. (unary)
$\sim$	Creates a tremelo effect on the individual note, all Notes in the Chord, or all Notes in the Track that it operates on. (unary)
$\wedge$	Creates a vibratro effect on the individual note, all Notes in the Chord, or all Notes in the Track that it operates on. (unary)
:	Parallel Add: adds Notes, Chords, or Tracks in parallel. When used on Notes, returns a new Chord containing both Notes; when used on Chords, returns a new Chord representing the union of both original Chords; when used with Tracks, returns a new Track such that Chords are added in parallel by corresponding time tick, with no added offset. (binary)
.	Serial Add: both operands must be Tracks. The right operand is concatenated to the first, and a third, new Track is returned. Notes are elevated to size-one Chords and Chords are elevated to Tracks before concatenating. (binary)
$=$	Assignment operator. (binary)
$+=$	Integer Add-in-place. (binary)
$\vert$	Conditional OR. (binary)
&	Conditional AND. (binary)
$==$	Logical equality (deep). (binary)

4.3 Functions

vol(<int>)	Change Chord/Note/Track volume (integer value 0-99). (absolute)
dur(<int>)	Change Chord/Note duration (number of beats). (absolute)
loop(<int>)	Loops a given Note, Chord, or Track the over number of beats specified. If given a number of beats fewer than the total track size (n.b. implicit elevation occurs as necessary), first <int> beats will be included.
repeat(<int>)	Repeats a given Note, Chord, or Track <int> times, returning a new Track.
add(<chord>)	Adds a Chord to a Track.
strip(<chord>)	Removes all instances of Chord from a Track.
remove(<int>)	Removes Chord from Track at designated location.

4.4 Reserved Words and Conditionals

if (*expr*) {...} else {...}	Paired control flow statement that acts upon the logical expression within the `if` statement parentheses. If the expression evaluates to true, the control flow will continue to the code contained within the braces of the `if` body. If the argument is false, then control flow moves on to the code in the braces of the `else` body.
return	Terminates control flow of the current function and returns control flow to the calling function, passing immediately subsequent primitive to calling function.
null	Undefined object identifier; used in declaring non-`return`ing functions.
int, Array Note, Rest, Chord, Track	Type declaration specifiers.
SONG {}	Conventional "main" function declaration, with unspecified return type, which indicates program outset to the compiler.

5 Examples

5.1 Example 1: Arpeggio

```
//the main function
SONG {
        s = Track[1];
        s[0] = t;

        num_beats = 1;
        c = 60;
        vol = 50;
        piano = 1;

        //a for loop
        for(i = 0; i <= 8; i++) {
                //make a new note with incremental pitch
                Note n = {c + i, piano, vol, num_beats, 0, 0, 0};
                //concatenate that note to the first (only) track of the song
                s[0].n;
        }
}
```

5.2 Example 2: Loop With Effects

```
Track loopEffects () {

        int pitchA = 60; //pitch of a will be middle C
        int pitchB = 62; //up a full step for b
        int pitchC = 65; // up a step and a half for a minor/dissonant something
        int volume = 50; //volume 50 - right in the middle
        int instr = 1; //use a piano -- mapped instrument 1
        int duration = 2;

        Note a, b, c;
        a = {pitchA, instr, volume, duration, 0, 0, 0};
        b = {pitchB, instr, volume, duration, 0, 0, 0};
        c = {pitchC, instr, volume, duration, 0, 0, 0};

        Chord ch = a : b : c;

        Track t = ch.repeat(50);

        for(int i = 0; i < t.size(); i += 2) { //iterate over every other chord in t
                t[i][0]~; //for every other chord in t, add a tremolo to the 0th Note
                t[i+1][0].vol(t[i+1][0].vol + 5); //for the rest of the chords, increase its v
        }
        return t;
}
```

5.3 Example 3: Add/Remove Notes & Chords

```
null reverseAddFancy{
        //create tracks track, adds and remove chords
        Note a, b, c, d, e, f;

        //the note pitches
        int midC = 60; //pitch 60 is usually around middle C
        int upabit = 62;
        int downabit = 40;
        int sumthinElse = 88;
        int lyfe = 42;

        //some other note attributes
        int volume = 20; //nice and quiet
        int oh = 47; //use an Orchestral Harp -- General MIDI mapping
        int shortish = 2;
        int longer = 5;

        //define the notes
        a = {midC, oh, volume, shortish};
        b = {lyfe, oh, volume, longer};
        c = {sumthinElse, oh, volume, longer};

        d = {upabit, oh, volume, shortish};
        e = {downabit, oh, volume, longer};
        f = {midC, oh, volume, shortish};


        Chord newChord = a : b : c; //parallel add to make a chord
        Chord oldChord = d : (f : e);
        Track newTrack = newChord.oldChord; //add track with serial add
        newTrack.strip(newChord); //remove all instances of specific chord
        newTrack.newChord; // add newChord back;
        newTrack.remove(0); // removes oldChord;
        newTrack[0] < 5; //pitchbend newChord up 5
}
```

10 Appendix B: Language Reference Manual

COMSW4115: Programming Languages and Translators
The DJ Language Reference Manual

William Falk-Wallace (wgf2104), Hila Gutfreund (hg2287),
Emily Lemonier (eql2001), Thomas Elling (tee2103)

December 21, 2013

Contents

1 Introduction

We propose a procedural scripting language, DJ, which provides a programming paradigm for algorithmic music production. Through its utilization of themes and motifs, music is naturally repetitive and often dynamic. DJ provides control-flow mechanisms, including `for` and `loop` functions, which simplify the development of structured iterative music. The DJ Language also makes use of conditional logic and offers built-in effects (including pitch bend, tremolo and vibrato). Our goal in the specification of The DJ Language is to abstract away the intricacies and limitations of the MIDI specification, including channeling, patch-maps and instrumentation, allowing the artist to focus on her or his work: composing music.

2 Lexical Conventions

2.1 Comments

Comments are initialized by the character sequence `/*` and terminated by the first following character sequence `*/`.

2.2 Identifiers

An identifier is a sequence of letters, underscores and digits; note that in identifiers, uppercase and lowercase letters correspond to different characters. The first character of an identifier is a letter [‘a’-‘z’] or [‘A’ - ‘Z’].

2.3 Keywords

Keywords are reserved identifiers and may not be redefined. They are used for control structure, constants, as well as system level function calls.

`double`	`note`	`rest`
`chord`	`track`	`score`
`song`	`if`	`else`
`for`	`return`	`loop`
`print`	`vol`	`dur`
`pitch`		

2.4 Separators

A separator distinguishes tokens. White space is a separator and is discussed in the next section, but it is not a token. All other separators are single-character tokens. These separators include `( ) { } ;`. Note that `;` is used to indicate the end of an expression or statement.

2.5 White Space

White space collectively refers to the space character, the tab character, and the newline character. White space is used to separate tokens and is otherwise ignored. Wherever one space is allowed, any length of white space is allowed. For example: $y{=}x{+}5$ is equivalent to $y{\sqcup}{=}{\sqcup}{\sqcup}x{\sqcup}{+}{\sqcup}5$, since y, x, 5, +, and = are all complete tokens.

3 Fundamental Data Types

DJ supports inline initialization for all data structures.

3.1 Doubles

A double is a primitive data type which represents some finite subset of the mathematical numbers. If ‘-’ is prepended to the double, the value of the double is considered negative (ex: `double x = -22`). A double may take a value that can be represented by 64 bits. DJ supports values with any number of digits and an optional decimal point. (Ex: 4.0, 2, 2., .0003) The double can be declared and initialized as so: `double x = 423.3`. The benefit of the double data type is that it can specify pitch frequencies according to the MIDI standard. This allows for greater precision than a simple integer.

3.2 Note

Note literals are atomic structures representing characteristic attributes of a musical note including pitch, volume, and duration (in beats or time-ticks). Notes are the most basic musical data type in DJ and take three formal arguments in this order: pitch, vol (volume), dur (duration). Note attributes may be retrieved throughout the program with an accessor (eg: `double p = n -> pitch;`). The accessor can only be used with the key words "pitch", "vol", and "dur".
To construct a note you must use the following syntax: `note n = note (pitch, volume, duration);`. Be aware that the arguments to a note must be expressions that evaluate to a double. Any other type of expression (including another note creation) will not be accepted and will return a compilation error.

3.3 Rest

A rest literal is an atomic unit of a composition (and DJ program) that doesn't have a pitch or volume but does maintain a duration. Rests allow for a tonal pause in a song.
To construct a rest you must use the following syntax: `rest r = rest (restduration);`. Just as the note constructor, the single argument to a rest must be an expression that resolves to a double. Any other type of expression will not be accepted and will return a compilation error.

3.4 Chord

A primitive data type representing a collection of notes which begin on the same beat.
To construct a chord, there are two options:

1. `chord c = chord (n1, n2, n3...);` where n1, n2, n3... is an arbitrary length list of notes. The chord constructor only takes a list of notes. Any other expression will result in a compilation error.

2. `c = c : n2` where c is previously declared with at least one note as an argument. The `:` operator is the parallel add and adds notes to a chord in parrallel since a chord is defined as a collection of notes beginning on the same beat.

3.5 Track

A series of chords which are played sequentially by the same instrument.
To construct a track: `track t = (piano);` where the single argument to a track must be an expression that resolves to a double. This double represents an instrument in the MIDI library. You can find a list of the MIDI instuments attached in the final document.
You can also serial add chords to tracks: `t = t.c;` where `.` is the serial add operator. The serial add operator adds chords sequentially to a track.

3.6 Score

A series of tracks. A score must be returned in the main `song()` function. To construct a score: `score s = score(t1, t2...);` where t1, t2... are previously defined tracks. The score takes an arbitrary number of tracks. A score can also be declared without any argument `score s = score();` this represents an empty score.

4 Expressions and Operators

An *operator* is a special token that specifies an action performed on either one or two operands. Operator precedence is specified in the order of appearance in the following sections of this document; directional associativity is also specified for each operator. The order of evaluation of all other expressions is left to the compiler and is not guaranteed. An *lvalue* is a manipulable object. Identifiers are typical *lvalues* but *lvalues* are also returned by some functions, including serial and parallel add for example.

4.1 Variable Declaration

Declarations dictate the type of identifiers. Declarations take the form `type-specifier identifier`, and are optionally followed by declarators of the form `type-specifier (expression)`.

4.2 Primary Expressions

Fundamental expressions consist of function calls and those expressions accessed using `->` (described below); these are grouped rightwardly.

4.2.1 Identifiers

Identifiers are primary expressions whose types and values are specified in their declarations.

4.2.2 Constants

Double, note, rest, array, chord, track, and score constants are primary expressions.

4.2.3 (expression)

A parenthesized expression is a primary expression and is in all ways equivalent to the non-parenthesized expression.

4.2.4 primary (args...)

A parenthesized expression following a primary expression is a primary expression. It specifies a function call which may accept a variable-length, comma-separated list of parameters `args`.

4.2.5 primary -> attribute

A primary expression which evaluates to a note followed by `->` and an attribute name is a primary expression. It specifies primitive data type attribute access. The expression evaluates to a double representing the attribute value. (ex: `double p = n -> pitch` returns the pitch, an double value, for a previously declared note n.)

4.3 Unary Operators

Unary Operators are left-to-right associative, except for '−', which is right-to-left associative.

4.3.1 − expression

If the expression resolves to an integer data-type, the '-' operator causes the expression to be considered as a negative value.

4.3.2 lvalue ++

This expression behaves as a shorthand for taking the expression result and incrementing its value. For double types this means an incremental increase in values.

4.3.3 lvalue −−

This expression behaves as above, decrementing instead of incrementing.

4.4 Multiplicative Operators

Multiplicative operators are left-to-right associative.

4.4.1 expression ∗ expression

Double multiplication acts as expected, returning a double which is the result of the multiplication of the two provided doubles.

4.4.2 expression / expression

Double division returns the result `expr / expr`.

4.5 Additive Operators

Additive operators are left-to-right associative.

4.5.1 expression + expression

This expression takes the expression result of the left operand and, depending on its type, increases its value by the amount specified in the right operand: for double types this means an additive increase in value.

4.5.2 expression − expression

This expression behaves like `expression + expression` above, except the left operand is decreased by the right operand.

4.5.3 expression . expression

This expression takes the expression in the right operand, a chord, and concatenates it to the expression in the left operand, a track, returning a new track.

4.5.4 expression : expression

This expression takes the right hand operand, a note, and adds it in parallel to the left hand expression, which is a chord. This statement returns a new chord containing both the chord and the note added in parallel by corresponding time tick, with no added offset.

4.6 Relational Operators

Relational operators are left-to-right associative.

4.6.1 expression < expression

The < operator takes doubles as input. The < operator returns an double 1 if the double on the left is less than the double on the right and 0 otherwise.

4.6.2 expression > expression

The > operator takes doubles as input. If the > operator returns an double 1 if the double on the left is greater than the double on the right and 0 otherwise.

4.6.3 expression <= expression

The <= operator takes doubles as input. The <= operator returns an double 1 if the double on the left is less than or equal to the double on the right and 0 otherwise.

4.6.4 expression >= expression

The > operator takes doubles as input expressions. The >= operator returns a double 1 if the double on the left is greater or equal to than the double on the right and 0 otherwise.

4.6.5 expression == expression

The == operator takes doubles as input expressions. The == operator returns a double 1 if the double on the left is equal to the double on the right and 0 otherwise.

4.6.6 expression ! = expression

The ! = operator takes doubles as input expressions. If the ! = operator is applied to doubles, it returns a double 1 if the double on the left is not equal to the double on the right and 0 otherwise.

4.7 Assignment Operators

Assignment operators are right-to-left associative.

4.7.1 lvalue = expression

The assignment operator stores the result of the evaluation of the right operand expression in the lvalue.

5 Statements

Statements cause actions and are responsible for control flow within your programs.

5.1 Expression Statement

Any statement can turn into an expression by adding a semicolon to the end of the expression (ex: 2+2;).

5.2 The `if-than-else` Statement

We use the `if-than-else` statement to conditionally execute part of a program, based on the truth value of a given expression.
General form of `if` statement:

```
if (conditional-test) {
statement }
else {
statement }
```

The `else` keyword and following, dependent `statement` are optional.

5.3 The `for` Statement

We use the `for` statement to loop over part of a program, based on variable initialization, expression testing, and variable modification. It is easy to use the form for making counter controlled loops.
General form of the `for` statement:

```
for (initialize; test; step) {
statement }
```

5.4 The `loop` Statement

Functions similar to the for control flow statement except it takes a single double argument and loops over the musical phrase according to the double argument.

```
loop (5) {
statement }
```

The advantage of the loop statement is that it is specifically tailored for acting on a musical phrase and can be very simply repeated by the single double argument.

5.5 The BLOCK Statement

5.6 The `return` Statement

Causes the current function call to end in the current sub-routine and return to where the function was called. The return function can return nothing (`return;`) or a return value can be passed back to the calling function (`return expression;`).

6 Functions

6.1 Defining Functions

Functions are defined by a function name and return type followed by parenthesis that contains function parameters separated by commas. All functions must have a `return` statement. The function body is contained between a curly brace at the beginning and a curly brace at the end of the function.

```
mergeTrack track (track track1, track track2) {
/*stuff*/
return newtrack;
}
```

6.2 The `song` Function

The `song` function is where the tracks a user has created will be modified and/or combined to form a score. This is where the music is essentially created. The `song` function returns a `score` which represents the complete song.

6.3 Reserved Functions

`print(expression)`	print to console
`loop(double d) { ... }`	Loops a given note, chord, or track the over number of beats specified by `d`.

6.4 Block Scoping

Braces, { and }, determine the scope of a set of statements and corresponding function and variable definitions. For example, if a variable is declared within a block, it is a local variable contained in that block and can only be accessed within that block and for so long as that block is active in the program environment. Blocks are used to specify function definition and conditional and control-flow operation scope. Local variables defined within the block of a function definition are accessible only within that function, during its operation.

7 Compile Process and Output Files

7.1 JAVA and MIDI

All programs consist of a single main function, called `song`, that returns an array of tracks, intended to start simultaneously and be played in parallel. Each array element can be considered as a polyphonic MIDI channel. This array of tracks is compiled into a CSV file containing the complete set of notes with corresponding time-tick organized into tracks with corresponding instrument mappings required to produce the programmed song. A third party CSV-to-MIDI JAVA library will be used to produce the final sound file.[1]

7.2 Compiler Options

Song-wide properties are specified to the compiler. Attributes like channel looping and transformation from beats to proper-time using attributes such as tempo/beats per minute are available as compiler options.

8 Hopes and Dreams

This structure, as well as the use of the MIDI specification and interface, allows for a fairly extensible language and production capability. For example, through the manipulation or linking of sound banks, new sounds and samples are able to be incorporated to produce rich and interesting programmatic music.

[1] " CSV2MIDI.java Sourcecode" MIDILC Language, 28 Oct 2013. Web. 28 Oct 2013. <https://midilc.googlecode.com/svn-history/r122/trunk/src/components/CSV2MIDI.java>.

9 Examples

9.1 Example 1: Arpeggio

```
//the main function
song score {

        track t = track(0);

        num_beats = 1;
        c = 60;
        volume = 50;
        piano = 0;

        //a for loop
        for(i = 0; i <= 8; i++) {
                //make a new note with incremental pitch
                Note n = note (c + i, volume, num_beats);
                //concatenate that note to the first (only) track of the song
                t = t.n;
        }
    score s = score (t)
        return s;
}
```

9.2 Example 2: Loop With Effects

```
Track loopEffects () {

        int pitchA = 60; //pitch of a will be middle C
        int pitchB = 62.33; //up a full step for b
        int pitchC = 65; // up a step and a half for a minor/dissonant something
        int volume = 50; //volume 50 - right in the middle
        int instr = 0; //use a piano -- mapped instrument 1
        int duration = 2;

        Note a, b, c;
        a = note (pitchA, volume, duration);
        b = note (pitchB, volume, duration);
        c = note (pitchC, volume, duration);

        Chord ch = a : b : c;
        Track t = track (instr);

        loop (50) {
        t = t . ch; //probably sounds a lot like
                        //the rite of spring now...

    }
        for(int i = 0; i < t.size(); i += 2) { //iterate over every other chord in t
                print ( t[i][0] -> pitch ); //for every other chord in t, print the pitch of t
        }
        return t;
}

song score () {
        score s = score ( loopEffects() );
}
```

9.3 Example 3: Add/Remove Notes & Chords

```
null reverseAddFancy{
        //create tracks track, adds and remove chords
        note a, b, c, d, e, f;

        //the note pitches
        double midC = 60; //pitch 60 is usually around middle C
        double upabit = 62;
        double downabit = 40;
        double sumthinElse = 88;
        double lyfe = 42;

        //some other note attributes
        double volume = 20; //nice and quiet
        double oh = 47; //use an Orchestral Harp -- General MIDI mapping
        double shortish = 2;
        double longer = 5;

        //define the notes
        a = note (midC, volume, shortish);
        b = note (lyfe, volume, longer);
        c = note (sumthinElse, volume, longer);

        d = note (upabit, volume, shortish);
        e = note (downabit, volume, longer);
        f = note (midC, volume, shortish);


        chord newChord = a : b : c; //parallel add to make a chord
        chord oldChord = d : (f : e);
        track newTrack = newChord.oldChord; //add track with serial add
}

song score () {
        reverseAddFancy();
        return score ();
}
```

11 Appendix C: Source Code

11.1 WDJC Source

wdjc.ml

```
type action = Ast | Compile | Java | Sast

let _ =
  let action = if Array.length Sys.argv > 1 then
    List.assoc Sys.argv.(1) [
            ("-a", Ast);
            ("-s", Sast);
            ("-j", Java);
            ("-c", Compile) ]
  else Compile in

    let output_name =
      if Array.length Sys.argv > 2 then
        Sys.argv.(2)
      else "song" in

  let lexbuf = Lexing.from_channel stdin in
  let program = Parser.program Scanner.token lexbuf in
  match action with
      Ast -> let listing = Ast.string_of_program program
          in print_string listing
    | Sast -> let program_t = Semcheck.sc_program program in
          let listing = Sast.string_of_program_t program_t
          in print_string listing
    | Java -> let listing = Compile.string_of_program output_name
      (Semcheck.sc_program program) in
          (* let compile = Sys.command("javac -classpath
             tests:java/jMusic/jMusic1.6.4.jar:java/jMusic/inst/:. tests/" ^
             output_name ^ "dj.java") in
            print_int compile; *)
          print_endline listing
    | Compile -> let listing = Compile.string_of_program output_name
      (Semcheck.sc_program program) in
          (* let output = Sys.command("./compile " ^ output_name ^ "dj") in *)
          (* let compile = Sys.command("javac -classpath
             tests:java/jMusic/jMusic1.6.4.jar:java/jMusic/inst/:. tests/" ^
             output_name ^ "dj.java") in
          let run = Sys.command("java -classpath
             tests:java/jMusic/jMusic1.6.4.jar:java/jMusic/inst/:. " ^ output_name
             ^ "dj") in *)
            (* print_endline listing;  *)
            ignore( listing );
            print_int 0
```

scanner.mll

```
{ open Parser }

let Decimal = '.' ['0'-'9']+

rule token = parse
  [' ' '\t' '\r' '\n'] { token lexbuf } (* Whitespace *)
| "/*"                 { comment lexbuf }       (* Comments *)
| '['                  { LBRACK }
```

```
| ']'                 { RBRACK }
| '('                 { LPAREN }
| ')'                 { RPAREN }
| '{'                 { LBRACE }
| '}'                 { RBRACE }
| ';'                 { SEMI }
| ','                 { COMMA }
| '^'                 { VIB }
| '~'                 { TREM }
| '+'                 { PLUS }
| "++"                { INCR }
| '-'                 { MINUS }
| "--"                { DECR }
| '*'                 { TIMES }
| '/'                 { DIVIDE }
| '.'                 { SERIAL }
| ':'                 { PARALLEL }
| "->"                { ARROW }
| '='                 { ASSIGN }
| "=="                { EQ }
| "!="                { NEQ }
| '<'                 { LT }
| "<="                { LEQ }
| '>'                 { GT }
| ">="                { GEQ }
| "if"                { IF }
| "else"              { ELSE }
| "for"               { FOR }
| "while"             { WHILE }
| "loop"              { LOOP }
| "return"            { RETURN }
| "fun"               { FUN }
| "vol"               { VOL }
| "dur"               { DUR }
| "pitch"             { PITCH }
| "double"            { DOUBLE }
| "note"              { NOTE }
| "rest"              { REST }
| "track"             { TRACK }
| "chord"             { CHORD }
| "score"             { SCORE }
| "print"             { PRINT }
(*
| "array"             { ARRAY }
*)
| '-'? Decimal as lxm { LITERAL(lxm) } (* Note in dj literals are really only
    doubles *)
| '-'? ['0'-'9']+ Decimal? as lxm { LITERAL(lxm) } (* Note in dj literals are
    really only doubles *)
| ['a'-'z' 'A'-'Z']['a'-'z' 'A'-'Z' '0'-'9' '_']* as lxm { ID(lxm) }
| eof { EOF }
| _ as char { raise (Failure("Illegal character " ^ Char.escaped char)) }

and comment = parse
  "*/" { token lexbuf }
| _    { comment lexbuf }
```

parser.mly

```
%{ open Ast %}

%token LBRACK RBRACK LPAREN RPAREN LBRACE RBRACE
%token COMMA SEMI ASSIGN
%token PLUS MINUS TIMES DIVIDE
%token SERIAL PARALLEL
%token VIB TREM ARROW
%token EQ NEQ INCR DECR
%token LT LEQ GT GEQ
%token IF ELSE FOR WHILE LOOP RETURN DOUBLE PRINT
%token FUN VOL DUR PITCH INSTR
%token <string> LITERAL
%token <string> ID
%token NOTE REST CHORD TRACK SCORE
%token EOF

/*ie TIMES DIVIDE is higher precedence than ASSIGN*/
%nonassoc NOELSE
%nonassoc ELSE
/*Right associative because if you have a = b = c you want
to do (a = (b = c))*/
%right ASSIGN
/* Equals/neq association: (a == b) == c */
%left EQ NEQ
%left LT GT LEQ GEQ
/*SERIAL/PARALLEL defaulted to PLUS/MINUS Associativity*/
%left SERIAL PARALLEL
%left PLUS MINUS
%left TIMES DIVIDE
%left VIB TREM
/*incr - incrememnt (++); decr - decrement  (--) */
/*Ex: (note++)++ */
%left INCR DECR



%start program
%type <Ast.program> program

%%

program:
   /* nothing */ { [], [] }
 | program vdecl { ($2 :: fst $1), snd $1 }
 | program fdecl { fst $1, ($2 :: snd $1) }


/*  --- FUNCTION --- */
fdecl:

  ID DOUBLE LPAREN formals_opt RPAREN LBRACE stmt_list RBRACE
    {{
        rtype = Double;
        fname = $1;
              formals = $4;
              body = List.rev $7
```

```
57      }}
58      | ID NOTE LPAREN formals_opt RPAREN LBRACE stmt_list RBRACE
59      {{
60         rtype = Note;
61         fname = $1;
62         formals = $4;
63         body = List.rev $7
64      }}
65      | ID CHORD LPAREN formals_opt RPAREN LBRACE stmt_list RBRACE
66      {{
67         rtype = Chord;
68         fname = $1;
69         formals = $4;
70         body = List.rev $7
71      }}
72
73      | ID REST LPAREN formals_opt RPAREN LBRACE stmt_list RBRACE
74      {{
75         rtype = Rest;
76         fname = $1;
77         formals = $4;
78         body = List.rev $7
79      }}
80      | ID TRACK LPAREN formals_opt RPAREN LBRACE stmt_list RBRACE
81      {{
82         rtype = Track;
83         fname = $1;
84         formals = $4;
85         body = List.rev $7
86      }}
87      | ID SCORE LPAREN formals_opt RPAREN LBRACE stmt_list RBRACE
88      {{
89         rtype = Score;
90         fname = $1;
91         formals = $4;
92         body = List.rev $7
93      }}
94
95
96  /* --- FORMALS --- */
97  /* formals to be vdecl */
98  formal:
99    vdecl { $1 }
100
101
102 /* optional function arguments */
103 formals_opt:
104     /* nothing */ { [] }
105   | formal_list   { List.rev $1 }
106
107 formal_list:
108     formal                   { [$1] }
109   | formal_list COMMA formal { $3 :: $1 }
110
111
112 /* --- VARIABLE DECLARATIONS --- */
113 vdecl:
```

```
114     dType ID      { { vType = $1;   vName = $2; } }
115
116
117 /*
118 vdecl_list:
119     */ /* nothing *//*     { [] }
120   | vdecl_list vdecl { $2 :: $1 }
121 */
122
123 vinit:
124     vdecl ASSIGN expr { Vinit($1, $3) }
125
126 assign:
127     expr ASSIGN expr { Assign($1, $3) }
128
129 /* --- SCORE -- */
130 score_cr:
131     SCORE LPAREN RPAREN { SCORE_CR ([]) }
132     | SCORE LPAREN score_list RPAREN { SCORE_CR ( List.rev $3 ) }
133
134 score_list:
135     expr { [$1] }
136     | score_list COMMA expr { $3 :: $1 }
137
138 /* --- TRACK -- */
139 track_cr:
140   TRACK LPAREN expr RPAREN { TRACK_CR( $3 ) }
141
142 /* --- REST --- */
143 rest_cr:
144   REST LPAREN expr RPAREN { REST_CR( $3 ) }
145
146 /*  --- NOTE  --- */
147 note_cr:
148   NOTE LPAREN expr COMMA expr COMMA expr RPAREN { NOTE_CR($3, $5, $7) }
149
150 /* --- CHORD --- */
151 chord_cr:
152     CHORD LPAREN RPAREN { CHORD_CR ([]) }
153     | CHORD LPAREN chord_list RPAREN { CHORD_CR ( List.rev $3 ) }
154
155 chord_list:
156     expr { [$1] }
157     | chord_list COMMA expr { $3 :: $1 }
158
159 /* --- ACCESSOR --- */
160
161 accessor:
162   ID ARROW note_attribute { ACCESSOR(Id($1), $3) }
163
164 /*
165 accessor:
166   data_type_acc { $1 }
167
168 data_type_acc:
169   note_cr ARROW note_attribute { ACCESSOR($1, $3) }
170 */
```

```
171
172 /* List of note attributes */
173 note_attribute:
174   PITCH {Pitch}
175   | VOL {Vol}
176   | DUR {Dur}
177
178 dType:
179    DOUBLE {Double}
180  | NOTE {Note}
181  | CHORD {Chord}
182  | TRACK {Track}
183  | REST {Rest}
184  | SCORE {Score}
185
186
187 /* --- MODIFIERS --- */
188 /*
189 modifier:
190
191 modifier_options:
192   BEND     {$1}
193   | VIB    {$1}
194   | TREM   {$1}
195
196 */
197 /* --- STATEMENTS --- */
198
199 stmt:
200     expr SEMI { Expr($1) }
201   | vinit SEMI { $1 }
202   | vdecl SEMI { Vdecl($1) }
203   | RETURN expr SEMI { Return($2) }
204   | PRINT LPAREN expr RPAREN SEMI { Print($3) }
205   | LBRACE stmt_list RBRACE { Block(List.rev $2) }
206   | IF LPAREN expr RPAREN stmt %prec NOELSE { If($3, $5, Block([])) }
207   | IF LPAREN expr RPAREN stmt ELSE stmt    { If($3, $5, $7) }
208   | FOR LPAREN expr_opt SEMI expr_opt SEMI expr_opt RPAREN stmt { For($3, $5, $7,
        $9) }
209   | LOOP LPAREN expr RPAREN stmt { Loop($3, $5) }
210   | WHILE LPAREN expr RPAREN stmt { While($3, $5) }
211
212 stmt_list:
213     /* nothing */  { [] }
214   | stmt_list stmt { $2 :: $1 }
215
216 /* --- EXPRESSIONS --- */
217
218 expr_opt:
219     /* nothing */ { Noexpr }
220   | expr          { $1 }
221
222 expr:
223     LITERAL          { Literal($1) }
224   | ID               { Id($1) }
225   | assign           { $1 }
226   | accessor         { $1 }
```

```
227   | chord_cr          { $1 }
228   | note_cr           { $1 }
229   | rest_cr           { $1 }
230   | track_cr          { $1 }
231   | score_cr          { $1 }
232   | expr PLUS   expr { Binop($1, Add,   $3) }
233   | expr MINUS  expr { Binop($1, Sub,   $3) }
234   | expr TIMES  expr { Binop($1, Mult,  $3) }
235   | expr DIVIDE expr { Binop($1, Div,   $3) }
236   | expr EQ     expr { Binop($1, Equal, $3) }
237   | expr NEQ    expr { Binop($1, Neq,   $3) }
238   | expr LT     expr { Binop($1, Less,  $3) }
239   | expr LEQ    expr { Binop($1, Leq,   $3) }
240   | expr GT     expr { Binop($1, Greater, $3) }
241   | expr GEQ    expr { Binop($1, Geq,   $3) }
242   | expr SERIAL expr { Binop($1, Ser, $3) }
243   | expr PARALLEL expr { Binop ($1, Par, $3) }
244   | expr INCR        { Modifier($1, Incr) }
245   | expr DECR        { Modifier($1, Decr) }
246   | expr VIB         { Modifier($1, Vib) }
247   | expr TREM        { Modifier($1, Trem) }
248   | ID LPAREN actuals_opt RPAREN { Call($1, $3) }
249   | LPAREN expr RPAREN { $2 }
250   | ID LBRACK expr RBRACK { Address(Id($1), $3) }
251   /*| LBRACKET actuals_opt RBRACKET { Array($?) } */
252
253  /* actuals - When you call the function you use actuals_opt?? */
254   actuals_opt:
255     /* nothing */ { [] }
256   | actuals_list  { List.rev $1 }
257
258   actuals_list:
259     expr                    { [$1] }
260   | actuals_list COMMA expr { $3 :: $1 }
```

ast.ml

```
1  (* AST *)
2  type modif = Vib | Trem | Incr | Decr
3
4  (* Not sure if I should make this a string *)
5  type note_attribute = Pitch | Vol | Dur
6
7  (*our data types*)
8  type dType = Double | Note | Chord | Track | Rest | Score
9
10 (* operation types *)
11 type op =   Add  | Sub
12           | Mult | Div
13           | Ser  | Par
14           | Equal | Neq | Geq | Leq | Greater | Less
15
16 (* Expression type *)
17 type expr =
18     Literal of string
19   | Id of string
20   | NOTE_CR of expr * expr * expr
21   | REST_CR of expr
```

```
22    | TRACK_CR of expr
23    | CHORD_CR of expr list
24    | SCORE_CR of expr list
25    | ACCESSOR of expr * note_attribute
26    | Binop of expr * op * expr
27    | Modifier of expr * modif
28    | Assign of expr * expr
29    | Address of expr * expr
30    | Call of string * expr list
31    | Noexpr
32
33    (* | Array of expr list *)
34    (*an array can be a list of expressions*)
35
36  (*variable declaration*)
37  type var_decl = {
38    vType : dType;
39    vName : string;
40  }
41
42  type var_init = {
43    vDecl : var_decl;
44    vExpr : expr;
45  }
46
47  (*need to decide if we are keeping loop or not*)
48  type stmt =
49      Block of stmt list
50    | Expr of expr
51    | Return of expr
52    | Print of expr
53    | If of expr * stmt * stmt
54    | For of expr * expr * expr * stmt
55    | Loop of expr * stmt
56    | While of expr * stmt
57    (* | Assign of var_decl * expr *)
58    | Vdecl of var_decl
59    | Vinit of var_decl * expr
60   (* | Loop of expr * expr * stmt *)
61
62
63  (* funciton declaration *)
64  type func_decl = {
65      rtype : dType;
66      fname : string;
67      formals : var_decl list;
68      body : stmt list;
69    }
70
71
72  (*ast is a list of variables and list of function dels*)
73  type program = var_decl list * func_decl list
74
75  (*pretty print for expr*)
76  (*TODO need to decide on arrays*)
77  let rec string_of_expr = function
78      Literal(l) -> l
```

```
79    | Id(s) -> s
80    | NOTE_CR(a, b, c) ->
81        "(" ^ string_of_expr a ^ ", " ^ string_of_expr b ^ ", " ^ string_of_expr c ^
              ")"
82    | REST_CR(expr) -> "(" ^ string_of_expr expr ^ ")" (* should this really be
        string of literal or something? *)
83    | TRACK_CR(expr) -> "(" ^ string_of_expr expr ^ ")"
84    | SCORE_CR(expr_list) ->
85        "(" ^ String.concat ". ." (List.map string_of_expr expr_list) ^ ")"
86    | ACCESSOR(a, b) ->
87        (string_of_expr a) ^ "->" ^ (
88        match b with
89          Pitch -> "pitch" | Vol -> "vol" | Dur -> "dur"
90        )
91    | Assign(id, expr) -> string_of_expr id ^ "=" ^ string_of_expr expr
92    | Address(id, expr) -> string_of_expr id ^ "[" ^ string_of_expr expr ^ "]"
93    | CHORD_CR(expr_list) ->
94        "(" ^ String.concat ":" (List.map string_of_expr expr_list) ^ ")"
95    | Binop(e1, o, e2) ->
96        string_of_expr e1 ^ " " ^
97        (match o with
98              Add -> "+" | Sub -> "-" | Mult -> "*" | Div -> "/"
99        | Equal -> "==" | Neq -> "!="
100       | Less -> "<" | Leq -> "<=" | Greater -> ">" | Geq -> ">="
101       | Ser -> "." | Par -> ":") ^ " " ^
102       string_of_expr e2
103   (*again, not sure about this section*)
104   | Modifier(e1, modif) ->
105       string_of_expr e1 ^
106       (match modif with
107       Vib -> "^" | Trem -> "~" | Incr -> "++" | Decr -> "--")
108   | Call(f, el) ->
109       f ^ "(" ^ String.concat ", " (List.map string_of_expr el) ^ ")"
110   | Noexpr -> ""
111 (*| Array*)
112
113 let string_of_vdecl v =
114   (match v.vType with
115     Double -> "double "
116     | Note -> "note "
117     | Chord -> "chord "
118     | Track -> "track "
119     | Rest -> "rest "
120     | Score -> "score " ) ^ v.vName
121
122 (*
123 let string_of_cr_type t =
124     (match )
125 *)
126 (*pretty print for stmts*)
127 (*TODO need to do loop*)
128 let rec string_of_stmt = function
129     Block(stmts) ->
130       "{\n" ^ String.concat "" (List.map string_of_stmt stmts) ^ "}\n"
131   | Expr(expr) -> string_of_expr expr ^ ";\n";
132   | Return(expr) -> "return " ^ string_of_expr expr ^ ";\n";
133   | Print(expr) -> "print (" ^ string_of_expr expr ^ ");\n";
```

```
  | If(e, s, Block([])) -> "if (" ^ string_of_expr e ^ ")\n" ^ string_of_stmt s
  | If(e, s1, s2) ->  "if (" ^ string_of_expr e ^ ")\n" ^
      string_of_stmt s1 ^ "else\n" ^ string_of_stmt s2
  | For(e1, e2, e3, s) ->
      "for (" ^ string_of_expr e1  ^ " ; " ^ string_of_expr e2 ^ " ; " ^
      string_of_expr e3  ^ ") " ^ string_of_stmt s
  | Loop(e, s) -> "loop (" ^ string_of_expr e ^ ") " ^ string_of_stmt s
  | While(e, s) -> "while (" ^ string_of_expr e ^ ") " ^ string_of_stmt s
  (* | Assign(v, e) -> string_of_vdecl v ^ " = " ^ string_of_expr e *)
  | Vdecl(v) -> string_of_vdecl v ^ ";\n"
  | Vinit(v, e) -> string_of_vdecl v ^ " = " ^ string_of_expr e ^ ";\n"

 (*| Loop*)


let string_of_fdecl fdecl =
   (match fdecl.rtype with
    Double -> "double "
    | Note -> "note "
    | Chord -> "chord "
    | Track -> "track "
    | Rest -> "rest "
    | Score ->  "score ") ^ fdecl.fname ^ "(" ^ String.concat ", " (List.map
      string_of_vdecl fdecl.formals) ^ ")\n{\n" ^
  String.concat "" (List.map string_of_stmt fdecl.body) ^
  "}\n"

(*pretty print for program*)
let string_of_program (vars, funcs) =
  String.concat "" (List.map string_of_vdecl vars) ^ "\n" ^
  String.concat "\n" (List.map string_of_fdecl funcs)
```

semcheck.ml

```
(*

semcheck.ml

- semantically checks the input file
- checks for type errors
- undefined variable & function errors
-converts from ast to sast

*)
open Sast
open Ast




(*NOTE:
map.find: returns the value associated with a key
map.mem: checks if value exists for a given key
*)


module StringMap = Map.Make(String)

```

```
type env = {
functions:      string list StringMap.t;
globals:        string StringMap.t;
locals:         string StringMap.t;
}


(* TYPE CONVERSIONS *)

(* var type -> string *)
let string_of_vartype = function
Ast.Double -> "double"
| Ast.Note -> "note"
| Ast.Rest -> "rest"
| Ast.Chord -> "chord"
| Ast.Track -> "track"
| Ast.Score -> "score"

(* ast -> sast type*)
let ast_to_sast_note_attr = function
Ast.Pitch -> Sast.Pitch_t
| Ast.Vol -> Sast.Vol_t
| Ast.Dur -> Sast.Dur_t
(* | _ -> raise (Failure ("Mismatch Note Attribute Type")) *)

(* ast -> sast type*)
let ast_to_sast_op = function
Ast.Add -> Sast.Add_t
| Ast.Sub -> Sast.Sub_t
| Ast.Mult -> Sast.Mult_t
| Ast.Div -> Sast.Div_t
| Ast.Ser -> Sast.Ser_t
| Ast.Par -> Sast.Par_t
| Ast.Equal -> Sast.Equal_t
| Ast.Neq -> Sast.Neq_t
| Ast.Geq -> Sast.Geq_t
| Ast.Leq -> Sast.Leq_t
| Ast.Greater -> Sast.Greater_t
| Ast.Less -> Sast.Less_t
(* | _ -> raise (Failure ("Mismatch Operator Type")) *)

let ast_to_sast_mod = function
Ast.Vib -> Sast.Vib_t
| Ast.Trem -> Sast.Trem_t
| Ast.Incr -> Sast.Incr_t
| Ast.Decr -> Sast.Decr_t
(* | _ -> raise (Failure ("Mismatch Modifier Type")) *)

(* ast -> sast type*)
let ast_to_sast_type = function
Ast.Double -> Sast.Double_t
| Ast.Note -> Sast.Note_t
| Ast.Rest -> Sast.Rest_t
| Ast.Chord -> Sast.Chord_t
| Ast.Track -> Sast.Track_t
| Ast.Score -> Sast.Score_t
(* | _ -> raise (Failure ("Mismatch Variable Type Type")) *)
```

```
(*
let ast_to_sast_vdecl vdecl =
let sast_type = (* ast_to_sast_type *) vdecl.vType in
Sast.var_decl_t( {vType=sast_type; vName=vdecl.vName;} )
*)

(* we may need a variable total conversion from
ast to sast *)

(*need for locals, formals, and global variabes*)
let convert_types vardecl =
(* Sast.Vdecl_t( {vType_t= (ast_to_sast_type vardecl.vType);
   vName_t=vardecl.vName;} ) *)
( {vType_t= (ast_to_sast_type vardecl.vType); vName_t=vardecl.vName;} )

(* TYPES - do we need this? *)
let get_type = function
var -> string_of_vartype var.vType


(* HELPFUL FUNCTIONS TO GET AND ADD VARIABLES (GLOBAL & LOCAL), FUNCIONS TO
   ENVIRONMENT *)

(*
get_variable vname env
vname - variable name
env - environment stringmap
Looks to find variable name in env's local list.
If it doesn't find it, it checks the env's global list.
If not found, raises error.
*)
let get_variable_name vname env =
try StringMap.find vname env.locals
with Not_found -> try StringMap.find vname env.globals
with Not_found -> raise (Failure ("Undeclared variable " ^ vname))

let get_variable_type vname env =
try StringMap.find vname env.locals
with Not_found -> try StringMap.find vname env.globals
with Not_found -> raise (Failure ("Untyped variable " ^ vname))


(*
get_function vname env
vname - function name
env - environment stringmap
Looks to find function name in env's function list.
If not found, raises error.
*)
let get_function fname env =
try StringMap.find fname env.functions
with Not_found -> raise (Failure ("Undeclared function " ^ fname))


(*
add_local var_type name env
var_type - variable type
```

```
137 name - variable name
138 env - environment stringmap
139 Checks to see if the name is in env's local list.
140 If it doesn't contain it, it adds it to the env's local list.
141 *)
142 let add_local var_type name env =
143 if StringMap.mem name env.locals then raise (Failure ("Local variable " ^ name ^
    "is already defined"))
144 else StringMap.add name (string_of_vartype var_type) env.locals
145
146 (*
147 add_global var_type name env
148 var_type - variable type
149 name - gariable name
150 env - environment stringmap
151
152 Checks to see if the add_localname is in the env's global list.
153 If it dlesn't contain it, it adds it to the env's global list.
154 *)
155 let add_global var_type name env =
156 (* if name exists in env.globals, return empty stringmap *)
157 if StringMap.mem name env.globals then raise (Failure ("Global variable " ^ name ^
    " is already defined."))
158 (*  else; add to env.globals:
159 key = name
160 value = vartype
161 *)
162 else StringMap.add name (string_of_vartype var_type) env.globals
163
164 (*
165 CONFUSED ON THE GET_TYPE
166 add_function fname return formals env
167 fname - function name
168 rtype - return type
169 formals - formal arguments
170 env - environment stringmap
171
172 Checks to see if the fname is in env's function list
173 if not- it gets the types of the formals, adds:
174 name, vartype of return, formals to environemt's function
175 *)
176 let add_function fname rtype formals env =
177 if StringMap.mem fname env.functions then raise (Failure ("function " ^ fname ^ "
    is already defined."))
178 else let fmls = List.map get_type formals in
179 (* weird parenthesis...*)
180 StringMap.add fname (string_of_vartype (rtype) :: fmls) env.functions
181 (*Strinmap.add, parse locals, add to env*)
182
183
184 (* SEMANTIC CHECKING FUNCTIONS *)
185
186
187
188 let rec build_expr = function
189 Ast.Literal(i) -> Sast.Literal_t(i)
190 | Ast.Id(i) -> Sast.Id_t(i)
```

```
191 | Ast.ACCESSOR(expr, note_attr) -> Sast.ACCESSOR_t( (build_expr expr),
      (ast_to_sast_note_attr note_attr) )
192 | Ast.NOTE_CR(expr1, expr2, expr3) -> Sast.NOTE_CR_t( (build_expr expr1),
      (build_expr expr2), (build_expr expr3) )
193 | Ast.REST_CR(expr) -> Sast.REST_CR_t( (build_expr expr) )
194 | Ast.CHORD_CR(expr_list) -> Sast.CHORD_CR_t( (build_expr_list expr_list) )
195 | Ast.TRACK_CR(expr) -> Sast.TRACK_CR_t( (build_expr expr) )
196 | Ast.SCORE_CR(expr_list) -> Sast.SCORE_CR_t( (build_expr_list expr_list) )
197 | Ast.Binop(expr1, op, expr2) -> Sast.Binop_t( (build_expr expr1), (ast_to_sast_op
      op) , (build_expr expr2) )
198 | Ast.Modifier(expr, m) -> Sast.Modifier_t( (build_expr expr), (ast_to_sast_mod m) )
199 | Ast.Assign(expr1, expr2) -> Sast.Assign_t( (build_expr expr1), (build_expr expr2)
      )
200 | Ast.Address(expr1, expr2) -> Sast.Address_t( (build_expr expr1), (build_expr
      expr2) )
201 | Ast.Call(str, expr_list) -> Sast.Call_t( str, (build_expr_list expr_list) )
202 | Ast.Noexpr -> Sast.Noexpr_t
203
204 and build_expr_list expr_list =
205 match expr_list with
206 [] -> []
207 | hd::tl -> let sast_expr_list = (build_expr hd) in
      sast_expr_list::(build_expr_list tl)
208
209 let rec build_stmt = function
210 Ast.Block(stmt_list) -> Sast.Block_t( (build_stmt_list stmt_list) )
211 | Ast.Expr(expr) -> Sast.Expr_t( (build_expr expr) )
212 | Ast.Return(expr) -> Sast.Return_t( (build_expr expr) )
213 | Ast.Print(expr) -> Sast.Print_t( (build_expr expr) )
214 | Ast.If(expr, stmt1, stmt2) -> Sast.If_t( (build_expr expr), (build_stmt stmt1),
      (build_stmt stmt2) )
215 | Ast.For(expr1, expr2, expr3, stmt) -> Sast.For_t( (build_expr expr1), (build_expr
      expr2), (build_expr expr3), (build_stmt stmt) )
216 | Ast.Loop(expr, stmt) -> Sast.Loop_t( (build_expr expr), (build_stmt stmt) )
217 | Ast.While(expr, stmt) -> Sast.While_t( (build_expr expr), (build_stmt stmt) )
218 | Ast.Vdecl( vardecl ) -> Sast.Vdecl_t( {vType_t=(ast_to_sast_type vardecl.vType);
      vName_t=vardecl.vName;} )
219 | Ast.Vinit(decl, expr) -> Sast.Vinit_t( {vType_t=(ast_to_sast_type decl.vType);
      vName_t=decl.vName;} , (build_expr expr) )
220
221 and build_stmt_list stmt_list =
222 match stmt_list with
223 [] -> []
224 | hd::tl -> let sast_stmt_list = (build_stmt hd) in
      sast_stmt_list::(build_stmt_list tl) (* returns SAST body which is a SAST stmt
      list *)
225
226 (* let sc_stmt_list func env =
227 (* match func.body with
228 [] -> []
229 | _ -> *) (* ignore type_stmt_list func env func.body; *)
230 build_stmt_list func.body *)
231
232 let is_id expr =
233 match expr with
234 Ast.Id(i) -> []
235 | _ -> raise (Failure ("Mismatch Expression type: \n" ^
```

```
"LHS of assign must be of type ID."))

let rec match_expr_list_types env types_list expr_list =
match expr_list with
[] -> (match types_list with
[] -> []
| _ -> raise (Failure ("Mismatch arguments number: function expects more arguments than supplied.")))
| hd::tl -> ignore (type_expr (List.hd types_list) env hd); match_expr_list_types env (List.tl types_list) tl

and type_call typestring env name_str expr_list =
let func_types_list = try StringMap.find name_str env.functions
with Not_found -> raise (Failure ("Undefined function: " ^ name_str))
in
let rtype = (List.hd func_types_list) in
if rtype != typestring && typestring <> "any"
then raise (Failure ("Mismatch Expression type: \n" ^
"function has return type " ^ rtype ^ ".\n" ^
"an expression of type " ^ typestring ^ " was expected."))
else match_expr_list_types env (List.tl func_types_list) expr_list

and type_binop typestring env expr1 op expr2 =
match op with
Ast.Add -> ignore (type_expr "double" env expr1);
ignore (type_expr "double" env expr2);
"double"
| Ast.Sub -> ignore (type_expr "double" env expr1);
ignore (type_expr "double" env expr2);
"double"
| Ast.Mult -> ignore (type_expr "double" env expr1);
ignore (type_expr "double" env expr2);
"double"
| Ast.Div -> ignore (type_expr "double" env expr1);
ignore (type_expr "double" env expr2);
"double"
(* TODO boolean should take note or chord or track too*)
| Ast.Equal -> ignore (type_expr "primitive" env expr1);
ignore (type_expr "primitive" env expr2);
"boolean"
| Ast.Neq -> ignore (type_expr "primitive" env expr1);
ignore (type_expr "primitive" env expr2);
"boolean"
| Ast.Geq -> ignore (type_expr "primitive" env expr1);
ignore (type_expr "primitive" env expr2);
"boolean"
| Ast.Leq -> ignore (type_expr "primitive" env expr1);
ignore (type_expr "primitive" env expr2);
"boolean"
| Ast.Greater -> ignore (type_expr "primitive" env expr1);
ignore (type_expr "primitive" env expr2);
"boolean"
| Ast.Less -> ignore (type_expr "primitive" env expr1);
ignore (type_expr "primitive" env expr2);
"boolean"
(* TODO either has to be chord OR note OR track *)
| Ast.Ser -> (match typestring with
```

```
| "track" ->
ignore (type_expr "track" env expr1);
ignore (type_expr "chord" env expr2);
"track"
| "any" ->
ignore (type_expr "track" env expr1);
ignore (type_expr "chord" env expr2);
"track"
| _ -> raise (Failure ("Mismatch Expression type: \n" ^
"expression was of type track.\n" ^
"but an expression of type " ^ typestring ^ " was expected.")) )
(* TODO either has to be chord OR note OR track OR score *)
| Ast.Par -> (match typestring with
"score" ->
ignore (type_expr "score" env expr1);
ignore (type_expr "track" env expr2);
"score"
| "chord" ->
ignore (type_expr "chord" env expr1);
ignore (type_expr "chord_or_note_or_rest" env expr2);
"chord"
| "any" -> (try
ignore (type_expr "chord" env expr1);
ignore (type_expr "chord_or_note_or_rest" env expr2);
"chord"
with Failure cause ->
try
ignore (type_expr "score" env expr1);
ignore (type_expr "score_or_track" env expr2);
"score"
with Failure cause -> raise (Failure ("Mismatch Expression type: \n" ^
"expression was required to be of type score or chord.\n" ^
"but an expression of type " ^ typestring ^ " was expected.")) )
| _ -> raise (Failure ("Mismatch Expression type: \n" ^
"expression was required to be of type score or chord.\n" ^
"but an expression of type " ^ typestring ^ " was expected.")) )

and type_expr typestring env expr =
match expr with
Ast.Literal(i) -> if typestring <> "double" && typestring <> "any" && typestring <>
    "primitive"
then raise (Failure ("Mismatch Expression type: \n" ^
"expression was of type double.\n" ^
"an expression of type " ^ typestring ^ " was expected."))
else env
| Ast.Id(i) -> let id_type = get_variable_type i env in
if typestring = "primitive"
then
if id_type <> "note" && id_type <> "chord" && id_type <> "track" && id_type <>
    "score" && id_type <> "double"
then raise (Failure ("Mismatch Expression type: \n" ^
"expression was of type " ^ id_type ^ ".\n" ^
"an expression of type " ^ typestring ^ " was expected."))
else env
else (match typestring with
"any" -> env
| "chord_or_note_or_rest" -> (match id_type with
```

```
"chord" -> env
| "note" -> env
| "rest" -> env
| _ -> raise (Failure ("Mismatch Expression type: \n" ^
"expression was of type " ^ id_type ^ ".\n" ^
"an expression of type " ^ typestring ^ " was expected.")) )
| "score_or_track" -> (match id_type with
"score" -> env
| "track" -> env
| _ -> raise (Failure ("Mismatch Expression type: \n" ^
"expression was of type " ^ id_type ^ ".\n" ^
"an expression of type " ^ typestring ^ " was expected.")) )
| "track_or_chord" -> (match id_type with
"track" -> env
| "chord" -> env
| _ -> raise (Failure ("Mismatch Expression type: \n" ^
"expression was of type " ^ id_type ^ ".\n" ^
"an expression of type " ^ typestring ^ " was expected.")) )
| _ -> if typestring <> id_type
then raise (Failure ("Mismatch Expression type: \n" ^
"expression was of type " ^ id_type ^ ".\n" ^
"an expression of type " ^ typestring ^ " was expected."))
else env )
| Ast.ACCESSOR(expr, note_attr) -> ignore (type_expr "note" env expr);
if typestring <> "double" && typestring <> "any"
then raise (Failure ("Mismatch Expression type: \n" ^
"expression was of type double.\n" ^
"an expression of type " ^ typestring ^ " was expected."))
else env
| Ast.NOTE_CR(expr1, expr2, expr3) -> if typestring <> "primitive" && typestring <>
    "note" && typestring <> "chord_or_note_or_rest" && typestring <> "any"
then raise (Failure ("Mismatch Expression type: \n" ^
"expression was of type note.\n" ^
"an expression of type " ^ typestring ^ " was expected."))
else ignore (type_expr "double" env expr1);
ignore (type_expr "double" env expr2);
ignore (type_expr "double" env expr3);
env
| Ast.REST_CR(expr) -> if typestring <> "primitive" && typestring <> "rest" &&
    typestring <> "chord_or_note_or_rest" && typestring <> "any"
then raise (Failure ("Mismatch Expression type: \n" ^
"expression was of type rest.\n" ^
"an expression of type " ^ typestring ^ " was expected."))
else ignore (type_expr "double" env expr);
env
| Ast.CHORD_CR(expr_list) -> if typestring <> "primitive" && typestring <> "chord"
    && typestring <> "chord_or_note_or_rest" && typestring <> "track_or_chord" &&
    typestring <> "any"
then raise (Failure ("Mismatch Expression type: \n" ^
"expression was of type chord.\n" ^
"an expression of type " ^ typestring ^ " was expected."))
else ignore (type_expr_list "chord_or_note_or_rest" env expr_list);
env
| Ast.TRACK_CR(expr) -> if typestring <> "primitive" && typestring <> "track" &&
    typestring <> "track_or_chord" && typestring <> "score_or_track" && typestring
    <> "any"
then raise (Failure ("Mismatch Expression type: \n" ^
```

```
397 "expression was of type track.\n" ^
398 "an expression of type " ^ typestring ^ " was expected."))
399 else ignore (type_expr "double" env expr);
400 env
401 | Ast.SCORE_CR(expr_list) -> if typestring <> "primitive" && typestring <> "score"
      && typestring <> "score_or_track" && typestring <> "any"
402 then raise (Failure ("Mismatch Expression type: \n" ^
403 "expression was of type track.\n" ^
404 "an expression of type " ^ typestring ^ " was expected."))
405 else ignore (type_expr_list "track" env expr_list);
406 env
407 | Ast.Binop(expr1, op, expr2) -> let binop_type = type_binop typestring env expr1
      op expr2 in
408 if typestring <> binop_type && typestring <> "any"
409 then raise (Failure ("Mismatch Expression type: \n" ^
410 "expression was of type " ^ binop_type ^ ".\n" ^
411 "an expression of type " ^ typestring ^ " was expected."))
412 else env
413 | Ast.Modifier(expr, m) -> ignore (type_expr "primitive" env expr); env
414 | Ast.Assign(expr1, expr2) -> ignore (is_id expr1);
415 ignore (type_expr typestring env expr1);
416 ignore (type_expr typestring env expr2);
417 (* TODO update environment with initialized boolean *)
418 (* TODO update environment? *)
419 env
420 | Ast.Address(expr1, expr2) -> (match typestring with
421 "track" -> ignore (type_expr "score" env expr1);
422 ignore (type_expr "double" env expr2);
423 env
424 | "chord" -> ignore (type_expr "track" env expr1);
425 ignore (type_expr "double" env expr2);
426 env
427 | "note" -> ignore (type_expr "chord" env expr1);
428 ignore (type_expr "double" env expr2);
429 env
430 | "rest" -> ignore (type_expr "chord" env expr1);
431 ignore (type_expr "double" env expr2);
432 env
433 | "any" -> ignore (type_expr "any" env expr1);
434 ignore (type_expr "double" env expr2);
435 env
436 | _ -> raise (Failure ("Mismatch Expression type: \n" ^
437 "expression was of the wrong type.\n" ^
438 "an expression of type " ^ typestring ^ " was expected.")) )
439 | Ast.Call(name_str, expr_list) -> ignore (type_call typestring env name_str
      expr_list);
440 env
441 | Ast.Noexpr -> env
442
443 and type_expr_list typestring env = function
444 [] -> []
445 | hd::tl -> let new_env = (type_expr typestring env hd) in type_expr_list
      typestring new_env tl
446
447
448 (* function matches a STATEMENT *)
449 (* || func.fname = "Song"*)
```

```
450 let rec type_stmt func env stmt =
451 match stmt with
452 Ast.Block(stmt_list) -> type_stmt_list func env stmt_list
453 | Ast.Expr(expr) -> type_expr "any" env expr
454 | Ast.Print(expr) -> type_expr "any" env expr
455 | Ast.Return(expr) -> if func.fname != "song"
456 then type_expr (string_of_vartype func.rtype) env expr
457 else
458 let rtn_type = string_of_vartype func.rtype in
459 if rtn_type != "score"
460 then raise (Failure ("Return type of song function must be of type score."))
461 else type_expr (string_of_vartype func.rtype) env expr
462 (* reordered! expr comes last (after stmts) becuase its the only one that can
      change the environment outside the block *)
463 | Ast.If(expr, stmt1, stmt2) -> ignore (type_stmt func env stmt1);
464 ignore (type_stmt func env stmt2);
465 type_expr "boolean" env expr
466 (* expr1=assign, expr2=boolean, expr3=junk *)
467 | Ast.For(expr1, expr2, expr3, stmt) -> let for_env = type_expr "double" env expr1
      in
468 ignore (type_expr "any" for_env expr2);
469 ignore (type_expr "any" for_env expr3);
470 ignore (type_stmt func for_env stmt);
471 env
472 | Ast.Loop(expr, stmt) -> let loop_env = type_expr "double" env expr in
473 ignore (type_stmt func loop_env stmt);
474 env
475 | Ast.While(expr, stmt) -> let while_env = type_expr "boolean" env expr in
476 ignore (type_stmt func while_env stmt);
477 env
478 | Ast.Vdecl(vardecl) -> let new_locals_stringmap = add_local vardecl.vType
      vardecl.vName env in
479 let new_env =
480 {
481 locals = new_locals_stringmap;
482 globals = env.globals;
483 functions = env.functions
484 } in
485 new_env
486 | Ast.Vinit(vardecl, expr) -> let new_locals_stringmap = add_local vardecl.vType
      vardecl.vName env in
487 let new_env =
488 {
489 locals = new_locals_stringmap;
490 globals = env.globals;
491 functions = env.functions
492 } in
493 type_expr (string_of_vartype vardecl.vType) new_env expr
494
495 and type_stmt_list func env = function
496 [] -> env
497 | hd::tl -> let new_env = (type_stmt func env hd) in type_stmt_list func new_env tl
498
499
500 (* let rec sc_local_vars func env =  *)
501
502 (* check the expression type can be used for
```

```
503 * the corresponding argument according to definition
504 * return the new expression list in expr_t for sast *)
505 let sc_func_arg lst expr arg_t =
506 if (snd expr) = arg_t then (fst expr)::lst else
507 raise (Failure("function_arguments_do_not_match"))
508 (* FUNCTIONS - EMILY *)
509 (*checks function arguments, then updates env*)
510 let sc_formal formal env =
511 (*fstrently, formals are var_decls*)
512 let new_locals_stringmap = add_local formal.vType formal.vName env in
513 let env =
514 {
515 locals = new_locals_stringmap;
516 globals = env.globals;
517 functions = env.functions
518 } in
519 convert_types formal, env
520 (* check function arguments *)
521
522
523 (* updates formals from fst context *)
524 (* in = function formals + env *)
525 let rec sc_formals formals env =
526 match formals with
527 [] -> []
528 | h::t -> let f, new_env = (sc_formal h env) in (f, new_env)::(sc_formals t new_env)
529
530
531 (* sc_function
532 returns updated formals + body
533 returns type, name, locals
534 *)
535 let rec sc_function fn env =
536 match List.hd (List.rev fn.body) with
537 (* check there is a return statement at the end of the function *)
538 (* TODO only song needs a return! *)
539 Return(_) ->
540 (* updating this function's personal envirnment *)
541 (* let env =
542 {
543 locals = StringMap.empty;
544 globals = env.globals;
545 functions = env.functions;
546 }
547 fill up env_new with functions;
548 change name possibly to something more intuitive
549 new_fn_sm - new function stringmap
550
551 in *)
552 let new_function_stringmap = add_function fn.fname fn.rtype fn.formals env in
553 let env =
554 {
555 locals = StringMap.empty;
556 globals = env.globals;
557 functions = new_function_stringmap (* new function env *)
558 }
559
```

```
(* check formal arguments with sc_formals
formals_env
- returns formal list appended w/ new environment as tuples
*)
in
let function_environment_tuple_list = sc_formals fn.formals env in (* f is tuple
    (formals, env) *)
(* formals = list of formals *)
let formals_list = List.map (fun formal -> fst formal )
    function_environment_tuple_list in
(match formals_list with
(* empty, no formals *)
[] -> ignore (type_stmt_list fn env fn.body);
let sast_body = build_stmt_list fn.body in
{
Sast.rtype_t = ast_to_sast_type fn.rtype;
Sast.fname_t = fn.fname;
Sast.formals_t = formals_list; (* ie empty *)
Sast.body_t = sast_body
}, env
|_ -> let new_env = snd (List.hd (List.rev function_environment_tuple_list)) in
ignore (type_stmt_list fn new_env fn.body);
let sast_body = build_stmt_list fn.body in
{
Sast.rtype_t = ast_to_sast_type fn.rtype;
Sast.fname_t = fn.fname;
Sast.formals_t = formals_list; (* ie empty *)
Sast.body_t = sast_body
}, new_env
)
|_ -> raise (Failure ("The last statement must be a return statement"))
(*let f = sc_formals fn.formals env i stopped fu nv stuff at ln 196*)


(* check function list *)
let rec sc_functions fns env =
match fns with
(* if no functions, return empty list *)
[] -> []
(* otherwise, go through and create a list of function, environment
pairs; the last element in the list is the most up-to-date env *)
| h::t -> let f, e = (sc_function h env) in f::(sc_functions t e)


(* TOM - I don't know what this is so I didn't want to change it *)
(*invokes a function and returns updated formals and block from env. Needs to also
update the symbol table for global variables*)
(*let functions_checker env func =

let rec functions_update env funcs =
*)


(* GLOBALS - EMILY *)

(* sem check global *)
let sc_global global env =
```

```
615 (* add_global returns updated stringmap *)
616 let new_global_stringmap = add_global global.vType global.vName env in
617 let env =
618 {
619 locals = env.locals;
620 globals = new_global_stringmap;
621 functions = env.functions
622 } in
623 (*
624 RETURN: global + env
625 *)
626 convert_types global, env
627
628 (* sem check list of globals *)
629 let rec sc_globals globals env =
630 match globals with
631 (* empty list of globals*)
632 [] -> []
633 (*
634 - iterate through list of globals
635 - semantically check each individual global
636 - (g, e) end up being pairs of globals + respective environments
637 - the last (g,e) pair has env with info from all globals
638 *)
639 | h::t -> let g, e = (sc_global h env) in (g,e)::(sc_globals t e)
640
641
642 (* semantically check program - Emily *)
643 let sc_program (globals, functions) =
644 (*  initialize empty env *)
645 let env =
646 {       locals    = StringMap.empty;
647 globals   = StringMap.empty;
648 functions = StringMap.empty
649 } in
650 (*
651 sc_globals returns list: [(g1,e1), (g2,e2), (g3,e3)....(gn,en)]
652 where g = global , e = environment
653 *)
654 let g = sc_globals globals env in
655 (* make a list of globals *)
656 (* note: fun = function pattern matching *)
657 (* note: elementss returned are in form (g, e)
658 -fst global returns g
659 -snd global returns e
660 *)
661
662 let globals = List.map (fun global -> fst global) g in
663 match g with
664 (* no globals; thus our environment stays the same *)
665 [] -> (globals, (sc_functions (List.rev functions) env))
666 | _ -> let new_env = snd (List.hd (List.rev g)) in
667 (*let new_functions = (fst(List.rev (sc_functions (List.rev functions) new_env))) in
668 new_globals,  new_functions*)
669 (globals, (sc_functions (List.rev functions) new_env))
670
671
```

```
(*         let globals = List.map (fun global -> fst global) g in
match g with
(* no globals *)
[] -> (globals, (check_functions env (List.rev funcs)))
(* get the envirnment from the last global *)
| _ -> let e = snd (List.hd (List.rev g)) in (globals, (check_functions e (List.rev
    funcs)))


*)
```

sast.ml

```
(* SAST *)
type modif_t = Vib_t | Trem_t | Incr_t | Decr_t

(* Not sure if I should make this a string *)
type note_attribute_t = Pitch_t | Vol_t | Dur_t

type dType_t = Double_t | Note_t | Chord_t | Track_t | Rest_t | Score_t

(* operation types *)
type op_t =   Add_t  | Sub_t
            | Mult_t | Div_t
            | Ser_t  | Par_t
            | Equal_t | Neq_t | Geq_t | Leq_t | Greater_t | Less_t

(* Expression type *)
(* Expression type *)
(*
type expr_t =
    Literal of int
  | Id of string
  | NOTE_CR of string * string * string
  | REST_CR of string
  | CHORD_CR of string list
  | TRACK_CR of string
  | ACCESSOR of string * note_attribute_t
  | Binop of expr_t * op_t * expr_t
  | Modifier of expr_t * modif_t
  | Assign of string * expr_t
  | Call of string * expr_t list
  | Noexpr
*)
type expr_t =
    Literal_t of string
  | Id_t of string
  | NOTE_CR_t of expr_t * expr_t * expr_t
  | REST_CR_t of expr_t
  | TRACK_CR_t of expr_t
  | CHORD_CR_t of expr_t list
  | SCORE_CR_t of expr_t list
  | ACCESSOR_t of expr_t * note_attribute_t
  | Binop_t of expr_t * op_t * expr_t
  | Modifier_t of expr_t * modif_t
  | Assign_t of expr_t * expr_t
  | Address_t of expr_t * expr_t
  | Call_t of string * expr_t list
```

```
46    | Noexpr_t
47
48
49    (* | Array of expr list *)
50    (*an array can be a list of expressions*)
51
52  (*variable declaration*)
53  type var_decl_t = {
54    vType_t : dType_t;
55    vName_t : string;
56  }
57
58  type var_init_t = {
59    vDecl_t : var_decl_t;
60    vExpr_t : expr_t;
61  }
62
63  (*need to decide if we are keeping loop or not*)
64  type stmt_t =
65      Block_t of stmt_t list
66    | Expr_t of expr_t
67    | Return_t of expr_t
68    | Print_t of expr_t
69    | If_t of expr_t * stmt_t * stmt_t
70    | For_t of expr_t * expr_t * expr_t * stmt_t
71    | Loop_t of expr_t * stmt_t
72    | While_t of expr_t * stmt_t
73    | Vdecl_t of var_decl_t
74    | Vinit_t of var_decl_t * expr_t
75
76
77  (* funciton declaration *)
78  type func_decl_t = {
79      rtype_t : dType_t;
80      fname_t : string;
81      formals_t : var_decl_t list;
82      body_t : stmt_t list;
83    }
84
85  (*ast is a list of variables and list of function dels*)
86  type program_t = var_decl_t list * func_decl_t list
87
88
89  let rec string_of_expr_t = function
90      Literal_t(l) -> l
91    | Id_t(s) -> s
92    | NOTE_CR_t(a, b, c) ->
93        "note (" ^ string_of_expr_t a ^ ", " ^ string_of_expr_t b ^ ", " ^
              string_of_expr_t c ^ ")"
94    | REST_CR_t(r) -> "rest (" ^ string_of_expr_t r ^ ")"
95    | TRACK_CR_t(track) -> "track (" ^ string_of_expr_t track ^ ")"
96    | SCORE_CR_t(score_list) ->
97    "score (" ^ String.concat " : " (List.map string_of_expr_t score_list) ^ ")"
98    | ACCESSOR_t(a, b) ->
99        (string_of_expr_t a) ^ " -> " ^ (
100       match b with
101         Pitch_t -> "pitch" | Vol_t -> "vol" | Dur_t -> "dur"
```

```
102       )
103   | Assign_t(id, expr) -> string_of_expr_t id ^ " = " ^ string_of_expr_t expr
104   | Address_t(id, expr) -> string_of_expr_t id ^ " [" ^ string_of_expr_t expr ^ "]"
105   | CHORD_CR_t(note_list) ->
106       "chord (" ^ String.concat " : " (List.map string_of_expr_t note_list) ^ ")"
107   | Binop_t(e1, o, e2) ->
108       string_of_expr_t e1 ^ " " ^
109       (match o with
110       Add_t -> "+" | Sub_t -> "-" | Mult_t -> "*" | Div_t -> "/"
111       | Equal_t -> "==" | Neq_t -> "!="
112       | Less_t -> "<" | Leq_t -> "<=" | Greater_t -> ">" | Geq_t -> ">="
113       | Ser_t -> "." | Par_t -> ":") ^ " " ^
114       string_of_expr_t e2
115   | Modifier_t(e1, modif) ->
116       string_of_expr_t e1 ^
117       (match modif with
118       Vib_t -> "^" | Trem_t -> "~" | Incr_t -> "++" | Decr_t -> "--")
119   | Call_t(f, el) ->
120       f ^ "(" ^ String.concat ", " (List.map string_of_expr_t el) ^ ")"
121   | Noexpr_t -> ""
122
123
124 let string_of_vdecl_t v =
125   (match v.vType_t with
126     Double_t -> "double "
127     | Note_t -> "note "
128     | Chord_t -> "chord "
129     | Track_t -> "track "
130     | Rest_t -> "rest "
131     | Score_t -> "score ") ^ v.vName_t
132
133
134 let rec string_of_stmt_t = function
135     Block_t(stmts) ->
136       "{\n" ^ String.concat "" (List.map string_of_stmt_t stmts) ^ "}\n"
137   | Expr_t(expr) -> string_of_expr_t expr ^ ";\n";
138   | Return_t(expr) -> "return " ^ string_of_expr_t expr ^ ";\n";
139   | Print_t(expr) -> "print (" ^ string_of_expr_t expr ^ ");\n";
140   | If_t(e, s, Block_t([])) -> "if (" ^ string_of_expr_t e ^ ")\n" ^
        string_of_stmt_t s
141   | If_t(e, s1, s2) ->  "if (" ^ string_of_expr_t e ^ ")\n" ^
142       string_of_stmt_t s1 ^ "else\n" ^ string_of_stmt_t s2
143   | For_t(e1, e2, e3, s) ->
144       "for (" ^ string_of_expr_t e1  ^ " ; " ^ string_of_expr_t e2 ^ " ; " ^
145       string_of_expr_t e3  ^ ") " ^ string_of_stmt_t s
146   | Loop_t(e, s) -> "loop (" ^ string_of_expr_t e ^ ") " ^ string_of_stmt_t s
147   | While_t(e, s) -> "while (" ^ string_of_expr_t e ^ ") " ^ string_of_stmt_t s
148   | Vdecl_t(v) -> string_of_vdecl_t v ^ ";\n"
149   | Vinit_t(v, e) -> string_of_vdecl_t v ^ " = " ^ string_of_expr_t e ^ ";\n"
150
151
152 let string_of_fdecl_t fdecl =
153   fdecl.fname_t ^
154    (match fdecl.rtype_t with
155     Double_t -> " double "
156     | Note_t -> " note "
157     | Chord_t -> " chord "
```

```
158     | Track_t -> " track "
159     | Rest_t -> " rest "
160     | Score_t -> " score ") ^ "(" ^ String.concat ", " (List.map string_of_vdecl_t
          fdecl.formals_t) ^ ")\n{\n" ^
161   String.concat "" (List.map string_of_stmt_t fdecl.body_t) ^
162   "}\n"
163
164 (*pretty print for program*)
165 let string_of_program_t (vars, funcs) =
166   String.concat "" (List.map string_of_vdecl_t vars) ^ "\n" ^
167   String.concat "\n" (List.map string_of_fdecl_t funcs)
```

compile.ml

```
1  open Sast
2  open Printf
3
4  (* WRITE PROGRAM TO FILE  *)
5  let rec write_to_file file programString =
6    let oc = open_out ("tests/" ^ file ^ "dj.java") in
7    fprintf oc "%s" programString;
8  (*   close_out oc in *)
9
10 and string_of_program file (vars, funcs)=
11   let global_string = write_global_string vars in
12     let func_string = write_func_string file funcs global_string in
13   let out = sprintf
14     "import java.util.*;\nimport jm.JMC;\nimport jm.music.data.*;\nimport
         jm.util.*;\n\npublic class %s implements JMC {\n%s\n}"
15     (file^"dj") func_string in
16     write_to_file file out;
17     out
18
19 and write_global_string vars =
20   let gs = parse_global vars in
21   if List.length gs >= 1
22     then sprintf "%s" ((String.concat ";\n" gs) ^ ";\n")
23   else
24     sprintf "%s" (String.concat ";\n" gs)
25
26 and write_func_string file funcs global_string =
27   let fs = parse_funcs file global_string funcs in
28     sprintf "%s" (String.concat "" fs)
29
30 and parse_global = function
31   [] -> []
32   | h::t -> let global_string = (write_vdecl h) in global_string::(parse_global t)
33
34 and parse_funcs file global_string = function
35   [] -> []
36   | h::t -> let funcs_string = (write_fdecl file global_string h) in
        funcs_string::(parse_funcs file global_string t)
37
38 and write_vdecl v =
39   (match v.vType_t with
40     Double_t -> "\t\tdouble "
41     | Note_t -> "Note "
42     | Chord_t -> "CPhrase "
```

```
43      | Track_t -> "Part "
44      | Rest_t-> "Note "
45      | Score_t -> "Score ") ^ v.vName_t
46
47  and write_vdecl_name v =  v.vName_t
48
49  and write_fdecl file global_string f =
50    (* no song function has arguments *)
51    let stmt_list = write_stmt_list file f.fname_t f.body_t in
52      let stmt_string = String.concat "" stmt_list in
53    (* SONG FUNCTION *)
54    if f.fname_t = "song"
55      then
56          "public static void main(String[] args){\n\tNote [] notes_array;\n" ^
57          global_string ^ "\n" ^
58          stmt_string ^
59          "\n\t\t}\n"
60    (* NON-SONG FUNCTION *)
61    else
62        let formals_list = List.map write_vdecl f.formals_t in
63          let formals_str = String.concat ", " formals_list in
64            "private static " ^
65              (match f.rtype_t with
66                Double_t -> "double "
67                | Note_t -> "Note "
68                | Chord_t -> "CPhrase "
69                | Track_t -> "Part "
70                | Rest_t -> "Rest "
71                | Score_t -> "Score "
72                (* | _ -> "void" *)) ^
73              f.fname_t ^ "( " ^ formals_str ^ " )" ^
74              "\n{\n\tNote [] notes_array;\n" ^ stmt_string ^ "\n\t\t}\n"
75
76  and write_stmt_list file fname = function
77    [] -> []
78    | h::t -> let string_stmt_list = ((write_stmt file fname h)) in
        string_stmt_list::(write_stmt_list file fname t)
79
80  and write_stmt file f_name statement =
81    match statement with
82      Block_t(stmts) -> sprintf "%s" ("\t\t" ^ write_stmt_block file f_name stmts)
83    | Expr_t(expr) -> sprintf "%s;\n" ("\t\t" ^write_expr f_name expr)
84    | Return_t(expr) ->
85      let ex1 = write_expr "junk" expr in
86        if f_name = "song" then
87          sprintf "%s" "\t\tWrite.midi(" ^ ex1 ^",\"" ^ file ^ ".mid\");\n"
88        else sprintf "%s" "\t\treturn " ^ ex1 ^ ";\n"
89    | If_t(e, s, Block_t([])) ->
90        let ex1 = write_expr "junk" e in
91          sprintf "%s" "\t\tif (" ^ ex1 ^ ")\n" ^ write_stmt file f_name s
92    | If_t(e, s1, s2) ->
93        let ex1 = write_expr "junk" e in
94          let s1 = write_stmt file f_name s1 in
95            let s2 = write_stmt file f_name s2 in
96              sprintf "%s" "\t\tif (" ^ ex1 ^ ")\n" ^ s1 ^ "else\n" ^ s2
97    | For_t(e1, e2, e3, s) ->
98      let ex1 = write_expr "junk" e1 in
```

```
99        let ex2 = write_expr "junk" e2 in
100         let ex3 = write_expr "junk" e3 in
101           let s1 = write_stmt file f_name s in
102             sprintf "%s" "\t\tfor (" ^ ex1  ^ "; " ^ ex2 ^ "; " ^ ex3 ^ ") " ^ s1
103     | Print_t(e) -> sprintf "System.out.println(%s);\n" (write_expr f_name e)
104     | While_t(e, s) ->
105     let ex1 = write_expr "junk" e in
106       let s1 = write_stmt file f_name s in
107         sprintf "%s" ("\t\twhile (" ^ ex1^ ") " ^ s1)
108     | Loop_t(e, s) ->
109     let ex1 = write_expr "junk" e in
110       let s1 = write_stmt file f_name s in
111         sprintf "%s" "\t\tfor (int w = 0; w < " ^ ex1 ^ "; w ++) " ^ s1
112     | Vdecl_t(v) -> sprintf "%s" (write_vdecl v ^ ";\n")
113     | Vinit_t(v, e) ->
114       let var = write_vdecl v in
115         (* let name = write_expr "junk" v  *)
116         let ex1 = (write_expr v.vName_t e) in
117           sprintf "%s" ("\t\t" ^ var ^ " = " ^ ex1 ^ ";\n")
118
119 and write_stmt_block file f_name stmts =
120   let stmt_list = (write_stmt_list file f_name stmts) in
121     let stmt_string = String.concat "" stmt_list in
122       "{\n" ^ stmt_string ^ "}\n"
123
124 and write_expr v_name ex =
125   match ex with
126     Literal_t(l) -> sprintf "%s" l
127   | Id_t(s) -> sprintf "%s" s
128   | NOTE_CR_t(a, b, c) ->
129       let ex1 = write_expr "junk" a in
130         let ex2 = write_expr "junk" b in
131           let ex3 = write_expr "junk" c in
132      sprintf "%s" "new Note((double)" ^ ex1 ^ ", " ^  ex3 ^ ", (int) " ^ ex2 ^ ")"
133   | REST_CR_t(r) ->
134       let ex1  = write_expr "junk" r in
135       sprintf "%s" "new Note( REST, " ^ ex1 ^ ")"
136   | ACCESSOR_t(a, b) ->
137       let ex1 = write_expr "junk" a in
138         let access_type = (
139           match b with
140           Pitch_t -> "getFrequency()" | Vol_t -> "getDynamic()" |  Dur_t ->
                "getDuration()"
141           ) in
142       sprintf "%s" ex1 ^ "." ^ access_type
143   | Assign_t(id, expr) ->
144       let identifier = write_expr "junk" id in
145         let ex = write_expr identifier expr in
146           sprintf "%s" identifier ^ " = " ^ ex
147   | Address_t(id, expr) ->
148       let identifier = write_expr "junk" id in
149         let ex = write_expr identifier expr in
150           sprintf "%s.getPhrase((int)%s)" identifier ex
151   | CHORD_CR_t(note_list) ->
152     let notes = write_expr_list "junk" note_list in
153       let notes_string = String.concat ", " notes in
```

```
154         sprintf "%s" " new CPhrase();\n" ^ "\t\t" ^ v_name ^ ".setAppend(true);\n"^
               "\t\tnotes_array = new Note [] {" ^ notes_string ^ "};\n\t\t" ^
               v_name^ ".addChord(notes_array);"
155 (* What exactly is track.. track creation, because that's what I'm writing it as.
       also where is the instrument part*)
156   | TRACK_CR_t(instr) ->
157     let ex1  = write_expr "junk" instr in
158       sprintf "%s" "new_Part( (int) " ^ ex1 ^ ")"
159   (* GLOBAL VARIABLES???? *)
160   | SCORE_CR_t(track_list) ->
161     let track_adds = write_score_track_list v_name track_list in
162       let track_str = String.concat ";\n" track_adds in
163         sprintf "%s" ("new_Score();\n" ^ track_str)
164   | Binop_t(e1, o, e2) ->
165       let ex1 = write_expr "junk" e1 in
166         let ex2 = write_expr "junk" e2 in
167           let op = (
168             match o with
169               Add_t -> "+" | Sub_t -> "-" | Mult_t -> "*" | Div_t -> "/"
170               | Equal_t -> "==" | Neq_t -> "!="
171               | Less_t -> "<" | Leq_t -> "<=" | Greater_t -> ">" | Geq_t -> ">="
172               (* serial (.); track.chord, track.track? *)
173               | Ser_t -> "."
174               (* parallel (:); score:track, chord: note/rest*)
175               | Par_t -> ":" ) in
176         (*serial add*)
177     if op = "." then sprintf "%s" ("\t" ^ ex1 ^ ";\n" ^ v_name ^ ".addCPhrase(" ^
           ex2 ^ ")")
178     else if op = ":" then sprintf "%s" ("\t" ^ ex1 ^ ";\n" ^ "\tnotes_array = new
           Note [] {"^ ex2 ^ "};\n" ^ v_name ^ ".addChord( notes_array )")
179       else sprintf "%s" (ex1 ^ " " ^ op ^ " " ^ ex2)
180     (* if op = ":" then sprintf "%s" (ex1 ^ ".addPart(" ^ ex2 ^ ")") *)
181
182
183   | Modifier_t(e1, modif) ->
184       let modifier = (
185         match modif with
186         Vib_t -> ""
187         | Trem_t -> ""
188         | Incr_t -> "++"
189         | Decr_t -> "--") in
190         (* | Incr_t -> ".setPitch((" ^ write_expr "junk" e1 ^".getPitch()) + 50)"
191         | Decr_t -> ".setPitch((" ^ write_expr "junk" e1 ^".getPitch())  -50)") in
               *)
192       sprintf "%s" ((write_expr "junk" e1) ^ modifier)
193   | Call_t(f, el) ->
194         let calls = write_expr_list "junk" el in
195         sprintf "%s" (f ^ "(" ^ String.concat ", " calls ^ ")")
196   | Noexpr_t -> sprintf "%s" ""
197
198 and write_score_track_list vname = function
199   [] -> []
200   | h::t -> let track_str_list = ("\t\t" ^ vname ^ ".addPart(" ^ (write_expr "junk"
        h) ^ ")") in track_str_list::(write_score_track_list vname t)
201
202 and write_chord_list vname chord_list =
203   match chord_list with
```

```
    [] -> []
  | h::t -> let track_str_list = ("\t\t" ^ vname ^ ".addCPhrase(" ^ (write_expr
      "junk" h) ^ ")") in track_str_list::(write_score_track_list vname t)


and write_expr_list v_name expr_list =
  match expr_list with
   [] -> []
   | hd::tl -> let string_expr_list = (write_expr v_name hd) in
      string_expr_list::(write_expr_list v_name tl)
```

compile (.sh)

```
# the main class name
MAIN=$1

if [[ $1 =~ 'clean' ]]; then
        echo "Cleaning up"
        echo "rm -rf tests/*.java tests/*.class tests/*.mid"
        rm -rf tests/*.java tests/*.class tests/*.mid
else
        # jMusic Jars
        #JM_JAR="java/jMusic/jMusic1.6.4.jar"
        #JM_INSTR="java/jMusic/inst/"

        # Set the CLASSPATH
        #CLASSPATH="tests:$JM_JAR:$JM_INSTR:."

        # Java tools
        #JFLAGS="-classpath $CLASSPATH"
        #JCFLAGS="-sourcepath tests -d tests -classpath $CLASSPATH"

        echo "Compiling"
        echo "javac -sourcepath tests -classpath
            tests:java/jMusic/jMusic1.6.4.jar:java/jMusic/inst/:. tests/$MAIN.java"
        javac -sourcepath tests -classpath
            tests:java/jMusic/jMusic1.6.4.jar:java/jMusic/inst/:. tests/$MAIN.java

        echo "Running"
        echo "java -classpath tests:java/jMusic/jMusic1.6.4.jar:java/jMusic/inst/:.
            $MAIN"
        java -classpath tests:java/jMusic/jMusic1.6.4.jar:java/jMusic/inst/:. $MAIN
fi
```

Makefile

```
#MAKEFILE
# name of compiler: wdjc
# name of file extension: .dj

# FILES NEEDED

# TESTS = \
#  ....


OBJS = scanner.cmo parser.cmo ast.cmo sast.cmo semcheck.cmo compile.cmo  wdjc.cmo

```

```
wdjc : $(OBJS)
	ocamlc -o wdjc $(OBJS)


.PHONY : test
test : wdjc testall.sh
	./testall.sh


scanner.ml : scanner.mll
	ocamllex scanner.mll

parser.ml parser.mli : parser.mly
	ocamlyacc parser.mly

%.cmo : %.ml
	ocamlc -c $<

%.cmi : %.mli
	ocamlc -c $<

.PHONY :          clean
clean :
	rm -f wdjc parser.ml parser.mli scanner.ml testall.log \
	*.cmo *.cmi *.out *.diff

# Generated by ocamldep *.ml *.mli
ast.cmo:
ast.cmx:
sast.cmo:
sast.cmx:
semcheck.cmo: ast.cmi sast.cmi
semcheck.cmx: ast.cmx sast.cmx
parser.cmo: ast.cmo parser.cmi
parser.cmx: ast.cmx parser.cmi
parser.cmi: ast.cmo
scanner.cmo: parser.cmi
scanner.cmx: parser.cmx

compile.cmo: ast.cmo
compile.cmx: ast.cmx

wdjc.cmo: scanner.cmo parser.cmi ast.cmo sast.cmo semcheck.cmo compile.cmo
wdjc.cmx: scanner.cmx parser.cmx ast.cmx sast.cmx semcheck.cmx compile.cmx
```

12 Appendix D: Git Log

git log

```
commit 2717bac58e24af585b3754fb45a4352203915b40
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Fri Dec 20 03:52:25 2013 -0500

    adds midi samples

commit ab83b8c311de559330cfcaec457f1b905080a95f
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Thu Dec 19 13:10:03 2013 -0500

    cleanup

commit 0c530c91cd5dd15592eb08e8294d02140573b27d
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Thu Dec 19 10:54:38 2013 -0500

    test fixes

commit 2171288362a2e0c20b8523f38b7437e4d2e1ff39
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Thu Dec 19 09:15:24 2013 -0500

    ZELDA IS DONE ENOUGH

commit 57d3b6dc93fbc14620ba6dbaea53f94c03257fb7
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Wed Dec 18 17:01:52 2013 -0500

    dissonance

commit 1b3ef4d0d33898af8ecd0452584e3c90c7c3b315
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Wed Dec 18 16:41:58 2013 -0500

    ZELDA MUST BE DONE. ALLCAPS

commit 8227c73e22548c3ccffc9c442a70e69a5f276147
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Wed Dec 18 15:11:34 2013 -0500

    dumb save

commit 402f674c77252dc58f20f5458f7557e93eb86573
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Wed Dec 18 15:10:37 2013 -0500

    prettify's map

commit c302510d9f9d6006e18b7df9f16c0e9f1b6ff1f2
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Wed Dec 18 15:00:58 2013 -0500

    PRETTYMAKE

commit 3ca1ab1f7bb552ff4d51d9c5fc5cbd0079c8c785
Author: William Falk-Wallace <wfalkwallace@gmail.com>
```

```
Date:   Wed Dec 18 14:59:10 2013 -0500

    comments out sys.command javac problems

commit 3ee05f36e4634e74d849a4eb0e618a77b5b2120a
Merge: 0d1c2c5 b4bf1f4
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Wed Dec 18 14:53:21 2013 -0500

    MERGECOMPILEFINALLY

commit b4bf1f4355082e243a4fc11eeeac7cde1c1f3e05
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Wed Dec 18 14:45:13 2013 -0500

    hooray

commit c085d40ee726ad5d8e7e3f4fa67c013d5de33da8
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Wed Dec 18 14:33:06 2013 -0500

    FIX

commit bc271bd66a3f73c918c9a0972f8d5ab08646dc02
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Wed Dec 18 13:59:18 2013 -0500

    changes

commit 4e27eef7cb95e8e3628c7d74834bd6c50f6d8232
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Wed Dec 18 13:42:59 2013 -0500

    trouble shooting

commit 12358d6bbc98fdc57866cc5ccf6bebbf6f92d338
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Wed Dec 18 11:48:39 2013 -0500

    updates so midi arent kept and semicolon

commit 23715fdf0be4fe28f86db8f3e140026dfd627d79
Merge: f27e0dc 4a9ba69
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Wed Dec 18 11:41:22 2013 -0500

    MERGE JSERPAR

commit f27e0dc88e261472a515db8b1b13335584b17ff2
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Wed Dec 18 11:38:52 2013 -0500

    no dashes in java names

commit 4f2465e2f34bf9bb6616620c5b5efb03d94056f3
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Wed Dec 18 11:38:10 2013 -0500
```

```
    SO MUCH STUFF

commit f6421def6f0608a1dfa9c57c529ff265fe8cd866
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Wed Dec 18 04:13:39 2013 -0500

    junxors

commit d0fd463573ed202f4a21b58306974b4cb05cced2
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Wed Dec 18 04:03:14 2013 -0500

    stuff

commit d082e83cf2b0b8de5e2fc2fa4fdb0b4766360740
Merge: cb7b406 11fb1b6
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Wed Dec 18 03:45:49 2013 -0500

    mergez

commit cb7b406f00c90da7d11c03485545a91c9b9c5ad3
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Wed Dec 18 03:45:24 2013 -0500

    fixes underscore naming

commit 134a500d7d199f6ff9700e2c8e69ad5fd83d9884
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Wed Dec 18 03:40:54 2013 -0500

    stuff and changes

commit 11fb1b683eb57f9e82d9e1954cbcc22d20a4536a
Author: elemonier <emily.lemonier@gmail.com>
Date:   Wed Dec 18 03:31:12 2013 -0500

    correct while.dj test.

commit ffd9d5a1cc0b4bc421f20647131b85e4b7102b63
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Wed Dec 18 03:04:48 2013 -0500

    compile .sh change

commit e6e5eb9b8b2a7cc1ba3eaf07eb76974a2c7e286d
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Wed Dec 18 02:59:13 2013 -0500

    updates jcompiler

commit 7cd74f11d6262c97bdc0268e955a6eb5b07312c6
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Wed Dec 18 02:56:03 2013 -0500

    syncs tests with master
```

```
commit 8928a5d09fe7720c6264fe909a41920b383362c3
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Wed Dec 18 02:52:44 2013 -0500

    updares to automatic java file and compilation.

commit 4a9ba69158a6b8b6f8ea3804c4ac71ebace79e20
Author: elemonier <emily.lemonier@gmail.com>
Date:   Wed Dec 18 00:05:52 2013 -0500

    serial working. test at serial.dj

commit 41233df1cb3b65d59ca330d6c84e632252eab1c4
Author: elemonier <emily.lemonier@gmail.com>
Date:   Tue Dec 17 23:53:53 2013 -0500

    serial add is working.

commit 7bf50b79a85348ae11e54d26867821000c02fa65
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Tue Dec 17 23:07:34 2013 -0500

    fixes some junk in tests for new ser/par; and id_type checking error match

commit 4ce4f7140907e2e43abf9931369a7d1d8912afef
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Tue Dec 17 22:53:47 2013 -0500

    removes print junk

commit 620e4ec05c868fb3dd38f56bd20573f5284e2d0b
Author: elemonier <emily.lemonier@gmail.com>
Date:   Tue Dec 17 22:36:43 2013 -0500

    More complicated accessor example.

commit ca6d9c6e787e4003c81ae3d07b9d1e5e1bf67b2a
Author: elemonier <emily.lemonier@gmail.com>
Date:   Tue Dec 17 22:36:24 2013 -0500

    compile adds some serial implementaion.

commit 568e2c4f30898e19c30320fd098be028a44590b0
Merge: e923e94 62e24ad
Author: Thomas <tee2103@columbia.edu>
Date:   Tue Dec 17 22:30:30 2013 -0500

    handles merge conflict

commit e923e94d4b48cbcbabd0ce83f3f75fa51d700bf1
Author: Thomas <tee2103@columbia.edu>
Date:   Tue Dec 17 22:28:10 2013 -0500

    resolves shift/reduce errors. Accessor, Addressor modified

commit 4aa4469cfbbf6b509595b09d38b3d68b17a7b473
```

```
Author: elemonier <emily.lemonier@gmail.com>
Date:   Tue Dec 17 22:04:05 2013 -0500

    more complicated accessor. fixing small compile bugs.

commit 62e24adc1901b85b21cd53b3f90794c7c1b2e273
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Tue Dec 17 21:28:14 2013 -0500

    stuff

commit 65424bf13137706706eaf2c1d0a0655dfb6b1d56
Author: elemonier <emily.lemonier@gmail.com>
Date:   Tue Dec 17 21:23:32 2013 -0500

    parallel add only between score:track, not score:(track/score)

commit d815c71030ab43590d1711b517045b10339e181a
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Tue Dec 17 19:41:04 2013 -0500

    reups from master semcheck

commit 2c7ff528d9ff7e3b5e97f582e2dd31136d39a8cc
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Tue Dec 17 11:57:44 2013 -0500

    removes extra files

commit a0dab2c68476c340402636fd44d6bdc2351e0447
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Tue Dec 17 11:57:07 2013 -0500

    updates output location java code and parameter

commit 559899ab22a4b67211e9388ca315f0a97d0ec8d1
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Tue Dec 17 11:55:15 2013 -0500

    updates wdjc and java make to build test

commit 24991cc1815dd2ce4b124a9f9fcfd85f62c73e44
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Tue Dec 17 11:54:04 2013 -0500

    updares output file location

commit 7d552d0f31eb4986b558f73113163e2de527d676
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Tue Dec 17 11:53:25 2013 -0500

    updarted compile script

commit 6b1076966c66c8c841d8f9a97a1453d3b914d1c7
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Tue Dec 17 09:31:37 2013 -0500
```

```
    corrections

commit 0a5bfa7ed9cc4ee64405fcd0fb255b568c4d7351
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Tue Dec 17 04:16:46 2013 -0500

    automating compile test

commit 8bf742940e0816a66d6bc93c89656c6760479c7f
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Tue Dec 17 03:03:54 2013 -0500

    updates gitignore for make clean junk

commit d94f2edf2b1edb8794b69387141048e82d8a4e87
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Tue Dec 17 03:03:19 2013 -0500

    adds test flag template

commit da0dc220a04115a950fceebdff54c979ae41a1be
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Tue Dec 17 02:53:06 2013 -0500

    fixes track constructor issue; it's not a lot more like rest

commit 1744b836771ece78f8d9a75ebc7dca6154fa3f75
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Tue Dec 17 02:52:42 2013 -0500

    adds extra case to some check somewhere for no good reason. well. fine. its a
       good reason. I'm so tired I'm talking to git now.

commit 0d1c2c5aedc052370b0a576050cce143301dcf29
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Mon Dec 16 23:46:04 2013 -0500

    should fix contgeo

commit 6eb2e066e5b959ca20848cb8951f7866d16c8418
Merge: 2eb420d f14507e
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Mon Dec 16 23:41:16 2013 -0500

    merge\!

commit 2eb420df46600a292eabae3270360abbbb51e095
Author: elemonier <emily.lemonier@gmail.com>
Date:   Mon Dec 16 21:40:42 2013 -0500

    pulls from origin/master into compile

commit f14507eed7955682cbf1b29113159bbff22cc2c7
Author: elemonier <emily.lemonier@gmail.com>
Date:   Mon Dec 16 21:31:13 2013 -0500

    fixes -t to compile + prodice midi file
```

```
commit e7c2576c34365efe700409750e96c8cf02b92db2
Author: elemonier <emily.lemonier@gmail.com>
Date:   Mon Dec 16 20:50:38 2013 -0500

    Adds default name of test for any .java gen.

commit 40ec3efb2b123752b06b1893a0334c7d944d7917
Author: elemonier <emily.lemonier@gmail.com>
Date:   Mon Dec 16 20:49:08 2013 -0500

    Adds optional name functionality to java gen.

commit 82a396bdb7b507deff70524ed2caf8f541571336
Author: elemonier <emily.lemonier@gmail.com>
Date:   Mon Dec 16 16:51:03 2013 -0500

    complete hello world java gen

commit d7733d694a663a373d0151946624d48e62728a64
Author: elemonier <emily.lemonier@gmail.com>
Date:   Mon Dec 16 14:31:42 2013 -0500

    Deletes old to_str functions

commit 51b998fe669cd4bba4c72ea40fb6fc5d2704f1e6
Author: elemonier <emily.lemonier@gmail.com>
Date:   Mon Dec 16 14:22:58 2013 -0500

    Fleshes out stmt javagen

commit 170a80e54b00c22d839e3625e4421acf7daa4adf
Author: elemonier <emily.lemonier@gmail.com>
Date:   Mon Dec 16 13:51:26 2013 -0500

    Fleshes out write_expr and write_expr_list

commit 9a642387d6c5e3b15c1c5d8acae39fdbb8e34f84
Author: elemonier <emily.lemonier@gmail.com>
Date:   Mon Dec 16 13:25:15 2013 -0500

    implements basic javagen statements.

commit a67b1d7f405e14746a244268f071e88eecfb5c8d
Author: elemonier <emily.lemonier@gmail.com>
Date:   Mon Dec 16 13:10:50 2013 -0500

    Adds basic write function functionality. Fixes globals.

commit 64b8c9724dd712e4f40afa99410d5b9c80015e92
Author: elemonier <emily.lemonier@gmail.com>
Date:   Mon Dec 16 12:39:40 2013 -0500

    Writes globals. BOO-YAHH

commit 53303d625d687ff7fd410e655b95d5a8b9ac6caf
Author: elemonier <emily.lemonier@gmail.com>
```

```
Date:   Mon Dec 16 12:13:51 2013 -0500

    Adds write_global_string and write_func_string functions to compile.

commit bb5bc3d484adab63bd33a0a5b5e65c7757a4f072
Author: elemonier <emily.lemonier@gmail.com>
Date:   Mon Dec 16 12:08:06 2013 -0500

    feeds program into string_of_program

commit d5ae50aaaefc3df970d09cc6363ebfaf0ac7bed3
Merge: 8055115 9898225
Author: elemonier <emily.lemonier@gmail.com>
Date:   Mon Dec 16 11:57:26 2013 -0500

    fuck geo.

commit 80551153c4e3fd7d8fde7e4b64b130fc27e7643f
Author: elemonier <emily.lemonier@gmail.com>
Date:   Mon Dec 16 11:55:19 2013 -0500

    output file from current compile.ml

commit c99b8f2427b4edfca894692d77360e0b25efbcd4
Author: elemonier <emily.lemonier@gmail.com>
Date:   Mon Dec 16 11:55:02 2013 -0500

    makes -j flag write to java/Dj.java

commit f96c404fa5866cdcbb959a182e9267437dbad513
Author: elemonier <emily.lemonier@gmail.com>
Date:   Mon Dec 16 11:54:37 2013 -0500

    prelim print to java/DJ.java

commit 480c8d3dca6c154e57836364432b63eacbe7e391
Merge: 30ef034 e06e7a8
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Mon Dec 16 01:24:55 2013 -0800

    Merge pull request #30 from WHEF-PLT/access

    Access

commit e06e7a8fc449130345737ec25a0ee51dce349af4
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Mon Dec 16 04:24:19 2013 -0500

    ADDRESSOR WAHOO. fixes #4

commit f6f8317ac7596f50ccb7f1aab777ba81021dbe52
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Mon Dec 16 04:23:51 2013 -0500

    loop, addressor test file

commit 30ef034f040a4edb40960744b9a19c50876f54a3
```

```
Merge: 5531702 7d1ebbb
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Mon Dec 16 00:36:20 2013 -0800

    Merge pull request #29 from WHEF-PLT/print

    Print

commit 7d1ebbb9f5d4e0843ba3bcc7194e99195329c123
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Mon Dec 16 03:36:07 2013 -0500

    fixes #26, defines print type and typechecking and building

commit a57b5ba835e2a8b5844f30dd9eec5994e305aef4
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Mon Dec 16 03:28:52 2013 -0500

    func call return type can be any

commit 55317025c64aba724af23fb22c65f263686f3b63
Merge: 9d64648 e67ffea
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Mon Dec 16 00:09:25 2013 -0800

    Merge pull request #27 from WHEF-PLT/loop

    carries through loop like while; closes #21

commit e67ffea1935a51c78a0b44bae1c0fe52b568ec13
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Mon Dec 16 03:08:22 2013 -0500

    carries through loop like while; closes #21

commit 9d646485f718e4183e7c5d55af601b8e4f72bc2f
Merge: d7ffea5 72f9530
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Sun Dec 15 23:42:21 2013 -0800

    Merge pull request #24 from WHEF-PLT/bools

    fixes #19; boolean binops now allow primitve types

commit 72f95305921eb30d507a82b5bb20bdd63c7d001b
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Mon Dec 16 02:41:52 2013 -0500

    fixes #19; boolean binops now allow primitve types

commit d7ffea59942dc56c219d0b165e6fed108af6f3e4
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Mon Dec 16 02:32:54 2013 -0500

    adds zelda theme template

commit cf0b40b256648e99c0beacb252f6430e611d97c0
```

```
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Mon Dec 16 02:32:38 2013 -0500

    adds note frequency map text reference

commit 0e4ba42822e8bb6d1cdfaa3b364b5f87cee5f90a
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Sun Dec 15 23:02:38 2013 -0500

    fixes #5; serial and parallel add through type check

commit eac9508bd1c7df32a2bf1b0c7ae11b62d8ef96d8
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Sun Dec 15 18:16:36 2013 -0500

    fixes binop tc for ser/par

commit 9898225818b85f26a94aa4e6a82a449e0a048839
Author: elemonier <emily.lemonier@gmail.com>
Date:   Sun Dec 15 18:12:03 2013 -0500

    printing skeleton to a file.

commit 7681336bd0820d6e752401daa4ee3dd459507636
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Sun Dec 15 17:49:36 2013 -0500

    adds paralell binop typechecking

commit f9944004b91866eaf8c0b7ca6039642367d130ca
Author: elemonier <emily.lemonier@gmail.com>
Date:   Sun Dec 15 17:19:39 2013 -0500

    fixes make.

commit 108e4910a38a17ea9a526c06fde1b035d4ae12ae
Author: elemonier <emily.lemonier@gmail.com>
Date:   Sun Dec 15 17:17:42 2013 -0500

    dj java file.

commit 21b728a4db7a3e8cc494199af63997aaff3d0fbf
Author: elemonier <emily.lemonier@gmail.com>
Date:   Sun Dec 15 17:17:24 2013 -0500

    hello world dj

commit 04dc2dfd5dbc115c82f1a9e4c9079fcb4077c2ed
Author: elemonier <emily.lemonier@gmail.com>
Date:   Sun Dec 15 17:17:14 2013 -0500

    no fucking clue.

commit 17c417842fe97ff7d3d1a9de6006d62072fe79d9
Author: elemonier <emily.lemonier@gmail.com>
Date:   Sun Dec 15 17:16:59 2013 -0500
```

```
    java makefile only makes one target.

commit 1ba7bb326b1382ae7e74fd8eba8142428d698735
Author: elemonier <emily.lemonier@gmail.com>
Date:   Sun Dec 15 17:16:45 2013 -0500

    compile outputs hello world.

commit e303c91fa5a9d4251b567ea7dd48762586357b0d
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Sun Dec 15 16:41:05 2013 -0500

    updates parallel todo

commit 47d4eb0bfcec350f78a059e533eaf1f9975c57f4
Merge: 04b529e 07d6e06
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Sun Dec 15 12:46:18 2013 -0800

    Merge pull request #18 from WHEP-PLT/score_check

    Score check

commit 07d6e0686f7da09e20cba0e41b2494b1677dcd21
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Sun Dec 15 15:45:58 2013 -0500

    updates test for score type checking

commit 97188c2c7f2b2a0ab4bccbb00eacb757fdd7defe
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Sun Dec 15 15:42:02 2013 -0500

    fixes semcheck song return type checking

commit 2ec2f52c685393e16cf95f5a2ce1b09aedc3e63c
Merge: 981fa33 04b529e
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Sun Dec 15 15:13:59 2013 -0500

    updates all tests for score typechecking; adds multifunc tests

commit f3e88f6a1aea4b6e69224a424ce0e268569c3e03
Author: elemonier <emily.lemonier@gmail.com>
Date:   Sun Dec 15 15:11:18 2013 -0500

    tom is amazing. fixed vdecl name.

commit 04b529e51341df0b58a38c4c11f51771ef3c9218
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Sun Dec 15 15:06:26 2013 -0500

    FIXES ALL THE MERGEZ

commit 926c59d7e8a87a56684f42088646ee92da0c2cee
Merge: 1c8c0d6 259e756
Author: William Falk-Wallace <wfalkwallace@gmail.com>
```

```
Date:   Sun Dec 15 11:54:48 2013 -0800

    Merge pull request #16 from WHET-PLT/score

    Score

commit 259e756c2689abf074a3f8d3c920a0025aee4888
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Sun Dec 15 14:54:22 2013 -0500

    closes #12, fixes #15

commit f42d008adfb9b8439fff64ce6ccd7800ae55f6fd
Author: elemonier <emily.lemonier@gmail.com>
Date:   Sun Dec 15 14:46:54 2013 -0500

    modifies compile.ml to include score pretty print.

commit 981fa335aaf9adbf1259ea0f52d7ad407a8ff8c7
Author: Thomas <tee2103@columbia.edu>
Date:   Sun Dec 15 14:46:09 2013 -0500

    adds return type checking

commit 89d6bf4a9a19835e167f36b56b4b6c5d3dba0b04
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Sun Dec 15 14:45:23 2013 -0500

    fixes sast function type print order

commit e1cf14cf756da2cb974bc5ed1739f696e0d70d9a
Author: elemonier <emily.lemonier@gmail.com>
Date:   Sun Dec 15 14:45:16 2013 -0500

    fixes pretty print stupidity.

commit 46d2a64ba7d469fc639d1607507253fe784b7c97
Merge: 028aa32 1c8c0d6
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Sun Dec 15 14:44:16 2013 -0500

    merge in master changes

commit 4d7c070abfaa87cd076113d06de07d81fb4cde04
Author: elemonier <emily.lemonier@gmail.com>
Date:   Sun Dec 15 14:41:23 2013 -0500

    modifies compile and tests.

commit fafb7eaeec46f6b156f31f69b3487ad37094f55b
Merge: f293495 028aa32
Author: elemonier <emily.lemonier@gmail.com>
Date:   Sun Dec 15 14:40:53 2013 -0500

    functioning compile needs to fix v_name optional argument.

commit 1c8c0d6d8c05dcdee8b317fb6572b8edaea1ee53
```

```
Merge: 5a7d985 782ba00
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Sun Dec 15 11:31:23 2013 -0800

    Merge pull request #17 from WHEP-PLT/trk_constr

    Trk constr

commit 782ba00962910f03859e972121239a98f66b7857
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Sun Dec 15 14:26:46 2013 -0500

    updates track constructor typechecking on double. fixes #7. corrects initialize
       test for new constr

commit f293495c10db098338c947af23ccac72cbc31544
Author: elemonier <emily.lemonier@gmail.com>
Date:   Sun Dec 15 14:06:47 2013 -0500

    score test.

commit 8350275c17e18a0d39eda5c891c223aafd447e5a
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Sun Dec 15 14:05:16 2013 -0500

    adds track cr takes expr thoughrout

commit f799d461df1cd12a56d5a1b373c9607e6d2eec4b
Author: elemonier <emily.lemonier@gmail.com>
Date:   Sun Dec 15 14:01:59 2013 -0500

    Adds f_name param to string_of_stmt_t and modifies song return.

commit 028aa320411d503eaf7d9baa4a072d208ccd2833
Author: elemonier <emily.lemonier@gmail.com>
Date:   Sun Dec 15 02:00:26 2013 -0500

    deletes comments from semcheck

commit 078dc2866548812d54e633763babffefac974eb1
Author: elemonier <emily.lemonier@gmail.com>
Date:   Sun Dec 15 01:59:22 2013 -0500

    added doubles to hello world test.

commit 8518da856c5d26674186f78ea73d05b3350acc78
Author: elemonier <emily.lemonier@gmail.com>
Date:   Sun Dec 15 01:54:12 2013 -0500

    adds score primitive. fixes #15

commit f783fbc3ceef10b0a2e83aaf6d47d676de0e8930
Author: elemonier <emily.lemonier@gmail.com>
Date:   Sun Dec 15 01:49:22 2013 -0500

    score test.
```

```
commit d85f8033c32a138e262989553413c6fb8a156539
Merge: 322c7b6 7503c83
Author: elemonier <emily.lemonier@gmail.com>
Date:   Sun Dec 15 00:34:16 2013 -0500

    merge fix.

commit 322c7b61d1b97bfca254f0bfb65964ffcf407537
Author: elemonier <emily.lemonier@gmail.com>
Date:   Sun Dec 15 00:31:03 2013 -0500

    non-working optional args. Tom + i will try to fix.

commit 7503c83de47327add528abf854c5bb909e3898f3
Author: hilagutfreund <hila.gut@gmail.com>
Date:   Sat Dec 14 23:55:45 2013 -0500

    changed chord create a bit to work properly (does it with part) and added a
       score create with parts.

commit 5a7d985fbdbc6ac05566633af74712269f23f045
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Sat Dec 14 21:11:18 2013 -0500

    removes microc bytecode

commit bac9a764ee25f955954145481193eccabe33079d
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Sat Dec 14 19:40:04 2013 -0500

    renames test.sh to test BECAUSE ITS EASIER

commit f3a18d47769851d7b994a5def240651f51fe41a7
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Sat Dec 14 19:39:24 2013 -0500

    updates printing to notify which flag

commit 7daad5469b3c52a168b5310024fa475b9c9ac8bf
Merge: e046512 dd3ec80
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Sat Dec 14 16:26:09 2013 -0800

    Merge pull request #13 from WHET-PLT/floatify

    Floatify

commit dd3ec80577290aa2410568b49456b3c4012b9115
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Sat Dec 14 19:23:39 2013 -0500

    replaces int with double across tests

commit 7ae994d90fa00664568b1505a72c596ad751bd9d
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Sat Dec 14 19:21:39 2013 -0500
```

```
    changes int to double through sast

commit e514f66fb5a669ed9b2e09449bc530f089ca1c5f
Merge: 662725f ab7bbce
Author: elemonier <emily.lemonier@gmail.com>
Date:   Sat Dec 14 18:52:13 2013 -0500

    geo conflict. fuck all the things.

commit 662725fe9e081b46bf812ab3d8ef347b006fa70a
Author: elemonier <emily.lemonier@gmail.com>
Date:   Sat Dec 14 18:51:46 2013 -0500

    removes commments from compile.

commit 6064118c42cc9369953885f937a76b79825968dd
Author: elemonier <emily.lemonier@gmail.com>
Date:   Sat Dec 14 18:51:28 2013 -0500

    make clean. always.

commit ab7bbce1d56e11f8b36bb9ce68e608d37ac3d1ad
Merge: 0487229 cbe0864
Author: hilagutfreund <hila.gut@gmail.com>
Date:   Sat Dec 14 18:50:02 2013 -0500

    fuck geo.

commit 048722977d2b01771d916fdf9aaaff543b0656f8
Author: hilagutfreund <hila.gut@gmail.com>
Date:   Sat Dec 14 18:49:28 2013 -0500

    Fixes git rm and adds chord example.

commit cbe08644aef08361ed735e986298d47f1f245819
Merge: 659dfe7 7256aee
Author: elemonier <emily.lemonier@gmail.com>
Date:   Sat Dec 14 18:43:25 2013 -0500

    fucking geo.

commit 659dfe7f899dab77d958013ee79d63543358aeb3
Author: elemonier <emily.lemonier@gmail.com>
Date:   Sat Dec 14 18:42:57 2013 -0500

    modifies string_of_fdecl_t to handle special case of song function.

commit 7256aee6de132226563afda850bc1d00f501835f
Author: hilagutfreund <hila.gut@gmail.com>
Date:   Sat Dec 14 18:37:48 2013 -0500

    create chord ex added in java

commit 796856099c8b67e4e2813f82b36ed648d06682ba
Author: hilagutfreund <hila.gut@gmail.com>
Date:   Sat Dec 14 18:26:22 2013 -0500
```

```
    create chord ex added in java

commit 8c30f6d635bc8768bc2acb89694ba926ae25dad3
Author: hilagutfreund <hila.gut@gmail.com>
Date:   Sat Dec 14 17:42:12 2013 -0500

    changed REST

commit 8809b58f20abb7dbbbaf339fe00d592cc99ce9c1
Author: elemonier <emily.lemonier@gmail.com>
Date:   Sat Dec 14 17:11:46 2013 -0500

    modifies imports in compile.

commit dba89e761f383ad154fb9ea648e94c28bbc53556
Merge: a74bf7b 514412b
Author: elemonier <emily.lemonier@gmail.com>
Date:   Sat Dec 14 15:23:43 2013 -0500

    geo

commit 514412b2ad2c45a4fb5a3b6f71539b31505c37d3
Author: hilagutfreund <hila.gut@gmail.com>
Date:   Sat Dec 14 15:17:59 2013 -0500

    one note hello world

commit a74bf7b44891342cab975605e513440be42f665b
Author: elemonier <emily.lemonier@gmail.com>
Date:   Sat Dec 14 14:46:50 2013 -0500

    added imports to song construct.

commit f343a4320198f501d570d78f50a17235b4a973f7
Merge: ac9528d 1c397c9
Author: elemonier <emily.lemonier@gmail.com>
Date:   Sat Dec 14 14:37:25 2013 -0500

    fuck geo.

commit ac9528da8f368f4cea17d4cea4c7d2fb5ee96209
Author: elemonier <emily.lemonier@gmail.com>
Date:   Sat Dec 14 14:37:06 2013 -0500

    modified compile moderately.

commit 1c397c9bbe398e22222e3629039950c6e68a0187
Author: hilagutfreund <hila.gut@gmail.com>
Date:   Fri Dec 13 22:37:22 2013 -0500

    added simple hello world program: helloWorld-Note.dj

commit 97119d10007af8fab99825841c9723fb91d234f2
Merge: ca74af0 f57f39f
Author: hilagutfreund <hila.gut@gmail.com>
Date:   Fri Dec 13 22:08:04 2013 -0500
```

```
    changed java note creation slightly

commit ca74af0d3dba1089451c860d38a1c1bdfa76a44e
Author: hilagutfreund <hila.gut@gmail.com>
Date:   Fri Dec 13 22:07:14 2013 -0500

    slight changes to note create in java file

commit 4adc722a02a58189e85ed94823d590cb3d0d46d6
Author: elemonier <emily.lemonier@gmail.com>
Date:   Fri Dec 13 21:45:29 2013 -0500

    fixed odd chord printing.

commit f57f39ff644fd1d6a733fd1c22732f390d226f19
Author: elemonier <emily.lemonier@gmail.com>
Date:   Fri Dec 13 21:36:54 2013 -0500

    git semcheck version of ast, parser, sast, scanner

commit e046512af9e6d589c4b6f292ff2244d8f1f129f6
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Fri Dec 13 20:56:43 2013 -0500

    TYPECHECKING IS DONE and the examples all work; this corrects the tests. and
        incr

commit ea0dfce1406a4878f628c70da9bd2dbba560e56e
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Fri Dec 13 20:40:20 2013 -0500

    TYPECHECKING IS ALIVE

commit 1a8d97ed359e43740489f8b40caa0d2e8cf1893b
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Fri Dec 13 20:20:52 2013 -0500

    typecheck functions\!\!\!\! booyah

commit 6d64ba77cbd91daaca2b58aa7988c47769f83e40
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Fri Dec 13 19:37:43 2013 -0500

    tc through modifiers ; now WITH PRIMITIVE NONDET

commit 289f06b5c9844ac5244aba0f6a3f6018a406d1ef
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Fri Dec 13 19:37:20 2013 -0500

    removes arrow binop

commit a22b6817cf6660df8019dc7e2eccf0a52f7ccf7c
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Fri Dec 13 18:54:54 2013 -0500

    tc through track CR
```

```
commit fd722b0c32ddae84a226782ddafe84ea921fdd02
Author: elemonier <emily.lemonier@gmail.com>
Date:   Fri Dec 13 18:54:19 2013 -0500

    example of optionl name.

commit 0e21364a53a1651ab07032ea88b312979db06bb0
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Fri Dec 13 18:35:58 2013 -0500

    beginnings of type_expr

commit 7e41c2fe666129e6ed85398eb8c3fdcd63c6f57d
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Fri Dec 13 18:17:16 2013 -0500

    type statement working with scope checking/env double decl in note.dj

commit d2e076c88904fed357b004b64d57d576e58141c4
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Fri Dec 13 14:45:36 2013 -0500

    type stmt mostly there. errors maybe?

commit f293d3139b93a7299cbf3eb1128811d4b4646e44
Author: elemonier <emily.lemonier@gmail.com>
Date:   Fri Dec 13 12:22:18 2013 -0500

    functioning optional params for chord. Needs work on pretty printing.

commit 6e8a281b302be733e4c1a21dbaa7f01222339c04
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Thu Dec 12 23:05:30 2013 -0500

    type/scope infra

commit 97cdeb9a4b3f2176826b9027761d5607d1090989
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Thu Dec 12 22:59:55 2013 -0500

    environment built through functions, now onto statements

commit 57c88cdba3f366ec6fb8a3f6516e519f86ef4547
Author: elemonier <emily.lemonier@gmail.com>
Date:   Thu Dec 12 21:46:51 2013 -0500

    deleted unecessary clean.

commit 7b9869d5f5f68a4dcbfbf3fc25fc6b7e581172f3
Author: elemonier <emily.lemonier@gmail.com>
Date:   Thu Dec 12 21:44:53 2013 -0500

    Adds java gen functionality -j.

commit d303c0ac8269784759011a70f3185ac6c4e0858c
Author: elemonier <emily.lemonier@gmail.com>
Date:   Thu Dec 12 21:44:38 2013 -0500
```

```
    Adds optional param to compile. needs testing.

commit 098f9ebd3ca5cd3eb17991afc8e2be6cf093ab5e
Author: elemonier <emily.lemonier@gmail.com>
Date:   Thu Dec 12 21:44:04 2013 -0500

    Adds compile to makefile.

commit 3e887615a2bf7225d6b767f046bfcee3737ca983
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Thu Dec 12 21:35:00 2013 -0500

    globals n junk. vdecl has NO SEMI

commit 795905a67616b0d071a044947fab65fb84e04c5f
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Thu Dec 12 21:10:34 2013 -0500

    adds global test

commit 18b43710a8b7710dbb29264ccdca1e666c6a4699
Author: hilagutfreund <hila.gut@gmail.com>
Date:   Thu Dec 12 19:25:37 2013 -0500

    fixing SAST type problems and errors but still has warnings

commit fddfc3e57fbef05ba6b76ed90a240c88a6cb2746
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Thu Dec 12 19:16:06 2013 -0500

    case matching fixes

commit 5e9490d55942b286b6b23147c3d416d71e36ca8c
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Thu Dec 12 19:08:50 2013 -0500

    ALL THE TESTS WORK. BOOYAH

commit 44f4e2717377e760c837c5653e79f5dda803a2c0
Author: hilagutfreund <hila.gut@gmail.com>
Date:   Thu Dec 12 18:44:05 2013 -0500

    slight changes to javagen and error fixing

commit 5a48a245654d99157997f7493747d4e3d0c04cf3
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Thu Dec 12 18:43:58 2013 -0500

    eradicates bend

commit fdbd7e11ea8113cd3c4f0cf21962c165e9c660e7
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Thu Dec 12 18:43:44 2013 -0500

    recomments wdjc java
```

```
commit 949748e2951232860f1cabd23142fa497b9f9a49
Author: elemonier <emily.lemonier@gmail.com>
Date:   Thu Dec 12 18:19:43 2013 -0500

    fixed small errors. continue java gen.

commit 99951042e01cec72c9d3a6ccc1686f2f9051ae72
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Thu Dec 12 18:15:01 2013 -0500

    uncomments java wdjc

commit b50a1aada4164d3f3876ff516992453140f37a2c
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Thu Dec 12 18:08:41 2013 -0500

    scanner pitch bend

commit af037cff3f208d41a8f6744b1e12e7e76fb35877
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Thu Dec 12 18:01:32 2013 -0500

    merges in semcheck_mod

commit 0074d32520d7b4d4163b58840cdfedc81704014d
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Thu Dec 12 17:59:13 2013 -0500

    parser stuff and tests

commit 122144785e5379c1438ac540fa6120aa0a36801c
Author: elemonier <emily.lemonier@gmail.com>
Date:   Thu Dec 12 15:55:36 2013 -0500

    Thats how its done bitches. It compiles.

commit 23be05f94ef43d21cf805b7ca47f3f4a864c1d90
Author: elemonier <emily.lemonier@gmail.com>
Date:   Thu Dec 12 15:45:56 2013 -0500

    fixed program return error from semcheck. now semcheck returns a program.

commit 2ec490e2608b3eafc8ebd67de542c1212ec67bc5
Author: elemonier <emily.lemonier@gmail.com>
Date:   Thu Dec 12 12:04:56 2013 -0500

    tries to fix error.

commit b853fd184043e44fe208d193394e53f5a79aed54
Author: elemonier <emily.lemonier@gmail.com>
Date:   Thu Dec 12 02:55:08 2013 -0500

    Added shit.

commit 50a6d69a46bc04ed6f1084904565059ce6a912b0
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Thu Dec 12 00:17:56 2013 -0500
```

```
    global/globals

commit 9886abf5b831d4a9dc512ddeb7d8081ee5b93204
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Wed Dec 11 23:38:24 2013 -0500

    whys it an Sast.func_decl list???

commit c8d182e0ea0b10c6a635c4d66bea9dd005762f68
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Wed Dec 11 22:32:28 2013 -0500

    more errors

commit 6f57ebf3fd8ed06158aa2e0741999042c698811a
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Wed Dec 11 21:42:33 2013 -0500

    stuff

commit a963f744951133e069d86435329da17db2349df1
Author: hilagutfreund <hila.gut@gmail.com>
Date:   Wed Dec 11 21:24:28 2013 -0500

    final go through of doc to change to sast format

commit 9075a8b344dc56da7f18f2e7a849c04a60f9c12d
Author: hilagutfreund <hila.gut@gmail.com>
Date:   Wed Dec 11 21:22:10 2013 -0500

    more ast to sast

commit 658a26c256fce1d921079cc2585e5f606e757881
Author: hilagutfreund <hila.gut@gmail.com>
Date:   Wed Dec 11 21:14:35 2013 -0500

    more changes from ast to sast

commit 7b7dd53480e7600c9d05e7bfeb6cb80956d7f27a
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Wed Dec 11 21:12:39 2013 -0500

    rec and

commit 45671595d2ac499fee5b033d287812bfde94d080
Author: hilagutfreund <hila.gut@gmail.com>
Date:   Wed Dec 11 21:09:10 2013 -0500

    changes from ast to sast

commit 70497afd38e7b7176ec435c94a6727e393f3fe4f
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Wed Dec 11 21:00:19 2013 -0500

    fixinf syntax errors
```

```
commit 2e150087ac47e3094bb381417e5a3861313df540
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Wed Dec 11 20:34:47 2013 -0500

    pulls in semcheck sast

commit f4ad78e616ed87feda2c65ad5ff58fb35adde23f
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Wed Dec 11 20:33:26 2013 -0500

    stuff

commit 7249a42b76bc364f59d772eb345c432515298f26
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Wed Dec 11 19:30:53 2013 -0500

    CHANGES

commit 07b93b80062e0c3115d43b68d9faa113d9976c93
Author: hilagutfreund <hila.gut@gmail.com>
Date:   Wed Dec 11 18:52:55 2013 -0500

    minor changes to printing java stuff

commit 105fab751af746e88c74bd57038308495905862f
Author: Thomas <tee2103@columbia.edu>
Date:   Wed Dec 11 16:37:12 2013 -0500

    update semcheck

commit 33d0ffd17836b5cb32e89b3688c4a0bfc248bb04
Author: Thomas <tee2103@columbia.edu>
Date:   Wed Dec 11 16:24:53 2013 -0500

    updates semcheck and README TODO section

commit e50d72562c208a4f702ba4eff2e4cfd7675a1293
Author: Thomas <tee2103@columbia.edu>
Date:   Wed Dec 11 13:53:15 2013 -0500

    begins compiling and debugging semcheck

commit 4d19bc599d82960b8c9828e923d8e5f202440602
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Mon Dec 9 11:31:58 2013 -0500

    Update CONTRIBUTORS.geojson

commit ef88dc0ed8126a8e11ebe863498e88df28f57632
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Mon Dec 9 11:29:45 2013 -0500

    fixes geo

commit 4b77ba966413448826ebe12bf97c237e37761de0
Merge: 19fd323 6d45f68
Author: William Falk-Wallace <wfalkwallace@gmail.com>
```

```
Date:   Mon Dec 9 11:24:01 2013 -0500

    merges

commit 19fd323ac59943dc608e2fcce9711487abf0e8a4
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Mon Dec 9 11:23:45 2013 -0500

    expr work

commit 6d45f688fdd4a639b42cb0f1b9b2f6b65d22572b
Author: elemonier <emily.lemonier@gmail.com>
Date:   Sun Dec 8 21:44:33 2013 -0500

    Modifies functions to add environment with statements.

commit 2b2c7753507caee7021ba1b1ce6826cf2914b64d
Author: elemonier <emily.lemonier@gmail.com>
Date:   Sun Dec 8 21:40:34 2013 -0500

    Adds vinit check for locals.

commit 9cdb4ee9f73557ce51d36bc082bf091a34ed55ef
Author: elemonier <emily.lemonier@gmail.com>
Date:   Sun Dec 8 21:26:01 2013 -0500

    fix merge.

commit f05a24efe598f263592fda54fd5c606ee91d2102
Merge: 9e6c508 61bbb4a
Author: elemonier <emily.lemonier@gmail.com>
Date:   Sun Dec 8 21:24:38 2013 -0500

    merge conflict.

commit 9e6c508869711947c8d8108b96b5678fbedc65a5
Author: elemonier <emily.lemonier@gmail.com>
Date:   Sun Dec 8 21:23:45 2013 -0500

    semcheck change.

commit 61bbb4a61c9c7a99b38f84b63f1107a1a178bdc0
Author: Thomas <tee2103@columbia.edu>
Date:   Sun Dec 8 21:22:40 2013 -0500

    adds print checking for sast

commit 28df220422c3901b2fbb9c12b4227529bd6e4b48
Author: hilagutfreund <hila.gut@gmail.com>
Date:   Sun Dec 8 12:36:48 2013 -0500

    minor changes to basic printing in the beginning -- still working on trem/vib

commit 08a6ac2b76727ad4744472baa62b24a35729a326
Author: Thomas <tee2103@columbia.edu>
Date:   Sat Dec 7 15:25:01 2013 -0500
```

```
    adds draft of check_vinit_type function

commit 79c3b272a884f65ef9f961d809af4b41d926c0ee
Author: Thomas <tee2103@columbia.edu>
Date:   Sat Dec 7 14:48:28 2013 -0500

    begins adding test printing for sast

commit e273d51430f578f57917fde8b95138c992653d0d
Merge: 4f7cb68 50f8a1b
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Sat Dec 7 14:41:56 2013 -0500

    merge

commit 4f7cb68a0c4b6c696c4bd8137c6ef984591788c6
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Sat Dec 7 14:41:43 2013 -0500

    expr fleshing

commit 50f8a1bbf61b01403c223b5b6e4a8ffa6c84be7c
Author: Thomas <tee2103@columbia.edu>
Date:   Sat Dec 7 14:34:03 2013 -0500

    adds vinit and vdecl to sast. adding vdecl and vinit to stmt checker

commit 55e3e772f080327fccb172d036e032a15804bffe
Merge: e6bfe13 72dda25
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Sat Dec 7 14:19:30 2013 -0500

    merges in parser

commit e6bfe1339a1d9b7775249d8488a9abb7063859be
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Sat Dec 7 13:06:38 2013 -0500

    adds sc expr accessor

commit 0334f995a78ad2d092d200e6db3dc23447deb241
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Sat Dec 7 12:55:18 2013 -0500

    adds isnote(etc) methods for checki g type and keeping name inline

commit 461780331407ad565758538a4694efb1afb110ba
Author: Thomas <tee2103@columbia.edu>
Date:   Sat Dec 7 12:28:06 2013 -0500

    modifies note_cr. all in comments.

commit 24b7460071b46ccc00b4f6c0aff05aa08188116b
Author: Thomas <tee2103@columbia.edu>
Date:   Thu Dec 5 18:12:44 2013 -0500

    modifies match binop op section for ser and par.
```

```
commit ab483965c85e0d4c57b6e6e35b45980c084b425f
Author: hilagutfreund <hila.gut@gmail.com>
Date:   Wed Dec 4 09:28:43 2013 -0500

    added changes to track probably wrong

commit 8867080a31eab2edb7dadc33444496eaadb54d9d
Author: hilagutfreund <hila.gut@gmail.com>
Date:   Wed Dec 4 09:11:53 2013 -0500

    slightly changed note creation

commit 543c2f29d96459cbbc72a229aec8a7997ee027e3
Author: hilagutfreund <hila.gut@gmail.com>
Date:   Wed Dec 4 08:59:42 2013 -0500

    added note creation, rest creation, part of chord creation

commit 1a352011daad2084f71c9c5bb1dd251b307bf6b7
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Wed Dec 4 00:26:18 2013 -0500

    removes weird vinit dtype line

commit b2512401810a229f09ef03d8e06e54304098698c
Author: hilagutfreund <hila.gut@gmail.com>
Date:   Tue Dec 3 23:12:39 2013 -0500

    compile parser test.

commit 7ca88a00732b7158ed7f5c3066baf0f2e8430369
Author: Thomas <tee2103@columbia.edu>
Date:   Tue Dec 3 22:18:48 2013 -0500

    finishes stmt checking function. Need clarification on if, for, while.

commit 21d12427131daf44aa06aeb80840199c8f71e020
Author: Thomas <tee2103@columbia.edu>
Date:   Tue Dec 3 17:37:06 2013 -0500

    works on stmt section, adds comments for pulling locals(vdecl, vinit)

commit bcfe61de1ea53145028803b9e193301cf4be1c30
Author: hilagutfreund <hila.gut@gmail.com>
Date:   Tue Dec 3 16:45:31 2013 -0500

    copied over updated ast to compile to work on it. other ast not commented out
       yet..

commit 72dda2515621cd204b8c9974ac45ebc35bc86560
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Tue Dec 3 11:16:32 2013 -0500

    updatesnstufs

commit f8de87519677f38be6cf9affa281f43b299bd223
```

```
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Tue Dec 3 11:13:34 2013 -0500

    adds compile template

commit fc3cbd4be4fe8e31e8485f838920ea1f3703dc23
Merge: 7570595 d87cade
Author: elemonier <emily.lemonier@gmail.com>
Date:   Tue Dec 3 11:11:06 2013 -0500

    fucking geo.

commit 7570595364cd7ebdfa72470290baae963574525e
Merge: 5713b8a f93a0a0
Author: elemonier <emily.lemonier@gmail.com>
Date:   Tue Dec 3 11:10:16 2013 -0500

    confused semcheck add.

commit 5713b8a9f2bba8d14083cecba2efd9810afd7ad5
Author: elemonier <emily.lemonier@gmail.com>
Date:   Tue Dec 3 11:09:00 2013 -0500

    Made notes, etc. exprs. Also - notes only have 3 elements for constructrs.

commit d87cadee1d1d8338ee6012f306af4fa2322cf425
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Tue Dec 3 10:55:41 2013 -0500

    merges in compile's wdjc -j stuff

commit 30c00cf439a00d4b97936af731ff1bdb2dd662d0
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Tue Dec 3 10:52:50 2013 -0500

    updates semcheck entry name

commit f93a0a0dcf8041915dd22533c7029a404c28c16c
Author: elemonier <emily.lemonier@gmail.com>
Date:   Tue Dec 3 10:33:51 2013 -0500

    small modifications to semcheck.

commit ecc4018ef8258f6051106803e7b8787ec5ff19dd
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Tue Dec 3 10:25:42 2013 -0500

    fixes mergerz junk

commit 25f53b96042bc17b7bf07f53b9a114c4e02da0da
Merge: b07df11 b90d94a
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Tue Dec 3 10:24:26 2013 -0500

    mergerz

commit b07df115e2fb8d7b9676b6aba252b76400df6268
```

```
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Tue Dec 3 10:23:59 2013 -0500

    updates wdjc to run java off ast

commit 6c9b18ffeb66ac996016ca1babe1f42f08583855
Author: hilagutfreund <hila.gut@gmail.com>
Date:   Tue Dec 3 02:50:46 2013 -0500

    added java syntax for track

commit 68b508ed48b992b96c9c1f50a13fc71969eebd8e
Author: hilagutfreund <hila.gut@gmail.com>
Date:   Tue Dec 3 02:35:02 2013 -0500

    unsure about how to access ID properly, but update to chord_cr, rest, and
       Note_CR

commit 0d7f9f4ec542890ae25549828bcea224552ea981
Author: hilagutfreund <hila.gut@gmail.com>
Date:   Tue Dec 3 00:43:48 2013 -0500

    literal, id, note creation, rest, accessor, and assing to java synatx on compile

commit 39505f94b304928a32d29f6f3404451243fda5e1
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Mon Dec 2 23:36:05 2013 -0500

    updates wdjc compiler options update

commit b90d94a4cb62bf1da306c17f64e7db8565741807
Merge: de5bfe9 40dd7e6
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Mon Dec 2 23:34:45 2013 -0500

    merge back

commit de5bfe9645893f7609b8c32724089cfdb6da7450
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Mon Dec 2 23:33:51 2013 -0500

    updates wdjc to add java printing and compile templates

commit bb9e3b12cbc1ba270640d4a922100923ac8cc28a
Author: elemonier <emily.lemonier@gmail.com>
Date:   Mon Dec 2 18:22:39 2013 -0500

    serial/parallel info.

commit 40dd7e601182512525289a4d154a1610fb30bb6e
Merge: 6446d7c 9a350e3
Author: elemonier <emily.lemonier@gmail.com>
Date:   Mon Dec 2 18:13:39 2013 -0500

    I WILL FUCKING KILL GEO.

commit 6446d7c9d2e8b1ef670632642b3884d154d25b97
```

```
Author: elemonier <emily.lemonier@gmail.com>
Date:   Mon Dec 2 18:13:21 2013 -0500

    small parser modifications.

commit a62c7149eb104871b26a8a07e80b23232bd42133
Author: Thomas <tee2103@columbia.edu>
Date:   Mon Dec 2 17:57:08 2013 -0500

    updates comments at semcheck

commit 9a350e3bd5f65b5e092ba4941b6caf918e222c07
Merge: d411504 f2c45b3
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Mon Dec 2 17:52:17 2013 -0500

    merging

commit d411504bad848092267e97c08f1002125bdb549c
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Mon Dec 2 17:47:20 2013 -0500

    a little bit of cleanup

commit f2c45b36cbe1a421903a903d041354dc499ed755
Author: elemonier <emily.lemonier@gmail.com>
Date:   Mon Dec 2 17:43:41 2013 -0500

    Modifies rest.

commit 5ce84632eb6e2b167ca3790826cb269a4c671288
Author: hilagutfreund <hila.gut@gmail.com>
Date:   Mon Dec 2 11:59:20 2013 -0500

    added fprintf for jvariables - might be wrong place, not sure yet

commit c72c3fa7349f8ad71fc65bc38a93b4f33fd5c557
Author: hilagutfreund <hila.gut@gmail.com>
Date:   Mon Dec 2 11:55:19 2013 -0500

    added breakdown for variable dec loop and type+id creation for declaration

commit 2c85d385d484ee1507197974c7ae52563fe0e70b
Author: hilagutfreund <hila.gut@gmail.com>
Date:   Mon Dec 2 10:58:53 2013 -0500

    added breakdown for variable dec

commit 1e8fd7966fa87e2a113a3728ccc0d8982356f17c
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Mon Dec 2 01:36:41 2013 -0500

    fixes pretty printer for 2 vars

commit e0054cabca59f4c32ba88726ef906dabb9962401
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Mon Dec 2 01:21:13 2013 -0500
```

```
    initialization and assignment to accessor

commit 3eeb8ff13b3adb56e8a94fcf83e6094beeda1f13
Merge: b2a1ace afea6fc
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Sun Dec 1 20:44:29 2013 -0800

    Merge pull request #3 from WHEF-PLT/parser_track

    adds updated rest and track to the parser. more to be done in semcheck

commit afea6fc52bbdcfcfa09353f3744ea4922b06cf84
Author: elemonier <emily.lemonier@gmail.com>
Date:   Sun Dec 1 18:52:12 2013 -0800

    Added rest and track functionality.

commit 7a77e5c9cb5492bf09ce201a322100a7e0589f3c
Author: elemonier <emily.lemonier@gmail.com>
Date:   Sun Dec 1 18:51:41 2013 -0800

    Added rest create functionality. where rest = RPAREN LITERAL LPAREN

commit 9a35fe80051e15c497e0aefa8bc7b8f4e173e389
Author: elemonier <emily.lemonier@gmail.com>
Date:   Sun Dec 1 17:56:20 2013 -0800

    adds comments for stuff I have to do.

commit 36c2beeab539c28dac20425bb0c74fc0eb299b16
Merge: 4314f54 5cd22d2
Author: hilagutfreund <hila.gut@gmail.com>
Date:   Sun Dec 1 16:25:11 2013 -0500

    deleted last 10 lines of compile.ml since it is not our code

commit 4314f545faef52a725c85c46a2f12f5cb6d7e060
Author: hilagutfreund <hila.gut@gmail.com>
Date:   Sun Dec 1 16:23:41 2013 -0500

    deleted last 10 lines of compile.ml since it is not our code

commit 5cd22d250d2a166f7f8eb0adb48c0e78ae5d97c5
Merge: 861e399 38af6ff
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Sun Dec 1 14:13:27 2013 -0500

    merges in master changes

commit b2a1ace48536467e474564920ed1e56947bd219d
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Sun Dec 1 13:56:55 2013 -0500

    removes locals from fdecl and fdecl printer

commit 2d6e904ec348a264acf52b0e50fdb3b66cfad45a
```

```
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Sun Dec 1 13:56:38 2013 -0500

    removes locals and vdecl list from funcdecl

commit 623cb65e6a050e9fbd84ec5f1e286794ecb35780
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Sun Dec 1 13:48:08 2013 -0500

    uncomments string of vdecl stuff

commit c3182173f9b5a7686346e75aee47a14e3c609ebe
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Sun Dec 1 13:35:58 2013 -0500

    NOERRORS formal is NOW a VDECL. WAHOO

commit 409c3d0eee65194202ecaa92e00112e746ba9ced
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Sun Dec 1 13:34:05 2013 -0500

    uncomments vdecls stmt constructor in the ast

commit 492197955d3656e1520b0a40ccde73236991d15a
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Sun Dec 1 13:26:14 2013 -0500

    formals and vdecls together at last....

commit 9f80472deb9cd7c4f93db1196a921d1c6f1d3fdd
Merge: f369a4a 38af6ff
Author: elemonier <emily.lemonier@gmail.com>
Date:   Sat Nov 30 21:47:05 2013 -0800

    merged with origin master.

commit f369a4a40345ebf53a2a21f4a37bb36a646fecd5
Author: elemonier <emily.lemonier@gmail.com>
Date:   Sat Nov 30 21:40:42 2013 -0800

    modified minor Makefile

commit dcef12651a9e066531cb14a7cccafc94698677f7
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Sun Dec 1 00:39:59 2013 -0500

    branching huh?

commit 38af6ff96fd536ded9a80b833ac40f12d5f5dff9
Merge: 4ce4d00 6cde6b0
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Sat Nov 30 21:27:28 2013 -0800

    Merge pull request #2 from WHET-PLT/parser

    Parser
```

```
commit 6cde6b0b0ee1d0e1ff9307e7c99ef250cf88a726
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Sat Nov 30 23:55:31 2013 -0500

    fixes test file function return type

commit 7722d04c60c4b9127a9a6e5122ada681b774a517
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Sat Nov 30 23:42:39 2013 -0500

    typo

commit 3ee2db0b87fb7af1f6265c8f6194b62cc6d8a7b1
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Sat Nov 30 23:41:41 2013 -0500

    removes tar junk and types from make

commit 1898bca0ed9ba56ade177893bac3150d662b930c
Merge: 9457ee9 4ce4d00
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Sat Nov 30 23:39:41 2013 -0500

    merges in assign makefile

commit 9457ee9bd4c44238223c7d3bd9923227116f182d
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Sat Nov 30 20:26:24 2013 -0500

    empties microc compile which was throwing errors

commit f598bc86e639d08724cb21d8b7fe214cdad51fa2
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Sat Nov 30 20:26:03 2013 -0500

    removes make execute rules

commit b00855e5d2602fc1dcf98b3d61d228fd87c778b6
Merge: e7df74c ab3b672
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Sat Nov 30 20:13:13 2013 -0500

    merges in assignment branch changes after turkey

commit e7df74c9f1b04cd7842e2ed344125afeac832404
Author: elemonier <emily.lemonier@gmail.com>
Date:   Sat Nov 30 13:53:01 2013 -0800

    Changed makefile to actually print. Now shows errors.

commit ab3b67270966fae7de07c9fef811697396b77ea2
Author: elemonier <emily.lemonier@gmail.com>
Date:   Sat Nov 30 13:34:40 2013 -0800

    Adds some expression checking functionality.

commit 0c941a869f1c436c702f00d83c62263bb913e77b
```

```
Author: elemonier <emily.lemonier@gmail.com>
Date:   Sat Nov 30 11:07:34 2013 -0800

    Adds formal semantic checking.

commit 3edd409b3afad88e68c4ccc95970720bf7c3eeec
Merge: 70bf705 6742d5f
Author: elemonier <emily.lemonier@gmail.com>
Date:   Sat Nov 30 10:59:24 2013 -0800

    geo

commit 70bf70539d800b4598744454946a161b4e2e0ee7
Author: elemonier <emily.lemonier@gmail.com>
Date:   Sat Nov 30 10:58:55 2013 -0800

    Completed semantic check for function.

commit 6742d5f7aa5235cb47e1fd7d551238c136b6707f
Author: Thomas <tee2103@columbia.edu>
Date:   Sat Nov 30 11:24:02 2013 -0500

    adds checking for binop and modifiers. begins stmt checking

commit 4ce4d006cb33e0c4c7c96c3a9728ec47293942fa
Author: hilagutfreund <hila.gut@gmail.com>
Date:   Sat Nov 30 05:55:28 2013 -0500

    added phrase to arpeggio1.java to see if I can push from Israel

commit 1a9ae23c29fdef09072e200f075f7c5c88cc3162
Merge: 5a2eb8f b3957cf
Author: hilagutfreund <hila.gut@gmail.com>
Date:   Sat Nov 30 04:29:45 2013 -0500

    merge conflict? (not sure why but here it is..) Merge branch 'master' of
        https://github.com/WHET-PLT/wdjc

commit 7062090ffcc2c6c5d77757249562db4905e22fe9
Author: elemonier <emily.lemonier@gmail.com>
Date:   Fri Nov 29 19:22:02 2013 -0800

    Some changes.

commit 861e3999f965f3823a7ed53b038949ca57a304c4
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Fri Nov 29 15:33:13 2013 -0500

    minors

commit a9da85d7fb6810fc538ff9e42753aa894d5961ea
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Fri Nov 29 15:15:16 2013 -0500

    formals/params to string, rec'ly plus relies on expression redux

commit ca455a147677bca5cb9be093cdf8435ac52bddb0
```

```
Merge: 3647f4c 1a3bf3c
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Fri Nov 29 14:44:42 2013 -0500

    Merge branch 'compile' of https://github.com/WHET-PLT/wdjc into compile

commit 3647f4c6df5665b9d4a9cf5ffdb722053afcdb6a
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Fri Nov 29 14:42:16 2013 -0500

    smore templates; stmt and expr recursions

commit 898f3fe03233d8ee0bc8548460f9830e3c501175
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Fri Nov 29 12:23:33 2013 -0500

    adds type print

commit 1a3bf3c1c27f60d4e1591550fcdddae30ea807b7
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Fri Nov 29 11:19:22 2013 -0500

    BETTER

commit 09ae77a9b9268bffe770dfe16b94c998816e1f2a
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Fri Nov 29 11:17:45 2013 -0500

    MAPS PRETTY!

commit 5a99c61917cc62cca4e90f3c01093660fbd764c0
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Fri Nov 29 11:08:59 2013 -0500

    silly sublime text, timestamps are for changes

commit f235062604f66a3f3517292847fa4c4cf55fed3e
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Fri Nov 29 10:58:30 2013 -0500

    updates template from sast

commit aa2c021f1748c57eae22250ec2c910b7eec168f7
Merge: 51a9bde b9901d7
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Fri Nov 29 10:42:10 2013 -0500

    merge assign

commit 51a9bde42922bcd8b969ca7860e3134d63b022cf
Merge: 396e797 8e2803f
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Thu Nov 28 13:43:37 2013 -0500

    merges in assignment/compile branch

commit b9901d7954689a04fa9f61f69331b7947c4e2de8
```

```
Author: elemonier <emily.lemonier@gmail.com>
Date:   Wed Nov 27 17:13:50 2013 -0800

    modifies sc_functions.

commit 4a13c2e9fa9ba8f3d3fadb9c3f0840bb10645844
Author: elemonier <emily.lemonier@gmail.com>
Date:   Wed Nov 27 16:52:30 2013 -0800

    modifies global info.

commit 3d9643e7194e8c19ab74ca6629823627077e20e7
Author: elemonier <emily.lemonier@gmail.com>
Date:   Wed Nov 27 16:52:19 2013 -0800

    Adds to to-dos.

commit 0f3546ddeef10998ed8aae731f816cfd02c0de03
Author: elemonier <emily.lemonier@gmail.com>
Date:   Wed Nov 27 13:31:25 2013 -0800

    Minimal modification of semcheck.

commit 8e2803f82582b5e0bc5b5c5611f6d7a514240a87
Merge: c0b8a9e bf789d7
Author: elemonier <emily.lemonier@gmail.com>
Date:   Wed Nov 27 09:09:04 2013 -0800

    Merge branch 'assignment_and_new_compile' of https://github.com/WHET-PLT/wdjc
        into assignment_and_new_compile

commit c0b8a9ec4b9f43cd48a7793dcb54eec07a62872e
Author: elemonier <emily.lemonier@gmail.com>
Date:   Wed Nov 27 12:03:10 2013 -0500

    Added function and program checking.

commit bf789d736c7fdedc32d2f0aa74c9d9a81a55c260
Author: Thomas <tee2103@columbia.edu>
Date:   Tue Nov 26 16:51:42 2013 -0500

    adds return to sast

commit 396e79711c9bd335d7deaa04e60f08d52e2da3ed
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Mon Nov 25 03:19:27 2013 -0500

    compile.ml function to string filewriting

commit 0a18fd22499bdfe5564ad46794ace4dbf5443e87
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Mon Nov 25 03:03:54 2013 -0500

    compile notes

commit f00012684b2cadde102c5aa416a62f2b7bf9fc50
Author: William Falk-Wallace <wfalkwallace@gmail.com>
```

```
Date:   Mon Nov 25 02:40:57 2013 -0500

    nevermind on the jast, probably too similar to the sast anyway?

commit 5e5e02533f7e2b6f762779be8f410dd6f86bd082
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Mon Nov 25 02:20:02 2013 -0500

    adds jast java tree file

commit 6130cd5f6ec6ac5a2dc4ea73c88b4ac1573fa2f0
Author: Thomas <tee2103@columbia.edu>
Date:   Sun Nov 24 23:12:16 2013 -0500

    adds formal checking

commit 307a76e43e7926f4d40a29e0a1868f163fbceded
Author: elemonier <emily.lemonier@gmail.com>
Date:   Sun Nov 24 21:03:13 2013 -0500

    modified part of types section.

commit c73285cb83cd4a4b6c98435b2eb866cbee041337
Author: elemonier <emily.lemonier@gmail.com>
Date:   Sun Nov 24 19:49:54 2013 -0500

    Separated Tom, Emily into semcheck ops.

commit 7b4f3e8533900ca2037c74cadef3967c5107eeca
Author: elemonier <emily.lemonier@gmail.com>
Date:   Sun Nov 24 19:40:48 2013 -0500

    Commented and clarified semcheck.

commit 54cb63fceea0a67b3cfb554f7ee122f0909cd252
Author: Thomas <tee2103@columbia.edu>
Date:   Sun Nov 24 18:44:11 2013 -0500

    combines types.ml into semcheck.ml. symbol tableand type checking done here

commit b3957cf04ab8d42382914b93acee7328322c97c0
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Sun Nov 24 18:17:22 2013 -0500

    rms mli

commit bc715ec4586337f74d4dfa457caff8f6f3ec0560
Author: Thomas <tee2103@columbia.edu>
Date:   Sat Nov 23 15:43:18 2013 -0500

    modifies semcheck

commit 0e2d675bd3e407d8e9448c9568919958c77c1814
Author: Thomas <tee2103@columbia.edu>
Date:   Sat Nov 23 15:23:25 2013 -0500

    adds comments to types
```

```
commit 0c57adff1bc72f79ca5fee52bdd2c1eb52999e91
Merge: 08e9c56 05fc4c9
Author: elemonier <emily.lemonier@gmail.com>
Date:   Sat Nov 23 15:20:12 2013 -0500

    geo

commit 08e9c568772a02dd4f67fdf2b1af3098c90a71e4
Author: elemonier <emily.lemonier@gmail.com>
Date:   Sat Nov 23 15:19:44 2013 -0500

    Adds return types to functions.

commit 05fc4c90f9077e5ea125c6d8c38980a710074c12
Author: Thomas <tee2103@columbia.edu>
Date:   Sat Nov 23 15:10:57 2013 -0500

    modifies makefile for sast,semcheck,types

commit ae748405747ebe749a71fe9af764a9cb73330cf1
Author: Thomas <tee2103@columbia.edu>
Date:   Sat Nov 23 14:55:59 2013 -0500

    modifies sast, semcheck, types

commit 93d924feced3678c98460fdb782f37867d4cc58a
Author: elemonier <emily.lemonier@gmail.com>
Date:   Sat Nov 23 13:32:26 2013 -0500

    Adding assignment functionality to ast.ml and parser.mly

commit 5a2eb8fb517ae92e2f551b85d1cee23d8dc43feb
Author: Thomas <tee2103@columbia.edu>
Date:   Wed Nov 20 22:09:43 2013 -0500

    builds out beginning symbol table and types table. Need review of types in ast

commit 33050995d88c4e048c5d189953cccd6f07dd95e3
Author: Thomas <tee2103@columbia.edu>
Date:   Wed Nov 20 18:04:22 2013 -0500

    adds type.ml for type checking and open lines

commit 1de46f848f6d9d0f63ce032a4dc05cd5e089d450
Merge: 2331846 485738a
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Tue Nov 19 19:26:30 2013 -0500

    geo and merge

commit 2331846f299b028d02d4cfc073bc1ba67141205b
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Tue Nov 19 19:26:11 2013 -0500

    renames freq to java
```

```
commit 36136831fb5baf0484e7e70977f7d1bc7b62926e
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Tue Nov 19 19:25:43 2013 -0500

    updates make for freq

commit 3a8fb3a6d19e17b71ab5c51e76ae078eb57b72e1
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Tue Nov 19 19:25:10 2013 -0500

    updates to make for freq notes

commit 485738a38ddf00bdcadb99cdd3c3fd00a3d369c8
Author: hilagutfreund <hila.gut@gmail.com>
Date:   Tue Nov 19 19:22:58 2013 -0500

    changed file name for notes freq again

commit f28db22fcb30eec54f193a6a10fc41fa1fc93e5c
Author: hilagutfreund <hila.gut@gmail.com>
Date:   Tue Nov 19 19:15:35 2013 -0500

    added freq note creation

commit 006d00038baac4b865254eee254483fdc540cede
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Tue Nov 19 19:00:05 2013 -0500

    minor updates

commit ce559a133934dd3b46b755853b525000fa9faf6b
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Tue Nov 19 18:55:39 2013 -0500

    renames ast's to mli

commit df66b2ef65eeb3f000429479a985d04de9e4ed88
Author: Thomas <tee2103@columbia.edu>
Date:   Tue Nov 19 18:42:32 2013 -0500

    modifies sast.

commit 5602700d270bd2bfde0a58389dcbab200a3db721
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Tue Nov 19 18:20:06 2013 -0500

    sast template copy

commit ca95920dfd11763e949e600c8811bdc63f7b5fa3
Merge: 3733b52 c6612e7
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Tue Nov 19 18:19:51 2013 -0500

    merge

commit c6612e7ca3df32ca905fba6e7f7cc002ffddf073
Author: Thomas <tee2103@columbia.edu>
```

```
Date:   Tue Nov 19 18:12:56 2013 -0500

    adds modifier.dj test. vib and trem work

commit 3733b521d88cf151eeff38820552a0e171b52156
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Tue Nov 19 18:10:49 2013 -0500

    adds sast file

commit 20a506e2c65f7ed21f24a6d9524a411408635c52
Merge: 84ad022 65d78bb
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Tue Nov 19 18:10:31 2013 -0500

    fixes merge

commit 84ad0227bdd43970da9a2ebcc116f43437004ff0
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Tue Nov 19 18:09:59 2013 -0500

    updates m to modif for clarity

commit 65d78bb44bb788354cb936973630cc35bf54bab1
Author: elemonier <emily.lemonier@gmail.com>
Date:   Tue Nov 19 18:00:09 2013 -0500

    Fixed modifier error.

commit 94bf379564ffcf86b8cca5f94eb5da38f2ccb7f3
Merge: df613cc ea4d363
Author: elemonier <emily.lemonier@gmail.com>
Date:   Tue Nov 19 17:58:10 2013 -0500

    geo

commit df613cc6cb53ea39dedd0e2daddb82e6263ce41e
Author: elemonier <emily.lemonier@gmail.com>
Date:   Tue Nov 19 17:57:48 2013 -0500

    Parsing.

commit ea4d363e1edec53070c6350c010a42d112e2c977
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Tue Nov 19 14:55:40 2013 -0500

    moves emily's todo.txt to the readme and answers them

commit 81f2583e52359640ea7cc106a6bb64a327d9a841
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Tue Nov 19 14:45:27 2013 -0500

    makes todo into list

commit d4d097d77dff9cfb662589fae858f5d7391dbc80
Author: Thomas <tee2103@columbia.edu>
Date:   Mon Nov 18 18:05:18 2013 -0500
```

```
    fixes incr/decr. now works with test.

commit 0f310a1393b6e8e3abe793321692a34fb57684cd
Author: Thomas <tee2103@columbia.edu>
Date:   Mon Nov 18 17:49:58 2013 -0500

    modifies incr/decr test

commit ed8a612056e290db5a443ba985de371d7593ab4d
Author: elemonier <emily.lemonier@gmail.com>
Date:   Mon Nov 18 13:06:27 2013 -0500

    merge conflict fix.

commit cc5d6e280cc3cf0fae9c9f56703e5a6abd228a41
Merge: 2358cd9 a72ee92
Author: elemonier <emily.lemonier@gmail.com>
Date:   Mon Nov 18 13:04:45 2013 -0500

    geo

commit 2358cd9862fd668dd42789bb5fe5212251564ff8
Author: elemonier <emily.lemonier@gmail.com>
Date:   Mon Nov 18 13:04:26 2013 -0500

    Adds accessor test for all note attributes.

commit f667d29ccf488368b070ee290109366aced31a54
Author: elemonier <emily.lemonier@gmail.com>
Date:   Mon Nov 18 13:02:33 2013 -0500

    Added accessor (->) functionality.

commit c74b2e2d5c62fdba2d2d62add792c34eaa2e11d1
Author: elemonier <emily.lemonier@gmail.com>
Date:   Mon Nov 18 13:02:08 2013 -0500

    Changed pit to pitch. Sorry guys. I just kept tthinking of an armpit.

commit a72ee92d12be47aa0426828fad35ab9a2d7d9d29
Author: Thomas <tee2103@columbia.edu>
Date:   Mon Nov 18 12:39:04 2013 -0500

    adds increment(++) and decr(--) test.

commit f494ab8e2e9b0dda26023f6d2780683522c0f366
Author: elemonier <emily.lemonier@gmail.com>
Date:   Mon Nov 18 12:08:12 2013 -0500

    Adds functioning while test.

commit a284b1740ae327950db199e73ace688177ad1c1e
Author: elemonier <emily.lemonier@gmail.com>
Date:   Sun Nov 17 21:35:34 2013 -0500

    Adds (nonworking) initialize test.
```

```
commit 9af7e12da2a3b380de86db181b742de754af8918
Author: elemonier <emily.lemonier@gmail.com>
Date:   Sun Nov 17 21:29:38 2013 -0500

    Starting a todo list as we break up work into individual tasks.

commit f3e4dc0d79c14b03f11dd08ee1641946e4e0d1e3
Author: elemonier <emily.lemonier@gmail.com>
Date:   Sun Nov 17 21:29:03 2013 -0500

    Adds comprehensive if test.

commit 5e9ae7af76b870cec3c67a493b095443d155d423
Author: elemonier <emily.lemonier@gmail.com>
Date:   Sun Nov 17 21:15:37 2013 -0500

    Updates if test. Notes limitations.

commit 78528c5c29e41807e6f105f0741666334de3a421
Author: elemonier <emily.lemonier@gmail.com>
Date:   Sun Nov 17 21:11:58 2013 -0500

    if works. Add if test.

commit f7bcae79e898145146ff76e312de63605a69ff9c
Merge: 02426e3 e61799b
Author: elemonier <emily.lemonier@gmail.com>
Date:   Sun Nov 17 21:03:01 2013 -0500

    geo. java fix.

commit 02426e3573352dd5faad9c5b4be833d88b5fd649
Author: elemonier <emily.lemonier@gmail.com>
Date:   Sun Nov 17 21:01:58 2013 -0500

    Cleaning up old comments from ast and parser

commit e61799bdd1ce4564c7d7448e9083f110acfb3485
Author: hilagutfreund <hila.gut@gmail.com>
Date:   Sun Nov 17 18:08:58 2013 -0500

    added some comments

commit f047b5baa5a7695bafbeb7911703a0c86cd568c0
Merge: 1479d92 70bc3a9
Author: elemonier <emily.lemonier@gmail.com>
Date:   Sun Nov 17 18:04:21 2013 -0500

    geo

commit 1479d92eb918d14eaf378af53a14057d0f9f596e
Author: elemonier <emily.lemonier@gmail.com>
Date:   Sun Nov 17 18:03:48 2013 -0500

    Updates notes + chords tests.
```

```
commit ae0b990a77a878782370378d9e7f0633d5f39e6a
Author: elemonier <emily.lemonier@gmail.com>
Date:   Sun Nov 17 18:03:22 2013 -0500

    Adds chord datatype; it is a list of IDs.

commit 70bc3a91210a0635a5e10137272c7f96bd0442c5
Author: hilagutfreund <hila.gut@gmail.com>
Date:   Sun Nov 17 18:01:12 2013 -0500

    added row your boat with explanation of different sections

commit 1ae21011c24670dce12e1a664dfe5389dd442814
Author: Thomas <tee2103@columbia.edu>
Date:   Sun Nov 17 17:57:27 2013 -0500

    cleans up old comments in ast/parser for ease of read

commit f647558b57c7df6a2eb7414a883c55beef1e0d4d
Author: Thomas <tee2103@columbia.edu>
Date:   Sun Nov 17 17:41:38 2013 -0500

    fixes exhauative pattern matching in ast and bytecode

commit 0d79cba24d07eca4ea111aab1a330b4fef7cebe7
Author: hilagutfreund <hila.gut@gmail.com>
Date:   Sun Nov 17 17:38:18 2013 -0500

    added TwoParts.java

commit 90f3b171e3d6de719e4f3f35b962abb53a17459f
Merge: 2168f3b 3569051
Author: elemonier <emily.lemonier@gmail.com>
Date:   Sun Nov 17 16:54:08 2013 -0500

    geo

commit 2168f3b35da34cd4c6b965715817e9dc94d4b557
Author: elemonier <emily.lemonier@gmail.com>
Date:   Sun Nov 17 16:53:53 2013 -0500

    Working chord which takes expressions as args (May want to convert to IDs
       instead of expressions.

commit 35690512f95eee26ac858b726ff1e6da25cc6d05
Author: hilagutfreund <hila.gut@gmail.com>
Date:   Sun Nov 17 16:53:49 2013 -0500

    added chord example

commit 45a547ef83b968aaef2260f8731a58feffd8d407
Author: hilagutfreund <hila.gut@gmail.com>
Date:   Sun Nov 17 16:20:36 2013 -0500

    added example of how to make notes

commit a9007be1c161ce6bb3f05d7cd992af7f83a06d1c
```

```
Author: elemonier <emily.lemonier@gmail.com>
Date:   Sun Nov 17 14:02:52 2013 -0500

    git fuckin up.

commit 45fd819f63985927be05f7d343c0765c078e4e29
Merge: 808a6f4 f63c2d0
Author: elemonier <emily.lemonier@gmail.com>
Date:   Sun Nov 17 13:58:54 2013 -0500

    Merge.

commit 808a6f485a2d7ce99b3051fd7416b529074819d0
Merge: 4a1ddb8 e3a0764
Author: elemonier <emily.lemonier@gmail.com>
Date:   Sun Nov 17 13:57:17 2013 -0500

    Merging.

commit f63c2d035e6e6e0db10dc741e16ce1df8cd70920
Author: elemonier <emily.lemonier@gmail.com>
Date:   Sun Nov 17 13:55:48 2013 -0500

    Adds functioning note test.

commit 7fccd170d5cf8e733f67f218352757e73e31f2e4
Author: elemonier <emily.lemonier@gmail.com>
Date:   Sun Nov 17 13:55:27 2013 -0500

    Added note_cr type; is an expression.

commit dc16f197961d939fdf52069012b02c8c47b3263e
Author: elemonier <emily.lemonier@gmail.com>
Date:   Sun Nov 17 13:54:27 2013 -0500

    Added NOTE_CR to expr constructor, added NOTE_CR pretty print to string_of_expr.

commit e3a07641ace4e3c6eeebd6d744d98388e35f8aeb
Author: Thomas <tee2103@columbia.edu>
Date:   Sun Nov 17 13:20:52 2013 -0500

    creates simple test for 'for'. no parser errors

commit ddc7a2cbd86e5d3478bdbca575d2bcde510fe1a6
Author: Thomas <tee2103@columbia.edu>
Date:   Sun Nov 17 12:40:30 2013 -0500

    fixes for and while ambiguities. while part of the language for now

commit 505bd29c726deec8676763e61083bbe91b3167de
Author: elemonier <emily.lemonier@gmail.com>
Date:   Sat Nov 16 16:22:57 2013 -0500

    Attempts to incorporate note similarly to vdecl.

commit 4a1ddb87a4e47084c43914071dfb60267ccce538
Author: elemonier <emily.lemonier@gmail.com>
```

```
Date:   Sat Nov 16 16:22:13 2013 -0500

    Deleted initialize text.

commit 9d4b3c010a181c8ae221711111ce753e5f85d7db
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Sat Nov 16 16:07:02 2013 -0500

    updates makefile midi dir

commit 1951e2a8c1df4f26437667b1744601518de629ce
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Sat Nov 16 15:55:33 2013 -0500

    java reorg:

commit b26ba488bc2b02c1438679a264c43c5aa8649761
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Sat Nov 16 15:46:43 2013 -0500

    java makefile

commit 723dd06b5cc4867d94e624b454321a0ca2fe2b89
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Sat Nov 16 15:46:25 2013 -0500

    adds midi and java org to gitignore

commit d8ea0cfa5e8da7519380ce545e781a2d886e4340
Author: elemonier <emily.lemonier@gmail.com>
Date:   Sat Nov 16 15:33:46 2013 -0500

    modifies parser.mly; comments out unnecessary vinit.

commit 9a02f89c8f2eb90616166f18dd17f9eab93a29e8
Author: elemonier <emily.lemonier@gmail.com>
Date:   Sat Nov 16 14:59:23 2013 -0500

    Compiling ast + parser.

commit 3a6990d3ea135cd92c4518b361de482bf969aa24
Merge: 469ee5e 7ff1715
Author: elemonier <emily.lemonier@gmail.com>
Date:   Sat Nov 16 14:57:29 2013 -0500

    broken dec + init: EXAMPLE: int pitch = 60;

commit 469ee5e29554ebc41dd85fab251e0fcd6abde83d
Author: elemonier <emily.lemonier@gmail.com>
Date:   Sat Nov 16 14:56:32 2013 -0500

    Broken assignment.

commit 7ff1715b8348763b6063d1413a5be6416eacd7ce
Author: hilagutfreund <hila.gut@gmail.com>
Date:   Sat Nov 16 13:16:47 2013 -0500
```

```
    adding jsound example

commit 2dbad07be3e51406dc599ff6b630d7b874c6a6ac
Merge: 680d0dd 267c751
Author: elemonier <emily.lemonier@gmail.com>
Date:   Sat Nov 16 12:51:07 2013 -0500

    geo.

commit 680d0dd3ad62510c0772aa7159ca60a8caf4716c
Author: elemonier <emily.lemonier@gmail.com>
Date:   Sat Nov 16 12:50:50 2013 -0500

    initialize test non functional. issue with assign.

commit 267c751675314ca2585aeca36246ea8b6bd49e8c
Author: Thomas <tee2103@columbia.edu>
Date:   Sat Nov 16 12:43:14 2013 -0500

    add temporary test script. must go in wdjc dir

commit 0e71acb01c4053a622db75bb0dfa41cf8ddebb4f
Author: Emily Quinn Lemonier <emily.lemonier@gmail.com>
Date:   Sat Nov 16 12:40:16 2013 -0500

    Update ast.ml

commit debc8f185a080d7b773e4d7481f31ab5b9f2ec49
Author: Emily Quinn Lemonier <emily.lemonier@gmail.com>
Date:   Sat Nov 16 12:39:30 2013 -0500

    Update ast.ml

commit c9f75d6ac468fef4d07dd25bf454778993046e27
Author: Emily Quinn Lemonier <emily.lemonier@gmail.com>
Date:   Sat Nov 16 12:35:12 2013 -0500

    Update ast.ml

commit 4cc72124a9d4d940860d5ef4046e4b9bc8f8ac8d
Author: Emily Quinn Lemonier <emily.lemonier@gmail.com>
Date:   Sat Nov 16 12:34:54 2013 -0500

    fix merge conflict.

commit 8fa010bce409b42b1c2b6597c039c66b375bb99e
Author: elemonier <emily.lemonier@gmail.com>
Date:   Sat Nov 16 12:33:16 2013 -0500

    initialize test.

commit 30cac5505080b97bd8b97da3b21c3eccb8a53549
Merge: 42f0540 261833b
Author: elemonier <emily.lemonier@gmail.com>
Date:   Sat Nov 16 12:32:31 2013 -0500

    new java files.
```

```
commit 42f0540aa62f27ab0c81a46a4cd3e552aa221e23
Author: elemonier <emily.lemonier@gmail.com>
Date:   Sat Nov 16 12:25:33 2013 -0500

    Assignment + intialization for ints works.

commit 261833b37c35310313e11b2f72dbbab2441e22b6
Author: hilagutfreund <hila.gut@gmail.com>
Date:   Sat Nov 16 12:07:57 2013 -0500

    forced add jmusic jar

commit 5d8619ac443fefd464ebebe755281523cbb2910c
Merge: b1ccb57 5169f7d
Author: hilagutfreund <hila.gut@gmail.com>
Date:   Sat Nov 16 12:00:24 2013 -0500

    merges contgeo

commit 6e1b09362f44a948fc51a17cf53444c96c85e25a
Author: elemonier <emily.lemonier@gmail.com>
Date:   Sat Nov 16 11:59:22 2013 -0500

    Cleans up ast.ml and parser.mly

commit b1ccb57c34db0f536c284bf80724749448bf5700
Author: hilagutfreund <hila.gut@gmail.com>
Date:   Sat Nov 16 11:58:58 2013 -0500

    adding adding jmusic

commit 5169f7d0042cd42f29c8d552917c49057f57043a
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Sat Nov 16 11:58:31 2013 -0500

    reorgs original boilerplate

commit ff794b694b48025e0b9cd6b2c7b475019d3033c9
Merge: 7fd2293 96b033a
Author: hilagutfreund <hila.gut@gmail.com>
Date:   Sat Nov 16 11:58:11 2013 -0500

    merging my changes

commit 7fd2293fa79f31888f002da16689f11b0036c467
Author: hilagutfreund <hila.gut@gmail.com>
Date:   Sat Nov 16 11:56:09 2013 -0500

    added jMusic folder with instrument and jar file

commit 96b033af3d80f0cff8955cee9ef88c0732541ff0
Author: Thomas <tee2103@columbia.edu>
Date:   Sat Nov 16 11:53:28 2013 -0500

    creates type mod for vib, trem, bend in ast
```

```
commit ea1274c0d2cac0ba7bb0fae35430e56c8d0b1a5b
Author: elemonier <emily.lemonier@gmail.com>
Date:   Sat Nov 16 11:24:43 2013 -0500

    Adds assign test.

commit f259089581934f2b40e6cf213e8206c883712d5b
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Fri Nov 15 20:52:31 2013 -0500

    adds declare.dj stuff

commit f5fccf4b760298d8ce96f31881ffacaac52cd89b
Author: elemonier <emily.lemonier@gmail.com>
Date:   Fri Nov 15 20:47:02 2013 -0500

    Adding note test.

commit 050d9e26b3452fdda6c8f2882d26252ae0060d3f
Merge: 48f3417 f8e4f26
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Fri Nov 15 20:43:42 2013 -0500

    fixes merges

commit 48f3417d901da45d4a8e8ff62b0c25b3e2b6c270
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Fri Nov 15 20:43:03 2013 -0500

    totally fixes vdecl for primitives

commit f8e4f2639c8aa5b4237f70a5bbe47e4e33138c74
Author: elemonier <emily.lemonier@gmail.com>
Date:   Fri Nov 15 20:42:39 2013 -0500

    adds simple arith.

commit 7ed0849febeff5f6066deaab04902b6e73b3c9cb
Merge: 5fc2c9d 7a3221b
Author: elemonier <emily.lemonier@gmail.com>
Date:   Fri Nov 15 20:23:16 2013 -0500

    geo

commit 5fc2c9d99e2e81635cb616c3965f3c07323f30fb
Author: elemonier <emily.lemonier@gmail.com>
Date:   Fri Nov 15 20:23:02 2013 -0500

    Adds more tests.

commit c90832600174eb47f314a75e249613fa1aea8f99
Author: elemonier <emily.lemonier@gmail.com>
Date:   Fri Nov 15 20:20:58 2013 -0500

    Adds notes to scanner.

commit 7a3221b6def7315abcdbf4a7a4fbf67ebb1aa5e6
```

```
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Fri Nov 15 20:19:13 2013 -0500

    fixes primitive scanner arguments (lxm)

commit 282ab6d3c8a58558f5f290c2d650e45d00540eaa
Merge: 1f81c0d 4a989fe
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Fri Nov 15 20:09:12 2013 -0500

    fixes emily's screw up

commit 1f81c0d16dde64a797fe93dd34a59b5b164b2395
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Fri Nov 15 20:08:18 2013 -0500

    adds track and chord vdecl tempaltes

commit 4a989fe23427c96b7c698c5817f1312b23460d72
Merge: c4d46e4 62115f9
Author: elemonier <emily.lemonier@gmail.com>
Date:   Fri Nov 15 20:06:04 2013 -0500

    geojson

commit c4d46e4b2bb7aee8a2f0be7c191cf66f6151b71d
Author: elemonier <emily.lemonier@gmail.com>
Date:   Fri Nov 15 20:04:40 2013 -0500

    Modifies midi example.

commit 62115f97b94e3b4719a5d7a515187c1dd57d1494
Merge: 75d7a3a 735bb1e
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Fri Nov 15 20:04:09 2013 -0500

    fixes java and contgeo merge

commit 75d7a3a6a3c7385355e3306f76e694f605660ecd
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Fri Nov 15 20:02:33 2013 -0500

    adds primitive decls

commit b6fb6b2c10854e9a4760dcb29ed64b7a46e823fc
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Fri Nov 15 19:44:49 2013 -0500

    uncomments scanner note stuff

commit 735bb1e5477e54efd114ecf91e3dd7a216c2748c
Merge: cd58150 c8eb3f1
Author: hilagutfreund <hila.gut@gmail.com>
Date:   Fri Nov 15 19:41:20 2013 -0500

    Merge branch 'master' of https://github.com/WHET-PLT/wdjc
```

```
    Conflicts:
        CONTRIBUTORS.geojson

commit cd581501728656ff9fe0ac8b9c5f26ed005e88a6
Author: hilagutfreund <hila.gut@gmail.com>
Date:   Fri Nov 15 19:39:57 2013 -0500

    added comments for java example

commit c8eb3f175184d525c2304d048f0f7f35df97d66c
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Fri Nov 15 19:27:34 2013 -0500

    changes main to song and bytecodes\!

commit 1f1370a2dee5c9c61f586749d6ef546e8f2fb713
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Fri Nov 15 19:24:45 2013 -0500

    fixes makefile to utilize bytecode

commit 14b74247766f09a6053bd234c8ef0875aa3caaa5
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Fri Nov 15 19:21:52 2013 -0500

    adds bytecode object to makefile

commit b77736e278b38a74755e841d6d6aa94696b34482
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Fri Nov 15 19:19:19 2013 -0500

    removes weirdo s

commit 2482ecb8e8cf37b03c150c712a2e69d4a1fd110b
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Fri Nov 15 19:14:01 2013 -0500

    removes interpret cl arg

commit 2eb6f0b45deb4af413e3181a028b9a56016b91a5
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Fri Nov 15 19:08:59 2013 -0500

    fixes global bytecode something

commit 5ea983de1a5e25c6b344be2cb6802916e752097a
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Fri Nov 15 14:54:29 2013 -0500

    removes while and for from compile.ml

commit ff118127d34392bd855d9db3cfe3865f16958ee6
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Fri Nov 15 14:50:55 2013 -0500

    readds Noexpr
```

```
commit b4ed4c761c233643654cd8c60b6475b8fd5372b1
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Fri Nov 15 14:43:37 2013 -0500

    adds microc compiler template

commit 085685b412e05ad033a5c4a2ad7a3a7bbfcc2d2c
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Fri Nov 15 14:41:55 2013 -0500

    adds bytecode action to wdjc

commit b5909a39d1d77a8d1bde16cf5989497066133f74
Merge: 781bd64 b90bc46
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Fri Nov 15 14:40:26 2013 -0500

    merging with testall

commit 781bd640624a6906353db063abcdb3b82d93cf74
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Fri Nov 15 14:39:14 2013 -0500

    adds copied bytecode type

commit 0e7019bae9e3ed7b7802b1b089eb4dd54c40e157
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Fri Nov 15 14:38:49 2013 -0500

    adds bytcode directive to wdjc

commit b90bc46f5df7a2a700248295706975cd9f785b13
Author: Thomas <tee2103@columbia.edu>
Date:   Fri Nov 15 14:06:52 2013 -0500

    modifies shell script

commit 9517028ed0dff40ef7d451b9726c592da2b311f1
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Fri Nov 15 13:46:57 2013 -0500

    minor ast move arounds

commit 4af58a59f897dc4eb09c4944ba105ff36b4b273f
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Fri Nov 15 13:46:23 2013 -0500

    updates simple arith test to represent valid program

commit a456f7307198992408ffc32f95ad3a740b674638
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Fri Nov 15 13:36:03 2013 -0500

    updates wdjc to runnable

commit 1df4060286d1d205b0579f4c9508c2cc57b95c39
Author: William Falk-Wallace <wfalkwallace@gmail.com>
```

```
Date:   Fri Nov 15 13:35:55 2013 -0500

    removes wdjc stuff from compile

commit f5bb5203c24bb5bbf74c8d5e3b448026b6f1890e
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Fri Nov 15 13:35:37 2013 -0500

    removes print from arith test

commit 0e4a56c31627d4de5ab6b591c6f1dcb71b2aeef6
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Fri Nov 15 13:33:54 2013 -0500

    removes make products

commit 1bb7047ead8c19d9c92ed12057ecb7839639414a
Author: elemonier <emily.lemonier@gmail.com>
Date:   Fri Nov 15 13:05:06 2013 -0500

    modifies test.

commit f1dba845acd4d291d29912a8f609c939542124eb
Author: Thomas <tee2103@columbia.edu>
Date:   Fri Nov 15 13:03:09 2013 -0500

    comments out sections of makefile

commit d42f4bc284b741117979de4714f1acae315d054e
Author: Thomas <tee2103@columbia.edu>
Date:   Fri Nov 15 12:57:22 2013 -0500

    adds to compile.ml

commit 247ec24c9f640bc21f83b0a4a27c7aa69c267c8a
Author: Thomas <tee2103@columbia.edu>
Date:   Thu Nov 14 23:13:13 2013 -0500

    yaaaay i think the parser/scanner/ast compile

commit 732f957cca3a042423e1312714e02a3ef425f655
Author: Thomas <tee2103@columbia.edu>
Date:   Thu Nov 14 18:18:41 2013 -0500

    fixes no_expr section of parser. parser compiles

commit d62a6c5a5cd3849c5eceb1d3d433ed285f7f98fb
Author: Thomas <tee2103@columbia.edu>
Date:   Wed Nov 13 20:08:38 2013 -0500

    comments out loop for tests

commit a066de7df0d1df252d53b7b3011f9ae0c0b96e82
Author: Thomas <tee2103@columbia.edu>
Date:   Wed Nov 13 20:03:39 2013 -0500

    fixes compile errors from parser. parser currently compiles
```

```
commit fdbfb1ac6e02a8cf62ebc2b328d30ad35c81be9a
Author: Thomas <tee2103@columbia.edu>
Date:   Wed Nov 13 16:39:26 2013 -0500

    modifies ast. gets precedence errors.

commit 87db89c67720329f2a3740b27b8eda638fc08a64
Author: Thomas <tee2103@columbia.edu>
Date:   Wed Nov 13 16:34:52 2013 -0500

    comments out 'TARFILES' in makefile. adds testall and wdjc

commit 371d9f1f2dca8e168f023689a9746931436d1f0a
Author: Thomas <tee2103@columbia.edu>
Date:   Wed Nov 13 15:56:10 2013 -0500

    deletes while from ast

commit 8b84c95f6758ee4d76e66cb5891f823618de5a78
Author: Thomas <tee2103@columbia.edu>
Date:   Wed Nov 13 14:59:12 2013 -0500

    adds comments to parser. adds loop to ast section.

commit 1bd1179d5a99953f97cfb01aa481800962bfd345
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Wed Nov 13 14:42:36 2013 -0500

    parser questions

commit 7bb50ff19529841d0ecf91bb217e485e1fa28ddb
Merge: 2498f6c 27d80e4
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Tue Nov 12 13:16:14 2013 -0500

    merges contgeo

commit 27d80e464ab4293ea887bda18147b08421f10f0b
Author: Thomas <tee2103@columbia.edu>
Date:   Tue Nov 12 02:33:49 2013 -0500

    replies to comments in parser

commit 2498f6c7738a687f4d05559638705fbb40fcb2f9
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Mon Nov 11 23:38:01 2013 -0500

    missed one; but make isn't working?

commit d09a07ca2d2d92db0f3ec053fc6c8931c84c1384
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Mon Nov 11 23:37:15 2013 -0500

    updates references to microc to refer to wdjc

commit b85bc1cb249cf625f8a0039cfff1a92c7a554797
```

```
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Mon Nov 11 23:30:16 2013 -0500

    minor comments and questions

commit 8cbbf46658f6f0027f96d8aeca1ad6e1e9e802b0
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Mon Nov 11 23:23:12 2013 -0500

    adds pitch and instrument attr accessor tokens

commit e266b74eae0a2bfd4231cac22f500d971ea98c3b
Merge: 4426528 0d8f26e
Author: Hila Gutfreund <hg2287@columbia.edu>
Date:   Mon Nov 11 18:38:21 2013 -0500

    Merge branch 'master' of https://github.com/WHET-PLT/wdjc

    Conflicts:
        CONTRIBUTORS.geojson

commit 4426528354e4724c191f9caae7c53e55f72236ca
Author: Hila Gutfreund <hg2287@columbia.edu>
Date:   Mon Nov 11 18:37:08 2013 -0500

    commenty stuff for midi program

commit 0d8f26ea74edd9895f68ade9f0457332a6579e02
Author: Thomas Elling <telling2103@gmail.com>
Date:   Mon Nov 11 01:32:30 2013 -0500

    Update CONTRIBUTORS.geojson

commit f9754b00b5d79dac27daa6f0ab934f72f3b186b5
Merge: 09e1e89 76dc5d7
Author: Thomas <tee2103@columbia.edu>
Date:   Mon Nov 11 01:29:39 2013 -0500

    fixing merge conflict

commit 09e1e89e278e0e444ee52111808cd5098b30a8c8
Author: Thomas <tee2103@columbia.edu>
Date:   Mon Nov 11 01:28:15 2013 -0500

    adds a TODO section to readme for my own sanity

commit 76dc5d7a92b18d56e6a585c51bc539cbc0b80fc1
Author: Hila Gutfreund <hg2287@columbia.edu>
Date:   Sun Nov 10 20:09:28 2013 -0500

    a very simple java program that utilizes the midi library to create a one note
        midi file

commit 06ce68b2b16b6036e74ac22bf5ca376549d726f4
Author: Thomas <tee2103@columbia.edu>
Date:   Sun Nov 10 12:34:06 2013 -0500
```

```
    adds to expr section of parser. adds comments

commit 9850aec9921ad8880c9391a6ba22bca9f16b5a38
Author: Thomas Elling <telling2103@gmail.com>
Date:   Sun Nov 10 12:22:18 2013 -0500

    Update CONTRIBUTORS.geojson

commit 360ca6842a1748cc7209b879e4d90e966c54a7c8
Author: Thomas <tee2103@columbia.edu>
Date:   Sun Nov 10 12:20:29 2013 -0500

    adds 'Modifer' to type expr. needs review, see comments

commit 1aa5b26f71e7c74757ec202d2020b47c6005fda7
Author: Thomas <tee2103@columbia.edu>
Date:   Sun Nov 10 11:58:07 2013 -0500

    updates various parts of ast. adds comments and TODOS

commit fc23af7f34fe5ac5df40e494f7cde36c090ad64a
Author: elemonier <emily.lemonier@gmail.com>
Date:   Sat Nov 9 14:19:13 2013 -0500

    Rename dj.ml as compile.ml to match microc.

commit a5a80950f753be8df6d002768c73c84a947df65a
Author: Emily <emily.lemonier@gmail.com>
Date:   Sat Nov 9 14:09:20 2013 -0500

    Update CONTRIBUTORS.geojson

commit e157b25e842434b876d896b4dddbe81c67712d4d
Merge: 7bcdc8c 1aed238
Author: elemonier <emily.lemonier@gmail.com>
Date:   Sat Nov 9 14:08:31 2013 -0500

    Modifies CONTRIBUTORS.geojson.

commit 7bcdc8caec6df3b989aa8fe3a43485201f0314dc
Author: elemonier <emily.lemonier@gmail.com>
Date:   Sat Nov 9 14:07:56 2013 -0500

    Modifies Makefile.

commit 1aed23803286d980c964c16ecc09ee0577601fec
Author: Emily <emily.lemonier@gmail.com>
Date:   Sat Nov 9 14:04:51 2013 -0500

    Update CONTRIBUTORS.geojson

commit 89598ece9ae6014d32d563ecdccf85b99960bcd6
Author: elemonier <emily.lemonier@gmail.com>
Date:   Sat Nov 9 14:03:40 2013 -0500

    Updates makefile.
```

```
commit 10788513e72847d46bc118c4c0ff5b9d31d5facd
Author: Emily <emily.lemonier@gmail.com>
Date:   Sat Nov 9 14:02:17 2013 -0500

    Update CONTRIBUTORS.geojson

commit e08a9d02dbabb9a85460e8fc32df6d9a3903c9cc
Author: elemonier <emily.lemonier@gmail.com>
Date:   Sat Nov 9 14:01:09 2013 -0500

    Updates makefile.

commit 18473c8dd861c6702ec8be68d7bc4c8e5bc4d46d
Author: Emily <emily.lemonier@gmail.com>
Date:   Sat Nov 9 13:54:51 2013 -0500

    Update CONTRIBUTORS.geojson

commit f2dd7f97a7994b7dfb5655e81b8dc6388e75573a
Author: Emily <emily.lemonier@gmail.com>
Date:   Sat Nov 9 13:53:28 2013 -0500

    Update CONTRIBUTORS.geojson

commit 72ee3237b8ca49f82c0c4b2827c53e4e4aac512e
Merge: b9ec2dc bc1954d
Author: elemonier <emily.lemonier@gmail.com>
Date:   Sat Nov 9 13:49:34 2013 -0500

    Updates CONTRIBUTORS.geojson.

commit b9ec2dc286ca91e735caeaa28e1cb323d0bbd7fd
Author: elemonier <emily.lemonier@gmail.com>
Date:   Sat Nov 9 13:48:59 2013 -0500

    Updates makefile.

commit bc1954daacc1abff2177ca0bb9dc961e3b89c0f4
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Sat Nov 9 13:47:39 2013 -0500

    fixes my contgeo goof

commit 52b6561610af630c6edddfa02943189271e6ff97
Merge: 04268ba f4060b2
Author: elemonier <emily.lemonier@gmail.com>
Date:   Sat Nov 9 13:47:30 2013 -0500

    Updates CONTRIBUTORS.geojson to fix merge conflict.

commit 04268badbc8d0641c2a74c83bc7ae8b8225f7ee2
Author: elemonier <emily.lemonier@gmail.com>
Date:   Sat Nov 9 13:44:17 2013 -0500

    Creates basic Makefile.

commit f4060b2ce28863a3c3f950d3f1f89d6521742c17
```

```
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Sat Nov 9 13:44:15 2013 -0500

    fixes emily's contgeo goofs

commit 878266964d4d13bdba5e759da951080b979960bf
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Sat Nov 9 13:39:08 2013 -0500

    fixes contgeo merge junk

commit 7a98b2ef84b13cb06d0a60cf8f23308d61314f61
Merge: ec567de 4d552e2
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Sat Nov 9 13:35:21 2013 -0500

    silly sublime merging

commit ec567ded069bb2f0626834fbe1e18bbb90c2cb87
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Sat Nov 9 13:33:23 2013 -0500

    comments and updates java note/chord/track

commit 4d552e27639af156564a2cd729df492a64d8e318
Author: Thomas <tee2103@columbia.edu>
Date:   Sat Nov 9 13:31:07 2013 -0500

    updates type op section of ast. adds comments to ast

commit e64bfec6d29d303f1be17dee190af66c03e7a2f0
Author: Thomas <tee2103@columbia.edu>
Date:   Sat Nov 9 13:21:07 2013 -0500

    modifies loop in stmt

commit ed4717ad0a757910b2de02cce6f37335780a892e
Author: elemonier <emily.lemonier@gmail.com>
Date:   Sat Nov 9 13:19:35 2013 -0500

    Updates parser.mly with more in depth association list.

commit 166f5860005f236802e1eb9c29074dd00bc847a4
Author: elemonier <emily.lemonier@gmail.com>
Date:   Sat Nov 9 13:05:48 2013 -0500

    removes unnecessary midi test files.

commit ac115f44d7aee7ca954022a12ecd6b7fd16d985f
Merge: 28f83ba ebad86b
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Sat Nov 9 13:04:24 2013 -0500

    merges ignore, parser, contgeo

commit 28f83bab016488d80dc8e67226d1288280059a7e
Author: William Falk-Wallace <wfalkwallace@gmail.com>
```

```
Date:   Sat Nov 9 13:03:29 2013 -0500

    fixes track constructor ref/val

commit c14985e7c02f515d18ebc775ceb374b7532368aa
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Sat Nov 9 13:02:49 2013 -0500

    fixes chord constructor ref/val

commit ebad86b47af259d4bf89d1b4af94f6466ea3c1df
Author: Thomas Elling <telling2103@gmail.com>
Date:   Sat Nov 9 13:02:36 2013 -0500

    Update .gitignore

commit ea5e203039e864e8d8c619c3c342d4ba00f5c84d
Author: Thomas Elling <telling2103@gmail.com>
Date:   Sat Nov 9 13:01:46 2013 -0500

    Delete .DS_Store

commit b09837784fdf1970cc82e3d580ba98b8462445a2
Author: Thomas <tee2103@columbia.edu>
Date:   Sat Nov 9 12:58:37 2013 -0500

    added serial and parallel to expr section

commit 54279f8abafa0a2bf0917557c460a6fd48022123
Merge: f449e9d 0c9e6a8
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Sat Nov 9 12:52:33 2013 -0500

    updates track serial add and merges track stuff

commit f449e9d7efdef39bf17d6a58925929d535533149
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Sat Nov 9 12:50:03 2013 -0500

    fixes parallel add

commit 0c9e6a80750b565db67bb93438bfc556cc556d63
Author: hilagutfreund <hila.gut@gmail.com>
Date:   Sat Nov 9 12:47:39 2013 -0500

    fixed return type for serial add track

commit f49605a1253269bf05ed2aa03eb8d5ad726ffdb0
Author: Emily <emily.lemonier@gmail.com>
Date:   Sat Nov 9 12:40:52 2013 -0500

    Delete midifile.class

commit eb1167a80e3386010a5c35a5aaddf5215fe874c3
Author: elemonier <emily.lemonier@gmail.com>
Date:   Sat Nov 9 12:40:20 2013 -0500
```

```
    Added java-midi example.

commit 4df08a20fa9bc20e3b3ecc3c0b04e3a892de9fd8
Author: Thomas <tee2103@columbia.edu>
Date:   Sat Nov 9 12:33:46 2013 -0500

    added expr section. clarification on arrays needed

commit 02b5b4452afbefb56a0bf1fe67cb554065f514d7
Merge: 6ffc72f af83bdb
Author: Hila Gutfreund <A@dyn-207-10-141-136.dyn.columbia.edu>
Date:   Sat Nov 9 12:32:10 2013 -0500

    Merge branch 'master' of https://github.com/WHET-PLT/wdjc

commit 6ffc72f5e9cf9e2fb7ebb9158f6bf6261f0435df
Author: Hila Gutfreund <A@dyn-207-10-141-136.dyn.columbia.edu>
Date:   Sat Nov 9 12:30:41 2013 -0500

    changed serial add for track

commit af83bdb16410e6206350d56bbc39b53f48d8aa86
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Sat Nov 9 12:30:40 2013 -0500

    adds instrument attribute to note class boilerplate

commit 96e6cea18d1479535de5ec4d7cd35aa7dc7d78df
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Sat Nov 9 12:27:05 2013 -0500

    adds track constructor on chord

commit c9a4101698cf21e19d7b7f15b566153693151182
Merge: 7c50aad 8a5a2f1
Author: Hila Gutfreund <A@dyn-207-10-141-136.dyn.columbia.edu>
Date:   Sat Nov 9 12:24:39 2013 -0500

    Merge branch 'master' of https://github.com/WHET-PLT/wdjc

commit 8a5a2f1a6454dd21283b47e957f815e635966705
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Sat Nov 9 12:24:03 2013 -0500

    updates contgeo

commit 7c50aad10786b3c307c9fcc4cc0c868351f5116b
Author: Hila Gutfreund <A@dyn-207-10-141-136.dyn.columbia.edu>
Date:   Sat Nov 9 12:24:01 2013 -0500

    changed track.java serial add - hila

commit 7d02d371b0f0b350e2e6454cbf3208627c2cb48e
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Sat Nov 9 12:22:12 2013 -0500

    fixes chord serial/parallel add methods
```

```
commit 06e58fdd1d68f996a35e9cf24ae3915688b6d12c
Merge: d3ca78d 92988a6
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Sat Nov 9 12:12:51 2013 -0500

    merges contgeo and parser

commit d3ca78d0b5e63a0bc05692d18dbe6f36ca61ee26
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Sat Nov 9 12:12:23 2013 -0500

    updates to java chord template

commit 4bcd820143c4d83c029639dede22804716c5abca
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Sat Nov 9 12:11:23 2013 -0500

    adds java Note template

commit 92988a6a92b9a0443878f4452e615a6977e21c08
Author: Thomas <tee2103@columbia.edu>
Date:   Sat Nov 9 12:10:41 2013 -0500

    added fdecl and vdecl

commit aa61c55a507f55f51bccd18dafed27d181e28759
Author: Thomas <tee2103@columbia.edu>
Date:   Fri Nov 8 17:29:37 2013 -0500

    added comments to scanner and parser. need clarification on precedence order
       for the associativity section of parser

commit 03c8cf4bca68a0f818f10f440c97cbc3f4a39876
Author: Thomas <tee2103@columbia.edu>
Date:   Fri Nov 8 17:20:18 2013 -0500

    added comma and loop to scanner. built out 'token' section of parser

commit 4c92d7c5ea6e652abf4ad1232194af8f603a8c63
Author: Thomas <tee2103@columbia.edu>
Date:   Wed Nov 6 22:28:10 2013 -0500

    created 'test' directory. sorry for the messy updates

commit 20870d86038193312d7fc9b5974bf9beacf071c0
Author: Thomas <tee2103@columbia.edu>
Date:   Wed Nov 6 22:22:48 2013 -0500

    sample arithmetic test case. let me know what you think

commit 7d52fc86b1ee6f2dc3f868703577b8304b21b7e5
Author: Thomas <tee2103@columbia.edu>
Date:   Wed Nov 6 22:17:36 2013 -0500

    fixed comment in ast
```

```
commit a1ccc97bd624945736beb24b44594a33297d6653
Author: Thomas <tee2103@columbia.edu>
Date:   Wed Nov 6 22:16:40 2013 -0500

    created dj.ml. added to ast.ml.

commit d31e87f59c8e04b77d100d149f23262a4afca7ba
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Wed Nov 6 19:35:21 2013 -0500

    adds java boilerplate classes; needs interfaces

commit c4922944f0ceb4ff76c5cd48663ea2e6f2667923
Author: Thomas <tee2103@columbia.edu>
Date:   Sun Nov 3 17:57:00 2013 -0500

    started to fill in ast/parser. pathetic makefile.

commit c8bf19704391f0a10465e9f7eae8a4cd0b0ead33
Merge: 51d3f78 1075a81
Author: Thomas <tee2103@columbia.edu>
Date:   Sun Nov 3 15:27:18 2013 -0500

    Merge branch 'master' of https://github.com/WHET-PLT/wdjc

commit 51d3f78f4672172d8cc758728faaef240bd7b182
Author: Thomas <tee2103@columbia.edu>
Date:   Sun Nov 3 15:26:57 2013 -0500

    added [mostly] blank ast.ml and parser.mly

commit 1075a81d07de00ab32537efd61cfb4eaa03ebc38
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Fri Nov 1 15:50:26 2013 -0400

    Update CONTRIBUTORS.geojson

commit fe772067b97160dadf08200916c4c82beaf803dd
Author: Thomas <tee2103@columbia.edu>
Date:   Fri Nov 1 15:48:49 2013 -0400

    cleanup from merge

commit ac261bda3448612b8e1dd96093748ae3e007b5e5
Merge: 67ec8c6 ed55e49
Author: Thomas <tee2103@columbia.edu>
Date:   Fri Nov 1 15:45:31 2013 -0400

    removes s typo

commit 67ec8c6c1c00d0a4414648e8db4d3d98e4a667d2
Author: Thomas <tee2103@columbia.edu>
Date:   Fri Nov 1 15:43:36 2013 -0400

    small edits

commit ed55e49ca786d5884e88ba6bdf0e83abdc127828
```

```
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Fri Nov 1 15:33:33 2013 -0400

    formatting a bit

commit fac4657984a45822f4dd00a87d9e958954dee6ad
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Fri Nov 1 15:24:35 2013 -0400

    ...and there was the scanner

commit 7f2faa6a63820a316998cf8411cd9e11d90eaacf
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Fri Nov 1 15:00:57 2013 -0400

    adds microc template and all of our reserved words

commit 96bfe91b64621bba373c8f100a09c9277bd9dc31
Author: William Falk-Wallace <wfalkwallace@gmail.com>
Date:   Fri Nov 1 11:28:30 2013 -0700

    Initial commit
```