corgi

musical alg’rhythms

The Team

- Philippe-Guillaume Losembe
- Alisha Sindhwani
- Justin Zhao
- Melissa O’Sullivan

Motivation

- Music is complex, but there are interesting patterns
- Patterns in notes and harmonies that can be analyzed
- Top-down and Bottom-up approaches
- Our goal was to develop a language to algorithmically generate music, and analyze these patterns in music.

Uses

- corgi’s main selling point is its ability to search through music.
- Data structures make it easy to identify and return the location of specific instances in a given composition
- Ability to programmatically generate music

Hello World

```
int main() {
    print("Hello, world!");
}
```

Types

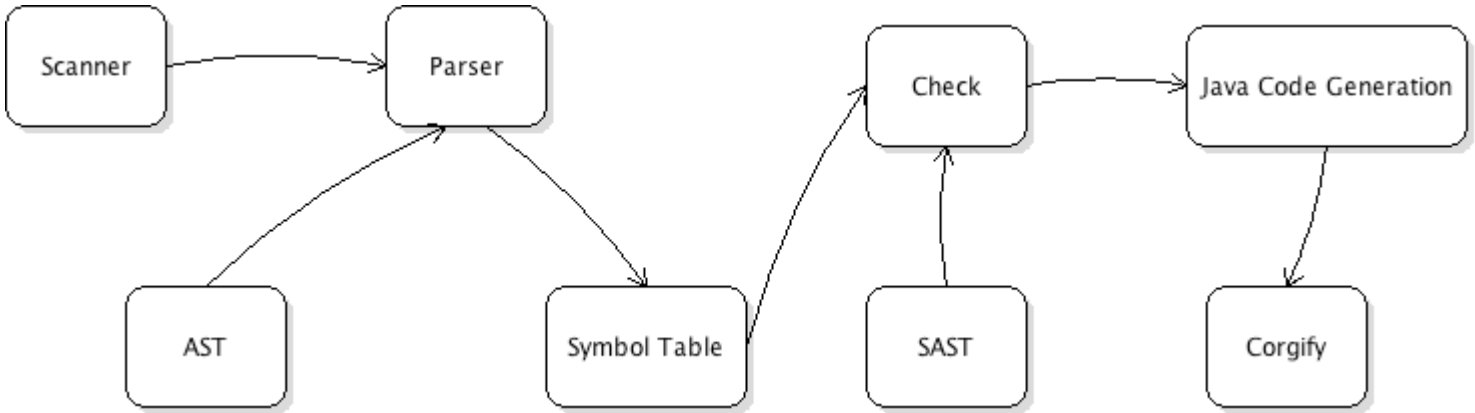
- Fractions
- Durations
- Pitch
- Pitch/Duration Tuples
- Chords
- Track
- Composition

Flexible Data Type Conversion

```
duration d;
d = $1/2$;
fraction f;
f = $1/3$
pitch p;
p = 5;
```

```
chord c;
c = [(5,$1/4$), (3,d), (p,f)];
```

Structure


Scanner
Parser
Check
Java Code Generation
AST
Symbol Table
SAST
Corgify

Java Implementation

- Use of the jFugue Library (not that great)
  - Limitations
- Translate well into Java class objects
- Added flexibility for greater abstraction

Lessons Learned

- Identify individual strengths earlier
- Start earlier, don’t procrastinate
- Do not underestimate how much time it takes to do even the small things
- Testing along the way is essential
- The more you distribute, the more you have to unify
- Be mindful of the limitations of the libraries that you use