# m

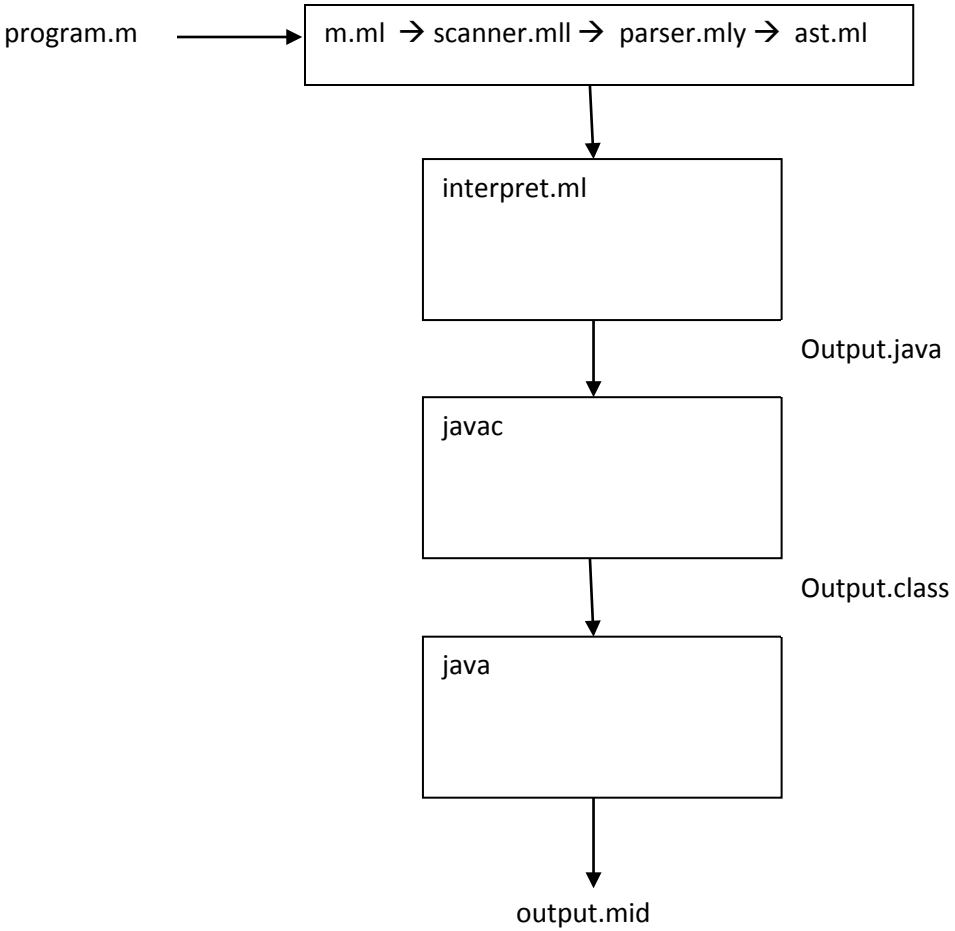
*A language for music generation.*

Yiling Hu | Monica Ramirez-Santana | Jiaying Xu

# Introduction

- m is a language specifically designed for algorithmic music composition
- Gives the programmer the following functionality:
  - translation of traditional musical concepts into arithmetic types
  - arithmetic operators for manipulation
  - typical control structures
  - randomization functions

# Compiler Design



# Hello World Tutorial

```
/*   The 'Hello World' program of the m algorithmic music composition
     language. Plays middle C. */

void main()
{
     note n;
     chord c;
     staff s;
     part p;

     n.pitch = C4;
     n.duration = 1.0;
     n.intensity = 100;

     s.instrument = 0;

     p.bpm = 60;
     p.beatsig = 0.25;

     add(c, n);
     add(s, c);
     add(p, s);

     play(p);
}
```

More types

C-like syntax

Set all derived type fields -Define note and song characteristics

Pitch literal – shorthand for the programmer

Compose the song

Print this song to Java source file

# Derived Types

- Translation of traditional music concepts into arithmetic types
- Hierarchical in nature
- Standard library functions to interact with them

| Type | Members | add() | play() |
|---|---|---|---|
| `note` | pitch<br>intensity<br>duration | Cannot add anything to type `note` | Play the pitch at the intensity defined for the duration defined. |
| `chord` | Collection of notes | Add type `note` | Play its notes simultaneously. |
| `staff` | Collection of notes and chords<br>instrument | Add type`chord` | Play its chords in the order they were added. |
| `part` | bpm<br>beat signature | Add type `staff` | Play its staffs simultaneously. |

# Lessons Learned

- Start early
- Learn Ocaml
- Meet often
- Source code version control
- Testing is good
- Catching errors early on is hard

# Conclusions

- Simple way to make simple music
- Great for elevators, coffee shops, places that have music that no one really pays attentions
- Meet often
- Making a language is fun!

# DEMO