E-CATZ

Ethiopia Mengesha, Chianna Cohen, Annie Sui, Tim Vallancourt

*:.*✧ (๑ↀ ᦿ ๑)ノ☆☆*♩*✧・゚
E-CATZ Overview 🐱

Goal: To enable users to explore their musical artistry through code.

Notable Features:

- Note Objects

  ![G4](image1)

- Writing Notes to LilyPond

  ```
  Note[] output = [[D3, E3, F#3, G3]];
  ```

- Sequences and Harmonies

  ```
  [A3, C4, D4, E4, C4, D4, E4, G4]
  ```

- Sequences of Harmonies

  ```
  [[E G C], [F A C], [F B D]]
  ```
E-CATZ in One Slide

```
#include "stdlib.catz"

/* 'invert' the array by flipping the distances between each note */

def Note[] inverse(Note[] input) {
    Note[] output = new Note[input.length];
    output[0] = input[0];
    int initialPitch = noteToInt(input[0]);
    for (int i = 1; i < input.length; i = i + 1) {
        int delta = noteToInt(input[i-1]) - noteToInt(input[i]);
        Note newPitch = addToNote(output[i-1], delta);
        output[i] = new Note(newPitch.pitch, input[i].rhythm);
    }
    return output;
}

Note[] bluesScale = [C4, D4, D#4, E4, G4, A5];
write(inverse(bluesScale), "filename");
```

Include statement for preprocessing

Include statement for preprocessing

Function definition

Function definition

Note array instantiation

Array access and assignment

Call to helper function in StdLib

Array literal

Write function takes an array and outputs a LilyPond file
• Represented as structs with a pitch and rhythm literal

```
Note myNote = new Note(C#5, hf);
```

• Access and modify the pitch and rhythm of a Note using dot (.) operator

```
Note x = new Note(F#3, wh);
Note y = new Note(x.pitch, hf);
x.rhythm = ts;
```
Arrays

- Construct arrays by specifying a length or using an array literal

```java
int[] x = new int[10]; /* empty array of length 10 */
int[] y = [3, 5, 6, 7]; /* array literal with 4 values */
```

- Use nested arrays to create harmonies

```java
int[][] a = new int[][][5]; /* array of 5 empty int arrays */
int[][] b = [[3, 5, 7], [1, 2]]; /* array of 2 arrays of varied length */
```
Standard Library

- Written in E-CATZ
- Contains useful built-in functions
  - `noteToInt()`
Write()

- Written in C
- Translates E-CATZ output into LilyPond
- Inserts LilyPond notation for Notes, Harmonies, and Sequences
How to Use E-CATZ 🐱

1. Set up Docker environment

2. Build E-CATZ compiler
   - `make`

3. Preprocess & execute program
   - `./run-ecatz.sh [program.catz]`

4. Acquire MIDI file

5. Open the .ly program outputted by an E-CATZ program in LilyPond

6. Listen to your work!
We will now show you a MIDI soundtrack we made from E-CATZ!
Future Work

Extended Features

- Capability to play notes of different rhythms in the same chord
- Dotted-note rhythms
- Time Signatures
- Rests
- Ability to read a LilyPond file into a program that one can manipulate
- More efficient memory management

Questions?