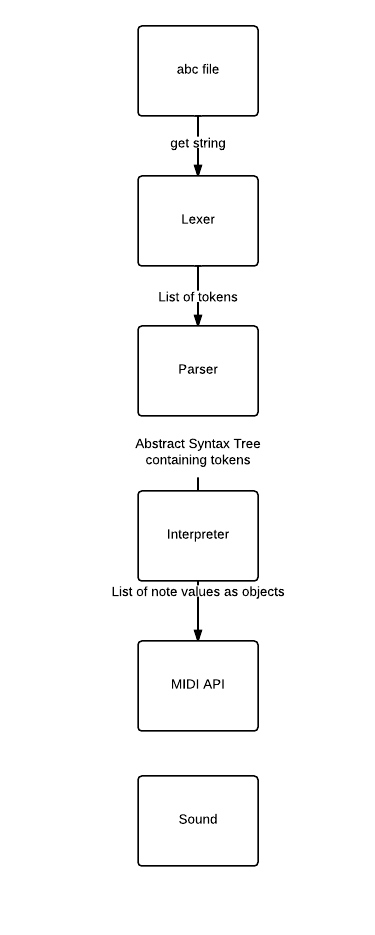
ABC Player

Author: Yonglin Wu, Yang Chen, Menghsuan Pan

Introduction

ABC Player is a Java application that plays an abc file. It takes one abc file at a time, parse it and feed it into Java MIDI API. It has three major parts: lexer, parser and interpreter. The function of each part is explained in details in this document.

Flow Chart:



Lexer

We read the input file line by line and convert each line into a string, Lexer will convert all the strings together into a header(a type defined in header.java) and a list of tokens.

**Header**

The header has the following fields: C,K,L,M,Q,T,X and it contains information about the whole piece, namely, composer(C), key(K), default length of a note(L), meter(sums of durations of all notes within a bar), Q(tempo), T(title of the piece) and X(index number).

**Token**

Tokens are generated from characters or substrings from the list. If we see a matching

A token has the following fields:

* Type. It specifies what type of token it is. It is an enum type, which is listed below with each type’s corresponding string:

LINE: |

ACCIDENTAL: ^ or \_ or =

NOTE: a-gA-G

OCTAVE: ' or ,

LEFTBRA: [

RIGHTBRA: ]

LEFTPAR: (

COLON: :

Rest: zZ

LENGTH: number that represent the length of the note, such as 2, 1/3.

VOICE: v (followed by the name of the voice)

* String. The original string value of the token is stored in this field.

**Semantic Checking and Exception Thrown**

In general we don’t handle grammar error (such as “((“ or “^^”) in Lexer. In Lexer we simply return all the tokens are the parser will throw exception later. We ignore all the white spaces and all the characters that don’t exist in abc grammar, such as ‘&’, ‘+’, etc.

Something else that is worth noting is that a % symbol will cause the remainder of any input line to be ignored.

Lexer throws only two types of Exception:

IllegalArgumentException.

It is thrown when the header of the input file doesn’t follow the following format:

* The first field in the header must be the index number ('X').
* The second field in the header must be the title ('T').
* The last field in the header must be the key ('K').
* Each field in the header occurs on a separate line.

IOException:

It is thrown when the application cannot find the file or cannot open the file.

Parser

Parser parses the output of a Lexer (a header and a list of tokens) into an abstract syntax tree. The datatype definition/grammar is shown as below. There are two types of datatypes: terminals, which are the tokens produced from Lexer, and non-terminals, which are recursively defined by terminals and non-terminals.

**Datatype Definition/ Grammar**

***Terminals:***

LINE: |

ACCIDENTAL: ^ or \_ or =

NOTE: a-gA-G

OCTAVE: ' or ,

LEFTBRA: [

RIGHTBRA: ]

LEFTPAR: (

COLON: :

Rest: z

LENGTH: number that represent the length of the note, such as 2, 1/3.

VOICE: v (followed by the name of the voice)

***Non-terminals:***

Music = Voice\*

Voice = (Bar|Repeat)\*

Bar = Notes LINE

Notes = (Single|Chord|Tuplet|Triplet|Rest)\*

Single = ACCIDENTAL? NOTE OCTAVE\* LENGTH?

Chord = LEFTBRA Single+ RIGHTBRA

Tuplet = LEFTPAR [2] Single{2}

Triplet = LEFTPAR [3] Single{3}

Rest = [z] Length?

**Graph**

**Semantic Checking and throwing exceptions**

If the input file has incorrect syntax, we will throw exceptions. Here are a list of exceptions and when to throw them. All of the exceptions extend RuntimeException class.

WrongGrammarException: When a set of tokens don’t fit into the grammar. (Eg. NOTE LENGTH LENGTH, LINE LINE)

BarLengthException: When the number of notes in a bar is incorrect. (Eg. When the meter is 4/4 and default note length is ¼ and there are only 3 full length note in a bar)

Interpreter