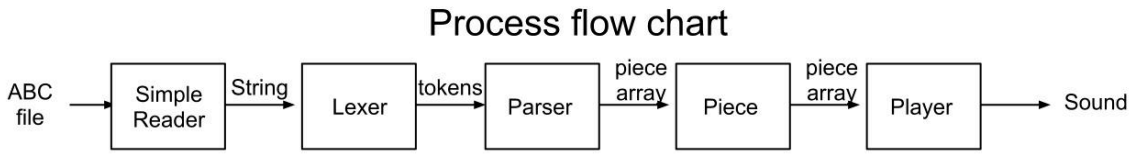
ABC Player Design:



Data types:

Tokens – enums:

Header tokens:

Token
X_INDEX
T_TITLE
K_KEY
C_COMPOSER
M_METER
L_NOTELENGTH
Q_TEMPO

Body tokens:

Token
NOTE
CHORD_OPEN
CORD_CLOSE
TUPLET
REST
VIOCE
BAR
REPEATSTART
REPEATEND
REPEAT1
REPAT2
MAJORSECTION
EOF

Classes and interface:

Functional classes:

- Token- enums.
- simpleReader- takes an ABC file, return a string
- Lexer- takes a string, return tokens.
- Parser- takes tokens, return a piece ArrayList.

Music classes:

- Piece – contains an ArrayList of major sections.
- Major section- contains an ArrayList of bars.,
- Bar- contains an ArrayList of: notes,rests, chords and tuplets.
- Chord- contains notes.
- Tuplet- contains notes.
- Rest.
- Note.
-

Interface:

- musicExpression – chords, tuplets, notes and rests implement this interface.

Additional information about classes:

Parser:

- Parse: will call parse header and parse voice.

Parse header:

- First is X , second is T last is K – throw an error if not
- After a key token will stip.

Parse voice:

- Will create arrayList for each voice. If no voices assume we have one voice.
- Call parse body.

Parse body:

- Evaluates the body, and deals with repeat.

Header class:

- Has a get method for each part of the header, for example getTempo().
- Keep default values in the header class and override them if we get new values.
- Constructor will get an arrayList of all the headers value that we found.
- Throws an exception if missing mandatory fields

Note class:

- Constructor Note(String note)
- parse inside for note value- iterate over each char in the string, to build the right note. We should have octave up/down ,set ^_= and timing. The piece class in the end will pass (ticks/note) and starting tick.
- Will have a pitch value
- Will have a length value
- Store in either a chord , tuplets or a bar

Chord class:

- Constructor Chord(ArrayList notes)
- Figure out ticks per note

Tuplets class:

- constructor Tuplets(ArrayList notes)
- parser will call it when we see ([2-4]
- figure out ticks per note

Rest class:

- constructor Rest(String z)
- figure out ticks- from the timing and the ticks per note that get pass to it

If we see repeat start |: then we set the starting repeat after the last bar, when we see repeat end :| repeat from the start bar
If we see repeat1 we skip

Voice class:

Constructor Voice(arrayList bars, String voice_name)

Grammar:

header

Mandatory:

Token	Regex
Index	X\\:[1-9]+
Title	T\\:[^\\r\\n]*
Key	K\\:[A-G][m]?

Optional:

Token	Regex	Default value
Composer	C\\:[^\\r\\n]*	“Unknown”
Meter	M\\:[1-9]+[\\/][1-9]+	4/4
NoteLength	L\\:[1-9]+[\\/][1-9]+	1/8
Tempo	Q\\:[1-9]+[0-9]*	100
VOICE	V\\:[^\\r\\n]*	N/A

Body

Part	Grammar
Piece	[MajorSection]+
MajorSection	[Bar]+
Bar	Repeat? musicExpressions repeatEnd?]+[Bar]
MusicExpressions	[NoteChordTupletRest]

Token	Regex
Note	[[\\^_\\=]{1,2}]?[a-gA-G][\\']*[\\,]*[1-9]*[\\/]?[1-9]*
Tuplet	[\\(][2-4]
Tuplet	Tuplet Note+
Rest	[z][1-9]*[\\/]?[1-9]*
Repeat	[RepeatStartRepeatEndRepeat1Repeat2
Repeat1	\\[[1]
Repeat2	\\[[2]
RepeatEnd	\\:\\|
RepeatStart	\\|\\:
Major section	\\|\\]
Bar	\\|
OpenChord	\\[
CloseChord	\\][1-9]*[\\/]?[1-9]*
Chord	OpenChord Note+ CloseChord

Appendix A:
Important assumptions and specs:

- ABC notation does not require measures to be complete. We are going to accept incomplete measures (for example fur elise) and will not create pickups for incomplete measures.
- Based on a talk with a TA , we are assuming that a tuplet will only contain notes and rests inside of it , and there is no possibility of a tuplet of tuplets. The behavior of such a tuplet with tuplets is not defined.
- Based on a talk with a TA , we are assuming that a chord will only contain notes inside of it. The behavior for a chord that contains something else is not defined.
- We are also assuming that the abc body will only contain valid notes, rests , tupltes and chords. Invalid body will result in a failure