**Project 1 ABC player**

**Overviews:**

The Java file takes the filename of a ABC-music.abc file as an input. The Java Read class read the content into a String Object. Then the String is transfered to Lexer, which will create Tokens according to the grammar we provided. The parser parse the tokens, generating an abstract syntax tree afterwards. Then Visitor will process the AST into a Song object which can be interpreted by another Visitor. As the Song being processed, Notes will be added to SequencePlayer simultaneously. Finally, The SequencePlayer plays the music that User can hear.

**The snapshot diagram as bellows:**

**Grammars & Abstract Syntax Tree ：**

Grammar has not been changed significantly. The AST was built based on the given BNF grammar file by using ANTLR.

* We defined the implicit things in the given grammar ex, DIGIT
* We Changed LineFeed into End-Of-Line ::= newline | comment .

By translating the modified grammar into ANTLR to generate Parser and Lexer.

**Traversing the AST to create Song object:**

Once we have an AST, we can create a listener, The ASTvisitor to traverse the AST to generate NOTE, CHORD, TUPLET, REST object which contain PITCH and DURATION, MEASURE, VOICE(List of Measures), MUSIC(List of VOICE) , HEADER then the SONG object which is the input of SequenceVisitor, Visitor class.

**Abstract Syntax Tree:**

**Datatype definition:**

**Sequence(Interface) :** this represents any contiguous block of music which provides the method that invoke a visitor object to process a sound that SequencePlayer can player.

* Visitor Method: accept( ).

**Music (Interface )** : It represents the a music element that its duration can be changed

* Method : 
  + ChangeDuration();

**Song (implement Sequence):** represent a entire song, an output of a parser & an input of a visitor

* **Immutable**
* **Fields :**
  + Voices (List<Voice>)
  + Header
* **Method :**
  + Observor: getHeader(), getVoices();
  + Visitor method : accept();
  + Constructor method for player();

**Header:** contains the fields which an ABC header store and help set the value of instance variables like NOTE, DURATION

* Immutable
* Fields :
  + tempo, title, index and etc;
* Methods: 
  + Observer : getTempo, getMeter , getDefaultLength, getVoice, getKey, getIndex.

**Voice( implement Sequence)** this contains a list of measures in a single voice

* **Immutable**
* **Fields :**
  + List<Measure> measures
  + int VoiceNum
* **Method:**
  + Constructor for Voice
  + getVoiceNum(), getMeasures().
  + Visitor method : accept().

**Repeat :** represent a repeat section in ABC music, dealing with measures and endings

* **Immutable**
* **Fields:**
  + List<Measure> measures
  + List<Measure> endings
* **Method :**
  + constructor for Repeat
  + getRepeat() : return measures in a Repeat section

**Measure(implement Sequence):** represent a repeat /Non-repeat measure and all the music inside it such as Single Note, Chord, Tuplet

* Immutable
* Fields :
  + List<Music> element
* Method : 
  + Constructor for Measure
  + Observor : getElements().
  + Visitor method : accept().

**Chords****(implement Sequence) :** an object contains a list of Note object with same time duration

* + **Immutable**
  + **Fields :** 
    - List<Notes> Notes
    - Duration duration
  + **Method** : 
    - Creator: (SingleNotes, Duration..)
    - Observer: getNotesNum, getDuration, getType
    - Producer : changeTickTime()

**Tuplet (implement Sequence): A** Tuplet represents a list of Musics, Chords or Notes.

* **Immutable**
* **Fields:** 
  + List <Note> , List <chord>
  + Duration duration
  + Type of Tuplet
* **Methods :** 
  + **Constructor for tuplet**
  + **Observor : getMusics, getType, getDuration**
  + **Producer: changeDuration().** 
    - **return a new Tuplet with different Duration .**
  + **Visitor method: accept().**

**Note(implement Sequence):**

a note object which contains a single Note or a Rest

* + **Immutable**
  + **Fields:**
    - * Key Key
      * String Rest
      * Duration
  + Method: 
    - **Constructor for tuplet**
    - **Observor : getKey, getDuration**
    - **Producer: changeDuration().** 
      * **return a new Note with different duration .**
    - **Visitor method: accept().**

**Duration :** represent the duration in terms of the music length

* + **Immutable**
  + **Fields**: int numerator, denominator
  + **Method** : 
    - Creator for duration
    - observer : getNumerator(), getDenominator()
    - Producer: getLCM() // calculate the LCM of all duration

**Key :** a key represents the Key in a Note, containing Pitch, Octave, accidental , or mode minor

* + **immutable**
  + **Fields**
    - **Pitch pitch**
    - **String modeMinor**
  + **Methods:** 
    - **Constructor for Key**
    - **getPitch().**

**Pitches:**

#Already given in the original file

**Visitor :**

the visitor interface containing methods that visit object in a song.

**SequenceVisitor** :

the visitor class implements Visitor that transverse the Song, using information in Music element such as duration to calculate Tick Number of a element and finally add it to sequence player.

**ASTVisitor :**

A Visitor that transverse the AST produced by a Parser to generate Header, Note, Chords, Tuplet, Measure, Voice, and Song, and dealing with repetition and calculate the Tick Time Per Quarter of a Song and Duration of a Note.