Rakiah L. Grende

Professor Williams

Carroll College

CS-410 Operating Systems

November 9th, 2022

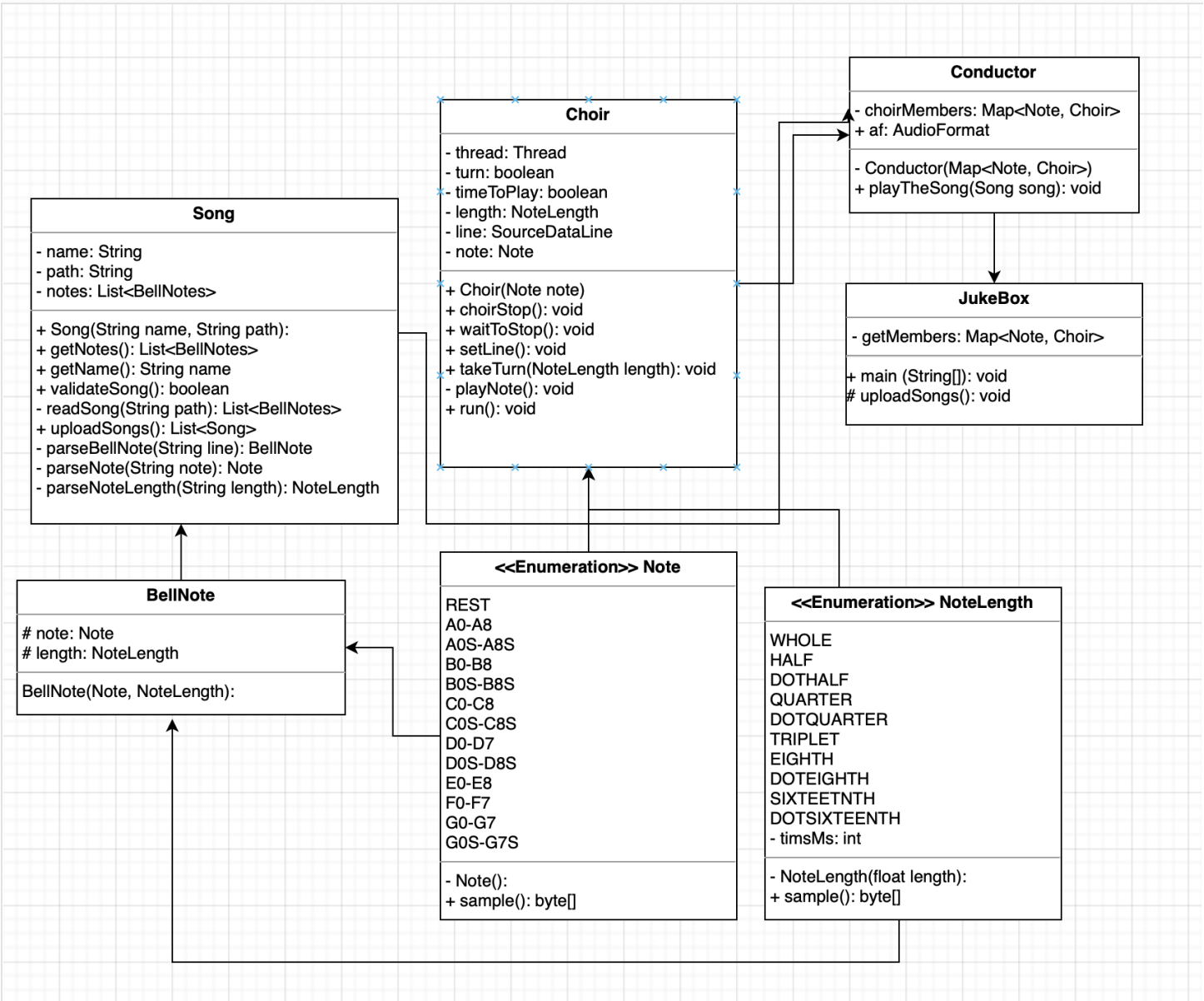
Lab 2: Bell Choir

I, Rakiah Grende, wrote 90% of the Bell Choir lab*.

X 

*I had exceptional help from Jake Grossé and Jaden Bathon and they allowed me to use all 8 of their octaves and a similar style in the main method of the JukeBox Class. They also introduced me to regular expression, or regex which was also implemented in my code. I did not copy and paste any code other than the list of 8 octaves and the regex line.

UML Diagram


Conductor
- choirMembers: Map<Note, Choir>
+ af: AudioFormat
- Conductor(Map<Note, Choir>)
+ playTheSong(Song song): void
Choir
- thread: Thread
- turn: boolean
- timeToPlay: boolean
- length: NoteLength
- line: SourceDataLine
- note: Note
+ Choir(Note note)
+ choirStop(): void
+ waitToStop(): void
+ setLine(): void
+ takeTurn(NoteLength length): void
- playNote(): void
+ run(): void
Song
- name: String
- path: String
- notes: List<BellNotes>
+ Song(String name, String path):
+ getNotes(): List<BellNotes>
+ getName(): String name
+ validateSong(): boolean
- readSong(String path): List<BellNotes>
+ uploadSongs(): List<Song>
- parseBellNote(String line): BellNote
- parseNote(String note): Note
- parseNoteLength(String length): NoteLength
JukeBox
- getMembers: Map<Note, Choir>
+ main (String[]): void
# uploadSongs(): void
BellNote
note: Note
length: NoteLength
BellNote(Note, NoteLength):
<<Enumeration>> Note
REST
A0-A8
A0S-A8S
B0-B8
B0S-B8S
C0-C8
C0S-C8S
D0-D7
D0S-D8S
E0-E8
F0-F7
G0-G7
G0S-G7S
- Note():
+ sample(): byte[]
<<Enumeration>> NoteLength
WHOLE
HALF
DOTHALF
QUARTER
DOTQUARTER
TRIPLET
EIGHTH
DOTEIGHTH
SIXTEETNTH
DOTSIXTEENTH
- timsMs: int
- NoteLength(float length):
+ sample(): byte[]

Documentation

My submitted program of Bell Choir meets the assignment requirements as it reads a file that contains a list of Bell Notes and each member of the choir plays one note at a time. Once the note has been assigned to a member, the note is only played by that member of the choir. I have a conductor class that controls the tempo of choir, in other words, they tell the members of choir when to play and for how long. During the playing of the song, only one note is played at once with the correct timing and length given by the user in the file. The program also reads and plays a variety of songs but only if given in the correct format. For example, if a line is read in the file that doesn't contain a note and/or notelength, the song will still play but ignore that line. The project was committed and pushed to GitHub and it plays the default song of "Mary Had a Little Lamb" with appropriate notes and timing. I felt that two seconds per measure suited the majority of the songs that were uploaded to the directory. Therefore, I changed the seconds per measure to two rather than one in my program. The biggest challenge that I encountered writing this lab is ensuring that I took all necessary precautions in terms of validation as there was much to consider during the beginning of the coding process. I am still not 100% confident in my validation and know that is an area where I can improve. In the end, I feel that my program validates properly and helps to invoke better performance throughout my program. Just like Lab 1, Jake Grossé and Jaden Bathon were extremely helpful in assisting me in organizing and laying a strong foundation for this lab. Unlike Juice Bottler, this lab had more moving pieces that needed to be organized in order for the choir to play songs correctly. This lab taught me to consider the unconsiderable and shoot for seamlessness which in the end will not only provide better program operations for me, but the user as well.