Jake Grosse
CS-410: Operating Systems
11 November 2021

Project Description of Meeting Requirements

I believe I met all of the requirements of the lab to the highest degree and with the following I hope you agree with me.

Writing the Code:

- All code which was not provided in class was written entirely by me, Jake Grosse. I theorized and talked about process and operations needed to complete tasks with Jaden and Josiah; however, there was no code exchanged which was used in my project or in either of theirs.
- The code, as evidenced, has been pushed to GitHub and is available to view and clone.

Running the Code:

- The entire program runs with the expected inputs, playing perfectly both Nate's "Mary Had a Little Lamb" file and my own "Imperial March" file, both with proper timing and recognizable tune. More songs may be added if I get bored at 2 AM.
- The program runs by assigning 88 unique notes, one to each of 88 unique threads. The threads are told when they can play and play on their own. They are not forced to ring their bell.
- The program should run with invalid inputs and simply not play some songs while printing error statements pointing the user to the line number. If a song cannot be played (is invalid), the program will inform the user of this and instruct them to fix the file error and restart the program.
- My error handling is so advanced that I only throw one error and the rest are taken care of right where they would ordinarily be thrown. It is implemented within the code that errored files and errored songs will not be able to be played but will instead remind the user to fix the errors and restart the program.
- The included build.xml file utilizes ANT to build and run the java files into a terminal based "Bellbox" selector.

Internal Code Documentation:

- All methods and variables are appropriately and intuitively named.
- Every class (including the enumeration, State) has a Javadoc comment which includes author and description.
- Every public method has a Javadoc comment with description, parameters if applicable, and returns if applicable.
- Every private method has comments within the code explaining the purpose of the method.
- Any areas perceived as obscure or areas which helped me learn have comments within the methods to explain either what is happening, why it needs to happen, or both.

Jake Grosse
CS-410: Operating Systems
11 November 2021
External Design and Documentation:

- This document is fantastically formatted, appealing, and detailed!
- The class diagram is up to UML standards, is all-encompassing, and is visually appealing. (Just look at that UML shadowing!)
- Every piece of the assignment is in the GitHub repository for this lab in one place.

Extra Credit Opportunities:

- I have created an “Imperial March” song file which plays, recognizably and in time, the chorus of “Imperial March” by John Williams. It has more than five unique notes and the file contains more than 15 notes total.
- I have added support for flat notes by interpreting any flat notation and converting it to sharp notation to be used in conjunction with the ordination of the Note enumeration.
- I spent 3 hours (of actual time, not distracted time) working on trying to get harmonies if that counts for anything. (Keep in mind that that amount of time is longer than it took me in total to write Juice Bottler :^]).

Jake Grosse
CS-410: Operating Systems
11 November 2021

Challenges Faced

For this lab, like many projects, my main challenge originated from the process of actually starting the project. I went through many phases of procrastination, dread, and partial loss of a will to live (partially in relation to the project, but mostly just in relation to the workload), before I was able to bring myself to start. Once started, things went quite smoothly. It wasn't until I was beginning input validation that I ran into an issue which stumped me for nearly 10 minutes! In ensuring that a note which was read from the file was indeed contained in the enumeration of notes, I accidentally forgot an exclamation point in front of a boolean operator in a conditional statement which I proceeded to pull my hair out over. It was only when I noticed that this was missing that the program worked as intended. After the challenge regarding a lack of pre-emptive punctuation, my only other challenge consisted of the time spent trying to harmonize bells. I thought it would be fairly straight forward to ring two bells at once, and it is, but the sound library I had been using the entire time doesn't like it when two sounds from two threads try to play onto the same SourceDataLine. It was at the point of discovering this that I gave up as I did not have the desire to delete all of my code and being from scratch by learning a new sound library at 1:00 AM on the day the project was due. For this reason, I did not overcome my last challenge. This could, potentially, be a future project; however, since it would be quite rad to be able to have my own means of creating electronic music.

Jake Grosse
CS-410: Operating Systems
11 November 2021

UML Class Diagram