Nick Clouse
April 4, 2025
CS-410 - Operating Systems - Homework 2

Requirements

Here is how I meet the requirements of the lab:

- **Project must be committed and pushed up to GitHub**

https://github.com/CanFam23/BellChoir

- **Must use ANT to build/run**

I used ant to build and run the program, as well as test it

- **Each Member must play each assigned note in a separate thread**

Each member has it's own separate thread which it uses to play its note

- **The assignment must be able to play the instructor provided song 'Mary Had a Little Lamb' with the sound output being properly recognizable with appropriate timing.**

Run `ant run -Dsong=MaryLamb.txt` to verify this

- **Student provided songs may be provided as additional song files to other students.**

My problem should handle these songs properly

- **Improper song files will be provided during the final instructor demonstration to determine how well the program behaves when given invalid data.**

Should handle these too

Challenges Faced

- **Making my own song**

I know nothing (nothing) about music and notes, so this took a lot of research and trial and error. Additionally, the song I chose (Never gonna give you up) was pretty complicated when using notes to create it, so that added some fun to it. I ended up finding a website (Last link in sources) that had the song letter notes for my song, which made it a little easier to convert into the format I needed.

- **Synchronization**

My first attempt at this project was a lot more complex, and brought a plethora of its own issues, such as thread execution order and members starting before the conductor was ready. This solution had the synchronization down, but the conductor wasn't actually controlling the members. I scratched that and restarted, and thought for a while about how to synchronize the threads in a simple way. I ended up pretty much using the exact code from the provided Player class, which made my program so, so much more simple and efficient.

- **Figuring out how to stop threads**

If I just set running to false, the threads waiting for their turn would still be waiting, and would therefore still be running. To solve this problem, I interrupt the threads when the stop method is called, which causes them to return out of their run method, effectively and efficiently terminating the thread.

- **JUnit with ant**

Since Im still learning ant, this took some research and trial/error too. I originally stumbled upon the Ant doc for the JUnit tag, and was trying to figure out how to use it for hours before realizing that it only works with JUnit 4, and JUnit 5 uses the Junitlauncher tag. Once I realized this, it was relatively easy to set it up, and send the system err/out messages to a separate file. Another challenge with this was deciding what targets should depend on who. Since the test class needs the library downloaded to install, I had to add a target to ensure the libraries are downloaded and installed correctly before compiling the project.

- **Program time spent playing the song**

I added a check in the run method of the conductor class to ensure the program isn't running for longer then it's supposed to. In the event that it does, the program is terminated. I don't think this could happen often, but I thought it would be good to add incase threads start waiting/running/etc for too long. I think it could be improved (If a giveTurn call somehow causes the program to wait forever, technically the program would never terminate. The check is good for when threads start taking longer, but still are completing their tasks.), or add a way to ensure threads don't wait for too long or never get to play, but I decided to leave that for a later time.

Sources

https://ant.apache.org/manual/Tasks/junitlauncher.html

https://junit.org/junit5/docs/5.0.0-M5/user-guide/

https://github.com/junit-team/junit4/wiki/Getting-started-%E2%80%93-Ant

https://noobnotes.net/never-gonna-give-rick-astley/?solfege=false