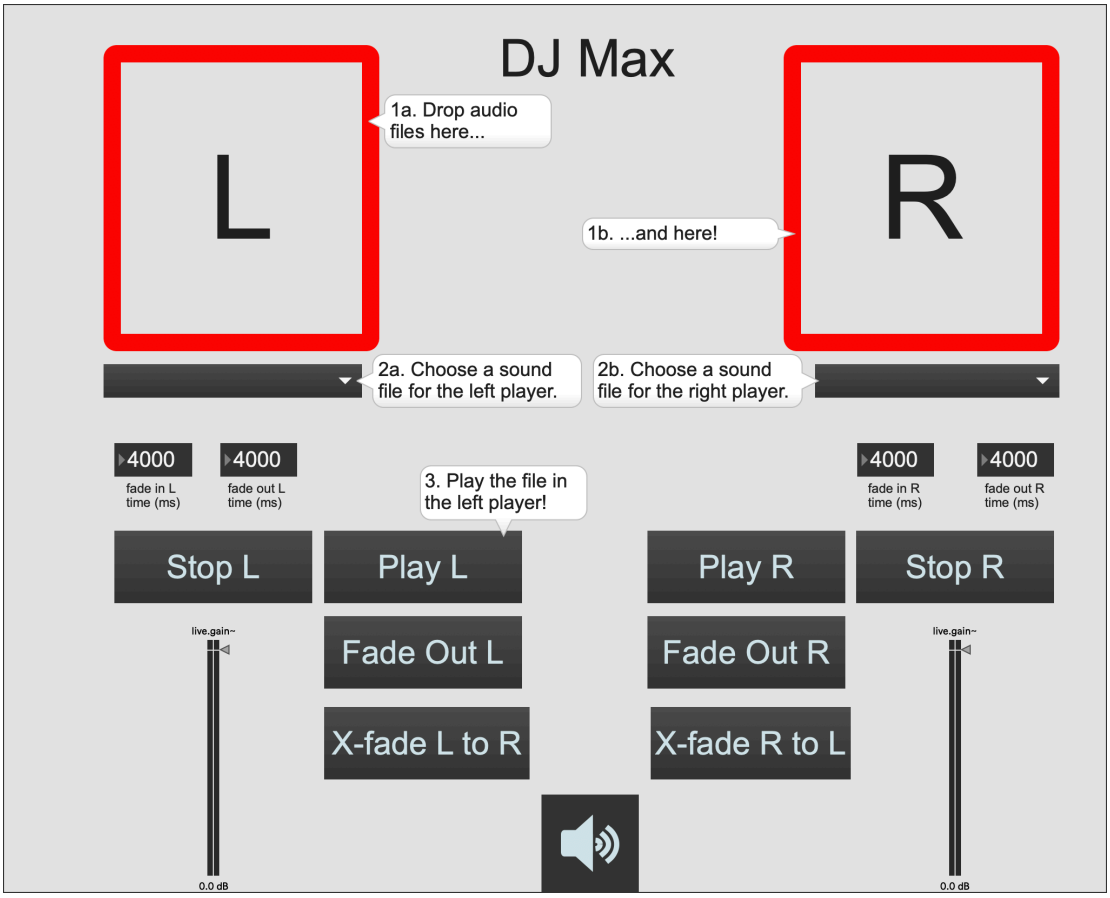
# DJ Max Assignment

**Description**
The goal of this project is to create a usable Max DJ machine which can alternate playing multiple audio files between two players through the use of cross-fading.




**Required Objects (Parent Patcher):**
- 2 dropfile objects
- 2 umenu objects
- recommended: sprintf object(s) for concatenating messages
- 2 polybuffer~ objects (one named “SoundL”, the other “SoundR”)
- 2 strippath objects
- textbuttons (for playing, stopping, x-fading, etc.)
- message objects
- prepend objects
- number objects (for fadein/fadeout times from user)
- 2 live.gain~ objects
- 1 ezdac~
- various arithmetic objects as needed (hint: +)

**Required Objects (AudioPlayer abstraction):**

- 1 inlet object
- 1 route object including the following arguments: <buffer>, <start>, <stop>, <intime>, <fadeout> and <outtime>
- message objects (start, stop, etc.)
- recommended: sprintf object(s) for packing messages
- 1 play~ object (name argument is "#1" and 2 channels
- 1 line~ object (for fading in or out)
- recommended: number~ (for testing the line~)
- *~ objects for multiplying audio signals as necessary
- 2 outlet objects (one for each channel of audio)

**Hints:**

- I would recommend beginning with your AudioPlayer patch. Test every feature **as you go**. Start with the basics: play~, an ezdac~, start and stop messages. Add a simple buffer~ for testing with a single sound file. Then, try adding the fadein and fadeout feature. Then, add your route infrastructure at the top of the patch and test it with messages.
- Once the audio player is working as a patch, convert it to an abstraction:
  - Replace the name of the buffer in the play~ object with the abstraction argument variable #1.
  - Add your inlet and two outlets and remove the buffer~ and ezdac~ that you used for testing.
- Then, I would build the parent patch.
- You'll need two copies of your AudioPlayer abstraction. This file should be located in the same directory as your "DJ Max.maxpat" parent patcher file.
- Build the dropfile, polybuffer~ and umenu infrastructure for one of the player and connect it to your AudioPlayer using the appropriate messages for the route system.
- Your AudioPlayer's name argument should be SoundL or SoundR depending on which one you're working on (in other words, it should match the corresponding polybuffer~'s name)
- The outlets from your AudioPlayer abstraction should connect to the single ezdac~ in your parent patch.
- Once you have this all working for one AudioPlayer, repeat with the other side of your patch.
- Next, add the text buttons which will trigger messages for playback, stopping, fading out, or x-fading L to R (or vice versa). Think about how to implement this! Which player should be fading out? Which should be fading in?
- Finally, create a nice presentation mode for your patch. Remember, if you resize objects in pres. mode, they will be resized in patching mode, too. Keep your patch looking clean and sleek in both modes. (Route all patch cables neatly; no spider webs!) … Be sure to add instructions for the user (you can use mine as a template).
- Lastly, make sure that your patch will open in presentation mode (cmd+shift+i, "open in presentation mode").