OndeSynth

concurrency requirements

There are four threads:

Thread[**Java Sound Event Dispatcher**,10,main] – shouldn't interfere

Thread[**Java Sound MidiInDevice** Thread,5,main] –
calls **recv** (defined in OndeSynth.**listen**() → midiListener.**routeMidiMessage**()

Thread[**OndesSynth – MidiListenerThread**,1,main]
→ calls **synth**.routeMidiMessage() - **noteON**() **noteOFF**() **sendChannelMessage**()

Thread[**OndeSynth - main** thread,5,main]

synchronized**(lock) – used in two places:**

- **run()** - the below analysis
- **routeMidiMessage() - see page 3**

Potential concurrency in OndeSynth.**run**()

WiredIntSupplierPool	**wires**	(private) *synchronize*: getWiredInSupplier(IntSupplier) reset()
Instant AnharmonicWaveGen	**clocks**	(private) *synchronize:* addPhaseClock() delPhaseClock() next()
MonoComponent	**inputs**	(protected) 10 usages, 8 of which can be replaced by a **sum**() function
MonoComponent	**namedInputs**	replace with inputSum("name")
ChannelVoicePool	**inUse** **available**	(both private) it should be possible to synchronize all modifications within ChannelVoicePool; Is a function-level lock OK, or create separate lock? *used in* c'tor - no synch necessary **getVoice**() - available.pop(), inUse.push() **release**() inUse.remove() available.pop()

VoiceTracker	**channelPlaying** = VoiceSet[16]	(private) VoiceSet is a 'typedef' for HashMap<Voice> *synchronize* **forEach**() - used in resetWires – potential deadlock. **addVoice**() **delVoice**() **getChannelPlaying**() - used in sendChannelMessage()
OndeSynth	**endedNoteQueue**	(private) used in queueNoteEnd() add a flushNoteEnds() function calls OndeSynth.**noteEnded**(chan,note) → ChannelVoicePool.releaseVoice() which calls **voice.pause**(), **inUse**.remove and **available**.add(). → VoiceTracker.delVoice() which removes from **channelPlaying**[chan] **voice.pause**() → calls MonoComponent.pause() for each component. * The filters use them to reset. * AnharmonicWaveGen deletes phase clocks from Instant. * WaveGen deletes the one phase clock from Instant → and… synth.getMainOutput().delInput(voiceMix.getMainOutput())
OndesSynth	**resetWires()**	voiceTracker.**forEach**(Voice::**resetWires**) → wiredInputSupplierPool.**reset**(); possible deadlock?
OndeSynth	**sendChannelMessage()**	→ Voice.**processMidiMessage**() these will never be note-ONs or offs, though a sustain pedal UP may eventually result in an envelope termination. → ChannelVoicePool.**updateState**()

ChannelState	**controllers** **afterKeys**	(private) Both HashMaps<Integer,Integer>() *synchronize* the three methods: * **getMessages**() * **reset**() * **update**()

Potential Concurrency issues in
OndeSynth.**routeMidiMessage**() (**Bold** comments copied from below)

noteOFF()		**- TODO - verify, but this should only affect envelopes** **- TODO - call queueNoteEnd() instead**
noteON()		**- TODO - check all MonoComponent.noteON() methods** **- TODO - note usage of available, inUse** **- TODO - be sure resume is synchronized with MonoMainMix.update()**
sendChannelMessage()		- **TODO - be sure it's safe** **- TODO - controllers, afterKeys cf. run() call tree**

Voice	**pause()** **resume()**	Both of these touch synth.getMainOutput() They connect the Junction voiceMix to it. Doesn't happen in the configure step: not until '**resume**()'

OndeSynth.noteOFF(MidiMessage) (ondes.synth)
VoiceTracker in OndeSynth.getVoice(int, int) (ondes.synth)
- reads from array location, so no issue

Voice.processMidiMessage(MidiMessage) (ondes.synth.voice)
- Voice.midiListeners - should be final, so no issue

MonoComponent.noteOFF(MidiMessage) (ondes.synth.component)
- TODO - verify, but this should only affect envelopes

OndeSynth.noteEnded(MidiMessage) (ondes.synth)
- TODO - call queueNoteEnd() instead

OndeSynth.noteON(MidiMessage) (ondes.synth)
VoiceTracker in OndeSynth.getVoice(int, int) (ondes.synth)
- reads from array, no issue
Voice.processMidiMessage(MidiMessage)(2 usages) (ondes.synth.voice)
MonoComponent.noteON(MidiMessage) (ondes.synth.component)
- calls noteON for a list of components, so hopefully no issue
TODO - check all MonoComponent.noteON() methods
ChannelVoicePool.getVoice() (ondes.synth.voice)
- TODO - note usage of available, inUse
VoiceMaker.getVoice(String, OndeSynth) (ondes.synth.voice)
- this call should be rare. only if none are available.
- calls new Voice(map, synth)... note that this only
configures the voice. new Voice() doesn't connect it.
That happens in resume();
ArrayDeque.push(E) (java.util)
- inUse (documented above)
ChannelState.getMessages() (ondes.synth.voice)
Voice.processMidiMessage(MidiMessage) (ondes.synth.voice)
Iterable.forEach(Consumer<? super T>) (java.lang)
- these align the voice with the channel state by
sending all the current state as MIDI messages
(e.g. sustain down or up)
Voice.resume() (ondes.synth.voice)
TODO - be sure resume is synchronized with
MonoMainMix.update()
VoiceTracker in OndeSynth.addVoice(Voice, int, int) (ondes.synth)
HashSet.add(E) (java.util)
- adds to channelPlaying (documented above)

OndeSynth.sendChannelMessage(MidiMessage) (ondes.synth)
VoiceTracker in OndeSynth.getChannelPlaying(int) (ondes.synth)
- accesses channelPlaying
- **TODO - be sure it's safe**
Voice.processMidiMessage(MidiMessage) (ondes.synth.voice)
- should be OK (see above under noteOFF)
- it's setting the controls for the static components, so no issue
ChannelVoicePool.updateState(MidiMessage) (ondes.synth.voice)
- calls ChannelState.update()
- TODO - controllers, afterKeys cf. run() call tree