

---

# Java Tick 6\*

A graphical display is just one method of communicating the evolution of a pattern in a Game of Life. In this tick you will experiment with audible representations of the game world and write a program which can produce a WAV audio file which represents the evolution of the game. Your workstation has a headphone port on the front. If you don't have any headphones please ask to borrow a pair.

A simple library for writing audio files has been provided to get you started. You should download this from the course website at <http://www.cl.cam.ac.uk/teaching/current/ProgJava/sounds.jar>. The Javadoc for this library is available at: <http://www.cl.cam.ac.uk/teaching/current/ProgJava/sounddocs>. It's a good idea to start by reading the documentation of the `AudioSequence` class.

There are three possible ways to get your tick:

- Invent an algorithm for choosing the pitch of notes based on the world state and use the `sound.jar` library to generate your tune. An example algorithm might be to iterate over the world and for each live cell add a new `SineWaveSound` to the `AudioSequence`. You can choose the pitch for your `SineWaveSound` using your own metrics; for example you might start by using the distance of the live cell from the centre of the board. Other variables to consider turning into sound might be the total population of the world combined in some way with the generation number. Feel free to experiment and see what, in your opinion, works best.
- Provide your own implementation of the `Sound` interface which can produce different sounds. You might like to experiment with saw-tooth shapes, or load in samples such as those available at <http://ibeat.org>).
- Dispense with the library altogether and write your own program. If you take this approach you must write your program in Java, and your program should output the audio representation of the Game of Life as a WAV file.

To gain the tick your submission must produce an audible output which is representative of the world state in some way. Your implementation should include a main method inside a class called `SoundLife` inside the package `uk.ac.cam.crsid.tick6star`. Your implementation of `SoundLife` should support at least three parameters on the command line:

- A string describing the state of the world in the Game of Life in the format we've used in this course
- The number of generations to play
- The name of the output WAV file

You should submit your favorite "tune" in a WAV file called `competition.wav` as part of your submission. You should also include a text file called `notes.txt` which contains instructions written with a sufficient level of detail to enable a competent programmer to re-implement your program. The best audible rendering of a Game of Life will be displayed on the course website next week. Once you are happy that your program works correctly, construct a jar file called `crsid-tick6star.jar` containing your code and email it as an attachment to `ticks1a-java@cl.cam.ac.uk`. Your jar file should contain at least:

```
META-INF/  
META-INF/MANIFEST.MF  
uk/ac/cam/crsid/tick6star/notes.txt  
uk/ac/cam/crsid/tick6star/competition.wav  
uk/ac/cam/crsid/tick6star/SoundLife.java  
uk/ac/cam/crsid/tick6star/SoundLife.class
```

You do not need to include any of the contents of the library `sounds.jar` in your submission but any other required classes should be included. Your jar file should have the entry point set so that the following renders six generations of a Glider pattern to the file `competition.wav`:

```
crsid@machine:~> java -jar crsid-tick6star.jar \  
"Glider:Richard Guy (1970):20:20:1:1:010 001 111" 6 competition.wav
```

---