1 The Microchip Factory

This is an example enterprise gen_server-based application that simulates well-audited microchip production.

1.1 Project structure

- `lib/`
  - `microchip_factory.ex` — script executing the experiment
  - `microchip_factory/`
    - `application.ex` — Elixir app and supervisor
    - `producer.ex` — the producer generic server
    - `inspector` — the inspector generic server
- `src/` — DDMon library

1.2 Services

Both services are implemented as simple generic servers.

1.2.1 Producer

`Producer` produces objects (representing microchips) according to some metadata which it was provided with during initialisation, and possibly other microchip components coming from other producers. Before a microchip is returned to the caller, it must be first inspected by an `Inspector` — only if it passes its meticulous audit, it can be sent back.

Aside from producing microchip, `Producer` may also provide the metadata of its produce.

1.2.2 Inspector

`Inspector` checks compliance of microchips against metadata of a reference microchip producer. In order to perform the audit, the inspector asks its reference producer for metadata and replies `ok` when certain conditions are met (in our scenario they are always met). Different inspectors may use different producers for their references.

1.3 Test cases

1.3.1 Two producers, two inspectors

Run the test:

```
mix run -e "MicrochipFactory.start_two"
```

The file `lib/microchip_factory.ex` describes a case of two producers and two inspectors. `Producer` 1 asks `Inspector` 1 for audits; similarly, `Producer` 2 asks `Inspector` 2. However, `Inspector` 1 uses `Producer` **2** as its reference; analogously, `Inspector` 2 refers to `Producer` 1 in its audits.

The test begins by requesting both producers to produce microchip. This, depending on the order of events, may result in either successful creation of two vegetables, or a deadlock.

1.3.2 Many producers, three inspectors

Run the test:

```
mix run -e "MicrochipFactory.start_many"
```

This case is different from the one above in that it has over 90 producers which call each other to obtain necessary microchip components. The system occasionally runs into deadlocks, but this time the deadlocks are of variable sizes.

Producers are divided into 3 categories: `:a`, `:b` and `:c`. Within each category, there are producers indexed from `0` to `30`. `n`-th producer (except the 30th) calls the `n+1`-th in order to produce its result. All producers in the category `:a` use an inspector whose reference is one of the producers in category `:b`. Similarly for `:b` and `:c`, as well as `:c` and `:a`.