GenServer - a cheat sheet

last version: https://elixir-lang.org/getting-started/mix-otp/cheat-sheet.pdf
reference: https://hexdocs.pm/elixir/GenServer.html

initialization: .start → `init/1`

client
```
def start_link(opts \\ []) do
  GenServer.start_link(__MODULE__, match_this, opts)
end
```

returns
```
{:ok, pid}
```

callback
```
def init(match_this) do
  # process input and compute result
  result
end
```

^result =
```
{:ok, state}
{:ok, state, then_what}

{:stop, reason}
:ignore!
```

^reason =

This applies wherever `result` matches `{:stop, reason}`, and to `reason` argument in `stop` message.

```
:normal
:shutdown
{:shutdown, any}
any
```

termination: .stop → `terminate/2`

client
```
def stop(pid, reason \\ :normal,
              timeout \\ :infinity) do
  GenServer.stop(pid, reason, timeout)
end
```

returns
```
:ok
```

callback
```
def terminate(reason, state) do
  # perform cleanup
  # result will not be used
end
```

asynchronous operation: .cast → `handle_cast/2`

client
```
def your_api_async_op(pid, args) do
  GenServer.cast(pid, match_this)
end
```

returns
```
:ok
```

callback
```
def handle_cast(match_this, state) do
  # process input and compute result
  result
end
```

^result =
```
{:noreply, state}
{:noreply, state, then_what}

{:stop, reason, state}
```

^then_what =

This applies wherever `result` matches `{_, state, then_what}`

```
milliseconds
:hibernate
{:continue, value}
```

synchronous operation: .call → **handle_call/3**

client
```
def your_api_sync_op(pid, args) do
  GenServer.call(pid, match_this)
end
```

callback
```
def handle_call(match_this, from, state) do
  # process input and compute result
  result
end
```

^result =
```
{:reply, reply, state}
{:reply, reply, state, then_what}
```
```
{:noreply, state}
{:noreply, state, then_what}
```
```
{:stop, reason, reply, state}
```

returns
waits for callback, receives `reply` if result matches `{:reply, reply, ...}` or `{:stop, _, reply, _}`.

^reply =
user defined

handling messages: → **handle_info/2**

client
```
def handle_info(match_this, state) do
  # process input and compute result
  result
end
```

^result =
```
{:noreply, state}
{:noreply, state, then_what}
```
```
{:stop, reason, state}
```

{**:continue, match_this**} → **handle_continue/2**

client
```
def handle_continue(match_this, state) do
  # process input and compute result
  result
end
```

^result =
```
{:noreply, state}
{:noreply, state, then_what}
```
```
{:stop, reason, state}
```

@impl true
Functions where `@impl true` applies.
```
init/1
terminate/2
handle_call/3
handle_cast/2
handle_info/2
handle_continue/2
code_change/3
format_status/2
```