initialization

client

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

immediate

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

callback

```
def init(match_this) do
 # compute state or fail; with or without reason
 result
end
```

^result =

```
{:ok, state}
{:ok, state, 5_000}
{:ok, state, :hibernate}
```

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

^reason =

This applies to all callback replies when `result` matches `{:stop, reason}`, and to `reason` argument in `stop` message.

```
:normal
:shutdown
{:shutdown, term}
term
```

termination

client

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

immediate

```
:ok
```

callback

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

synchronous operation - **`call`**

client

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

immediate

waits for callback and returns `reply` if returned value from callback matches `{:reply, reply, _}`

callback

```
def handle_call(match_this, from, state) do
 # compute reply, state or fail; with or without reason
 result
end
```

^result =

```
{:reply, reply, new_state}
{:reply, reply, new_state, 5_000}
{:reply, reply, new_state, :hibernate}
```

```
{:noreply, new_state}
{:noreply, new_state, 5_000}
{:noreply, new_state, :hibernate}
```

```
{:stop, reason}
```

asynchronous operation - **cast**

client

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

immediate

```
:ok
```

callback

```
def handle_cast(match_this, state) do
 # compute new state or fail to do so
 result
end
```

^result =

```
{:noreply, new_state}
{:noreply, new_state, 5_000}
{:noreply, new_state, :hibernate}
```

```
{:stop, reason}
```

handling messages - **info**

client

```
def handle_info(msg, state) do
 # compute new state or fail to do so
 result
end
```

^result =

```
{:noreply, new_state}
{:noreply, new_state, 5_000}
{:noreply, new_state, :hibernate}
```

```
{:stop, reason}
```

other

I defined a module with one `abc` function, preceded by the `@impl true` directive, as to force the compiler to give me the list of callback functions I could correctly define.
This first block we have handled.

* `GenServer.init/1 (function)`
* `GenServer.terminate/2 (function)`
* `GenServer.handle_call/3 (function)`
* `GenServer.handle_cast/2 (function)`
* `GenServer.handle_info/2 (function)`

Who can fill in the blanks for the following three functions?

* `GenServer.code_change/3 (function)`
* `GenServer.format_status/2 (function)`
* `GenServer.handle_continue/2 (function)`