initialization

client

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

immediate

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

callback

```
def init(:ok) do
 # initialize state, or fail to do so
 return_value
end
```

^return_value =

```
{:ok, state}
{:ok, state, 5_000}
{:ok, state, :hibernate}
{:stop, reason}
:ignore!
```

^reason =

This applies to all callback replies when `return_value` 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
 # no return value expected
end
```

synchronous operation

client

```
def sync_op(pid, args) do
 GenServer.call(pid, {:sync_op, args})
end
```

immediate

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

callback

```
def handle_call({:sync_op, args}, from, state) do
 new_state = f(state, args)
 return_value
end
```

^return_value =

```
{: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 , reply, new_state}
{:stop, reason , new_state}
```

asynchronous operation

client

```
def sync_op(pid, args) do
 GenServer.cast(pid, {:async_op, args})
end
```

immediate

```
:ok
```

callback

```
def handle_cast({:async_op, args}, state) do
 # compute new state or fail to do so
 return_value
end
```

^return_value =

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

{:stop, reason , new_state}
```

handling information messages

client

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

^return_value =

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

{:stop, reason , new_state}
```

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)
```