```elixir
defmodule NameServer6 do
  @moduledoc """
  Two ways to stop the server
  """
  use GenServer

  def start(name) do…
  end

  def add(serverName, name, place) do…
  end

  def find(serverName, name) do…
  end

  def shutdown(serverName) do
    GenServer.cast(serverName, :shutdown)
  end

  def shutdown1(serverName) do
    GenServer.stop(serverName, :normal)
  end

  @impl true
  def init(_) do…
  end

  @impl true
  def terminate(_reason, _state) do
    # Cleanup code
  end

  @impl true
  def handle_cast({:add, name, place}, state) do…
  end

  def handle_cast(:shutdown, state) do
    # do something
    {:stop, :normal, state}
  end

  @impl true
  def handle_call({:find, name}, _from, state) do…
  end
end
```

Interface

Implementation

Called when the server is terminating because of:

- Callback returns {:stop, reason, … }
- Callback raises an exception.
- Callback returns an invalid value.
- GenServer.stop(…) is called.
- …

GenServer Callbacks

- **init(args) (interface - start(), start_link())**
  - {:ok, state *[,timeout]*}
  - {:stop, reason}
- **handle_call(request, from, state) (interface - call())**
  - {:reply, reply, new_state *[, timeout]*}
  - {:stop, reason *[,reply]*, new_state}
- **handle_cast(request, state) (interface - cast())**
  - {:noreply, new_state *[, timeout]*}
  - {:stop, reason, new_state}
- **handle_info(msg, state) (async - message or timeout)**
  - msg = term or :timeout
  - {:noreply, new_state *[,timeout]*}
  - {:stop, reason, new_state}
- **terminate(reason, state) (Callback return = {:stop …..} or interface stop call)**