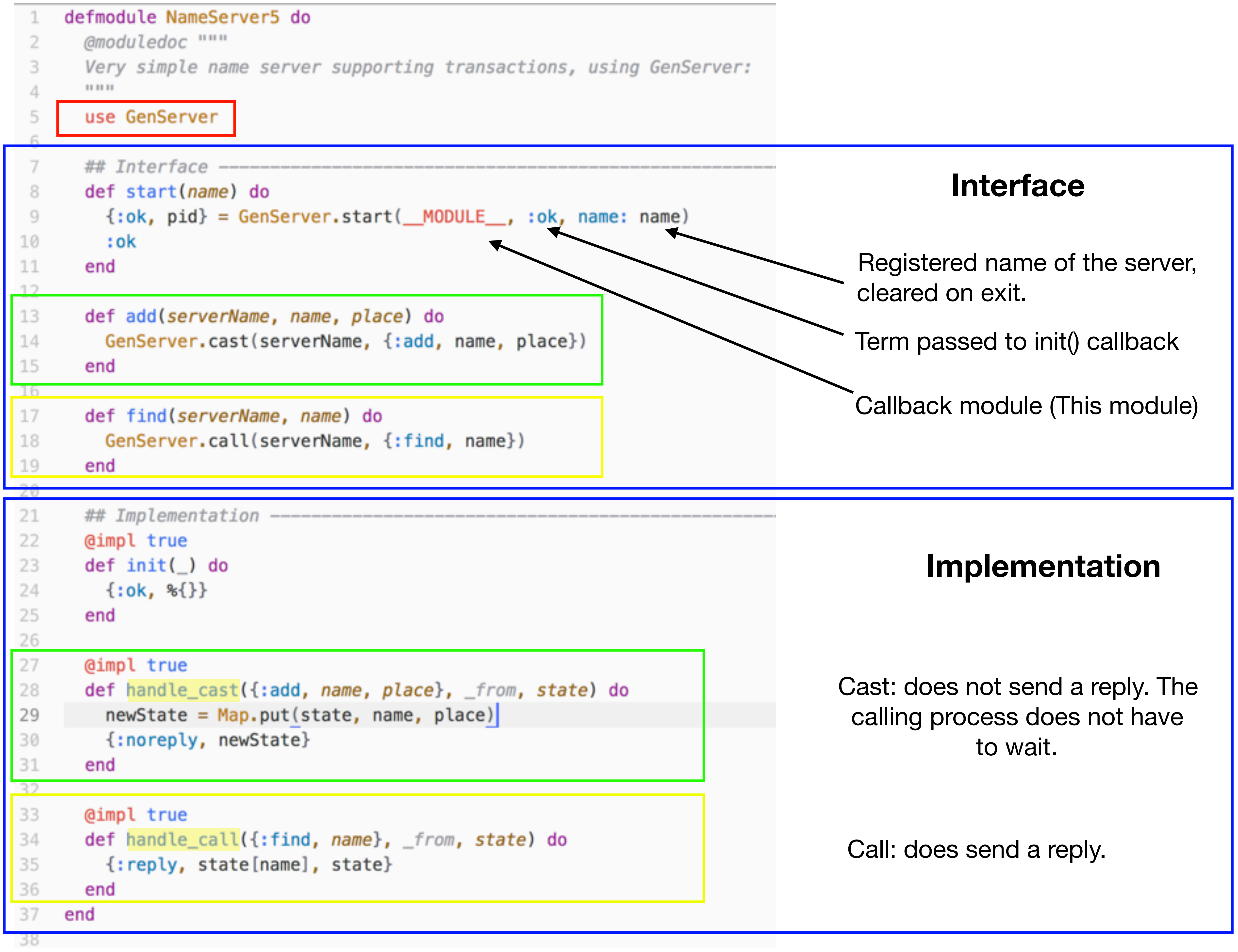

```
defmodule NameServer5 do
  @moduledoc """
  Very simple name server supporting transactions, using GenServer:
  """
  use GenServer

  ## Interface ---------------------------------------------------
  def start(name) do
    {:ok, pid} = GenServer.start(__MODULE__, :ok, name: name)
    :ok
  end

  def add(serverName, name, place) do
    GenServer.cast(serverName, {:add, name, place})
  end

  def find(serverName, name) do
    GenServer.call(serverName, {:find, name})
  end

  ## Implementation ----------------------------------------------
  @impl true
  def init(_) do
    {:ok, %{}}
  end

  @impl true
  def handle_cast({:add, name, place}, _from, state) do
    newState = Map.put(state, name, place)
    {:noreply, newState}
  end

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

Interface

Registered name of the server, cleared on exit.

Term passed to init() callback

Callback module (This module)

Implementation

Cast: does not send a reply. The calling process does not have to wait.

Call: does send a reply.

Nameserver5 Test

- Nothing Changed using GenServer

```elixir
defmodule NameServer3Test do
  use ExUnit.Case
  doctest NameServer3

  test "1 - start the server" do
    assert NameServer3.start(:my_server3) == :ok
    assert NameServer3.add(:my_server3, :dwayne, "Red Dwarf") == :ok
    assert NameServer3.find(:my_server3, :dwayne) == "Red Dwarf"
  end
end
```

```elixir
defmodule NameServer5Test do
  use ExUnit.Case
  doctest NameServer5

  test "1 - start the server" do
    assert NameServer5.start(:my_server5) == :ok
    assert NameServer5.add(:my_server5, :dwayne, "Red Dwarf") == :ok
    assert NameServer5.find(:my_server5, :dwayne) == "Red Dwarf"
  end
end
```