```elixir
#Modulo que permite crear un actor observador.
defmodule ObservableActor do
    #Permite iniciar un actor observador, la función start recibe un parametro el cual representa
    # el objeto que tendra el actor observador.
    def start(object) do
        spawn(__MODULE__, :reference, [object,true])
    end
    #Función que es evaluada por el actor observador, se encarga de recibir las peticiones de otros
    #actores, de tipo shared y exclusive. También se encarga de llevar el estado (Objeto y si esta
    #bloqueado o no).
    def reference(state, enable) do
        receive do
            #Caso en que el dominio del mensaje es shared.
            {:shared, sender, func, ref} ->
                if(enable) do
                    send sender, {func,get_domain_reference(state),ref}
                    reference(state, true)
                else
                    send sender, {:not_available,ref}
                    reference(state, false)
                end
            #Caso en que el dominio del mensaje es exclusive.
            {:exclusive, sender, func, ref} ->
                if(enable) do
                    #get_domain_reference envia la referencia en memoria del objeto que entra por
                    #parámetro.
                    send sender, {func,get_domain_reference(state), ref}
                    #Se bloquea al observador esperando que el sender termine de ejecutar el proceso
                    reference(state, false)
                else
                    send sender, {:not_available, ref}
                    reference(state, false)
                end
            #Caso en el que el sender notifica al observador que termino de ejecutar el proceso
            {:exclusive, :ok, ref} ->
                reference(state, true)
        end
    end
end
#Modulo que representa como un actor puede hacer una petición de la referencia al actor observador
defmodule RequestActor do
    #Función que le permite hacer un request al actor observador y desbloquearlo en caso de ser
    #necesario, la funcion recibe el observador
    def request(obs_actor,function, domain) do
        ref = make_ref()
        send obs_actor, {domain, self(), func, ref}
        receive do
            #Caso en que el actor observador envia la referencia.
            {^function, object_reference ,^ref} ->
                #get_object_by_reference es una función que trae el objeto que esta en la
                #referencia que entra por memoria.
                resp = function.(get_object_by_reference(object_reference))
                if(domain == :exclusive) do
                    send obs_actor, {domain, :ok, ref}
                end
                resp
            #Caso en que el actor observador estaba ocupado.
            {:not_available, ^ref} ->
                request(obs_actor,function, domain)
        end
    end
end
#Instrucciones para correr el programa.
observador=ObservableActor.start(%{:hydrogen => 1008, :carbon => 12.011, :sodium => 22.99})
spawn(fn -> RequestActor.request(observador, fn(object) -> Map.put_new(object, :neon, 20.17) end, :exclusive) end)
spawn(fn -> RequestActor.request(observador, fn(object) -> Map.put_new(object, :helium, 27) end, :shared) end)
spawn(fn -> RequestActor.request(observador, fn(object) -> Map.put_new(object, :lithium, 28) end, :shared) end)
```