



Memento

Padrões de projeto #7 - Live de Python # 255



1. Introdução lúdica

Fausto vs Diabão!

2. O padrão como padrão

a visão clássica e o UML

3. Implementações possíveis

Mutabilidade, consumo de memória, etc.

4. Pontos de atenção

Mutabilidade, consumo de memória, etc.



apoia.se/livedepython



pix.dunossauro@gmail.com



patreon.com/dunossauro



Ajude o projeto <3



Ademar Peixoto, Adilson Herculano, Alemão, Alexandre Harano, Alexandre Lima, Alexandre Takahashi, Alexandre Villares, Alfredo Braga, Alisson Souza, Alysso Oliveira, Andre Azevedo, Andre Mesquita, Andre Paula, Aniltair Filho, Antônio Filho, Arnaldo Turque, Aslay Clevisson, Aurelio Costa, Bárbara Grillo, Ber_dev_2099, Bernardo Fontes, Bruno Almeida, Bruno Barcellos, Bruno Batista, Bruno Freitas, Bruno Ramos, Caio Nascimento, Carlos Ramos, Cristian Firmino, Daniel Bianchi, Daniel Freitas, Daniel Real, Daniel Wojcickoski, Danilo Boas, Danilo Silva, David Couto, David Kwast, Davi Souza, Dead Milkman, Denis Bernardo, Diego Guimarães, Dino, Edgar, Edilson Ribeiro, Emerson Rafael, Ennio Ferreira, Erick Andrade, Érico Andrei, Everton Silva, Fabio Barros, Fábio Barros, Fabio Valente, Fabricio Biazotto, Felipe Augusto, Felipe Rodrigues, Fernanda Prado, Fernando Celmer, Flávio Meira, Francisco Silvério, Frederico Damian, Gabriel Espindola, Gabriel Mizuno, Gabriel Moreira, Gabriel Paiva, Gabriel Ramos, Gabriel Simonetto, Giovanna Teodoro, Giuliano Silva, Guilherme Beira, Guilherme Felitti, Guilherme Gall, Guilherme Ostrock, Guilherme Piccioni, Guilherme Silva, Gustavo Suto, Haelmo Almeida, Harold Gautschi, Heitor Fernandes, Helvio Rezende, Henrique Andrade, Hiago Couto, Higor Monteiro, Igor Taconi, Janael Pinheiro, Jean Victor, Jefferson Antunes, Joelson Sartori, Jonatas Leon, Jônatas Oliveira, Jônatas Silva, Jorge Silva, Jose Barroso, José Gomes, Joséito Júnior, Jose Mazolini, Josir Gomes, Juan Felipe, Juliana Machado, Julio Batista-silva, Julio Franco, Júlio Gazeta, Júlio Sanchez, Kaio Peixoto, Leandro Silva, Leandro Vieira, Leonardo Mello, Leonardo Nazareth, Lucas Carderelli, Lucas Lattari, Lucas Mello, Lucas Mendes, Lucas Nascimento, Lucas Schneider, Lucas Simon, Luciano Filho, Luciano Ratamero, Luciano Teixeira, Luis Eduardo, Luiz Carlos, Luiz Duarte, Luiz Lima, Luiz Paula, Mackilem Laan, Marcelo Araujo, Marcio Moises, Marcio Silva, Marco Mello, Marcos Gomes, Marina Passos, Mateus Lisboa, Matheus Ferreira, Matheus Silva, Matheus Vian, Mírian Batista, Mlevi Lsantos, Murilo Carvalho, Murilo Viana, Nathan Branco, Ocimar Zolin, Otávio Carneiro, Pedro Gomes, Pedro Henrique, Pedro Pereira, Peterson Santos, Rafael Araújo, Rafael Faccio, Rafael Lopes, Rafael Romão, Rafael Silva, Raimundo Ramos, Ramayana Menezes, Renato José, Renato Moraes, Renato Oliveira, Renê Barbosa, Rene Pessoto, Renne Rocha, Ricardo Combat, Ricardo Silva, Rinaldo Magalhaes, Riverfount, Rjribeiro, Rodrigo Barretos, Rodrigo Oliveira, Rodrigo Quiles, Rodrigo Santana, Rodrigo Vaccari, Rodrigo Vieira, Rogério Nogueira, Rui Jr, Samanta Cicilia, Samuel Santos, Santhiago Cristiano, Selmison Miranda, Téó Calvo, Thiago Araujo, Thiago Borges, Thiago Curvelo, Tony Dias, Tyrone Damasceno, Valdir, Valdir Tegon, Vinícius Costa, Vinicius Stein, Washington Teixeira, Willian Lopes, Wilson Duarte, Zeca Figueiredo




Obrigado você





Disclaimer






Padrões de projeto

Eduardo Mendes

Pública


10 vídeos 7.936 visualizações Atualizado hoje



▶ Reproduzir tu...


↻ Ordem aleató...

≡ Ordenar




Então você quer entender padrões de projeto? Nova série da live de Python

Eduardo Mendes • 5 mil visualizações • há 4 anos




Live de Python #115 - Introdução a padrões de projeto (parte 1)

Eduardo Mendes • 9,4 mil visualizações • Transmitido há 4 anos



Live de Python #115 - Introdução a padrões de projeto (parte 2)

Eduardo Mendes • 3,4 mil visualizações • Transmitido há 4 anos



Live de Python #116 - Observer / PubSub / Dispatcher - Padrões de projeto #02

Eduardo Mendes • 4,5 mil visualizações • Transmitido há 4 anos

<https://www.youtube.com/playlist?list=PLOQgLBuj2-3lPHFIBmqhtbM4vLJg9tob4>

Uma visão lúdica

Fausto vs Diabão

Um pouco de contexto sobre nossa história



Fausto



Diabão



Maldição



Fausto

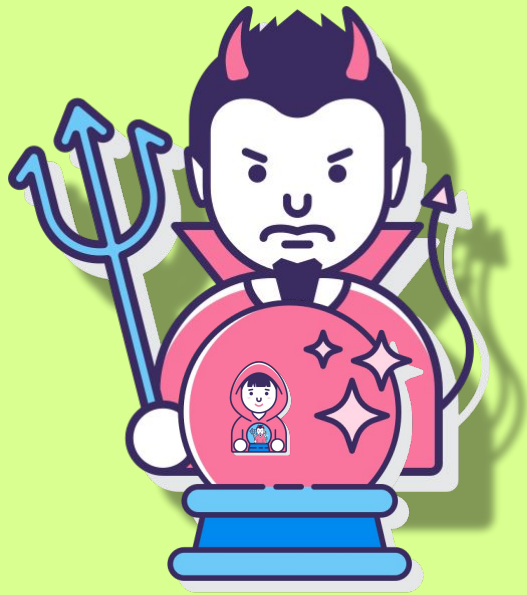
Fausto é um aprendiz de bruxos.

Um dos seus passatempos é descobrir onde objetos/pessoas estão usando sua bola mágica!



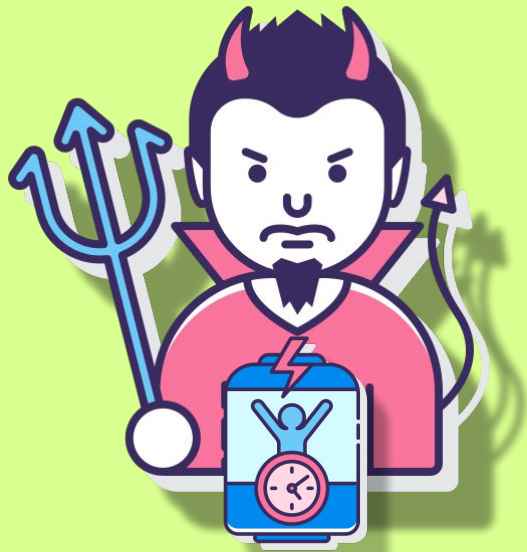
Fausto

Um dia, Fausto decidiu que gostaria de ver onde o Diabão estava!



Diabão

O que ele não contava era que o Diabão também tinha uma bola mágica. Com ela descobriu que Fausto o estava observando!



Diabão

Para que Fausto não usasse seus poderes para revelar sua posição. O Diabão acabou jogando uma maldição em Fausto.

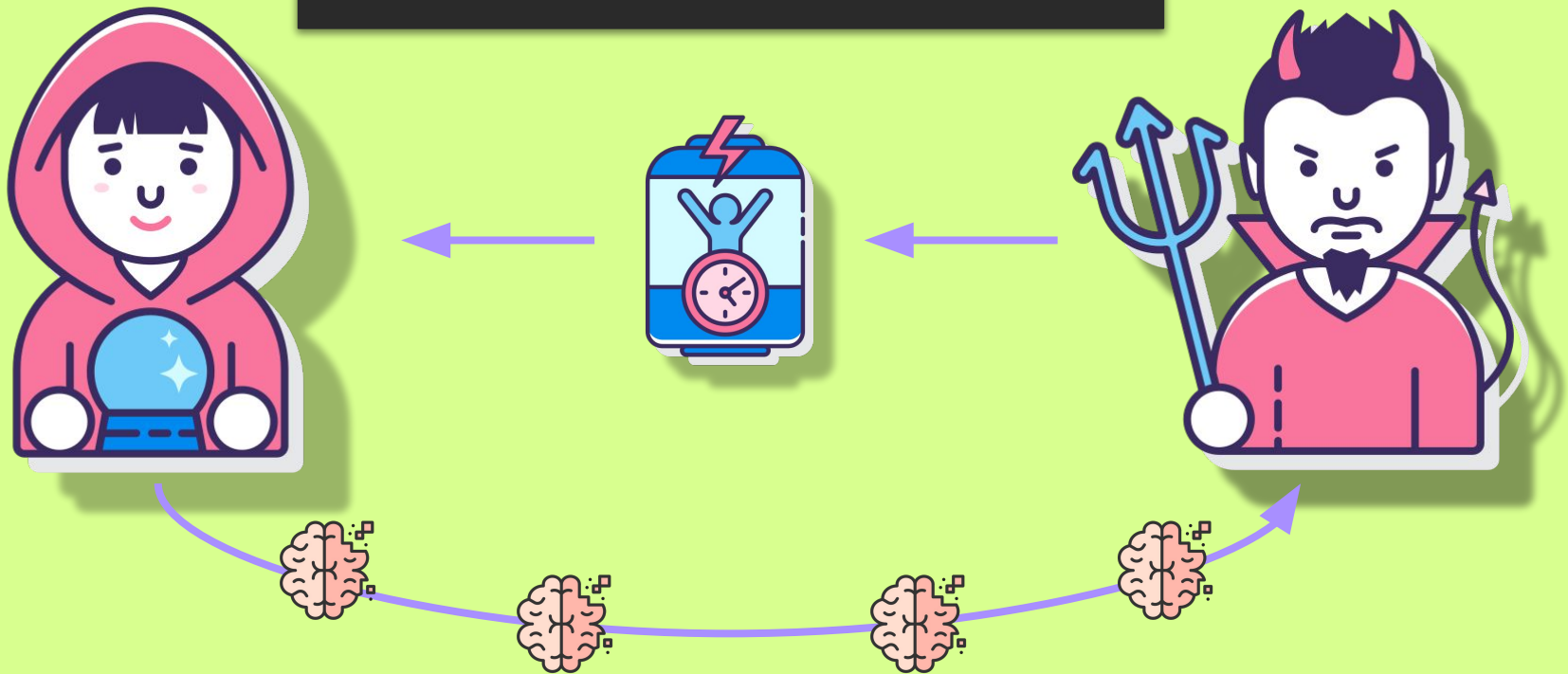


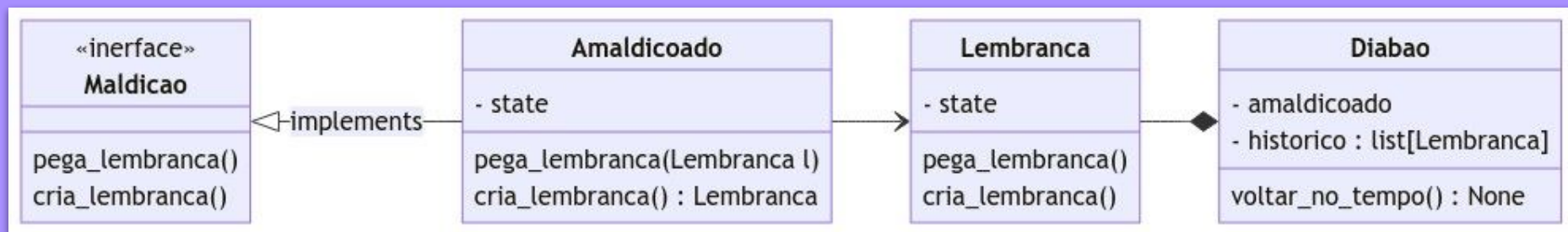
Maldição

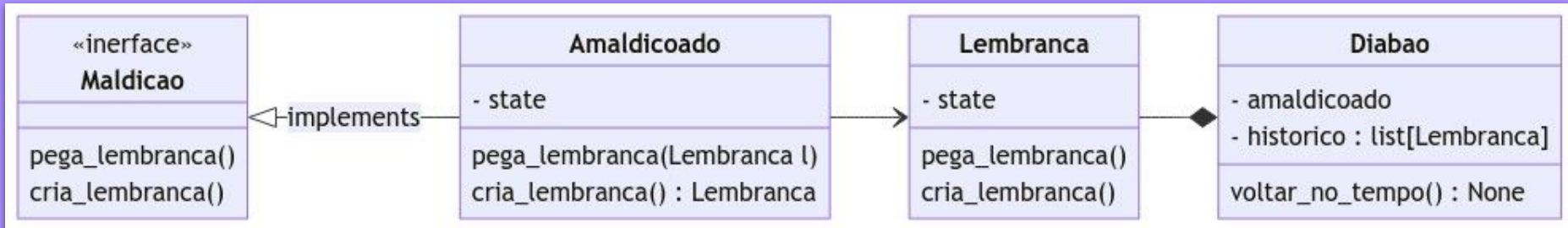
A maldição jogada é a maldição do **tempo e das lembranças**.

- Fausto criará uma memória ao acordar
- Se Fausto procurar pelo Diabão, ele voltará no tempo
- Quando voltar no tempo perderá todas as lembranças do dia

Quando a maldição foi liberada, um interface de lembranças foi criada no cérebro de Fausto. E todas as suas lembranças passaram a pertencer ao Diabão.







```
class Maldição(Protocol):
    def pega_lembança(self) -> Lembrança:
        ...

    def cria_lembança(self, lembrança: Lembrança):
        ...
```



Implementação do Amaldiçoado

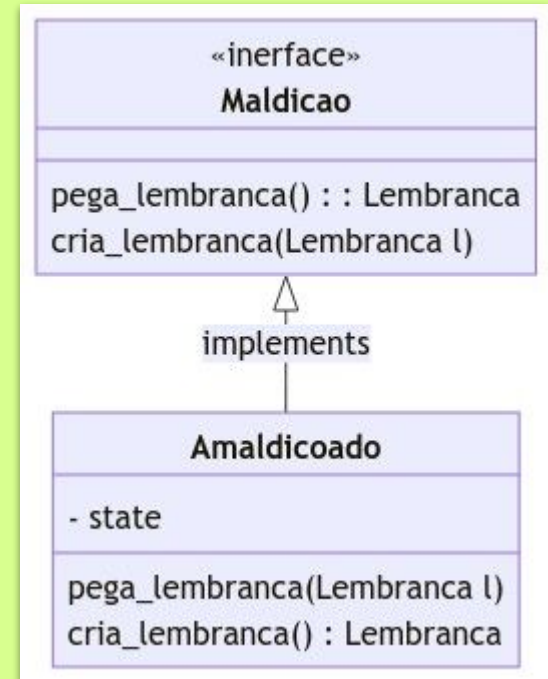
```
class Amaldiçoado:
    def __init__(self, nome):
        self.nome = nome
        self.proibido = False
        self.hora_atual = datetime.now()

    def ver_na_bola_de_cristal(self, algo):
        print(f'vendo {algo} na bola de cristal!')
        self.hora_atual = datetime.now()
        if algo == 'Diabão':
            self.proibido = True

    def cria_lemb branca(self) -> Lembrança:
        state = copy(self.__dict__)
        return Lembrança(state)

    def pega_lemb ranca(self, memento):
        self.__dict__.clear()
        self.__dict__.update(memento.get_estado())

    def __repr__(self):
        return f'Amaldiçoado({self.__dict__})'
```



A lembrança



```
class Lembrança:
    def __init__(self, estado) -> None:
        self.estado = estado

    def get_estado(self):
        return self.estado
```

Lembranca

- state

pega_lembanca()

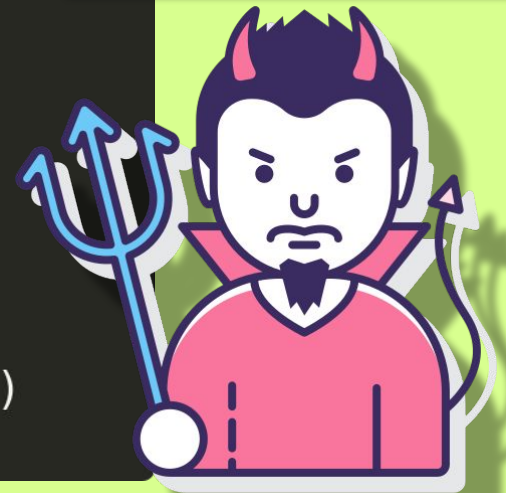
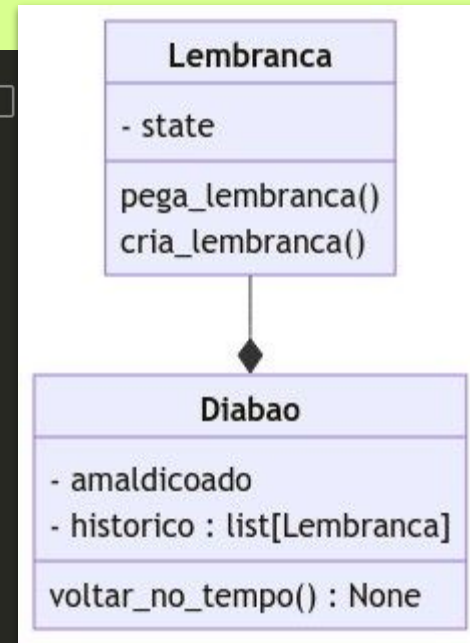
cria_lembanca()

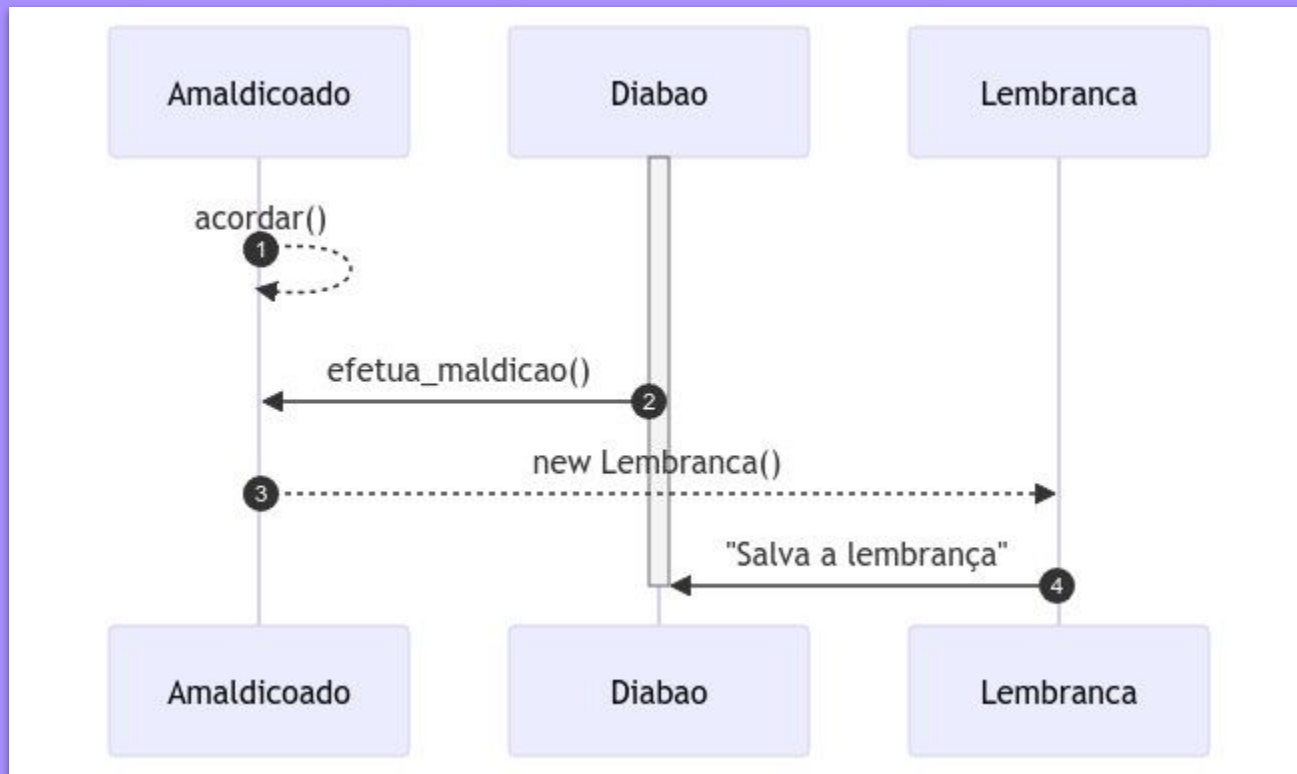


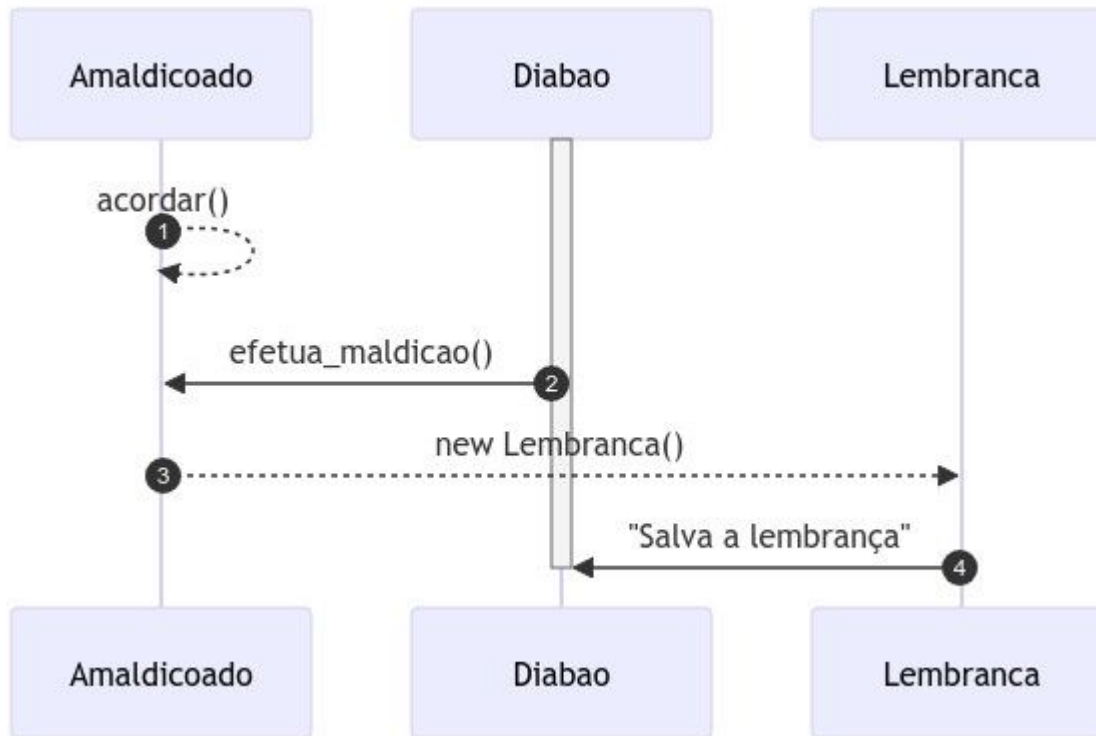
```
class Diabão:
    def __init__(self, amaldiçoado):
        self.amaldiçoado = amaldiçoado
        self.history = []

    def efetua_a_maldição(self):
        estado = self.amaldiçoado.acordar()
        self.history.append(estado)

    def voltar_no_tempo(self):
        estado = self.history.pop()
        self.amaldiçoado.voltar_no_tempo(estado)
```



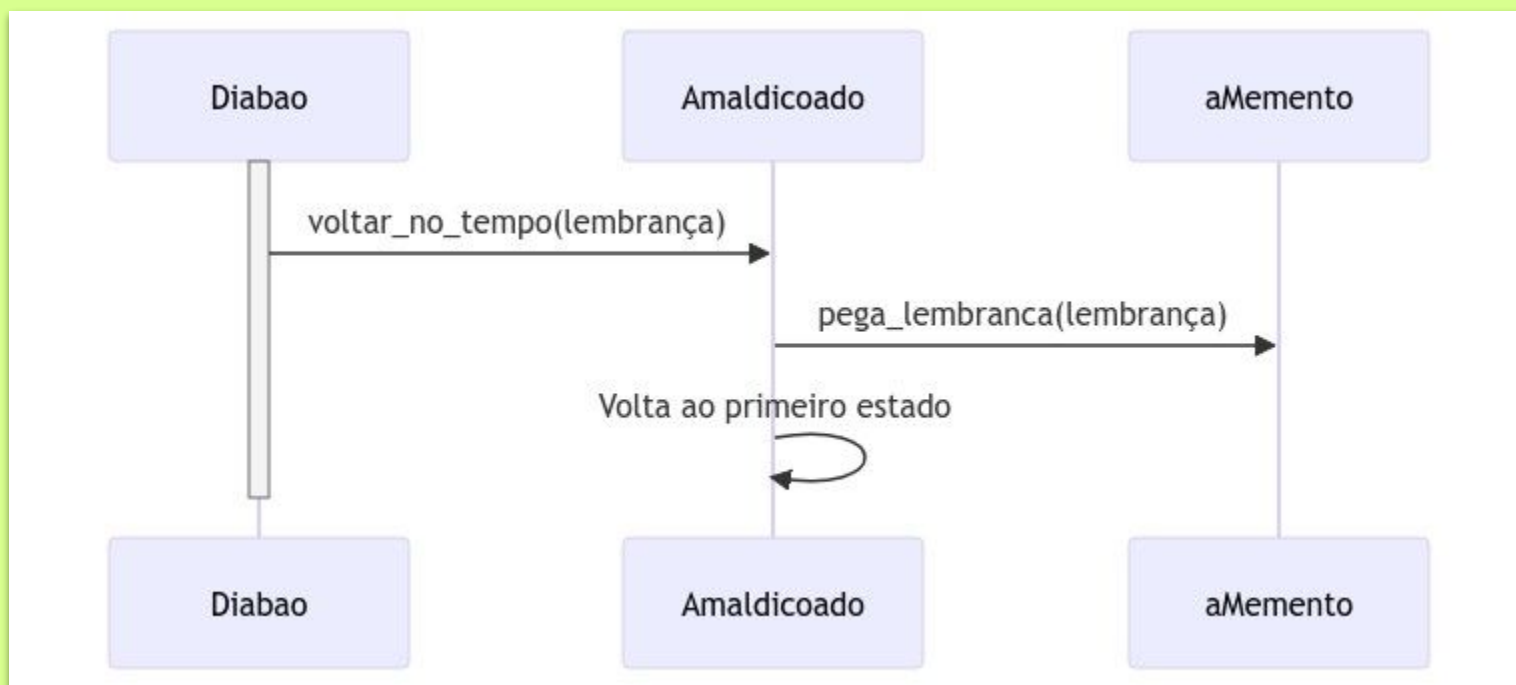




```
fausto = Amaldiçoado('Fausto')
diabo = Diabão(amaldiçoado=fausto)

fausto.acordar()
diabo.efetua_a_maldição()
```

A volta

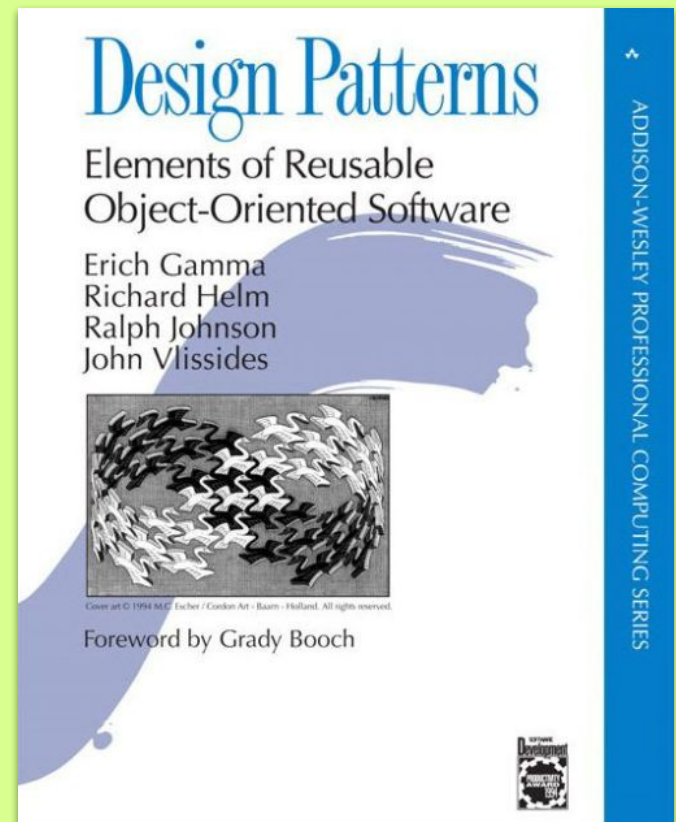


O padrão clássico

ME
MEN
TO



*"Sem violar o **encapsulamento**, capturar e externalizar um **estado interno** de um objeto, de maneira que o objeto possa ser **restaurado** para esse estado mais tarde"*

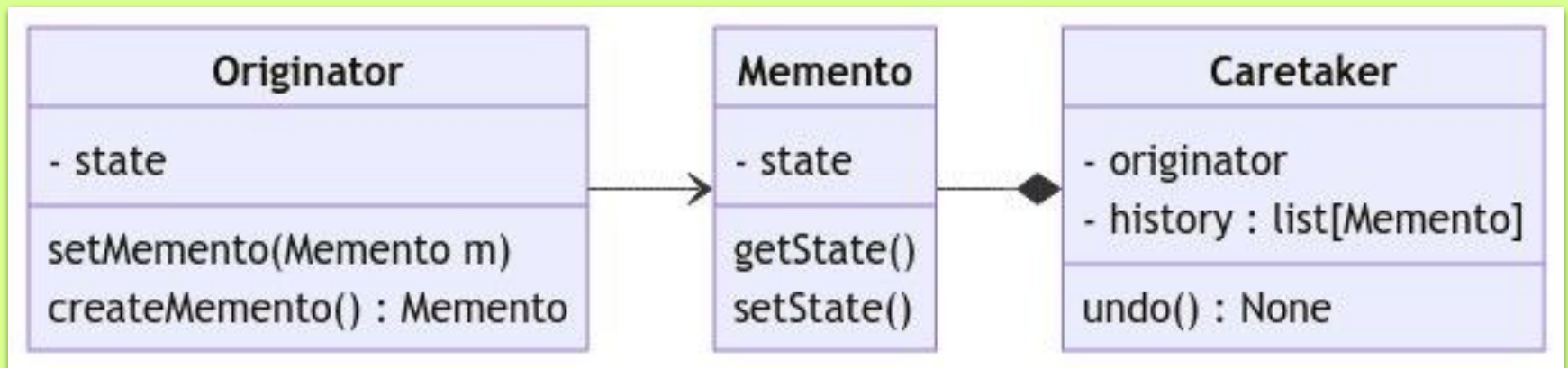


Registrar o estado do objeto

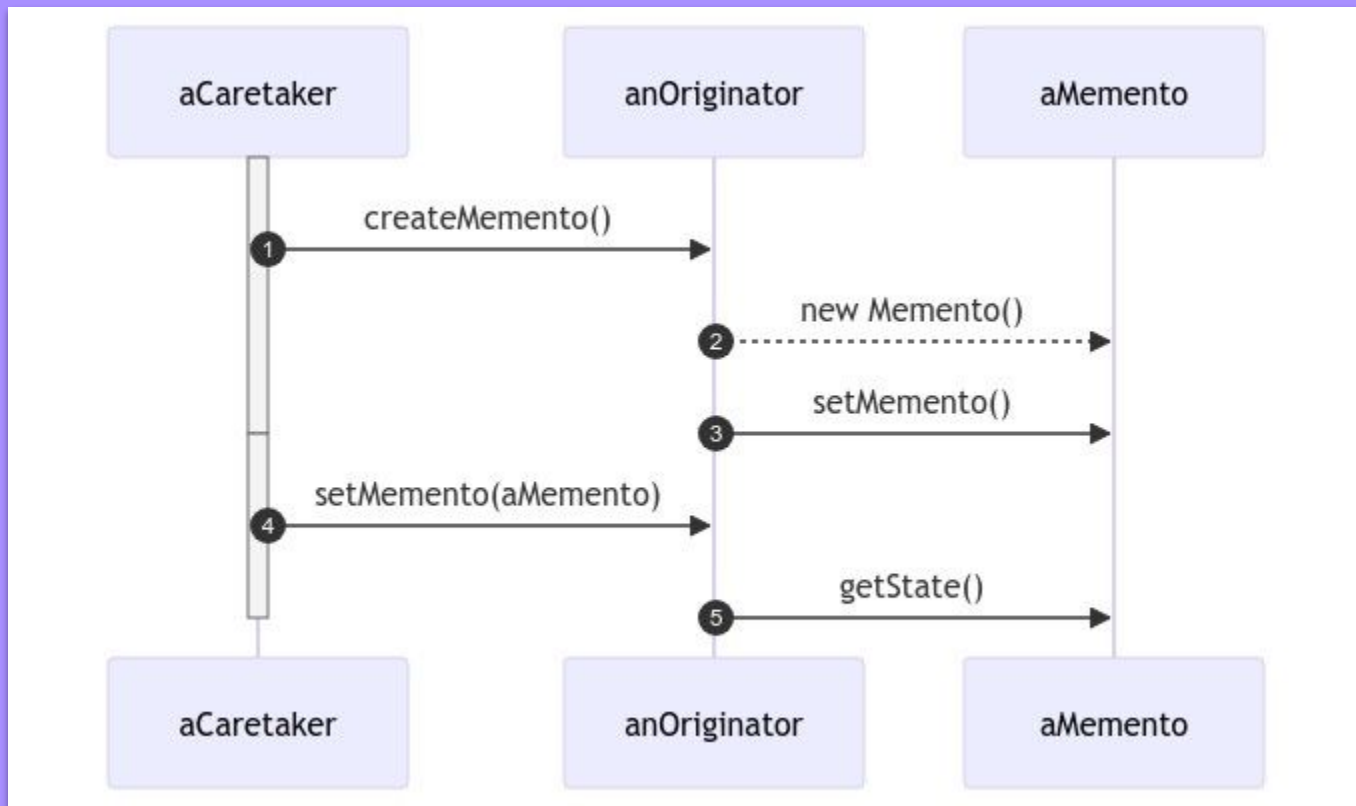


Abrir o GIMP!

0 diagrama original



0 fluxo



Implementações

Várias formas
possíveis

Bora codar!



Vimos pra isso, não?





Are Design Patterns Missing Language Features

On various places, it has been claimed that use of **DesignPatterns**, especially complex ones like **VisitorPattern**, are actually indicators that the language being used isn't powerful enough. Many **DesignPatterns** are by convention rather than encapsulable in a library or component, and as such contain repetition and thus violate **OnceAndOnlyOnce**. If it didn't contain at least *some* repetition, or something that could be Refactored out, then it wouldn't be a pattern.

Discussion on this topic culled from elsewhere on **WardsWiki**:

<https://wiki.c2.com/?AreDesignPatternsMissingLanguageFeatures>



Design Patterns in Dynamic Programming

Peter Norvig
Chief Designer, Adaptive Systems
Harlequin Inc.

<http://norvig.com/design-patterns/>

Atenção

!!!

Pontos importantes



- Linguagens dinâmicas não são "*seguras*"
- Em alguns contextos pode ser considerado um anti-padrão
 - Alternativa a ele: Post-State
 - Não suporta concorrência

Uma boa referência:

<https://freedium.cfd/https://medium.com/@bonnotguillaume/software-architecture-memento-is-an-anti-pattern-part-1-memento-c6c69cf11bee>

Pontos



- Linguagens dinâmicas não são "*seguras*"
- Em alguns contextos pode ser considerado um anti-padrão
 - Alternativa a ele: Post-State
 - Não suporta concorrência
- Memento depende de mutabilidade
 - Objetos imutáveis não são aplicáveis ao padrão
- Memento é um comilão de memória
 - <https://github.com/dunossauro/ldp-photo-editor>



apoia.se/livedepython



pix.dunossauro@gmail.com



patreon.com/dunossauro



Ajude o projeto <3

