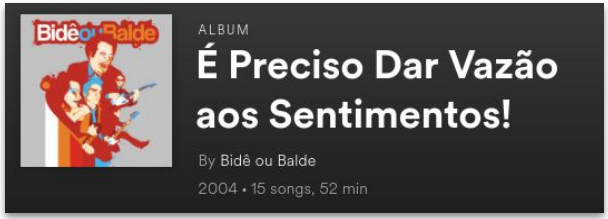


Live de Python #69

Programação orientada a objetos #6

Ajude a Live de Python

apoia.se/livedepython

picPay: @livedepython

Desconto 30% novatec: MENDESPY

Roteiro

- Protocolos (Revisão)
- Criando meu protocolo
- Filas ou quase isso

Referências:

- https://docs.python.org/3/library/abc.html
- https://github.com/python/cpython/blob/master/Lib/_collections_abc.py
- https://github.com/python/cpython/blob/master/Lib/abc.py
- https://github.com/python/cpython/blob/master/Lib/_py_abc.py

interfaces

“Interfaces são métodos ou atributos públicos que outros objetos possam usar para se **comunicar** com outros objetos”

Dusty Phillips

Interfaces (Definição informal)

“O subconjunto de métodos públicos de um objeto que lhe permitem desempenhar um papel específico em um sistema”

Ramalho

Criando meu protocolo

para criar meu protocolo, ou minhas ABCs, a classe abstrata só precisa herdar de ABC e colocar @abstractmethod nos métodos que necessariamente precisam ser reescritos.

- Aninhar decoradores é uma boa prática.

```python
from abc import ABC, abstractmethod

class MeuProtocolo(ABC):

    @abstractmethod
    def meu_metodo_abstrato(self):
        """Uma Docstring legal."""
        ...

    @staticmethod
    @abstractmethod
    def meu_metodo_abstrato_e_estatico(self):
        """Uma Docstring legal."""
        ...
```

Filas

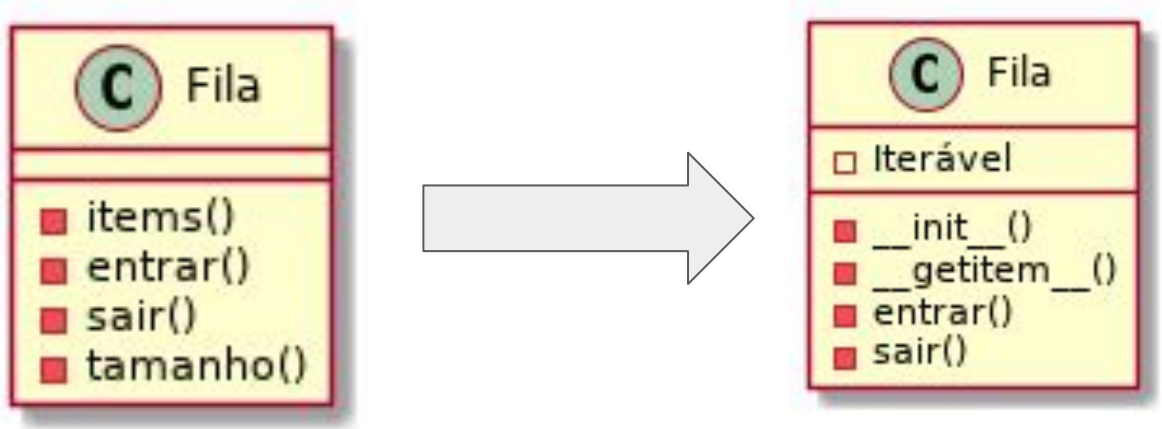
Filas seguem um determinado padrão e várias abstrações de processo. Como: Entrar, sair, tamanho, pessoas na fila... etc...

Filas

Filas seguem um determinado padrão e várias abstrações de processo. Como: Entrar, sair, tamanho, pessoas na fila... etc...

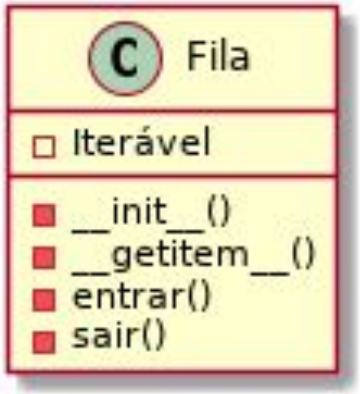
Filas

Filas seguem um determinado padrão e várias abstrações de processo. Como: Entrar, sair, tamanho, pessoas na fila... etc...



Filas

Filas seguem um determinado padrão e várias abstrações de processo. Como: Entrar, sair, tamanho, pessoas na fila... etc...

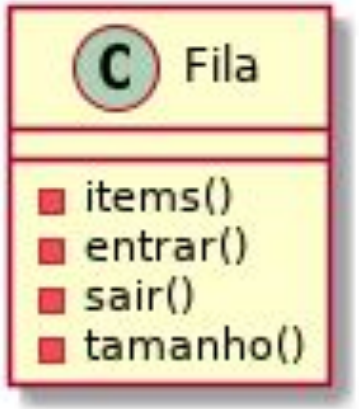


```
class Fila:
    def __init__(self, iterável):
        ...

    def __getitem__(self, pos):
        ...

    def entrar(self, item):
        ...

    def sair(self, pos):
        ...
```

Filas

Só que, vamos imaginar que temos 2 tipos de filas diferentes.

1. Fila do supermercado
2. Fila da padaria



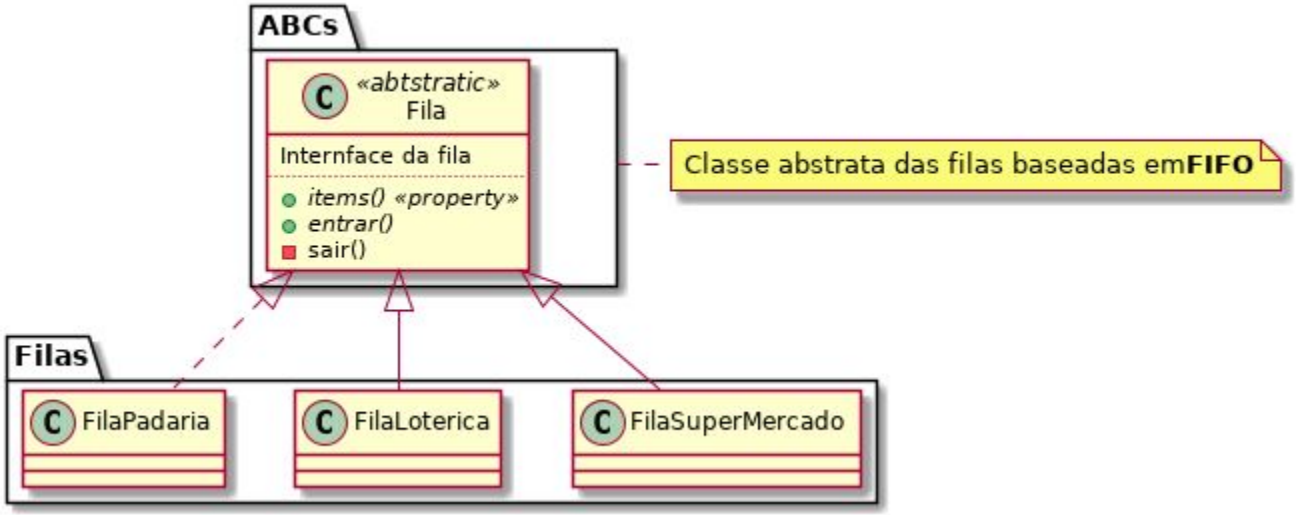
Filas

Só que, vamos imaginar que temos 2 tipos de filas diferentes.

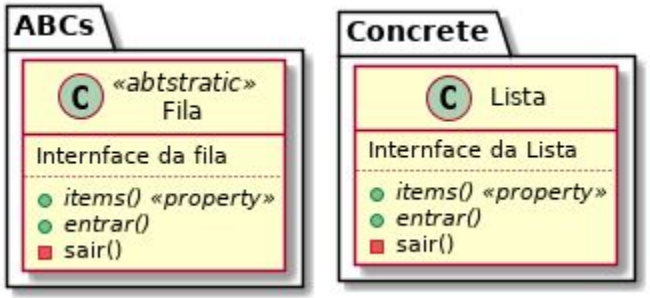
1. Fila do supermercado
    a. Prefencial
        i. Idade
        ii. Gravidez
        iii. Deficientes físicos
2. Fila da Lotérica
    a. Quem chegar primeiro, é atendido primeiro

Vamos codar um pouco

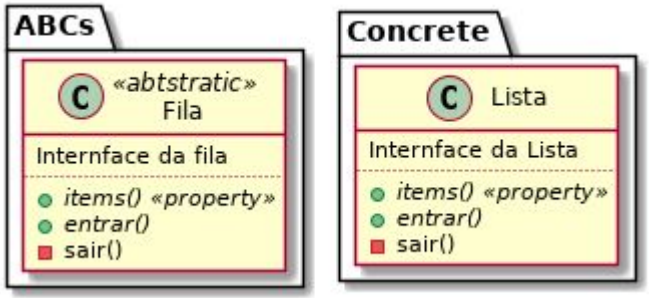
Fila que não herda, mas registra



Fila que não herda, mas é



Fila que não herda, mas é




```python
@classmethod
def __subclasshook__(cls, classe):
    if cls is Fila:
        return _check_methods(classe, 'entrar', 'items')
    return NotImplemented
```