Interfaces

Programação Orientada a Objetos

Python

Prof. Diógenes Furlan

Interface

- É um tipo de herança “fraca”.
- Somente é permitida a especificação de:
  - Constantes públicas
  - Métodos públicos e abstratos
- São proibidos:
  - Atributos
  - Métodos privados e protegidos
- Definem um comportamento.
  - Praticamente um “decorador”

Interface

- Em C++: é uma classe abstrata que contém SOMENTE métodos virtuais puros.

```
class glutOpengl {
public:
   virtual void init() = 0;
   virtual void draw() = 0;
   virtual void timer() = 0;
};
```

Interface

- Em Python: é uma classe abstrata que contém SOMENTE métodos abstratos.

```
class glutOpengl(ABC):
    @abstractmethod
    def init(self):
        pass
    @abstractmethod
    def draw(self):
        pass
    @abstractmethod
    def timer(self):
        pass
};
```

Exemplo

- Interface com 1 método
- Define comportamento

```
class OpMat(ABC):

    @abstractmethod
    def calcular(self,x,y):
        pass
```

Exemplo – Classe Conta

```python
from abc import ABC, abstractmethod

class Conta(ABC):
    def __init__(self):
        self._saldo = 0.0

    def depositar(self, valor: float):
        if valor > 0:
            self._saldo += valor
            print(f"Depositado: {valor}")
        else:
            print("Valor de depósito deve ser positivo.")

    def sacar(self, valor: float):
        if valor > 0:
            self._saldo -= valor
            print(f'Sacado: {valor}')
        else:
            print('Valor de saque deve ser positivo.')

    def consultar_saldo(self):
        print('Saldo: ', self._saldo)
```

Exemplo – Classe Conta

```
class ContaCorrente(Conta):
    def sacar(self, valor: float):
        if valor <= 700.0 and self._saldo >= valor:
            super().sacar(valor)
        else:
            if valor > self._saldo:
                print('Saldo insuficiente.')
            else:
                print('Saque superior a R$ 700,00 somente na conta especial.')
```

Exemplo – Classe Conta

```
class ContaEspecial(Conta):
    def __init__(self, lim=0):
        self._limite = lim
        super().__init__()

    def sacar(self, valor: float):
        if valor <= self._saldo:
            super().sacar(valor)
        else:
            if valor <= self._saldo + self._limite:
                print(f'Sacado {valor}. Utilizando o limite de \
                        crédito de {valor - self._saldo}.')
                self._saldo = - (valor - self._saldo) * 1.05
            else:
                print('Saldo insuficiente.')

    def consultar_saldo(self):
        print('Saldo: ', self._saldo)
        print('Limite disponível: ', max(0, self._limite + self._saldo))
```

Exemplo – Classe Conta

```
class ContaPoupança(Conta):
    def sacar(self, valor: float):
        if valor <= self._saldo:
            super().sacar(valor)
        else:
            print('Saldo insuficiente.')
```

Exemplo – Classe Conta

```
if __name__ == "__main__":
    cc = ContaCorrente()
    cc.depositar(1000.0)
    cc.sacar(-300.0)
    cc.sacar(1200.0)
    cc.sacar(800.0)
    cc.sacar(600.0)
    cc.consultar_saldo()

    ce = ContaEspecial(1000.0)
    ce.depositar(1000.0)
    ce.sacar(-300.0)
    ce.sacar(2500.0)
    ce.sacar(1200.0)
    ce.consultar_saldo()
    ce.sacar(1000.0)
    ce.consultar_saldo()
    ce.sacar(790.0)
    ce.consultar_saldo()
```

```
cp = ContaPoupança()
cp.depositar(1000.0)
cp.sacar(-300.0)
cp.sacar(1200.0)
cp.sacar(800.0)
cp.sacar(600.0)
cp.consultar_saldo()
```

Interface Pagador

- O banco paga taxas aos clientes

```
class Pagador(ABC):
    @abstractmethod
    def pagarTaxa(self):
        pass

class ContaCorrente(Conta, Pagador):
    # sem taxa
    def pagarTaxa(self):
        pass

class ContaEspecial(Conta, Pagador):
    # taxa de 1% para um cliente especial
    def pagarTaxa(self):
        self._saldo += self._saldo * 0.01

class ContaPoupança(Conta):
    # taxa de 5% para poupança
    def pagarTaxa(self):
        self._saldo += self._saldo * 0.05
```

Interface Pagador

```
# main
cc = ContaCorrente()
ce = ContaEspecial(1000.0)
cp = ContaPoupança()

cc.depositar(1000.0)
ce.depositar(1000.0)
cp.depositar(1000.0)

cc.pagarTaxa()
ce.pagarTaxa()
cp.pagarTaxa()

cc.consultar_saldo()
ce.consultar_saldo()
cp.consultar_saldo()
```

Interface Cobrador

- O banco recolhe taxas dos clientes

```
class Cobrador(ABC):
    @abstractmethod
    def cobrarTaxa(self):
        pass

class ContaCorrente(Conta):
    # taxa de R$10,00
    def cobrarTaxa(self):
        self._saldo -= 10.0

class ContaEspecial(Conta):
    # taxa de R$20,00 + 1% do limite
    def cobrarTaxa(self):
        self._saldo -= 20.0 + self._limite * 0.01

class ContaPoupança(Conta):
    # taxa de R$0,00
    def cobrarTaxa(self):
        pass
```

Interface Cobrador

```
# main
cc = ContaCorrente()
ce = ContaEspecial(1000.0)
cp = ContaPoupança()

cc.depositar(1000.0)
ce.depositar(1000.0)
cp.depositar(1000.0)

cc.cobrarTaxa()
ce.cobrarTaxa()
cp.cobrarTaxa()

cc.consultar_saldo()
ce.consultar_saldo()
cp.consultar_saldo()
```

Interface XML

- Faça uma interface que permita trabalhar com XML na classe Empregado.

- Entrada:
  - `Empregado e1("Isidoro", "1901-09-11", "5.000")`
- Saída:

```
<e1>
    <nome> Isidoro </nome>
    <dtnasc>
        <dia> 11 </dia>
        <mes<> 9 </mes>
        <ano> 1901 </ano>
    </dtnasc>
    <salario> 5000.00 </salario>
</e1>
```

Interface JSON

- Faça uma interface que permita trabalhar com JSON na classe Empregado.

```
class Empregado {
    string nome;
    Data dtNasc;
    double salario;
};
```

- Entrada:
  - `Empregado e1("Isidoro", "1901-09-11", "5.000")`
- Saída:
  - `e1:{"nome":"Isidoro", "dtnasc":{"dia":11, "mes":09, "ano":1901}, "salario":5000.00}`

Enviar em

https://forms.gle/tcB7G6YsgLGGkz2dA

EXERCÍCIOS

Exercício 1

- Sejam as seguintes classes:
  - Seguro vida
  - Previdência Privada
- Implemente a interface Cobrador para elas.
- Implemente a interface Pagador para elas.

Exercício 2

- Seja uma Locadora de Veículos, que trabalha com:
  - Motos
  - Carros de passeio
  - Caminhões
- Todo Veículo possui
  - Preço do veículo
  - Ano fabricação
- Carros de passeio possuem:
  - numero de passageiros
- Caminhões possuem:
  - peso máximo de carga

Exercício 2

- Interface Aluguel
  - diarista()
  - semanista()
  - mensalista()

- FQ: faixas de quilometragem
  - Deve ser adicionado um valor de combustível e outro para depreciação

Exercício 2

- Regra da Diária:
  - Moto – 0.5% valor veículo + seguro 500 + FQ
  - Carro – 0.2% valor + FQ
  - Caminhão 0.05% valor + 2% do valor da carga transportada + FQ
- Regra Semanal
  - conceder algum desconto sobre valor de 7 diárias
- Regra Mensal
  - conceder um desconto sobre 4,5 semanas

Exercício 2

- Existem 5 (FQ) faixas de quilometragem (semanal):
  - A primeira é para quem roda até 200 km na semana;
  - A segunda é para quem roda de 201 km a 400 km;
  - A terceira é para quem roda de 401 km a 1.250 km
  - A quarta é para quem roda de 1.251 km a 1.500 km
  - A quinta é para quem roda ilimitado: acima de 1.501 km até o infinito!