Disciplina: IPOO - Introdução à Programação Orientada a Objetos

Prof. Luiz Cláudio Dalmolin - luiz.dalmolin@udesc.br

Orientação a Objetos com Python:

Herança

In [1]:
```python
# Slides - parte 11
# Exemplo 1
class Pessoa:
    __nome = None
    __idade = None

    def __init__(self, nome, idade):
        self.__nome = nome
        self.__idade = idade

    def setnome(self, nome):
        self.__nome = nome

    def setidade(self, idade):
        self.__idade = idade

    def getnome(self):
        return self.__nome

    def getidade(self):
        return self.__idade

#from pessoa import Pessoa
class PessoaFisica(Pessoa):
    __CPF = None
    def __init__(self, CPF, nome, idade):
        super().__init__(nome, idade)
        self.__CPF = CPF

    def getCPF(self):
        return self.__CPF

    def setCPF(self, CPF):
        self.__CPF = CPF

#from pessoa import Pessoa
class PessoaJuridica(Pessoa):
    __CNPJ = None
    def __init__(self, CNPJ, nome, idade):
        super().__init__(nome, idade)
        self.__CNPJ = CNPJ

    def getCNPJ(self):
        return self.__CNPJ
```

```
    def setCNPJ(self, CNPJ):
        self.__CNPJ = CNPJ
###
pf = PessoaFisica('123.456.789-11', 'Fulano da Silva', 18)
pj = PessoaJuridica('12.123.123/0001-01', 'Empresa XYZ Ltda', 5)

print('CPF:', pf.getCPF(), '\nNome:', pf.getnome(), '\nIdade:', pf.getidade())
print('\nCNPJ:', pj.getCNPJ(), '\nNome:', pj.getnome(), '\nIdade:', pj.getidade(

p1 = Pessoa('Maria',22)
print('\nNome:', p1.getnome(), '\nIdade:', p1.getidade())
```

```
CPF: 123.456.789-11
Nome: Fulano da Silva
Idade: 18

CNPJ: 12.123.123/0001-01
Nome: Empresa XYZ Ltda
Idade: 5

Nome: Maria
Idade: 22
```

In [5]:

```
class Conta:
    # incluir os atributos do construtor, os getters e os setters
    def __init__ (self, clientes, numero, saldo=0):
        self._saldo = 0
        self.__clientes = clientes
        self.__numero = numero
        self._operacoes = []
        self.deposito(saldo)
    def resumo(self):
        print ("CC Numero: ", self.__numero, "Saldo:", self._saldo)
    def saque(self, valor):
        if self._saldo >= valor:
            self._saldo -= valor
            self._operacoes.append(["Saque", valor])
    def deposito(self, valor):
        self._saldo += valor
        self._operacoes.append(["Deposito",valor])
    def extrato(self):
        print("Extrato CC Nr: %3d" % self.__numero)
        for o in self._operacoes:
            print("%10s %10.2f" % (o[0],o[1]))
        print ("\nSaldo: %10.2f\n" % self._saldo)

class ContaEspecial(Conta):
    __limite = None
    def __init__(self,clientes,numero,saldo=0,limite=0):
        super().__init__(clientes, numero, saldo)
        self.__limite = limite
    def saque(self, valor):
        if self._saldo + self.__limite >= valor:
            self._saldo -= valor
            self._operacoes.append((["Saque", valor]))

class Cliente:
    def __init__ (self, nome, telefone):
        self.__nome = nome
```

```
        self.__telefone = telefone

###
#from clientes import Cliente
#from contas import Conta, ContaEspecial

joao = Cliente("Joao da Silva", "777-1234")
maria = Cliente("Maria da Silva", "555-4321")

conta1 = Conta([joao], 1, 1000)
conta2 = ContaEspecial([maria, joao], 2, 500, 1000)   # 1000 = limite
conta1.saque(50)
conta2.deposito(300)
conta1.saque(190)
conta2.deposito(95.15)
conta2.saque(1500)
conta1.extrato()
conta2.extrato()
```

```
Extrato CC Nr:   1
  Deposito    1000.00
     Saque      50.00
     Saque     190.00

Saldo:     760.00

Extrato CC Nr:   2
  Deposito     500.00
  Deposito     300.00
  Deposito      95.15
     Saque    1500.00

Saldo:    -604.85
```

In [2]:

```
#18. Crie uma classe chamada Ingresso que possui um valor em reais e um método
#    imprimeValor().
# a) crie uma classe VIP, que herda Ingresso e possui um valor adicional.
#    Crie um método que retorne o valor do ingresso VIP (com o adicional
#   incluído).
# b) crie uma classe Normal, que herda Ingresso e possui um método que imprime:
#    "Ingresso Normal".
# c) crie uma classe CamaroteInferior (que possui a localização do ingresso
#    e métodos para acessar e
#    imprimir esta localização) e uma classe CamaroteSuperior, que é mais
#    cara (possui valor adicional).
#    Esta última possui um método para retornar o valor do ingresso. Ambas
#    as classes herdam a classe VIP.

class Ingresso:
    valor = None

    def __init__(self,valor):
        self.valor = valor

    def imprimeValor(self):
        return self.valor

class Vip(Ingresso):
    valorAdicionalVip = None
```

```
    def __init__(self,valor,valorAd):
        super().__init__(valor)
        self.valorAdicionalVip = valorAd

    def getValorIngressoVip(self):
        return self.valorAdicionalVip + self.valor

class Normal(Ingresso):

    def __init__(self,valor):
        super().__init__(valor)

    def getIngressoNormal(self):
        return self.valor

class CamaroteInferior(Vip):
    localizacao = None

    def __init__(self,valor,valorAd,localizacao):
        super().__init__(valor,valorAd)
        self.localizacao = localizacao

    def getLocalizacao(self):
        return self.localizacao

    def setLocalizacao(self,localizacao):
        self.localizacao = localizacao

class CamaroteSuperior(Vip):
    valorAdicionalCamaroteSuperior = None

    def __init__(self,valor,valorAd,valorAdicionalCamaroteSuperior):
        super().__init__(valor,valorAd)
        self.valorAdicionalCamaroteSuperior = valorAdicionalCamaroteSuperior

    def getValorCamaroteSuperior(self):
        return self.valorAdicionalCamaroteSuperior + self.valorAdicionalVip + se

###

ingressoNormal = Normal(10.0)
ingressoVip = Vip(10.0,5.0)
ingressoCamaroteInferior = CamaroteInferior(10.0,5.0,"Setor A")
ingressoCamaroteSuperior = CamaroteSuperior(10.0,5.0,10.0)

# Tarefa: Com base no exemplo acima, escrever um programa que solicite ao
# usuário a quantidade de ingressos que deseja e imprimir o valor a ser pago,
# considerando os valores de ingressos acima e as classes criadas.
###
print("----- Tabela de Preços -----")
print("Preço ingresso normal - R$ %.2f" % ingressoNormal.getIngressoNormal())
print("Preço ingresso VIP - R$ %.2f" % ingressoVip.getValorIngressoVip())
print("Preço ingresso camarote Inferior - R$ %.2f - Localização: %s" % (ingresso
print("Preço ingresso camarote Superior - R$ %.2f" % (ingressoCamaroteSuperior.g
print("----------------------------")
print("")

tipo = int(input("Tipo de ingresso (1-Normal, 2-VIP, 3-Camarote Inf, 4-Camarote
qtde=int(input("Quantidade de ingressos: "))
```

```
if tipo == 1:
    print("==> %i ingressos tipo 'normal' = R$ %.2f" % (qtde, qtde*ingressoNorma
if tipo == 2:
    print("==> %i ingressos tipo 'VIP' = R$ %.2f" % (qtde, qtde*ingressoVip.getV
if tipo == 3:
    print("==> %i ingressos tipo 'Camarote Inferior' = R$ %.2f " % (qtde, qtde *
if tipo == 4:
    print("==> %i ingressos tipo 'Camarote Superior' = R$ %.2f " % (qtde, qtde *
```

```
----- Tabela de Preços -----
Preço ingresso normal - R$ 10.00
Preço ingresso VIP - R$ 15.00
Preço ingresso camarote Inferior - R$ 15.00 - Localização: Setor A
Preço ingresso camarote Superior - R$ 25.00
----------------------------

Tipo de ingresso (1-Normal, 2-VIP, 3-Camarote Inf, 4-Camarote Sup: )1
Quantidade de ingressos: 10
==> 10 ingressos tipo 'normal' = R$ 100.00
```

In [3]:

```
#19. Crie a classe Imovel, que possui um endereço e um preço.
# a) crie uma classe Novo, que herda Imovel e possui um adicional no preço.
#    Crie métodos de acesso (get/set) e impressão deste valor adicional.
# b) crie uma classe Usado, que herda de Imovel e possui um desconto no preço.
#    Crie métodos de acesso (get/set) e impressão para este desconto.

class Imovel:
    endereco = None
    preco = None
    def __init__(self,endereco, preco):
        self.endereco = endereco
        self.preco = preco

class Novo(Imovel):
    adicionalNoPreco = None
    def __init__(self,endereco, preco,adicionalNoPreco):
        super().__init__(endereco, preco)
        self.adicionalNoPreco = adicionalNoPreco
    def getPrecoNovo(self):
        return self.adicionalNoPreco + self.preco
    def setAdicionalNoPreco(self,adicionalNoPreco):
        self.adicionalNoPreco = adicionalNoPreco

class Usado(Imovel):
    descontoNoPreco = None
    def __init__(self,endereco, preco,descontoNoPreco):
        super().__init__(endereco, preco)
        self.descontoNoPreco = descontoNoPreco
    def getPrecoUsado(self):
        return self.preco - self.descontoNoPreco
    def setDescontoNoPreco(self,descontoNoPreco):
        self.descontoNoPreco = descontoNoPreco

i = Imovel("rua x",1000.0)
ino = Novo("Rua Y",1000.0,500.0)
ius = Usado("Rua Z",1000.0,200.0)
print("Preço do final do imóvel Novo - R$ %.2f" % ino.getPrecoNovo())
print("Preço do final do imóvel Usado - R$ %.2f" %ius.getPrecoUsado())
```

```
Preço do final do imóvel Novo - R$ 1500.00
Preço do final do imóvel Usado - R$ 800.00
```

In [14]:

```python
# 14. Crie classes para representar estados e cidades. Cada estado
#tem um nome, sigla e cidades. Cada cidade tem nome e
#população. Escreva um programa de testes que crie três estados
#com algumas cidades em cada um. Exiba a população de cada
#estado como a soma da população de suas cidades.

class Cidade:
    def __init__(self, nome, populacao):
        self.nome = nome
        self.populacao = populacao
    def getNome(self):
        return self.nome
    def getPopulacao(self):
        return self.populacao

class Estado:
    def __init__(self, nome, sigla):
        self.nome = nome
        self.sigla = sigla
        self.cidades = []
    def adiciona_cidade(self, cidade):
        self.cidades.append(cidade)

    def populacao(self):
        return sum([c.getPopulacao() for c in self.cidades])

    def getNome(self):
        return self.nome

###
c1 = Cidade("São Bento do Sul", 85000)
c2 = Cidade("Rio Negrinho", 60000)
c3 = Cidade("Curitiba", 1500000)
c4 = Cidade("Pien", 25000)
e1 = Estado("Paraná", "PR")
e2 = Estado("Santa Catarina", "SC")
e1.adiciona_cidade(c4)
e1.adiciona_cidade(c3)
e2.adiciona_cidade(c1)
e2.adiciona_cidade(c2)
print("Estado: %s - População %d " % (e1.getNome(), e1.populacao()))
print("Estado: %s - População %d " % (e2.getNome(), e2.populacao()))
print('ou')

# formatando números maiores...
texto_pop1 = f'{e1.populacao():_.0f}' # separando milhar com '_'
texto_pop1 = texto_pop1.replace('.',',').replace('_','.') # substituindo '_' por
texto_pop2 = f'{e2.populacao():_.0f}' # idem para e2
texto_pop2 = texto_pop2.replace('.',',').replace('_','.') # idem para e2

 # imprimindo valores formatados em Reais
print(f'Estado: {e1.getNome()}: População = {texto_pop1} hab.')
print(f'Estado: {e2.getNome()}: População = {texto_pop2} hab.')
```

```
Estado: Paraná - População 1525000
Estado: Santa Catarina - População 145000
ou
Estado: Paraná: População = 1.525.000 hab.
Estado: Santa Catarina: População = 145.000 hab.
```

In [25]:

```python
# formatação de moeda para sistema brasileiro 'BR'
from decimal import Decimal
import locale
locale.setlocale(locale.LC_ALL, 'pt_BR')

# Exemplo
valor =10254897.257876
print(locale.currency(valor, grouping=True))
```

```
R$ 10.254.897,26
```