

Avaliação 4 de POO

1ª Questão (10 Escores). Associe a cada item da 2ª coluna um valor que corresponde a um item da 1ª coluna.

*a) Permite que um objeto seja usado no lugar de outro. (C)
Encapsulamento*

b) Define a representação de um objeto. (H) Mensagem

*c) Separação de interface e implementação que permite que usuários de objetos possam utilizá-los sem conhecer detalhes de seu código. (I)
Herança*

d) Possui tamanho fixo. (A) Polimorfismo

e) Instância de uma classe. (F) Dependência

f) Forma de relacionamento entre classes onde objetos são instanciados no código. (J) Lista

g) Forma de relacionamento entre classes implementado por meio de coleções. (B) Classe

h) Forma de chamar um comportamento de um objeto. (E) Objeto

i) Reuso de código na formação de hierarquias de classes. (G) Composição

j) Permite inserções e remoções. (D) Array

2ª Questão (10 Escores). Aplique V para as afirmações verdadeiras e F para as afirmações falsas.

- a) Métodos construtores devem sempre ser explícitos. (**F**)
- b) A classe Professor tem um relacionamento de agregação com a classe Disciplina. (**V**)
- c) Quando uma classe possui como atributo uma referência para um objeto temos uma dependência. (**V**)
- d) Membros de classes static existem mesmo quando nenhum objeto dessa classe exista. (**V**)
- e) Um relacionamento 'tem um' é implementado via herança. (**F**)
- f) Uma classe Funcionário tem um relacionamento 'é um' com a classe Dependente. (**F**)
- g) Uma classe abstract pode ser instanciada. (**F**)
- h) Relacionamentos TODO-PARTE são tipos de associações. (**V**)
- i) Você implementa uma interface ao subscrever apropriada e concretamente todos os métodos definidos pela interface. (**V**)
- j) Um método static não é capaz de acessar uma variável de instância. (**F**)

3ª Questão (40 Escores). Escreva exemplos de código Python onde seja possível identificar os seguintes conceitos de POO.

a) Herança;

class Veiculo():

def __init__(self, cor, portas, marca):

self.cor = cor

self.portas = portas

self.marca = marca

def mostrar(self):

print("Cor:", self.cor)

print("Número de portas:", self.portas)

print("Marca:", self.marca)

b) Encapsulamento;

```
class ContaBancaria():  
    def _init_(self, saldo):  
        self.__saldo = saldo #Privado  
  
    def sacar(self, valor):  
        self.__saldo -= valor  
  
    def depositar(self, valor):  
        self.__saldo += valor  
  
    def getSaldo(self):  
        return self.__saldo  
  
    def setSaldo(self, novo):  
        self.__saldo = novo  
  
conta1 = ContaBancaria(5000)  
  
conta1.depositar(800)  
print("Valor total após o depósito:", conta1.getSaldo())  
  
conta1.sacar(600)  
print("Valor total após o saque:", conta1.getSaldo())
```

c) Polimorfismo;

```
class Super:  
    def hello(self):  
        print("Oi!")
```

```
class Sub (Super):  
    def hello(self):  
        print("Como vai?")
```

```
teste = Sub()  
teste.hello()
```

d) Variáveis de Instância;

```
class ContaBancaria():  
    saldoBanco = 38000  
    def _init_(self, saldo):  
        self.__saldo = saldo  
ContaBancaria.saldoBanco += saldo  
  
    def sacar(self, valor):  
        self.__saldo -= valor  
        ContaBancaria.saldoBanco -= valor  
  
    def depositar(self, valor):  
        self.__saldo += valor  
        ContaBancaria.saldoBanco += valor  
  
    def getSaldo(self):  
        return self.__saldo  
  
    def setSaldo(self, novo):  
        self.__saldo = novo
```

```
conta1 = ContaBancaria(3800)
```

```
conta1.depositar(600)print(conta1.getSaldo())
```

```
conta1.sacar(800)
```

```
print(conta1.getSaldo())
```

```
print(ContaBancaria.saldoBanco)
```

e) Métodos construtores

```
class Pessoa():
```

```
    def _init_(self, nome, idade):
```

```
        self.nome = nome
```

```
        self.idade = idade
```

```
Pessoa = ("João", 18)
```

g) Associação

```
class Musica():
```

```
    def _init_(self, nome, artista):
```

```
        self.nome = nome
```

```
        self.artista = artista
```

```
class Artista():
```

```
    def _init_(self, id, nome):
```

```
self.id = id  
self.nome = nome  
self.cancoes = list()
```

```
def adicionarMusica(self, musica):  
    self.cancoes.append(musica)
```

```
def mostrarMusicas(self):  
    for a in self.cancoes:  
        print("Nome:", a[1])  
    print()
```

```
artistaAurora = Artista(2, "Aurora")
```

```
musicaA = [1, "The Seed"]  
musicaB = [2, "Runaway"]  
musicaC = [3, "Warrior"]
```

```
artistaAna.adicionarMusica(musicaA)  
artistaAna.adicionarMusica(musicaB)  
artistaAna.adicionarMusica(musicaC)
```

```
artistaAurora.mostrarMusicas()
```

h) Relacionamento TODO-PARTE

```
class Pedido():  
    def __init__(self, id, *itens):  
        self.id = id
```

```
self.itens = list(map(lambda x : x._dict_, itens))
```

```
def pago(self):  
    print("Produtos:")  
    for a in self.itens:  
        print(a["nome"])  
    print("Total a pagar:", sum(list(map(lambda x : x["preco"], self.itens))))
```

```
class itemPedido():  
    def _init_(self, nome, preco):  
        self.nome = nome  
        self.preco = preco
```

```
pedido1 = Pedido(1, itemPedido("Refrigerante dois litros", 10),  
itemPedido("Arroz", 4), itemPedido("Morango", 5))  
pedido1.pagamento()
```

4ª Questão (20 Escores)

Escreva em Python uma classe Ponto que possui os atributos inteiros x e y. Escreva uma classe Reta que possui dois pontos a e b. Escreva os métodos construtores para a classe Ponto e para a Classe Reta. Escreva os métodos get e set para acessar e alterar os atributos da classe Ponto e da classe Reta. Escreva um método distancia que retorna um valor real da distancia entre os dois pontos da reta.

```
import math
```

```
class Ponto(object):  
    def init(self,x,y):  
        self.x=x  
        self.y=y
```

```
def getponto(self):  
    print(f'\nx: {self.x}\t y: {self.y}')
```



```
def set_x(self,x):  
    self.x=x
```



```
def set_y(self,y):  
    self.y=y
```



```
class Reta(object):  
    def init(self,ax,ay,bx,by):  
        self.ax = ax  
        self.ay = ay  
        self.bx = bx  
        self.by = by  
  
    def get_distancia(self):  
        d=math.sqrt((self.bx - self.ax) * (self.bx - self.ax)+(self.by - self.ay) *  
        (self.by - self.ay))  
        print("A distância é igual a {}".format(d))  
  
    def set_a(self,a):  
        self.a=a  
  
    def set_b(self,b):  
        self.b=b
```



```
if name == 'main':
```



```
x = int(input('escreva o valor de X: '))
y = int(input('escreva o valor de Y: '))
print('Forme o ponto a com ax e ay')
ax = int(input('escreva o valor de ax: '))
ay = int(input('escreva o valor de ay: '))
print('Forme o ponto b com bx e by')
bx = int(input('escreva o valor de bx: '))
by = int(input('escreva o valor de by: '))
oz = Ponto(x, y)
oz.getponto()
iz = Reta(ax,ay,bx,by)
iz.get_distancia()
```