

DEPARTAMENTO DE TELEMÁTICA
DISCIPLINA: PROGRAMAÇÃO ORIENTADA A OBJETO
LISTA EXERCÍCIO

ALUNO: Paulo Tomas Barboza da Silva

Data: 30/09/2021

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 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 inscrever 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 Animal():
    def __init__(self, nome, cor):
        self.__nome = nome
        self.__cor = cor

    def comer(self):
        print(f"O {self.__nome} está comendo")
```

b) Encapsulamento;

```
class Funcionario:
    def __init__(self, nome, cargo, valor_hora_trabalhada):
        self.nome = nome
        self.cargo = cargo
        self.valor_hora_trabalhada = valor_hora_trabalhada
        self.__horas_trabalhadas = 0
        self.__salario = 0

    def registra_hora_trabalhada(self):
        self.horas_trabalhadas += 1

    def calcula_salario(self):
        self.__salario = self.__horas_trabalhadas * self.valor_hora_trabalhada
```

c) Polimorfismo;

```
class ObjetoGrafico(object):
    def __init__(self, centro):
        super(ObjetoGrafico, self).__init__()
        self._centro = centro

    @abstractmethod
    def desenha(self):
        pass

    def apaga(self):
        self.setPenColor(self.BACKGROUND_COLOR)
        self.desenha()
        self.setPenColor(self.FOREGROUND_COLOR)

    def movePara(self, p):
        self.apaga()
        self._centro = p
        self.desenho()
```

d) Variáveis de Instância;

```
class testClass():
    list = ['foo']
    def __init__(self):
        self.list = []
        self.list.append('thing')
```

```
x = testClass()
print x.list
print testClass.list
del x.list
print x.list
```

e) Métodos construtores

```
class Carro(object):
    def __init__(self, modelo, ano):
        self.modelo = modelo
        self.ano = ano

meuCarro = Carro("Jeep 4x4", 2015)

print("Modelo: %s" % meuCarro.modelo)
print("Ano: %d" % meuCarro.ano)
```

f) Dependência

```
import time
class write():
    def escreve (self):
        print (time.ctime(time.time()))
class receive ():
    def __init__(self, x):
        print (x.escreve)

x = write()
y = receive(x)
```

g) Associação

```
class A(object):
    def __init__(self, a, b, c):
        self.a = a
        self.b = b
        self.c = c

    def addNums():
        self.b + self.c

class B(object):
    def __init__(self, d, e):
        self.d = d
        self.e = e

    def addAllNums(self, Ab, Ac):
        x = self.d + self.e + Ab + Ac
        return x

ting = A("yo", 2, 6)
ling = B(5, 9)

print ling.addAllNums(ting.b, ting.c)
```

h) Relacionamento TODO-PARTE

```
class A(object):
    def __init__(self, a, b, c):
        self.a = a
        self.b = b
        self.c = c

    def addNums():
        self.b + self.c

class B(object):
    def __init__(self, d, e):
        self.d = d
        self.e = e
        self.A = A("yo", 2, 6)
```

```

def addAllNums(self):
    x = self.d + self.e + self.A.b + self.A.c
    return x

ling = B(5, 9)

print ling.addAllNums()

```

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.

```

class Ponto():
    def __init__(self, x , y):
        self.x = x
        self.y = y
    def setX(self,new):
        self.x = new
    def setY(self,new):
        self.y=new
    def getX(self):
        return self.x
    def getY(self):
        return self.y

class Reta():
    def __init__(self, um, dois):
        self.um = um
        self.dois = dois
    def setUm(self, new):
        self.um =new
    def setDois(self, new):
        self.dois = new
    def getUm(self):
        return self.um
    def getDois(self):
        return self.dois
    def distancia (self):
        dist= (((self.getDois().getX() - self.getUm().getX())**2)+((self.getDois().getY() - self.getUm().getY())**2)) **
        0.5
        return dist

A = Ponto(1, 6)
B = Ponto(4, 8)

straightA= Reta(A, B)

print("\nA distância entre os pontos A(1,6) e B(4,8) é:", straightA.distancia())

```