INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
CEARA

DEPARTAMENTO DE TELEMÁTICA
DISCIPLINA: PROGRAMAÇÃO ORIENTADA A OBJETO
LISTA EXERCICIO

ALUNO: Davison Lennon de Lima Sousa **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.** (f)

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. (f)

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. (v)

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

a) Herança;

A)
```
class Ponto():
    def __init__(self, x, y):
        self.x = x
        self.y = y

class Circulo(Ponto):
    def __init__(self, x, y, r):
        super().__init__(x, y)
        self.r = r
```

b) Encapsulamento;

B)
```
class Ponto():
    def __init__(self, x, y):
        self.__x = x
        self.__y = y

    @property
    def x(self):
        return self.__x

    @property.setter
    def x(self, x):
        raise ValueError("Não eh possivel alterar o valor da propriedade 'x'.
              Utilize a funcao 'setx()'.")

    @property
    def y(self):
        return self.__y

    @property.setter
    def y(self, y):
        raise ValueError("Não eh possivel alterar o valor da propriedade 'y'.
              Utilize a funcao 'sety()'.")

    def setx(self, x):
        self.__x = x

    def sety(self, y):
        self.__y = y
```

c) Polimorfismo;

C)
```
class Ponto():
    def __init__(self, x, y):
        self.__x = x
        self.__y = y

    def imprime_posicao(self):
        print( "(" + str(self.__x) + "," + str(self.__y) + ")" )

class Circulo(Ponto):
    def __init__(self, x, y, r):
        super().__init__(x, y)
        self.__r = r

    def imprime_posicao(self):
        print( "(" + str(self.__x) + "," + str(self.__y) + ")" )
```

d) Variáveis de Instância;

F)
```
class Ponto():
    def __init__(self, x, y):
        self.__x = x
        self.__y = y

class Retangulo():
    def __init__(self, x1, y1, x2, y2):
        self.__p1 = Ponto(x1, y1)
        self.__p2 = Ponto(x2, y2)
```

e) Métodos construtores

D)
```
import math

class Ponto():
    def __init__(self, x, y):
        self.__x = x
        self.__y = y

    def distancia_da_origem(self):
        dx = self.__x
        dy = self.__y

        return math.sqrt(dx**2 + dy**2)
```

E)
```
import math

class Ponto():
    def __init__(self, x, y):
        self.__x = x
        self.__y = y

    def argumento(self):
        dx = self.__x
        dy = self.__y

        return math.atan2(dy, dx)
```

H)
```
class Locomotiva():
    def __init__(self, c, a, p):
        self.__comprimento = c
        self.__altura = a
        self.__potencia = p

class Vagao():
    def __init__(self, c, a):
        self.__comprimento = c
        self.__altura = a

class Trem():
    def __init__(self, loc, vag):
        self.__locomotiva = loc
        self.__vagao = vag
```

f) Dependência

g) Associação

G)
```
class Conta():
    def __init__(self, agencia, n_conta):
        self.__agencia = agencia
        self.__n_conta = n_conta

class ContaCorrente(Conta):
    def __init__(self, agencia, n_conta):
        super().__init__(agencia, n_conta)

class Cliente():
    def __init__(self, nome, agencia, n_conta):
        self.__conta_corrente = ContaCorrente(agencia, n_conta)
        self.__nome = nome
```

h) Relacionamento TODO-PARTE

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():
    def __init__(self, x, y):
        self.__x = x
        self.__y = y

    @property
    def x(self):
        return self.__x

    @x.setter
    def x(self, valor):
        self.__x = valor

    @property
    def y(self):
        return self.__y

    @y.setter
    def y(self, valor):
        self.__y = valor

class Reta():
    def __init__(self, ponto_a, ponto_b):
        self.__a = ponto_a
        self.__b = ponto_b

    @property
    def a(self):
        return self.__a

    @a.setter
    def a(self, ponto):
        self.__a = ponto

    @property
    def b(self):
        return self.__b

    @b.setter
    def b(self, ponto):
        self.__b = ponto

    def distancia(self):
        dx = self.__a.__x - self.__b.__x
        dy = self.__a.__y - self.__b.__y

        return math.sqrt(dx**2 + dy**2)
```