INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
CEARA

DEPARTAMENTO DE TELEMÁTICA
DISCIPLINA: PROGRAMAÇÃO ORIENTADA A OBJETO
LISTA EXERCICIO

ALUNO: RAFAEL ALEXANDRE LINHARES Data: 20/09/2021

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 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;

```python
class Pessoa:
    def __init__(self, nome, idade, data_nasc):
        self.nome = nome
        self.idade = idade
        self.data_nasc = data_nasc

class Aluno(Pessoa):
    def __init__(self, nome, idade, data_nasc, matr, curso):
        super().__init__(nome, idade, data_nasc)
        self.matr = matr
        self.curso = curso
```

b) Encapsulamento;

```python
class User:
    def __init__(self, username, password):
        self.username = username
        self.__password = password

    @property
    def password(self):
        return self.__password

    @password.setter
    def password(self, new_pass):
        raise ValueError("Impossible to change password directly, try calling
set_new_pass() method.")
```

```python
    def set_new_pass(self):
        login = input()
        if login == self.__password:
            print("Type new password: ")
            new_pass = input()
            print("Confirm password: ")
            confirm_pass = input()
            if new_pass == confirm_pass:
                self.__password = new_pass
        else:
```

```
            return "Wrong password."
```

c) Polimorfismo;

```
class Ave():
    def __init__(self, nome):
        self.nome = nome

    def movimento(self):
        print("Aves voam.")

class Peixe():
    def __init__(self, nome):
        self.nome = nome

    def movimento(self):
        print("Peixes nadam.")
```

```
    def movimento(self):
        print("Peixes nadam.")

animal1 = Ave("papagaio")
animal1.movimento()

animal2 = Peixe("peixe-espada")
animal2.movimento()
```

d) Variáveis de Instância;

```
class Triangulo:
    def __init__(self, ang1, ang2, ang3):
        self.ang1 = ang1
        self.ang2 = ang2
        self.ang3 = ang3
```

e) Métodos construtores

```
class Aluno:
    def __init__(self, nome, matr, curso):
        self.nome = nome
        self.matr = matr
        self.curso = curso
```

f) Dependência

```python
class Curso:
    def __init__(self, nome, carga):
        self.nome = nome
        self.carga = carga

class Aluno:
    def __init___(self, nome, matr, curso=type(Curso())):
```

g) Associação

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.

```python
import math

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

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

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

    def get_ponto(self):
        return "({}, {})".format(self.x, self.y)
```

```
class Reta:
    def __init__(self,a,b):
        self.a=a
        self.b=b

    def set_a(self,a):
        self.a=a

    def set_b(self,b):
        self.b=b

    def get_distancia(self):
```