1.

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

2.

a) F
b) V
c) V
d) V
e) F
f) F
g) F
h) V
i) V
j) F

3.

a)

```
# Herança
class Entidades:
    def __init__(self, tipo):
        self.tipo = tipo
# Classe Inimigo Herda a classe Entidades
class Inimigo(Entidades):
    def __int__(self, tipo, vida, dano):
        super().__init__(self, tipo)
        self.vida = vida
```

b)

```
# Encapsulamento
class Player:
    __vida = 5
```

c)

```
# Polimorfismo
class BemVindo:
    def mensagem(self):
        print('Seja bem vindo(a)!')

class BemVindoNovamente(BemVindo):
    def mensagem(self):
        print('Eai mano(a)')
```

d)

```
# Variaveis de Instancia
class Player():
    name = ''
    id = ''
    save = ''
```

e)

```
# Métodos Construtores
class Player:
    def __init__(self, name):
        self.name = name
```

f)

```
# Dependência
class Pular():
    def pularUmaVez(self):
        print('Pulei')

class Player2():
    def __init__(self, movimento):
        self.movimento = movimento
    def movimentando(self):
        self.movimento.pularUmaVez()
```

g)

```
# Associação
class Player():
    def __init__(self, name):
        self.__name = name
        self.__gear = None

class Weapon():
    def __init__(self, tipo):
        self.__tipo = tipo
```

```
    def usar(self):
        print('Usando a arma')
```

h)

```
# Relacionamento TODO-PARTE

class Lojinha():
    def __init__(self):
        self.items = []

    def new_item(self, items):
        self.items.append(items)

    def list_items(self):
        for x in self.items:
            print(x.nome, x.valor)

class Item():
    def __init__(self, nome, valor):
        self.nome = nome
        self.valor = valor

my_hand = Lojinha()
item1 = Item('Luvas', 15)
item2 = Item('Espada', 1490)
my_hand.new_item(item1)
my_hand.new_item(item2)
```

4.

```
from math import sqrt

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, x):
        self.__x = x

    @property
    def y(self):
```

```
        return self.__y

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


class Reta:
    def __init__(self, a: type(Ponto), b: type(Ponto)):
        self.__a = a
        self.__b = b

    def distancia(self):
        distancia = sqrt((self.__a.x - self.__a.x)**2 + (self.__a.y -
self.__b.y)**2)
        return distancia

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

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

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

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