```
# Programação Orientada a Objetos
# AC03 ADS-EaD - Implementação de classes, herança, polimorfismo e
lançamento de exceções.
#
# Email Impacta: _____________@aluno.faculdadeimpacta.com.br


class Produto:
        """
        Classe Produto: deve representar os elementos básicos de um
produto.
        """

        def __init__(self, nome, preco):
                """
                Inicializa os atributos privados nome e preco.

                Esses atributos não devem ser declarados diretamente
no construtor. Ao invés
                disso, utilize os setters "nome" e "preco" para
inicializá-los indiretamente,
                pois dessa forma eles serão validados (veja a
descrição dos setters na sequência).
                """
                pass


        @property
        def nome(self):
                """
                Property nome: devolve (retorna) o valor do atributo
privado nome.
                """
                pass


        @property
        def preco(self):
                """
                Property preco: devolve (retorna) o valor do atributo
privado preco.
                """
                pass


        @nome.setter
        def nome(self, novo_nome):
                """
                Setter nome: recebe um novo_nome e atualiza o valor do
atributo        privado
                nome com esse valor.

                Antes de modificar o valor do atributo privado nome,
verifique se o tamanho
                da string novo_nome é maior que zero. Se for igual a
zero, lance uma exceção
```

```
                do tipo ValueError.
                """
                pass


        @preco.setter
        def preco(self, novo_preco):
                """
                Setter preco: recebe um novo_preco e atualiza o valor
do atributo privado
                preco com esse valor.

                Antes de modificar o valor do atributo privado preco,
verifique se seu valor
                é do tipo int ou do tipo float (utilize a função
isinstance para fazer essa
                verificação). Caso não seja de um desses tipos, lance
uma exceção do tipo
                TypeError. Caso novo_preco seja int ou float,
verifique se seu valor é maior
                ou igual a zero. Se for menor que zero (negativo),
lance uma exceção do tipo
                ValueError.
                """
                pass


        def calcular_preco_com_frete(self):
                """
                Método que calcula o valor final do produto com o
frete incluso.
                Deve devolver (retornar) o valor do atributo privado
preco.
                """
                pass


class ProdutoFisico:
        """
        Classe ProdutoFisico: deve representar os elementos básicos de
um produto físico.
        Esta classe herda da classe Produto.
        """

        def __init__(self, nome, preco, peso):
                """
                Inicializa nome e preco utilizando o construtor da
superclasse Produto,
                (use a função super()), e também inicializa o atributo
privado peso.

                O atributo privado peso não deve ser declarado
diretamente no construtor.
                Ao invés disso, utilize o setter "peso" para
inicializá-lo indiretamente,
                pois dessa forma ele será validado.
                """
```

```
                pass


        @property
        def peso(self):
                """
                Property peso: devolve (retorna) o valor do atributo
privado peso.
                """
                pass


        @peso.setter
        def peso(self, novo_peso):
                """
                Setter peso: recebe um novo_peso e atualiza o valor do
atributo        privado
                peso com esse valor (que representa o peso do produto
em gramas).

                Antes de modificar o valor do atributo privado peso,
verifique se seu valor
                é do tipo int (utilize a função isinstance para fazer
essa verificação),
                caso contrário lance uma exceção do tipo TypeError.
                Caso novo_peso seja do tipo int, verifique se seu
valor é maior que zero.
                Se for menor ou igual a zero, lance uma exceção do
tipo ValueError.
                """
                pass


        def peso_em_kg(self):
                """
                Método que calcula o peso do produto em quilogramas.
                Deve devolver (retornar) o valor do peso convertido em
quilogramas.
                Exemplos:
                        - Se o valor do atributo privado peso for
1000, este método retorna 1;
                        - Se o valor do atributo privado peso for
7500, este método retorna 7.5;
                        - Se o valor do atributo privado peso for 600,
este método retorna 0.6;
                """
                pass


        def calcular_preco_com_frete(self):
                """
                Método que calcula o valor final do produto físico com
o frete incluso.
                Para cada quilograma no peso do produto, acrescente
R$5 ao seu valor final.

                Deve devolver (retornar) o valor final do produto
```

```
acrescido do frete (que depende
                do peso do produto em quilogramas, conforme descrito
acima).

                Exemplos:
                        - Se o produto (preço) custa R$100 e seu peso
é 1000 gramas, retorna R$105;
                        - Se o produto (preço) custa R$50 e seu peso é
2500 gramas, retorna R$62.5;
                        - Se o produto (preço) custa R$10 e seu peso é
100 gramas, retorna R$10.5;

                """
                pass


class ProdutoEletronico:
        """
        Classe ProdutoEletronico: deve representar os elementos
básicos de um produto eletrônico.
        Esta classe herda da classe ProdutoFisico.
        """

        def __init__(self, nome, preco, peso, tensao, tempo_garantia):
                """
                Inicializa nome, preco e peso utilizando o construtor
da superclasse ProdutoFisico,
                (use a função super()), e também inicializa os
atributos privados tensao e
                tempo_garantia da seguinte forma:
                        - O atributo privado tensao não deve ser
declarado diretamente no
                        construtor.     Ao invés disso, utilize o
setter "tensao" para inicializá-lo
                        indiretamente, pois dessa forma ele será
validado.

                        - O atributo privado tempo_garantia deve ser
inicializado diretamente
                        no construtor, sem necessidade de validação.
                """
                pass


        @property
        def tensao(self):
                """
                Property tensao: devolve (retorna) o valor do atributo
privado tensao.
                """
                pass


        @property
        def tempo_garantia(self):
                """
                Property tempo_garantia: devolve (retorna) o valor do
```

```
atributo privado tempo_garantia.
                """
                pass


        @tensao.setter
        def tensao(self, nova_tensao):
                """
                Setter tensao: recebe uma nova_tensao e atualiza o
valor do atributo privado
                tensao com esse valor (que representa a tensão de um
aparelho eletrônico,
                com os seguintes valores possíveis: 0, indicando que o
produto é bivolt,
                127 ou 220).

                Antes de modificar o valor do atributo privado tensao,
verifique se seu valor
                é do tipo int (utilize a função isinstance para fazer
essa verificação),
                caso contrário lance uma exceção do tipo TypeError.
                Caso nova_tensao seja do tipo int, verifique se seu
valor é igual a 0, ou
                127 ou 220. Caso nova_tensao seja diferente desses
valores, lance uma
                exceção do tipo ValueError.
                """
                pass


        def calcular_preco_com_frete(self):
                """
                Método que calcula o valor final do produto eletrônico
com o frete incluso.
                O cálculo é o mesmo que o produto físico, mas deverá
ser acrescido 1%
                ao valor final do frete.
                Dica: você pode reaproveitar o método
calcular_preco_com_frete() da
                superclasse (a classe ProdutoFisico), através da
função super(). Ou seja,
                obtenha o valor do frete do produto físico, depois
acrescente 1% e devolva
                (retorne) esse valor.

                Deve devolver (retornar) o valor final do produto
acrescido do frete (será
                o mesmo valor com frete do produto físico, com o
acréscimo de 1%).

                Exemplos:
                        - Se o produto (preço) custa R$100 e seu peso
é 1000 gramas, retorna R$106.05;
                        - Se o produto (preço) custa R$50 e seu peso é
2000 gramas, retorna R$60.6;
                        - Se o produto (preço) custa R$10 e seu peso é
800 gramas, retorna R$14.14;
```

```
                """
                pass


class Ebook:
        """
        Classe Ebook: deve representar os elementos básicos de um
ebook (livro digital).
        Esta classe herda da classe Produto.
        """

        def __init__(self, nome, preco, autor, numero_paginas):
                """
                Inicializa nome e preco utilizando o construtor da
superclasse Produto,
                (use a função super()), e também inicializa os
atributos privados autor e
                numero_paginas da seguinte forma:
                        - O atributo privado autor deve ser
inicializado diretamente
                        no construtor, sem necessidade de validação.

                        - O atributo privado numero_paginas não deve
ser declarado diretamente no
                        construtor.     Ao invés disso, utilize o
setter "numero_paginas" para
                        inicializá-lo indiretamente, pois dessa forma
ele será validado.
                """
                pass


        @property
        def nome_exibicao(self):
                """
                Property nome_exibicao: devolve (retorna) uma string
com o nome e autor
                do livro no seguinte formato (sem aspas): "Nome
(Autor)"

                Exemplos:
                        - Se nome é "Aprendendo Python" e autor é "Ana
Maria", deve devolver (retornar)
                         uma string com: "Aprendendo Python (Ana
Maria)";

                        - Se nome é "O senhor dos anéis" e autor é "J.
R. R. Tolkien", deve
                        devolver (retornar) uma string com: "O senhor
dos anéis (J. R. R. Tolkien)";
                """
                pass


        @property
        def numero_paginas(self):
                """
```

```
                Property numero_paginas: devolve (retorna) o valor do
atributo
                privado numero_paginas.
                """
                pass


        @numero_paginas.setter
        def numero_paginas(self, valor):
                """
                Setter numero_paginas: recebe um valor e atualiza o
atributo privado
                numero_paginas com esse valor.

                Antes de modificar o valor do atributo privado
numero_paginas, verifique
                se o valor é maior que zero. Caso contrário (se valor
for menor ou igual
                a zero), lance um erro do tipo ValueError.
                """
                pass
```