Linguagens de Programação ULP452-14
Exame Modelo 2

Nº Mecanográfico: _____

Nome: _____

1. Considere a seguinte classe que define um objeto que encapsula uma data:

```
import datetime
class Data:
    '''Define uma data com dia, mês e ano'''
    def __init__(self, dia, mes, ano):
        self.dia = dia
        self.mes = mes
        self.ano = ano

    def getDiffDias(self, other):
        '''Calcula a diferença de dias entre as datas self e other'''
        diff = datetime.datetime(self.ano, self.mes, self.dia)\
               - datetime.datetime(other.ano, other.mes, other.dia)
        return diff.days
        #Nota: a '\' permite quebrar uma linha em duas, mantendo a indentação.
```

a) Instancie dois objetos Data em duas variáveis `hoje` e `haUmAno`, respetivamente com as datas de hoje e de há um ano.

b) Imprima no ecrã quantos dias passaram desde `haUmAno` até `hoje`.

c) Crie um método privado que faça uma pre-validação da data inserida (devolve True/False). Para simplificar, garanta apenas que o dia está no intervalo [1, 31], o mês [1, 12] e o ano positivo (d.c.).

d) Faça o override ao método __str__ de forma a poder imprimir corretamente a data. Por exemplo: `print(str( hoje ))` deverá imprimir "26/1/2020".

e) Crie um método estático onde se passe um argumento com uma str no formato "26/1/2020" e que devolva um objeto do tipo `Data` com a respetiva data.

2. Num novo ficheiro, crie uma classe `EventoAbstrato`:

a) A classe deve armazenar dois atributos inicializados no construtor/inicializador
`data` – atributo público do tipo `Data` (ponto anterior).
`name` – atributo privado do tipo `str`.

b) A classe tem um método abstrato `duration_in_days` que irá devolver a duração do evento em dias (inteiro).

c) Crie um property com o nome `duration` que deverá devolver uma string com a duração do evento. Ex: Se a duração for 1 devolve "1 dia". Se a duração for 4 devolve "4 dias"

Nota: O método da alínea b) é abstrato, mas quando o objeto é instanciado ele já existe!

d) Faça o *override* ao método __str__ de forma a poder imprimir corretamente o evento. Por exemplo: "Exame|26/1/2020|1 dia".

e) Crie um método estático (is_overlapped) onde, dados 2 eventos (nos argumentos), retorna True se houver sobreposição das datas, caso contrário retorna False.
Nota: tem acesso ao método `duration_in_days` que retorna o número de dias do evento e ao método `getDiffDias` de `data` que retorna a diferença de dias entre duas datas.

3. Num novo ficheiro, crie duas classes:

a) Classes: `Evento_de_1_dia` e `Evento_de_2_dias`, que implementam o método abstrato definido no ponto anterior. Devem descrever respetivamente a duração de 1 dia e 2 dia.

b) Teste criando uma função (fora da classe) para onde se passa uma lista de eventos e que devolva a média de todas as durações constantes da lista.

4. Num novo ficheiro:

a) Crie uma classe `Agenda` que deve ser inicializada com uma lista privada - `eventos`. Nesta lista vazia irão ser colocados os eventos da agenda.

b) Crie um método `add_evento` que permita adicionar eventos novos à agenda.

c) Melhore o método anterior criando uma exceção para quando o evento (candidato) a ser inserido estiver sobreposto com algum já constante da lista. Em caso de sobreposição deve ser levantada uma exceção e o evento não será inserido.

Crie código para testar e apanhar a exceção gerada.

Nota: Tem funções dos exercícios anteriores que permitem saber se há sobreposição. Se não completou os exercícios anteriores, pode assumir que estão operacionais.

5. Num novo ficheiro, continuando a classe Agenda:

a) Adicione um método de instância `itera_eventos_cronologicamente`.

- Este método deverá implementar uma função geradora ou retornar uma expressão geradora de eventos armazenados na lista eventos.
- Os eventos deverão ser gerados cronologicamente.

Nota: Para ordenar uma lista pode consultar:
Se necessitar conjugar múltiplas keys, p.ex: ano, mês, dia (por esta ordem) pode conjugar da seguinte forma:

lista_ordenada = sorted(my_list, key=lambda i: (i.ano, i.mes, i.dia), reverse=False)