

Atividade Avaliativa REO2 : Paradigma Orientado a Objetos

Descrição

(100 pts.) Criar um sistema *simplificado de fins didáticos* de um prontuário médico, na linguagem Python, versão 3.*, com orientação a objetos. O sistema deve ter as seguintes características, agrupadas por classes:

- Classes, cada classe deve estar presente em um arquivo .py, conforme o nome indicado.
 - Pessoa (arquivo `pessoa.py`)
 - Essa classe deve ser abstrata e ter os seguintes métodos:
 - construtor, que recebe o nome da pessoa e armazena em uma variável privada;
 - um método *getter*, para retornar o nome da pessoa;
 - `definir_id`, método abstrato, que recebe um parâmetro (*identificacao*) a ser definido nas classes filhas.
 - Paciente (`paciente.py`)
 - Essa classe deve herdar a classe¹ Pessoa;
 - Deve ter uma variável de classe que seja incrementada a cada novo paciente instanciado (método construtor), e decrementada a cada paciente desinstanciado (método destrutor). Essa variável armazena a quantidade de objetos instanciados, no momento, da classe Paciente.
 - Implementar os seguintes métodos:
 - `definir_id`, que recebe um id de no máximo 5 caracteres alfa-numéricos. Caso o id tenha mais de 5 caracteres, exibir mensagem de erro, mas não interromper a execução do programa;
 - `definir_prontuário`, que recebe um objeto da classe `Prontuario`;
 - `pacientes_ativos`, um método de classe, que retorna o valor da variável de classe que armazena a quantidade de pacientes ativos, no momento.
 - Medico (`medico.py`)
 - Essa classe deve herdar a classe Pessoa.
 - Implementar os seguintes métodos:
 - `definir_id`, que recebe um id de no máximo 3 caracteres alfa-numéricos; Caso o id tenha mais de 3 caracteres, exibir mensagem de erro, mas não interromper a execução do programa;
 - `nome_medico`, que retorna o nome do médico.
 - Medicamento (`medicamento.py`)
 - Implementar os seguintes métodos:
 - construtor que recebe o nome do medicamento e armazena em variável privada;
 - `nome_medicamento`, que retorna o nome do medicamento.
 - Prontuário (`prontuario.py`)

¹ Para referenciar a classe Pessoa, que estará no arquivo `pessoa.py`, basta fazer **`from pessoa import Pessoa`** no início do código da classe que a herdar.

- Deve conter uma variável privada que armazenará uma lista de procedimentos ou registros no prontuário
- Deve implementar os seguintes métodos:
 - construtor, que inicia a variável privada de lista de procedimentos como lista vazia;
 - `insere_procedimento`, que recebe os parâmetros listados a seguir, cria um dicionário com esses valores e faz o `append` desse dicionário na lista de procedimentos. As chaves do dicionário podem ser iguais aos nomes dos parâmetros listados a seguir:
 - `medicamento`: objeto da classe `Medicamento`;
 - `posologia`: string indicando hipotética posologia do remédio utilizado;
 - `medico`: objeto da classe `Medico`;
 - `data`: string da data do procedimento no seguinte formato "dd-mm-aaaa", por exemplo '15/06/2020'. Dica em ²;
 - `hora`: string com horário do procedimento no formato "hh:mm", por exemplo '10:10'.
 - `exibe_prontuario`, que exibe todos os procedimentos armazenados na lista de procedimentos. Cada posição da lista terá um dicionário com as informações de um procedimento armazenado. Veja o caso de teste neste documento para definir o formato de saída deste método.

Execução

Seu programa será executado de maneira interativa, pelo docente, de forma a verificar se todos os requisitos foram atendidos. Para testar o seu desenvolvimento, siga o procedimento a seguir, semelhante ao que o docente realizará para avaliar a atividade:

Abra o terminal do seu computador, vá ao diretório em que se encontram os arquivos das classes foram criadas e digite:

```
python3
```

O interpretador Python será aberto, então, siga a lista de comandos a seguir e certifique-se de que cada comando funcionará:

Caso de teste 1

```
>>> from paciente import Paciente
>>> from medico import Medico
>>> from medicamento import Medicamento
>>> from prontuario import Prontuario
>>> paciente1 = Paciente('Paciente1')
>>> prontuario1 = Prontuario()
>>> paciente2 = Paciente('Paciente2')
>>> prontuario2 = Prontuario()
>>> medicamento1 = Medicamento('Omeprazol')
>>> medicamento2 = Medicamento('Tylenol')
>>> medicamento3 = Medicamento('Dipirona')
```

² <https://www.journaldev.com/23365/python-string-to-datetime-strptime> ou <https://dicasdepython.com.br/python-como-converter-string-em-date/>

```

>>> medico1 = Medico('Medico1')
>>> medico2 = Medico('Medico2')
>>> paciente1.definir_prontuario(prontuario1)
>>> paciente2.definir_prontuario(prontuario2)
>>> paciente1.prontuario.insere_procedimento(medicamento1, '68mcg',
medico1, "15-06-2020", "10:10")
>>> paciente1.prontuario.insere_procedimento(medicamento2, '100mg',
medico1, "15-06-2020", "11:40")
>>> paciente2.prontuario.insere_procedimento(medicamento2, '18mcg',
medico2, "14-06-2020", "12:00")
>>> paciente2.prontuario.insere_procedimento(medicamento3, '20mg',
medico2, "15-06-2020", "12:05")
>>> paciente1.prontuario.exibe_prontuario()
Omeprazol - 68mcg - Medico1 - 2020-06-15 - 10:10:00
Tylenol - 100mg - Medico1 - 2020-06-15 - 11:40:00
>>> paciente2.prontuario.exibe_prontuario()
Tylenol - 18mcg - Medico2 - 2020-06-14 - 12:00:00
Dipirona - 20mg - Medico2 - 2020-06-15 - 12:05:00
>>> Paciente.pacientes_ativos()
2
>>> del paciente2
>>> Paciente.pacientes_ativos()
1

```

Observação, o output do método `exibe_prontuario` é uma concatenação dos valores do dicionário de procedimentos, intercalados por ' - ', um procedimento por linha.

Entrega

A entrega deve ser feita, até o prazo permitido no Campus Virtual, 28/06/2020, anexando os 5 arquivos .py.