INSTITUTO FEDERAL
Catarinense
Campus Blumenau

INB0929 - Programação 1

Técnico em Informática integrado ao Ensino Médio

Prof. Dr. Paulo César Rodacki Gomes

Lista de exercícios - 07

Associação entre classes - parte 3

1 Descrição

O objetivo deste exercício é implementar a relação entre uma turma e os alunos matriculados nela. Para isto crie as duas classes mostradas no diagrama da figura 1:

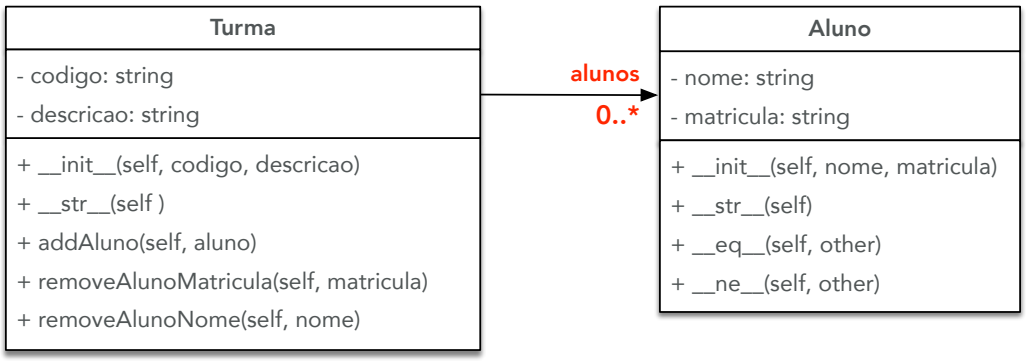

Figura 1: Diagrama de classes a ser implementado.

2 Roteiro de execução/observações

1. Todos os atributos das duas classes são privados (crie propriedades ***setter*** e ***getter*** para todos os atributos;

2. Além do método `__str__`, a classe classe `Aluno` deve implementar os seguintes métodos especiais:

   - `__eq__`: este método realiza a operação de comparação igualdade entre dois alunos. Duas instâncias de aluno (ou seja, dois objetos da classe `Aluno`) serão considerados iguais se tiverem o mesmo nome e o mesmo número de matrícula. Este método, portanto, emula o comportamento do operador `==`.
   - `__ne__`: este método realiza a operação de comparação de ***desigualdade*** entre dois alunos. Duas instâncias de aluno serão considerados diferentes se tiverem nomes ***ou*** números de matrícula diferentes. Este método, portanto, emula o comportamento do operador `!=`.
   - Consulte a documentação da linguagem para saber mais sobre este método especial.

3. A classe `Turma` tem os seguintes atributos: código, descrição e alunos (este é ***uma lista de objetos da classe*** `Aluno`). Os métodos a seguir devem ser implementados:

   - `addAluno(self, aluno)`: adiciona uma instância de `Aluno` na lista de alunos na turma. **Atenção**: você deve pode inserir o mesmo aluno mais de uma vez. Para evitar isso, você deve percorrer toda a lista de alunos e verificar de alum deles é igual ao aluno que você está tentando inserir na lista, faça isso da seguinte forma: `if novo_aluno == aluno_da_lista:` ou `if novo_aluno != aluno_da_lista:`. Note que, neste caso, os operadores `==` e `!=` irão automaticamente chamar os métodos `__eq__` ou `__ne__` da classe `Aluno`, respectivamente.

Se a lista já tiver um aluno igual ao novo aluno, seu método deve simplesmente ignorar e não inserir o novo objeto aluno na turma;

- `removeAlunoMatricula(self, matricula)`: remove um aluno da turma, tendo sido passado seu número de matrícula;
- `removeAlunoNime(self, nome)`: remove um aluno da turma, tendo sido passado seu nome;

4. As duas classes devem ter o método especial `__str__` que retorna uma string com a descrição textual do objeto. No caso da classe `Turma`, o método deve percorrer a lista de alunos e exibir os dados de todos os alunos;

5. Vamos tentar criar um programa com 3 arquivos fonte conforme descrito a seguir:

   - Crie um arquivo chamado `Aluno.py` contendo apenas a classe `Aluno`
   - Crie um arquivo chamado `Turma.py` para a classe `Turma`.
     No início deste arquivo, coloque a seguinte instrução:
     `from Aluno import Aluno`
   - Crie o arquivo chamado `TesteTurma.py`.
     No início deste arquivo, coloque as seguintes instruções:
     `from Aluno import Aluno`
     `from Turma import Turma`

6. No arquivo `TesteTurma.py`, crie alguns objetos objetos da classe `Turma`. Depois vá criando alunos e matriculando em turmas até que a palavra "Fim" seja informada para o número da conta. Após isso faça testes removendo alunos e imprimindo as turmas. Por exemplo:

```
#Turmas:
programacaoI = Turma("POO", "Programacao I")
historia = Turma("HIS", "História")
engenhariaSoft = Turma("ENG", "Engenharia de Software")

#Alunos:
joao = Aluno('001', 'João da Silva')
maria = Aluno('002', 'Maria dos Santos')
henrique = Aluno('003', 'Henrique Mattos')
pedro = Aluno('004', 'Pedro Gonçalves')
felipe = Aluno('004', 'Felipe Martins')

#Insercao de alunos
programacaoI.addAluno(joao)

#Matriculas e cancelamentos
print(programacaoI)
print(historia)
print(engenhariaSoft)
```

Observações: entregue seus arquivos no Google Classroom (entregue os 3 arquivos separados)