

**Data de Entrega:** 03/11/2021 (23:59)

**Grupos:** máximo de 6 alunos

---

### **DESCRIÇÃO DO MINIMUNDO:**

As escolas de magia formam vários bruxos em um determinado ano. Uma escola contém um nome e é formada por várias casas.

As casas também possuem um nome e um animal como símbolo. Em cada casa existe um diretor e um único aluno que é monitor.

Os alunos bruxos são identificados pelo seu nome e data de nascimento. Os alunos também são classificados em três tipos, de acordo com sua descendência, podendo ser *puro-sangue*, *trouxa* ou *mestiço*.

Um aluno bruxo ao entrar na escola é selecionado para morar em uma casa pelo chapéu seletor.

Na escola existe vários professores-bruxos que são identificados pelo seu nome e data de nascimento. Os professores podem ministrar várias disciplinas em um determinado ano.

Cada disciplina possui um nome e uma ementa. E os alunos podem frequentar várias disciplinas.

Durante as aulas os alunos recebem pontos dos professores, que podem ser positivos, quando realizam um *trunfo*, ou negativos, quando realizam um *mau-feito*.

Nas escolas de magia, o esporte preferido dos alunos é o *Quadribol*. Uma casa disputa torneios de quadribol contra outra casa, sendo que nessa disputa existe uma quantidade de pontos para cada casa. Vence o torneio a casa que tiver a maior quantidade de pontos.

A partir do cenário descrito e do diagrama de classes apresentado abaixo, implemente em Python as classes solicitadas, aplicando os conceitos de programação orientada a objetos.

---

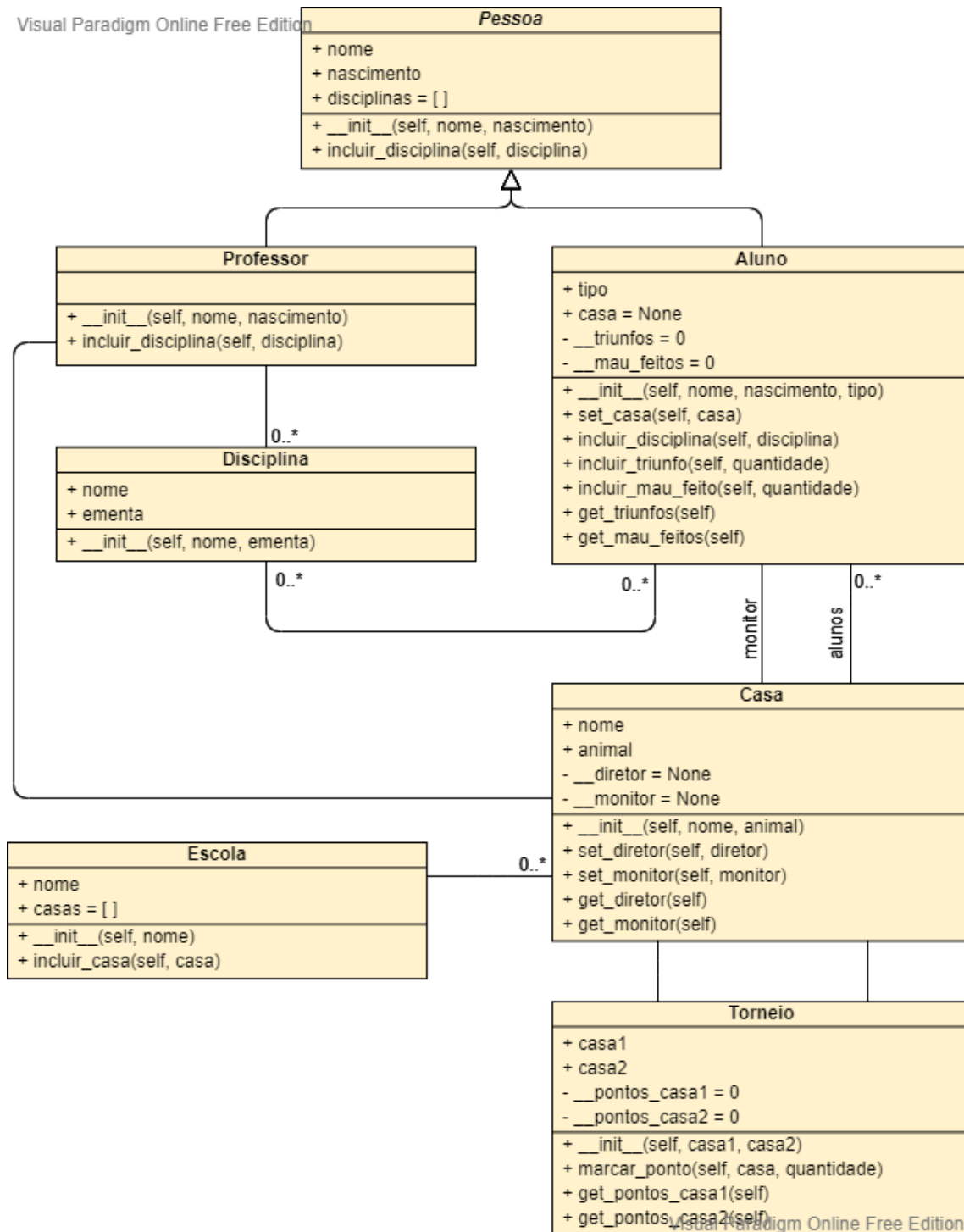
### **ATENÇÃO:**

- Não serão aceitos trabalhos entregues em atraso.
- Não serão aceitos trabalhos com mais de 6 alunos no grupo.
- Não deve ser utilizado input dentro das classes.
- A entregue pode ser feita por apenas um aluno do grupo, mas não esqueçam de colocar o nome de todos os integrantes no código do programa.
- Arquivos em outros formatos que não sejam .py, não serão aceitos (doc, pdf, txt, link para git, zip, etc)
- O arquivo `ac04_teste.py` contém um subconjunto de testes que serão usados na correção da atividade, para que vocês possam testar o funcionamento das classes implementadas e ter uma avaliação prévia do que estão fazendo antes de submeterem a atividade.
  - Outros testes poderão ser realizados na correção da atividade, para verificar se a implementação está de acordo com o solicitado.
- Se for identificada cópia de trabalhos entre os grupos, a nota será zerada.

## CRITÉRIOS DE AVALIAÇÃO:

- 8.0 pontos: Implementação e execução correta do programa conforme as especificações acima e conforme ilustrado no diagrama de classes.
- 2.0 pontos: Formatação e organização do código de acordo com o padrão PEP8.

## DIAGRAMA DE CLASSES:



### Classe Escola:

#### Atributos

- **nome:** nome da escola
- **casas:** lista de objetos da classe Casa. Definido no construtor como uma lista vazia.

#### Métodos

- **incluir\_casa:** recebe como parâmetro um objeto Casa e inclui na lista de casas

### Classe Casa:

#### Atributos

- **nome:** nome da casa
- **animal:** animal símbolo
- **\_\_diretor (privado):** objeto da classe Professor. Definido no construtor como None.
- **\_\_monitor (privado):** objeto da classe Aluno. Definido no construtor como None.

#### Métodos

- **set\_diretor:** recebe como parâmetro um objeto Professor e o define como diretor da casa.
- **set\_monitor:** recebe como parâmetro um objeto Aluno e o define como monitor de casa
- **get\_diretor:** retorna o objeto Professor que é diretor da casa.
- **get\_monitor:** retorna o objeto Aluno que é monitor da casa.

### Classe Disciplina:

#### Atributos

- **nome:** nome da disciplina
- **ementa:** ementa da disciplina

#### Métodos:

- Não possui.

### Classe Pessoa (classe abstrata):

#### Atributos

- **nome:** nome do professor
- **nascimento:** data de nascimento no formato "AAAAMMDD" (não precisa validar).
- **disciplinas:** lista de objetos da classe Disciplina. Definido no construtor como uma lista vazia.

#### Métodos:

- **incluir\_disciplina** (método abstrato).

### Classe Professor (herda da classe abstrata Pessoa):

#### Atributos

- Não possui.

#### Métodos:

- **incluir\_disciplina:** sobrescreve o método abstrato da classe Pessoa. Recebe como parâmetro um objeto Disciplina e inclui na lista de disciplinas.

### Classe Aluno (herda da classe abstrata Pessoa):

#### Atributos

- **tipo:** classificação do aluno em puro-sangue, trouxa ou mestiço.
- **casa:** objeto da classe Casa. Definido no construtor como None.
- **\_\_triunfos:** (*privado*) quantidade de triunfos. Definido no construtor como 0 (zero).
- **\_\_mau\_feitos:** (*privado*) quantidade de mau-feitos. Definido no construtor como 0 (zero).

#### Métodos:

- **incluir\_disciplina:** sobrescreve o método abstrato da classe Pessoa. Recebe como parâmetro um objeto Disciplina e inclui na lista de disciplinas.
- **set\_casa:** recebe como parâmetro um objeto Casa e o define como casa do aluno.
- **incluir\_triunfo:** recebe a quantidade de pontos e incrementa os triunfos do aluno.
- **incluir\_maufeito:** recebe a quantidade de pontos e incrementa os mau-feitos do aluno.
- **get\_triunfos:** retorna a quantidade de triunfos do aluno.
- **get\_mau\_feitos:** retorna a quantidade de mau-feitos do aluno.

### Classe Torneio:

#### Atributos:

- **casa1:** objeto da classe Casa
- **casa2:** objeto da classe Casa
- **\_\_pontos\_casa1:** (*privado*) quantidade de pontos da casa1. Definido no construtor como 0 (zero)
- **\_\_pontos\_casa2:** (*privado*) quantidade de pontos da casa2. Definido no construtor como 0 (zero)

#### Métodos:

- **marcar\_ponto:** recebe como parâmetro um objeto Casa e a quantidade de pontos e incrementa a pontuação da casa correspondente.
- **get\_pontos\_casa1:** retorna a quantidade de pontos da casa 1
- **get\_pontos\_casa2:** retorna a quantidade de pontos da casa 2