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 descendencia, 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.

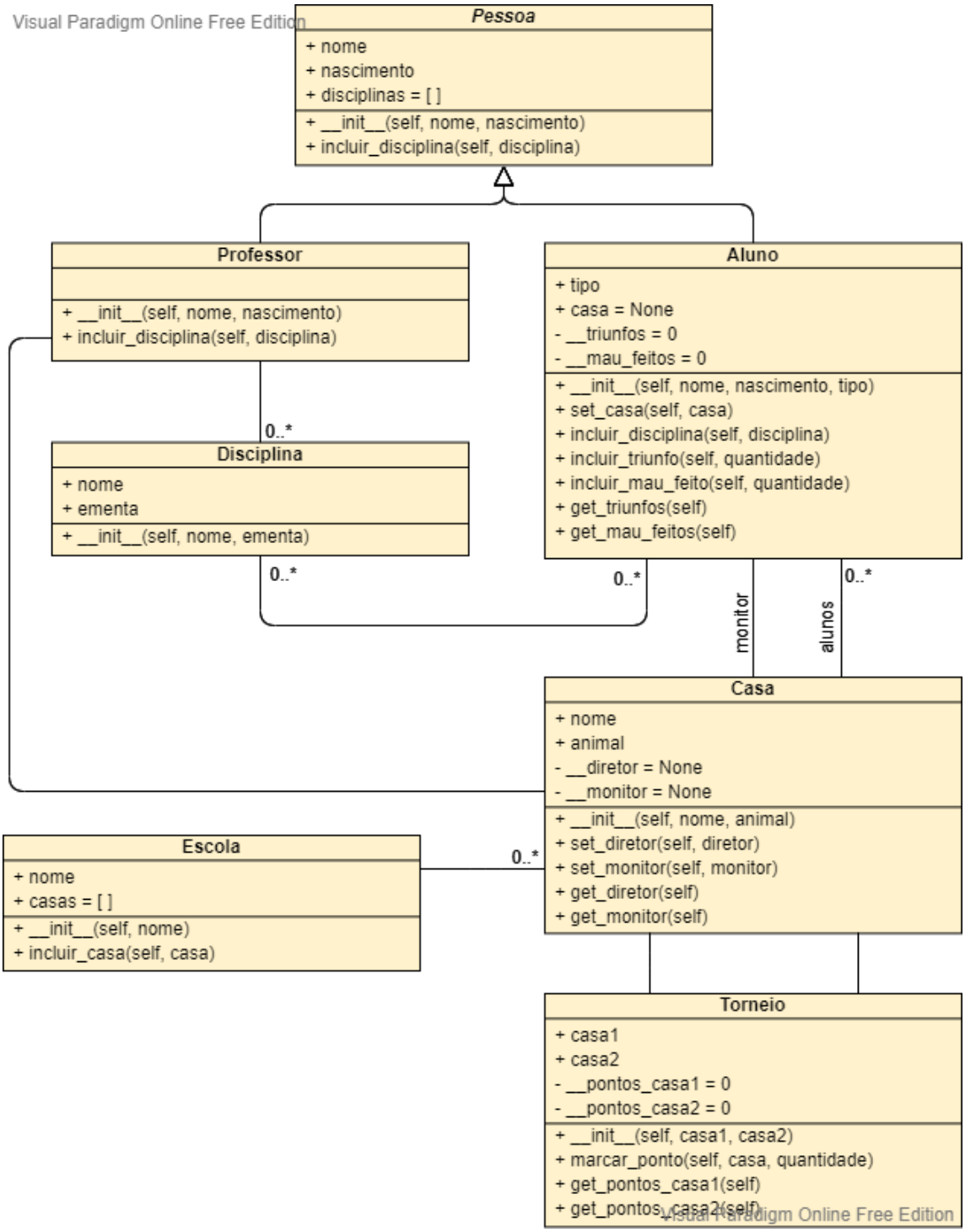
Cada disciplina possui um nome e uma ementa. E os alunos podem frequentar várias disciplinas.

Durante as aulas os alunos recebem pontos dos professores, que podem ser positivos, quando realizam um *triunfo*, 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.

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 contrutor como None.
- **__monitor (privado)**: objeto da classe Aluno. Definido no contrutor como None.

Métodos

- **set_diretor**: recebe como parâmetro um objeto Profesor 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 da pessoa
- **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 desciplinas.

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 desciplinas.
- **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_mau_feito**: 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 contrutor 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