Faculdade IMPACTA TECNOLOGIA

Linguagem de Programação 2

Classes em Python

Faculdade IMPACTA TECNOLOGIA

Objetivos

- Explicar o conceito fundamental de orientação a objetos
- Explicar o conceito de classes
- Entender como a criação de classes pode prover estruturação para programas complexos
- Definir e utilizar classes em Python
- Utilização de métodos especiais
- Escrever programas envolvendo definição de classes

Introdução - Classificação

Livros

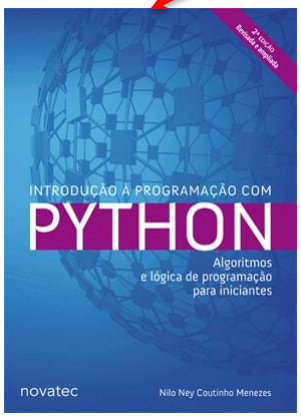
TI

Administração

Engenharia de Software

Desenvolvimento

Redes

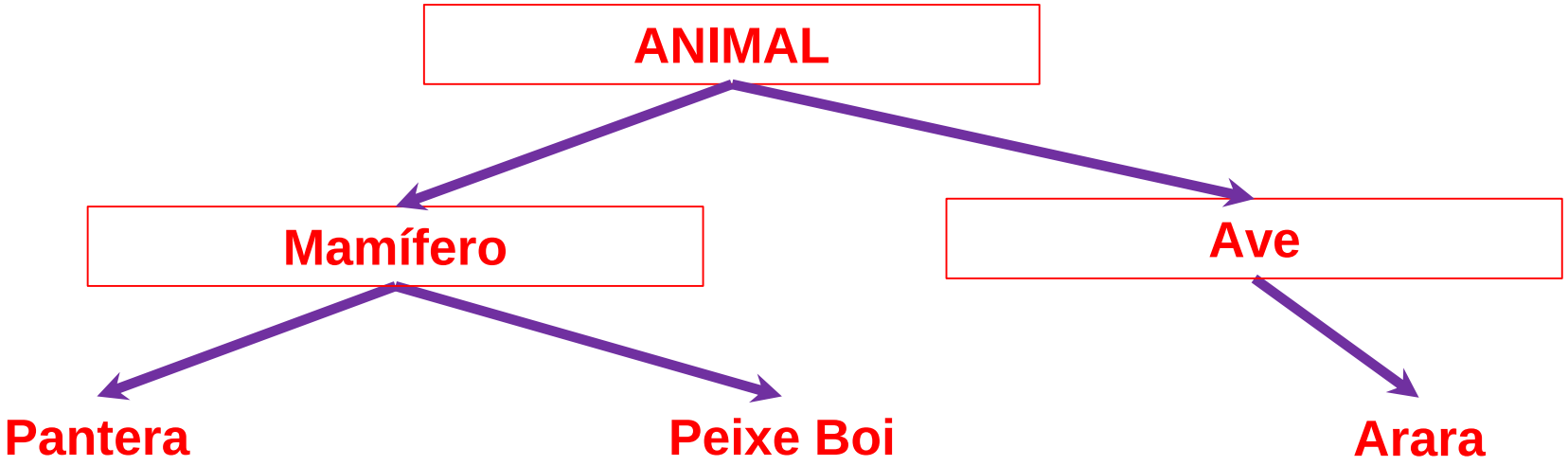


Introdução

- No mundo é comum trabalharmos com classificação
  - Agrupar coisas em categorias que tem algo em comum
- Na biblioteca, por exemplo, podemos agrupar os livros de acordo com o tema
  - Desenvolvimento, engenharia de software, redes, sistemas operacionais...
- Também é comum dividir em subcategorias, e estas em subcategorias...
  - Dentro de desenvolvimento, podemos classificar por linguagens de programação: Python, Java, C#, php...

Introdução

- Outro exemplo: podemos agrupar os animais de acordo com a classe



Faculdade IMPACTA TECNOLOGIA

Introdução

- Nesta aula veremos como a organização do código em tipos e subtipos de classes é utilizado em **Programação Orientada a Objetos**
- Estudamos na aula passada que tudo em Python é um objeto
  - Ou seja, utilizamos objetos todo o tempo na programação em Python
  - Entretanto, isso não significa que estamos utilizando programação orientadas a objetos
- Precisamos entender o que é uma classe e um objeto

Classes e objetos

- Programação orientada a objetos (POO) facilita a escrita e manutenção dos programas
  - Utilizam classes e objetos

Classe

Objetos (Instâncias)

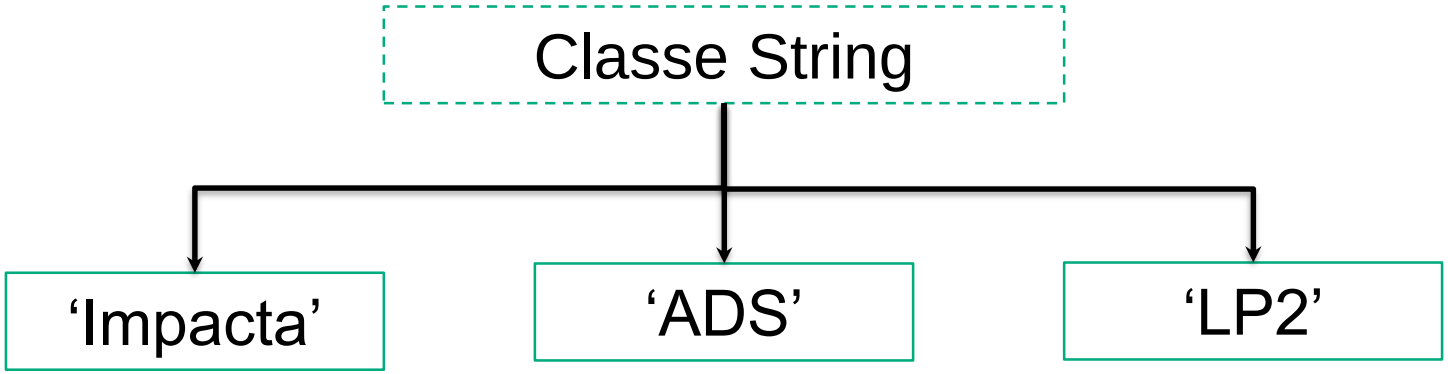
Faculdade IMPACTA TECNOLOGIA

Classes e objetos

- Uma **classe** é uma definição de um novo tipo de dado que não existe nos módulos do python
- Uma classe associa dados e operações em uma só estrutura, definindo um tipo de objeto
  - Como se fosse uma forma, para fazer quantos objetos forem necessários
  - Por exemplo, str (string), list (listas) e dict (dicionário) são classes do Python

Classes e objetos

- Por exemplo, as strings que você cria no seu programa são instâncias de objetos da classe String



  - String é a classe
  - 'Impacta', 'ADS', e 'LP2' são 3 objetos criados a partir da classe String

Classes e objetos

- Um **objeto** é como uma variável cujo tipo é uma **classe**
  - Em programação dizemos que um objeto é uma **instância** de uma classe
- A classe String foi definida com uma série de método
  - lower, upper, format...
- Cada objeto da classe String que for criado, terá estes métodos e poderá ser utilizado
  - O método é aplicado sobre o texto que está no objeto criado

Classes e objetos

- Exemplo

```
s1 = 'Impacta' # objeto String com o texto Impacta
s2 = 'ADS' # objeto String com o texto ADS

print(s1.lower()) # método lower de String
                  #aplicado em s1 (Impacta)

print(s2.lower()) # método lower de String
                  #aplicado em s1 (ADS)
```

  - Mesmo método, aplicado em textos diferentes

Classes e objetos

- Portanto, uma classe define um conjunto de atributos e métodos que podem ser utilizados pelos objetos da classe
  - Não é necessário implementar o método lower toda vez que for utilizar um texto.
  - O método já está definido na Classe e provê o comportamento para qualquer objeto desta classe
- Lembre-se uma classe é como se fosse uma forma

Faculdade IMPACTA TECNOLOGIA

Objetos e o mundo real

- Um **objeto** em Python pode ser pensado como uma **representação** de um **objeto do mundo real**
  - Faculdade, Pessoa, Aluno, Veículo, Carro, Roda...
- Essa representação é limitada pela quantidade de detalhes que representamos
  - **Abstração**
  - Nome e RA do aluno
  - Quantidade de salas da faculdade
  - Quantidade de Rodas de um Carro

Faculdade IMPACTA TECNOLOGIA

Objetos e o mundo real

- Vamos pensar em uma televisão. Quais as características de uma televisão?

Objetos e o mundo real

- Vamos pensar em uma televisão. Quais as características de uma televisão?
  - Cor
  - Marca
  - Modelo
  - Tamanho
  - ...

Faculdade IMPACTA TECNOLOGIA

Objetos e o mundo real

- Também podemos pensar no que podemos fazer com uma televisão:

Faculdade IMPACTA TECNOLOGIA

Objetos e o mundo real

- Também podemos pensar no que podemos fazer com uma televisão:
  - Ligar
  - Desligar
  - Mudar de canal
  - Mudar o volume
  - Sintonizar
  - ...

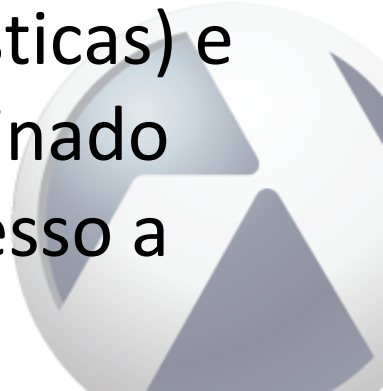
Faculdade IMPACTA TECNOLOGIA

Encapsulamento

- Encapsulamento é o meio agrupar variáveis de instâncias (atributos/características) e métodos (comportamento) em um determinado tipo
  - Uma **Classe**
- Encapsulamento é exatamente o que fizemos quando pensamos na TV
  - Colocamos suas características e comportamento em um “pacote”
  - Este pacote é a Classe Televisão

Encapsulamento

- No encapsulamento também podemos restringir o acesso a determinadas características, conhecido como características privadas
- Em uma classe Televisão, não podemos deixar que ninguém altere a marca ou o modelo
  - Poderia causar uma inconsistência
- Portanto, encapsulamento é o meio agrupar variáveis de instâncias (atributos/características) e métodos (comportamento) em um determinado tipo, e também um meio de restringir o acesso a determinados membros de uma classe



Classes em Pyhton – Mão na massa

- Podemos representar a Televisão, com suas **características** e **comportamento** em Python criando uma classe, da seguinte forma:

```
class Televisao:
    def __init__(self):
        self.ligada = False
        self.canal = 2
```

- Coloque este código em um arquivo chamado televisoes.py

Faculdade IMPACTA TECNOLOGIA

Instrução class

- A instrução **`class`** é utilizada para indicar a declaração de uma nova classe
- Após isso, coloca-se o nome da classe (por exemplo, Televisao) e finaliza com : para iniciar o bloco da classe
- Como qualquer bloco de execução em python, este deve ser **identado**
- Ao declarar uma classe estamos criando um novo tipo de dados
- O novo tipo define seus métodos e atributos (comportamento e características)

Faculdade IMPACTA TECNOLOGIA

Método __init__

- Classes em python tem um método especial chamado **`__init__`** (2 sublinhados no começo e no fim)
  - Métodos são funções associadas a uma classe
- **`__init__`** é chamado de **construtor**, uma vez que será chamado sempre que uma instância (objeto) da classe for criado
  - O construtor que inicia o novo objeto com os valores padrões
- **`__init__`** recebe um parâmetro chamado **self**
  - Uma representação do próprio objeto Televisao

Faculdade IMPACTA TECNOLOGIA

Atributos (self)

- Dentro do construtor os atributos do objeto são iniciados
- Ao utilizar **`self.ligada`**, por exemplo, estamos referenciando o atributo ligada do próprio objeto (**self**)
  - Portanto, **`self.ligada`** é um **atributo** do objeto
- Sempre que quisermos referenciar um atributo do objetos, utilizamos com o self
  - Se não colocar o self, será apenas uma variável local

```
self.ligada = False # atributo do objeto
ligada = True # variável local
```

Faculdade IMPACTA TECNOLOGIA

Atributos (self)

- Agora vamos instanciar (criar) um objeto de Televisao
- Crie um novo arquivo chamado main.py
- Faça a importação da classe e instancie um objeto da seguinte forma:

```
from televisoes import Televisao

tv = Televisao()
```

- tv é um objeto da classe Televisao
- tv é uma **instância** de Televisao

Faculdade IMPACTA TECNOLOGIA

Atributos (self)

- Com a instância criada, podemos acessar os atributos da classe da seguinte forma:

```
print(tv.ligada)

print(tv.canal)
```

Atributos (self)

- Podemos criar quantas instância quisermos da classe Televisao
  - Cada instância será independente, ou seja, terão seus próprios atributos e métodos
- Por exemplo, vamos criar uma nova instância de Televisao e armazenar na variável tv_sala

```
tv_sala = Televisao()
```

- Agora altere os atributos ligada e canal dessa nova instância da seguinte forma

```
tv_sala.ligada = True
tv_sala.canal = 5
```

Atributos (self)

- Mostre os valores ligado e canal das duas instâncias para ver as diferenças

```
print(tv_sala.ligada)
print(tv_sala.canal)
print(tv.ligada)
print(tv.canal)
```

- Estes valores, bem como seus objetos, são independentes
  - Cada um tem seus próprios valores, assim como o mundo real

Classes

- Ao criar um objeto de uma classe, ele tem todos os atributos e métodos da declaração da classe, e iniciados com o construtor
- Essa característica simplifica o desenvolvimento dos programas, uma vez que pode-se definir o **comportamento** de todos os objetos de uma classe (**métodos**), preservando os **características** individuais (**atributos**)

métodos == comportamento
atributos == características

Métodos

- Vamos adicionar alguns comportamentos à classe Televisao , adicionando 2 métodos:
  - aumenta_canal
  - diminui_canal
- A definição de um método é feita da mesma forma que define-se uma função
  - Instrução def
  - Nome do método
  - Dois pontos (:)
  - Bloco identado

Métodos

```
class Televisao:
    def __init__(self):
        self.ligada = False
        self.canal = 2

    def aumenta_canal(self):
        print('aumentar canal')

    def dimunui_canal(self):
        print(diminuir canal)
```

- Todo método da classe deve receber como primeiro parâmetro o self

Faculdade IMPACTA TECNOLOGIA

Métodos - exercício

- Complete os blocos dos dois métodos para aumentar e diminuir os canais

Métodos - chamada

- Para ser executado, o método deve ser chamado
  - Assim como uma função
- A chamada é feita utilizando a variável que armazena a instância e o nome do método, separados por pontos
- No módulo main.py, adicione os seguintes códigos para chamar os métodos de Televisao

```
tv.aumenta_canal()
tv.diminui_canal()
```

Métodos - chamada

- Repare que na chamada não há a passagem de nenhum parâmetro
  - A instância de tv (self) é colocada automaticamente pelo interpretador
- Também não é preciso enviar o número do canal como parâmetro
  - O canal utilizado está na própria classe
  - Características importantes ficam armazenadas nos atributos da classe, evitando passá-los na chamada

Construtor com parâmetros

- Na classe Televisao não existe controle de quantos canais podem existir
- O menor e o maior canal são características de uma televisão configuradas de fábrica
  - Ou seja quando ela é construída
- Um método construtor pode receber parâmetros como qualquer outro método.
- Os parâmetros do construtor são declarado logo após o self

Construtor com parâmetros

```
class Televisao:
    def __init__(self, min, max):
        self.ligada = False
        self.canal = 2
        self.cmin = min
        self.cmax = max
```

- min e max são parâmetros do construtor
- Dessa forma, eles devem ser enviados na chamada

```
tv_sala = Televisao(2,50)
```

Faculdade IMPACTA TECNOLOGIA

Construtor – parâmetros opcionais

- Existem situações que queremos instanciar 2 objetos da mesma classe de formas diferentes
- Por exemplo, a classe pode estar preparada para instanciar objetos cuja a quantidade de canais é fixa ou que não tenha limite de canais
- Portanto, seria necessário 2 formas de construir um objeto da classe:
  - Um que não recebe nenhum parâmetro além do self
  - Outro que recebe o canal mínimo e o canal máximo, além do self

Construtor – parâmetros opcionais

- Em python não é possível definir 2 métodos com o mesmo nome
- Para que a situação anterior seja possível, devemos colocar parâmetros opcionais no construtor, da seguinte forma:

Construtor com parâmetros

```
class Televisao:
    def __init__(self, min=None, max=None):
        self.ligada = False
        self.canal = 2
        if min != None:
            self.cmin = min
        if max != None:
            self.cmax = max
```

```
tv = Televisao()  # objeto criado com construtor padrao
tv_sala = Televisao(2,50) # objeto criado com o\
                          construtor com parâmetros
```

Faculdade IMPACTA TECNOLOGIA

Atributos privados

- O menor e o maior canal são características de uma televisão que não podem ser alteradas inadvertidamente
  - São características configuradas de fábrica
  - Devemos proteger estes atributos/características para que eles não sejam modificados por alguma operação externa à classe
- A proteção é feita utilizando atributos privados

Faculdade IMPACTA TECNOLOGIA

Atributos privados

- Um atributo privado de uma classe em Python é nomeado com dois sublinhas (__) no começo do nome do atributo
  - Canal máximo: `__cmax`
  - Canal mínimo: `__cmin`
- Vamos alterar o construtor com parâmetros para utilizar atributos privados

Construtor com parâmetros

```
class Televisao:
    def __init__(self, min=None, max=None):
        self.ligada = False
        self.canal = 2
        if min != None:
            self.__cmin = min
        if max != None:
            self.__cmax = max
```

Atributos privados

- Como sei que vai funcionar? Vamos fazer os seguintes testes:
  - Crie uma nova instância de TV utilizando o construtor com parâmetros

```
tv_sala = Televisao(2,50)
```

  - Acesse a variável de classe canal e mostre seu valor
    - Funciona

```
print(tv_sala.canal)
```

  - Agora tente acessar a variável de classe privada __cmin

```
print(tv_sala.__cmin)
```

    - Erro!

Atributos privados

- Como sei que vai funcionar? Vamos fazer os seguintes testes:
  - Agora tente alterar o valor de __cmin, e depois mostre o valor

```
tv_sala.__cmin = 20
print(tv_sala.__cmin)
```

  - Funcionou? Sim! Mas o que foi alterado não foi o atributo privado
    - Python cria um outro __cmin para poder alterar
  - Portanto, cuidado ao utilizar e acessar atributos privados!

Faculdade IMPACTA TECNOLOGIA

Atributos privados

- Usualmente, os atributos privados tem associados dois métodos: um para alterar (set) e um para retornar (set)
- Para terminar o exemplo anterior, crie um método na classe Televisao chamado get_cmin, que retorna o canal mínimo da Televisão e veja que o valor é diferente do que foi atribuído no comando **`tv_sala.__cmin = 20`**

Faculdade IMPACTA TECNOLOGIA

Atributos privados

```
class Televisao:
  def __init__(self, min=None, max=None):
    self.ligada = False
    self.canal = 2
    if min != None:
        self.cmin = min
    if max != None:
        self.cmax = max

  def get_cmin(self):
    return self.__cmin

tv = Televisao(2,50)
print(tv.canal)
tv.__cmin = 20
print("__cmin alterado fora do objeto: ", tv.__cmin)
print("__cmin do objeto: ", tv.get_cmin())
```

Faculdade IMPACTA TECNOLOGIA

Métodos - Parâmetros

- Os outros métodos das classes também podem receber parâmetros
- Altere os métodos aumenta_canal() e diminui_canal() para comportar as seguintes regras:
  - Quando o canal chegar ao canal máximo, deve voltar para o canal mínimo
  - Quando o canal chegar ao canal mínimo, deve voltar ao canal máximo

Faculdade IMPACTA TECNOLOGIA

Métodos - Parâmetros

```
class Televisao:
    def __init__(self, min=None, max=None):
        self.ligada = False
        self.canal = 2
        if min != None:
            self.cmin = min
        if max != None:
            self.cmax = max

    def aumenta_canal(self):
        if self.canal == self.__cmax:
            self.canal = self.__cmin
        else:
            self.canal += 1

    def dimunui_canal(self):
        if self.canal == self.__cmin:
            self.canal = self.__cmax
        else:
            self.canal -= 1
```

Faculdade IMPACTA TECNOLOGIA

Considerações e Conclusão

- Trabalhar com classes significa representar em Python uma abstração do problema
- Uma abstração reduz os detalhes do problema ao necessário para sua solução
- Fazendo isso estamos construindo um modelo, ou seja, modelando classes e objetos

Faculdade IMPACTA TECNOLOGIA

Considerações e Conclusão

- Cada pessoa modela de uma forma
- Uma das partes mais difíceis é decidir o quanto representar e onde limitar o modelo
  - No nosso modelo de TV não colocamos nada sobre tomada, volume, controle remoto, etc
- Dica: modele apenas as informações que precisa, acrescentando conforme a necessidade

Faculdade IMPACTA TECNOLOGIA

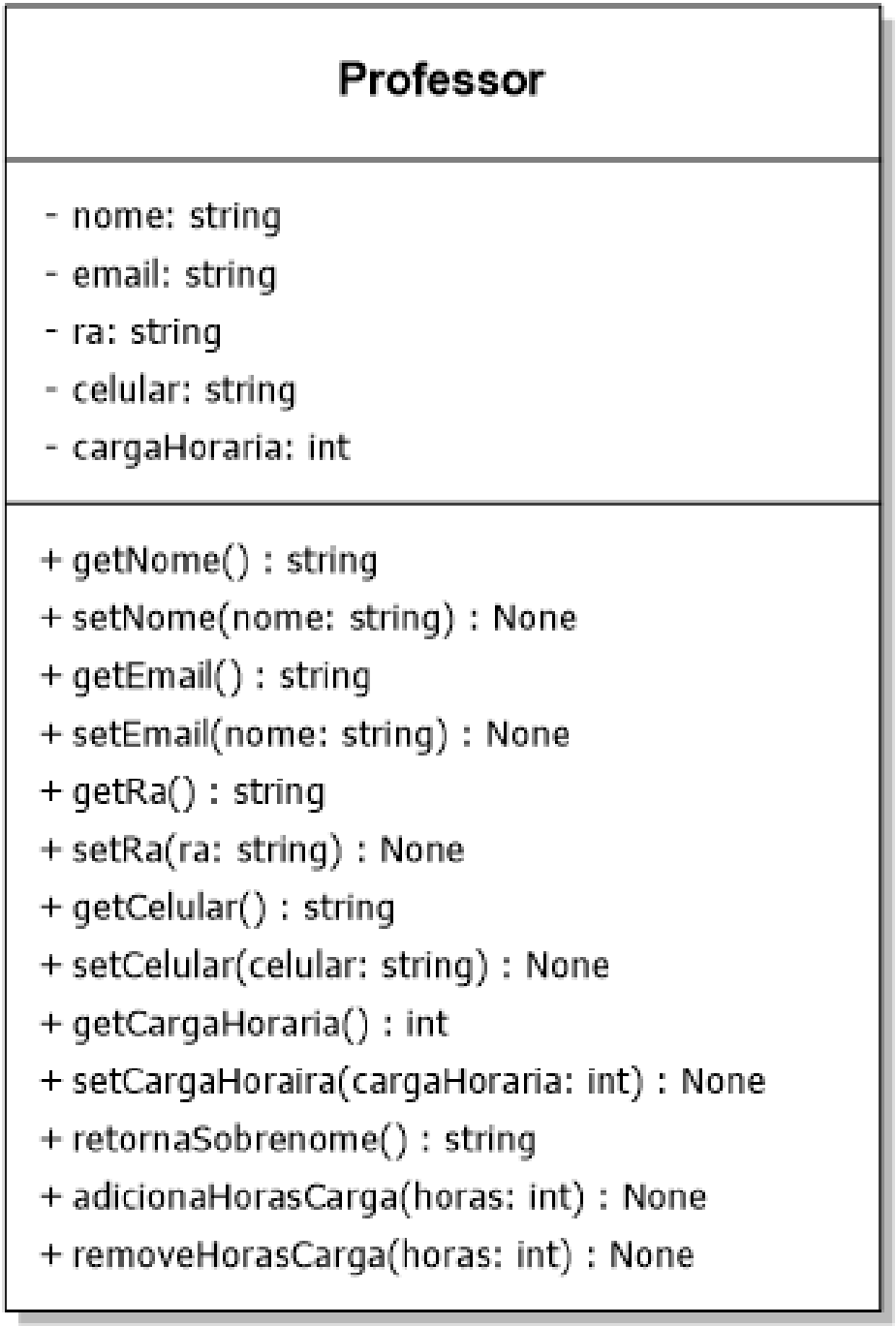
Considerações e Conclusão

- Métodos de classes são como funções
  - Pode aproveitar tudo que aprendeu
- Principais diferenças:
  - Um método é associado a uma classe e atua em um objeto
  - O primeiro parâmetro de um método é sempre self, e representa a instância sobre a qual o método atua
  - Na chamada não é preciso enviar self como parâmetro. Isso é feito automaticamente pelo interpretador

Modelando o LMS

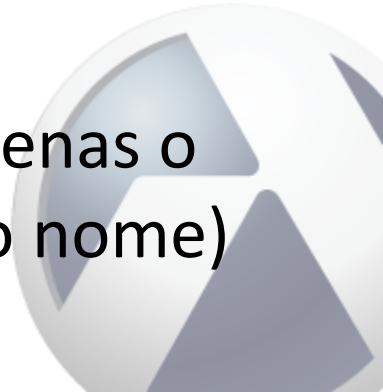
- Vamos modelar parte do trabalho da disciplina (LMS), para controlar Professores, Alunos e Disciplinas
- Começaremos modelando a classe Professor com o seguinte diagrama de classes:

Modelando o LMS



Modelando o LMS

- Todos os atributos da classe são privados
- A classe deve ter 2 construtores: 1 sem nenhum argumento e outro com argumentos para iniciar todos os atributos privados
- Os métodos get devem retornar os valores dos seus respectivos atributos privados
- Os métodos set devem alterar os valores dos respectivos atributos privados
- O método retornaSobrenome deve retorna apenas o sobrenome do professor (baseado no atributo nome)



Modelando o LMS

- O método adicionaHorasCarga(horas) deve adicionar a quantidade de horas do argumento horas no atributo carga_horaria. Caso a carga horária final passe de 40 horas, não deve adicionar e retornar uma mensagem de erro
- O método removeHorasCarga(horas) deve remover a quantidade de horas do argumento horas do atributo carga_horaria. Caso a carga horária final seja menor que 0, não deve diminuir e retornar uma mensagem de erro

Associação de classes

- Durante a modelagem de um sistema, é muito comum fazer a associação de classes.
- Ou seja, os atributos de uma classe referenciam instâncias de objetos de outras classes
- Por exemplo: um Professor leciona uma ou mais disciplinas.
  - Portanto, existe um atributo da classe Professor que será uma lista de elementos de outra classe: Disciplina

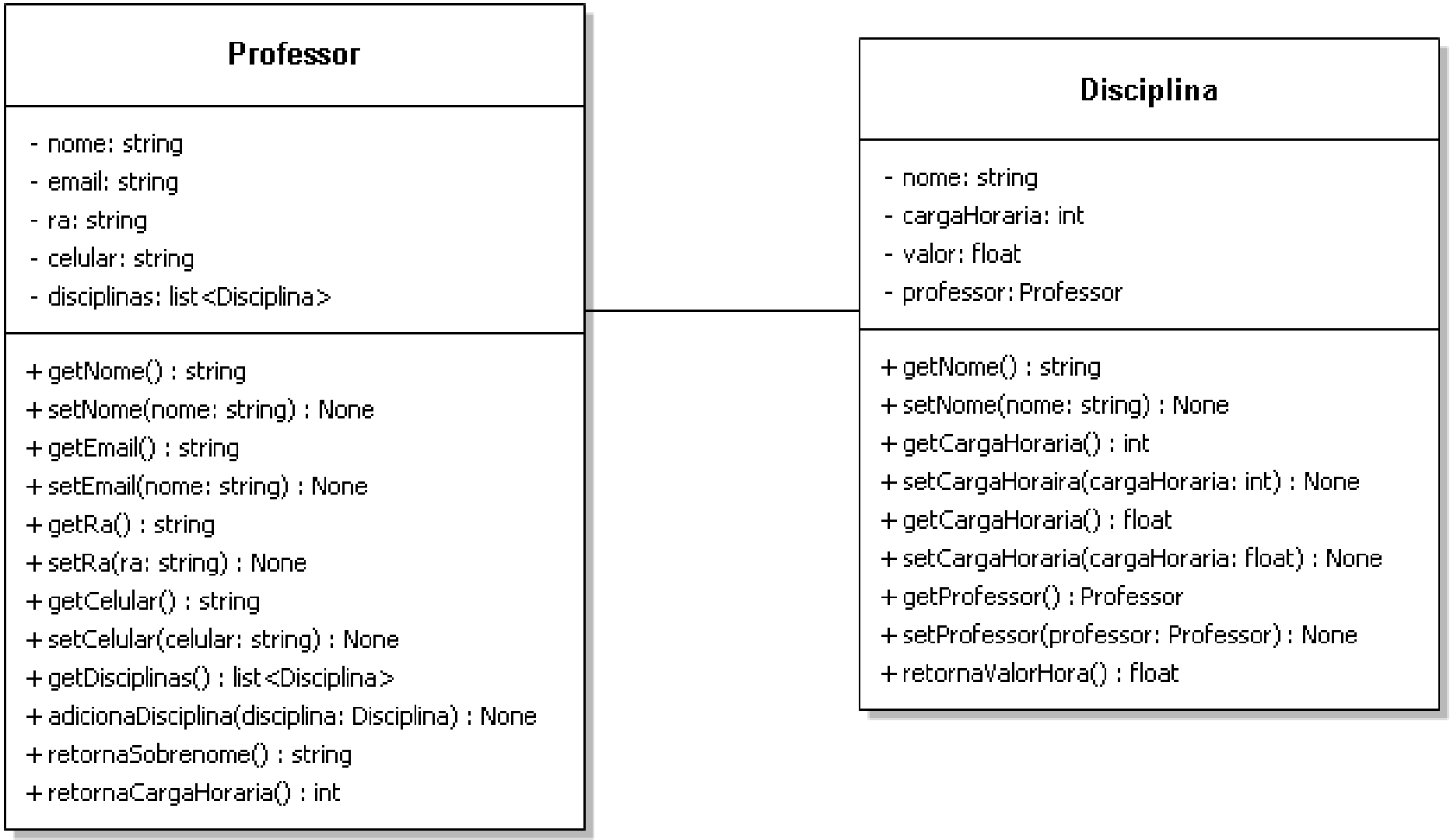
Faculdade IMPACTA TECNOLOGIA

Associação de classes

- Vamos alterar a modelagem para:
  - Incluir a classe Disciplina;
    - Um professor pode ministrar várias disciplinas, mas uma disciplina será ministrada por apenas um professor
  - Remover a carga horária do professor para colocar na disciplina
  - Calcular a carga horária de cada professor de acordo com o que está na disciplina

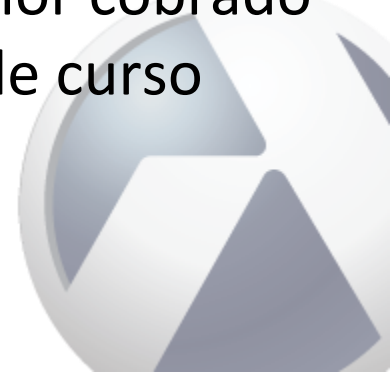
Faculdade IMPACTA TECNOLOGIA

Associação de classes



Associação de classes

- Na classe Disciplina:
  - Todos os atributos são privados
  - Deve ter 2 construtores: 1 sem nenhum argumento e outro com argumentos para iniciar todos os atributos privados
  - Os métodos get devem retornar os valores dos seus respectivos atributos privados
  - Os métodos set devem alterar os valores dos respectivos atributos privados
  - O método retornaValorHora deve retornar o valor cobrado pela hora da disciplina, considerando 6 meses de curso



Faculdade IMPACTA TECNOLOGIA

Associação de classes

- Na classe Professor
  - Não deve haver o campo cargaHoraria
  - O construtor não deve aceitar a disciplina
  - O atributo disciplina tem somente o método get
  - Utilize o método adiciona Disciplina para adicionar uma disciplina na lista associada ao professor. Só deve adicionar se o professor for o mesmo da instância (verificando pelo RA). Caso contrário, retorna uma mensagem de erro
  - O método retornaCargaHoraria deve retornar a carga horária semanal total do professor, baseado nas disciplinas que leciona e em 20 semanas de aulas

Avaliação Contínua

- AC 6

Faculdade IMPACTA TECNOLOGIA

Obrigado!

Prof. MSc. Fernando Sousa

professor.fsousa@gmail.com