Lógica de Programação

Aula 6

SENAC HUB ACADEMY

CENTRO DE INOVAÇÃO EM EDUCAÇÃO PROFISSIONAL

Exercícios funções

1 - Crie uma função que necessite de três argumentos, e que imprima o produto desses três argumentos.

Entendendo sobre Nomeclaturas

Fecomércio MS · Senac
Sistema Comércio

SENAC HUB ACADEMY
CENTRO DE INOVAÇÃO EM EDUCAÇÃO PROFISSIONAL

Quando falamos em programação, nem sempre as formas de digitarmos os códigos e os nomes são iguais às que já conhecemos, por isso é importante entendermos como isso funciona para termos mais facilidade na hora de programar.

Entendendo sobre Nomeclaturas

Fecomércio MS · Senac
Sistema Comércio

SENAC HUB ACADEMY
CENTRO DE INOVAÇÃO EM EDUCAÇÃO PROFISSIONAL

Primeiramente vamos pensar em palavras reservadas:

- Palavras reservadas são palavras que não podemos usar por nossa própria conta quando começamos a nomear variáveis dentro do nosso código
- Todas as linguagens possuem palavras reservadas
- As palavras reservadas normalmente são utilizadas para chamada de funções que já existem no interno da linguagem, como **print, def, int** etc...

Entendendo sobre Nomeclaturas

Agora vamos pensar em questões de declarações:

Quando declaramos variáveis e funções nós precisamos nomear isso para que passe a existir

nome=5
def funcao():

A partir do momento que criamos e nomeamos uma variável ela passa a existir.

Entendendo sobre Nomeclaturas

Fecomércio MS · Senac
Sistema Comércio

SENAC HUB ACADEMY
CENTRO DE INOVAÇÃO EM EDUCAÇÃO PROFISSIONAL

Mas para criarmos os nomes existem algumas regras que precisamos entender além da ideia das palavras reservadas:

Quando nomeamos uma variavel ou uma função, nunca usaremos uma letra maiúscula como nomes de pessoas:

def funcao(): **def Funcao():**

Além disso nós também não "espaçamos" as palavras dentro do nosso código:

def nome_da_funcao(): **def nome da funcao():**

E por fim, não utilizamos ç nem acentos quando criamos nomes:

def funcao(): **def funcão():**
def funçao():

Entendendo sobre Nomeclaturas

Fecomércio MS · Senac
Sistema Comércio

SENAC HUB ACADEMY
CENTRO DE INOVAÇÃO EM EDUCAÇÃO PROFISSIONAL

Lembrando que essas regras são válidas para nomeclatura de variáveis e de funções, quando começarmos a ver outros tipos de dados/estruturas veremos que cada um tem suas regras.
E também é importante pensar que essas regras são diferentes quando pensamos em Strings (trechos de textos demarcados por ' ' ou " ") dentro do espaço da declaração da String, somos livres para escrevermos da forma que estamos acostumados.

"Aqui é válido escrever assim porque é uma String"

def aqui_nao():

Entendendo sobre Nomeclaturas

Fecomércio MS · Senac
Sistema Comércio

SENAC HUB ACADEMY
CENTRO DE INOVAÇÃO EM EDUCAÇÃO PROFISSIONAL

Quando começamos a escrever nomes, também é sempre importante pensarmos em nomes com significados, porque caso façamos uma variavel "a" e outra pessoa necessitar desse código por qualquer motivo, para vias de manutenção esse código fica com uma legibilidade ruim.

```
a =3
a = input()
```

```
idade = 3
entrada_idade = input()
```

Vocês devem fazer um cálculo de soma utilizando 3 funções:
1 – Para pegar o primeiro valor
2 – Para o segundo valor
3 – Para realizar o cálculo e imprimir na tela o resultado

OBS: Quando pensamos em criar funções pensamos em formas de alocação de memória/código

Paradigma/Programação Orientada a Objetos

Como vimos em aulas anteriores a ideia da programação orientada a objetos é muito semelhante a ideia de objetos que temos no mundo real, o que torna esse modelo muito palpável.

Paradigma/Programação Orientada a Objetos

Fecomércio MS · Senac
Sistema Comércio

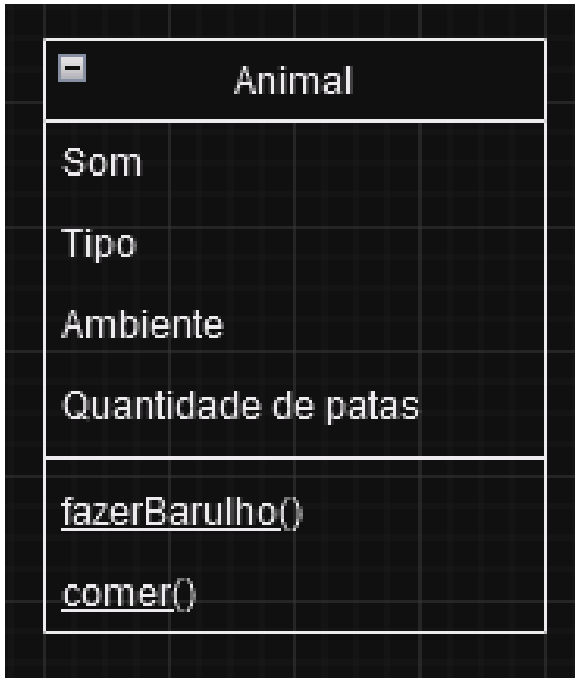
SENAC HUB ACADEMY
CENTRO DE INOVAÇÃO EM EDUCAÇÃO PROFISSIONAL

Dentro da orientação objeto veremos alguns novos conceitos:

Classe
Objeto
Herança
Encapsulamento

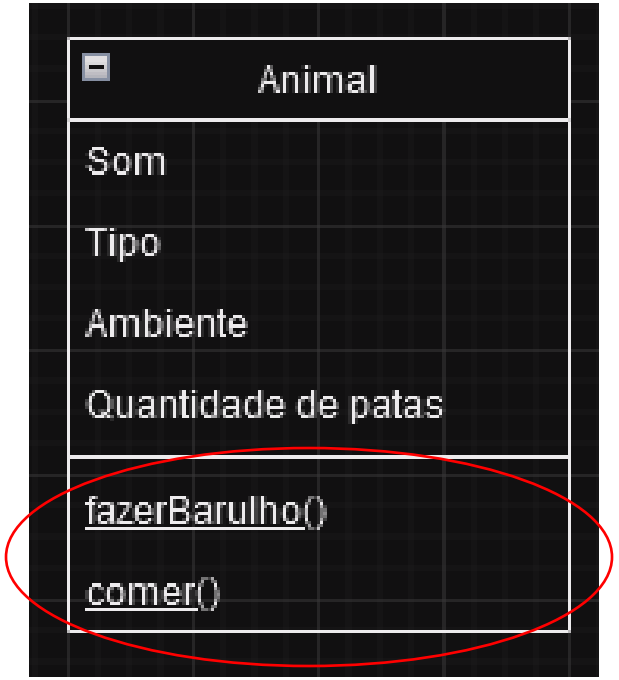
Classes:

Uma classe é uma forma de definir um tipo de dado em uma **linguagem orientada a objeto**. Ela é formada por dados e comportamentos.



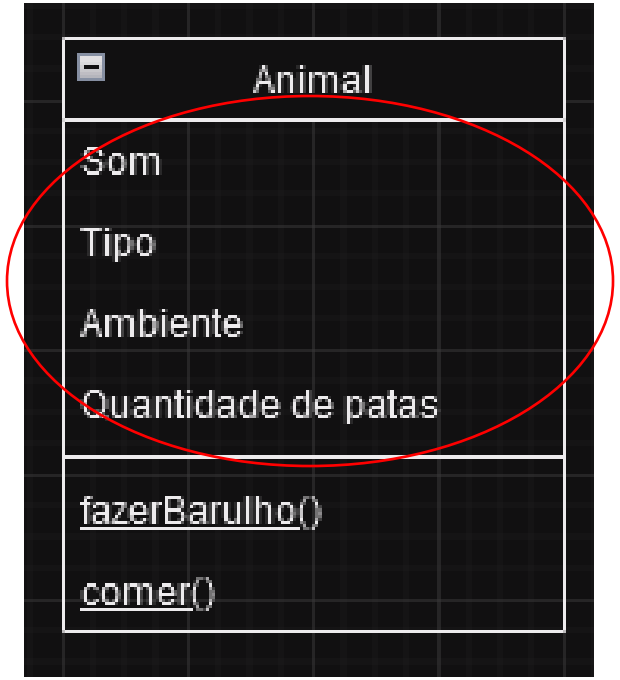
Classes:

Quando falamos dos comportamentos das classes eles podem se referir aos métodos da classe que equivalem as nossas funções vistas anteriormente.



Classes:

Quando pensamos em dados, podem se referir tanto às variáveis quanto a união com bancos de dados, as classes permitem inúmeras manipulações com os valores que são fornecidas a elas sempre voltadas a um objetivo específico.

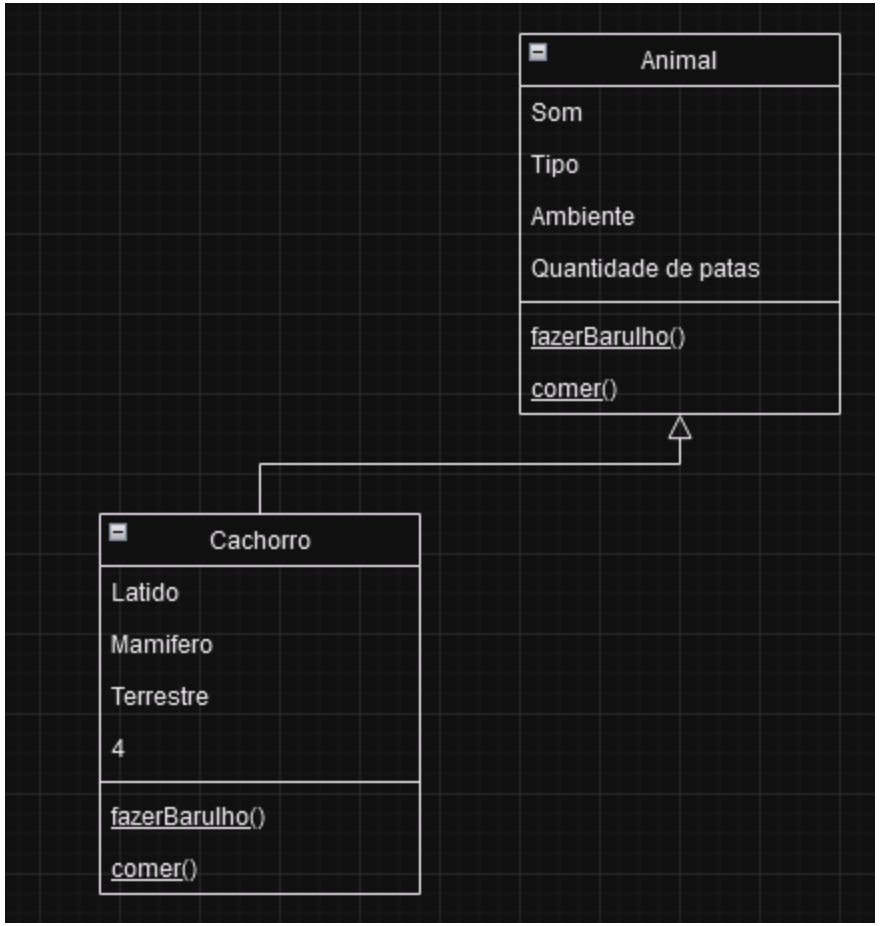


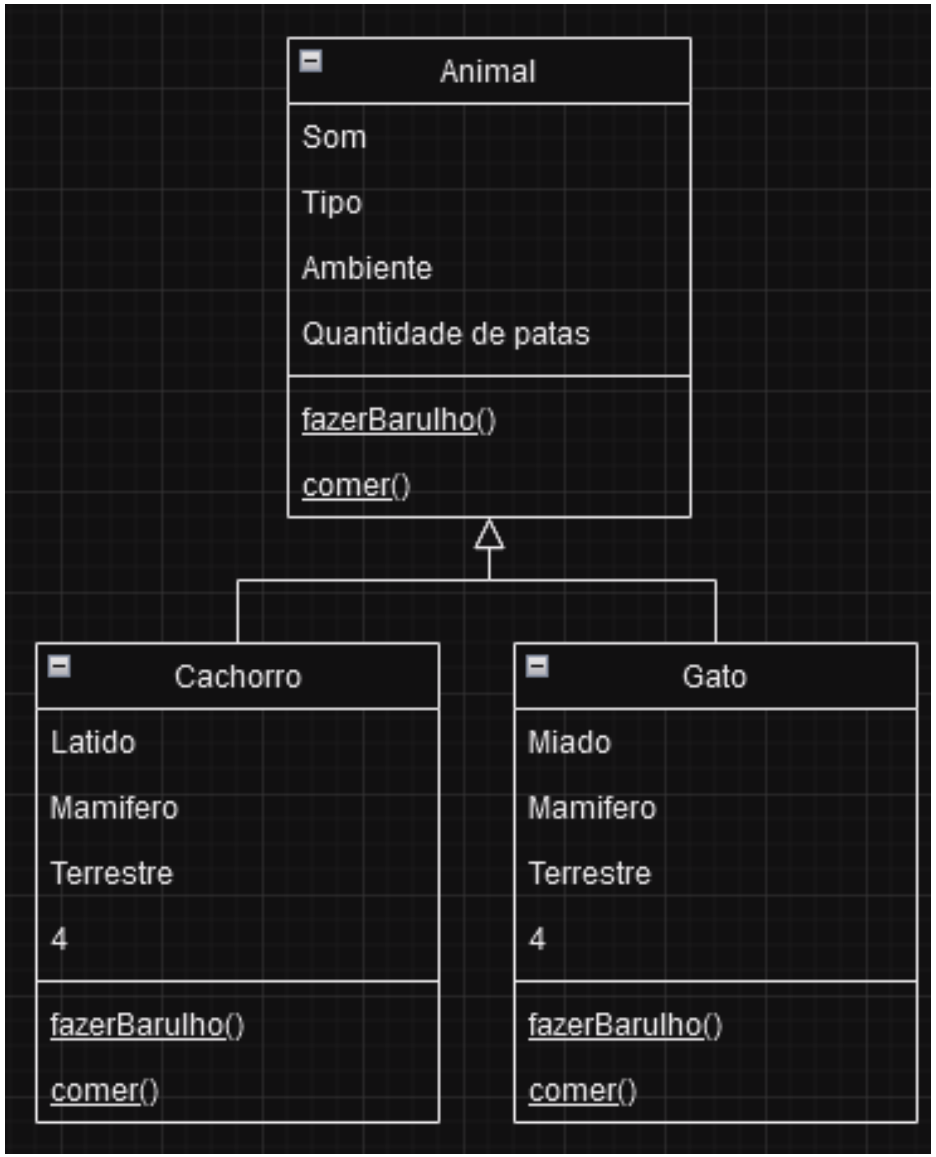
Crie uma Classe de Empresa com Atributos e Comportamentos.

Objetos:

Os objetos são modelos das classes que utilizamos para podermos criar relacionamentos

No exemplo ao lado Animal é um objeto de cachorro, porque cachorro o instância para poder manipular seus atributos



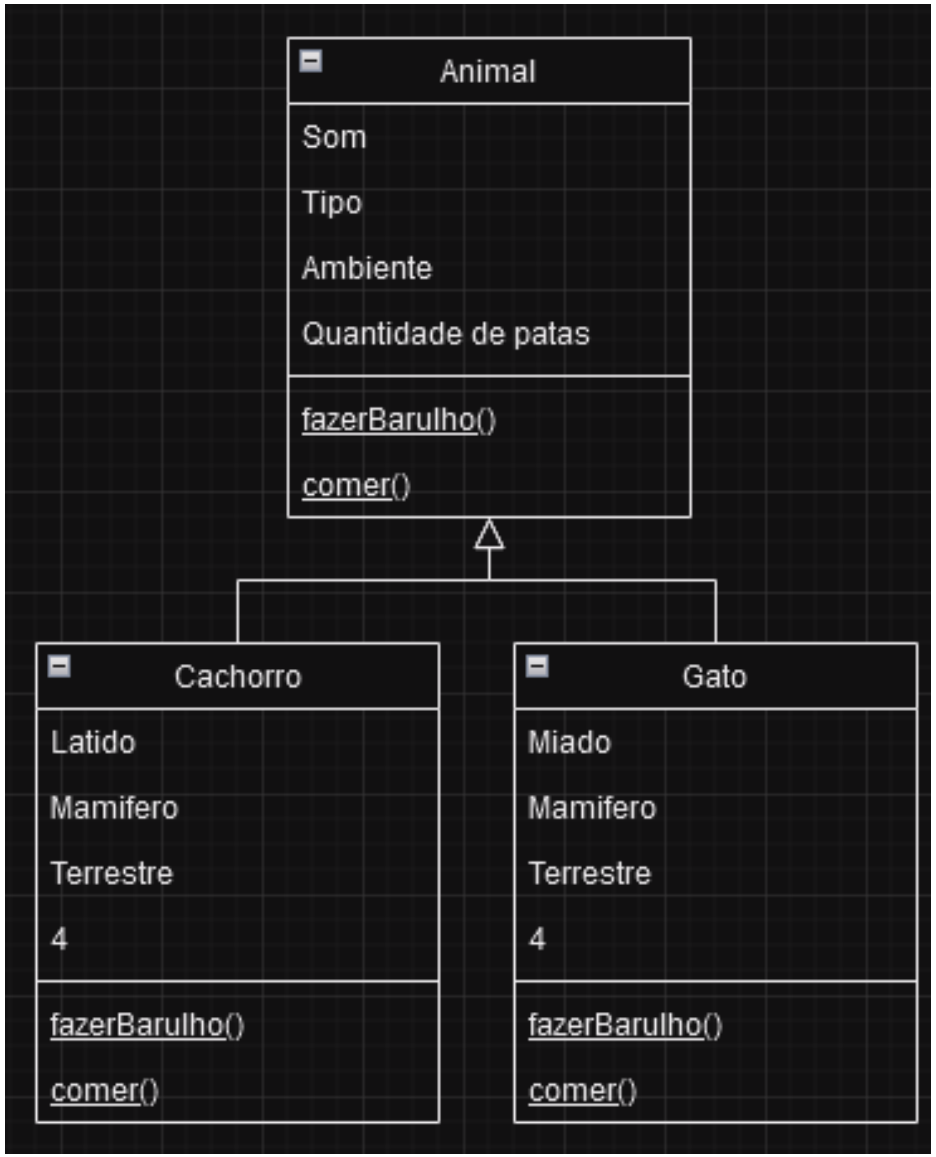

Animal
Som
Tipo
Ambiente
Quantidade de patas
fazerBarulho()
comer()
Cachorro
Latido
Mamifero
Terrestre
4
fazerBarulho()
comer()
Gato
Miado
Mamifero
Terrestre
4
fazerBarulho()
comer()

Herança:

Fecomércio MS · Senac
Sistema Comércio

SENAC HUB ACADEMY
CENTRO DE INOVAÇÃO EM EDUCAÇÃO PROFISSIONAL

Herança é o conceito que nos permite criar relacionamentos entre as classes, os objetos e tudo o que começamos a criar dentro dos códigos e programas que fazemos.
A Herança funciona de uma maneira diferente do que podemos imaginar na vida fora das máquinas, as heranças em código servem para economizarmos código e criarmos classes "pais" que permitem aos filhos implementarem valores que existem neles e assim tirar a necessidade de repetirmos modelos.


Animal
Som
Tipo
Ambiente
Quantidade de patas
fazerBarulho()
comer()
Cachorro
Latido
Mamifero
Terrestre
4
fazerBarulho()
comer()
Gato
Miado
Mamifero
Terrestre
4
fazerBarulho()
comer()

Criem uma Classe mãe com pelo menos duas Classes Filhas
É necessário que criem atributos e comportamentos para elas.

Encapsulamento:

O Encapsulamento é o que utilizamos quando queremos ou não que determinados atributos sejam acessados por outras classes ou outros métodos.

Public
Protected
Private

Public:

Public é o que permite que atributos e métodos sejam acessados por qualquer um, sendo assim não tem nenhuma restrição
Também o padrão quando começamos nosso código e não declaramos outro modificador.

Protected: (_)

Não são acessíveis a partir de fora da classe, mas podem ser utilizados pelas subclasses. O encapsulamento protegido é útil quando você deseja restringir o acesso direto a certos elementos da classe, mas ainda permitir que as subclasses os utilizem.

Private: (__)

Não são acessíveis a partir de fora da classe e nem pelas subclasses. O encapsulamento privado é o nível mais restritivo e é utilizado quando você deseja ocultar completamente os detalhes de implementação da classe.

Dúvidas?

SENAC HUB ACADEMY

CENTRO DE INOVAÇÃO EM EDUCAÇÃO PROFISSIONAL