

**INSTITUTO
FEDERAL**

Piauí

Campus
Pedro II



Programação orientada a objetos

Aula [02] : Abstração

Objetivos

Entender o que significa Abstração;

Contextualizar a Abstração no mundo O.O;

Exercitar teoricamente e na prática;

Conceito

Abstração é a técnica em que selecionamos as principais entidades (objetos) que vão compor determinado sistema, bem como as principais características e comportamentos a fim de implementar um sistema baseado em objetos.

- Seleção de itens relevantes;
- Ignorar itens não relevantes;

Abstração

Identificar **entidades** em potencial;

Técnicas como leitura de documentos, entrevistas, pesquisas, elucidação de requisitos, etc.

Selecionar **características** e **comportamentos** das entidades;

Traduzir para uma linguagem de programação orientada a objetos;

De onde abstrair ?

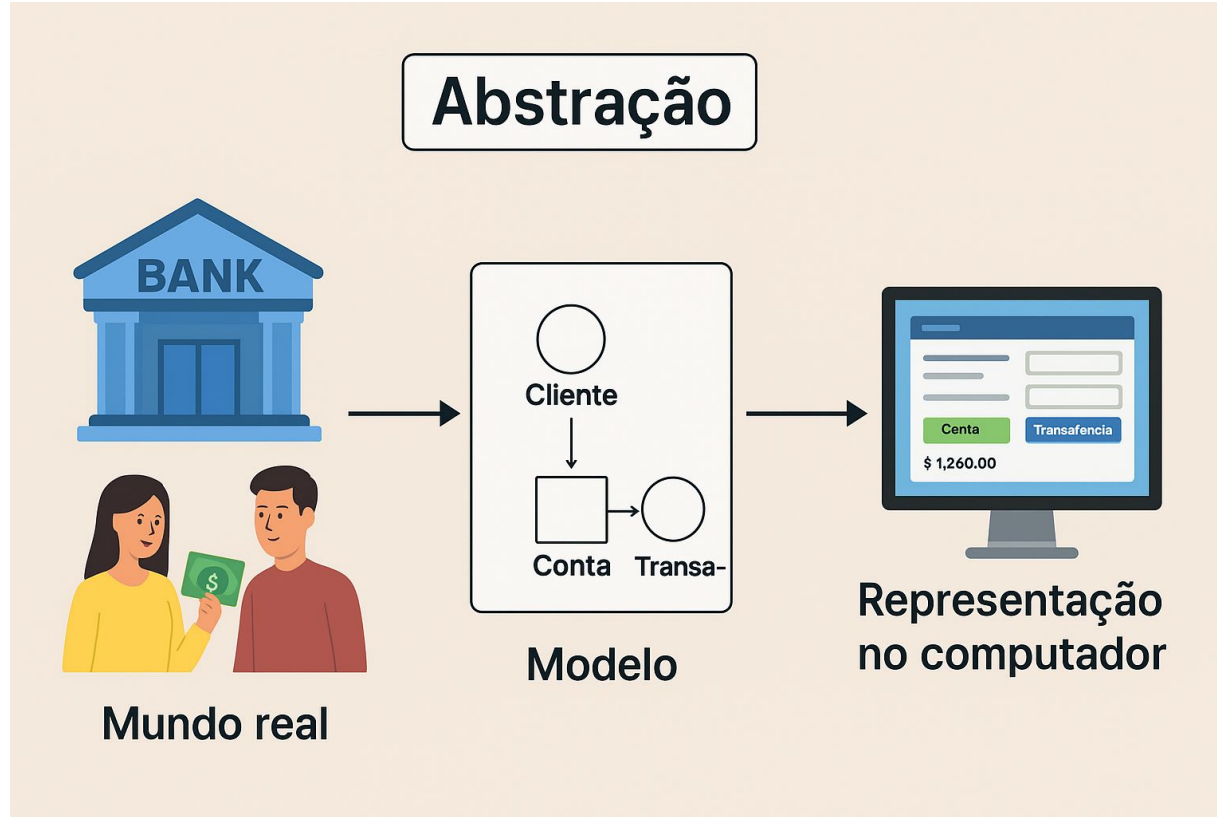
Normalmente, a partir de **descrições, requisitos, entrevistas, etc.** de sistemas a desenvolver.

Exemplos...

O que é importante ?

1. Abstração do mundo real;
2. Elencar os principais personagens (objetos), suas características e comportamentos (relação uns com os outros);
3. Construir os modelos (classes e derivados) que representam os projetos dos objetos;
4. Codificar

Abstração





Conceitos básicos



Orientação a objetos

1. Classe

Projeto de um objeto, onde são definidas, de forma clara e objetiva, atributos, métodos, construtor e detalhes relacionados com o relacionamento entre os objetos;

Uma classe geralmente representa um substantivo, por exemplo: uma pessoa, um lugar, algo que seja “abstrato”

Uma Classe representa um novo tipo criado pelo Programador;

Orientação a objetos

1. Classe (notação UML)



Orientação a objetos

2. Classe (Python)

```
class Pessoa:  
    pass
```

```
class Televisao:  
    pass
```

```
class Conta:  
    pass
```

```
class Banco:  
    pass
```

Orientação a objetos

2 . Atributo (característica ou estado de um objeto)

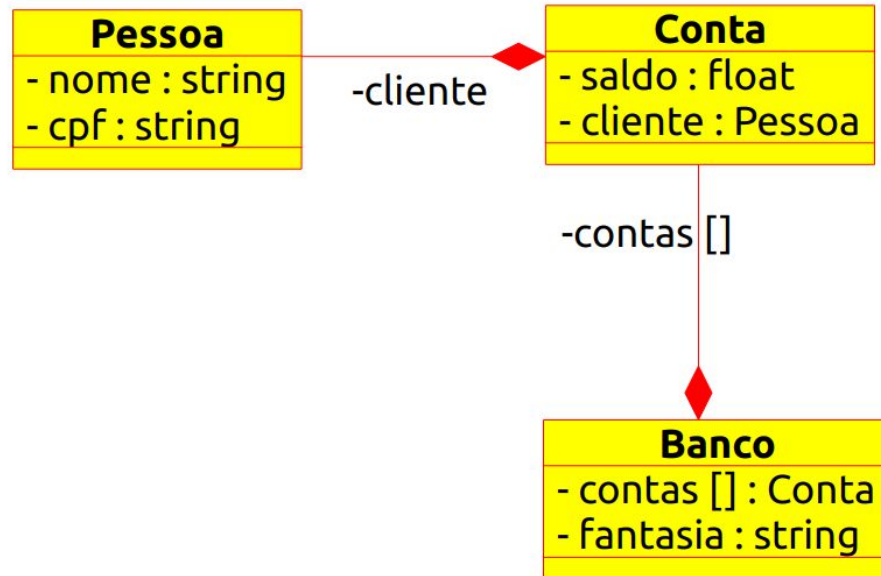
Os atributos são as propriedades de um objeto, também são conhecidos como variáveis ou campos.

Essas propriedades definem o estado de um objeto, fazendo com que esses valores possam sofrer alterações.

- a. Definidos na classe (via de regra);
- b. Valorados no objeto (via de regra);

Orientação a objetos

2 . Atributo (característica ou estado de um objeto) notação UML



Orientação a objetos

2 . Atributo (característica ou estado de um objeto) Python

```
class Pessoa:  
    nome=None  
    cpf=None
```

```
class Conta:  
    saldo = 0.0  
    cliente = Pessoa()
```

```
class Banco:  
    contas = []  
    fantasia = None
```

Orientação a Objetos

3 . Métodos (comportamento do objeto)

Os métodos são ações ou procedimentos, onde podem interagir e se comunicarem com outros objetos.

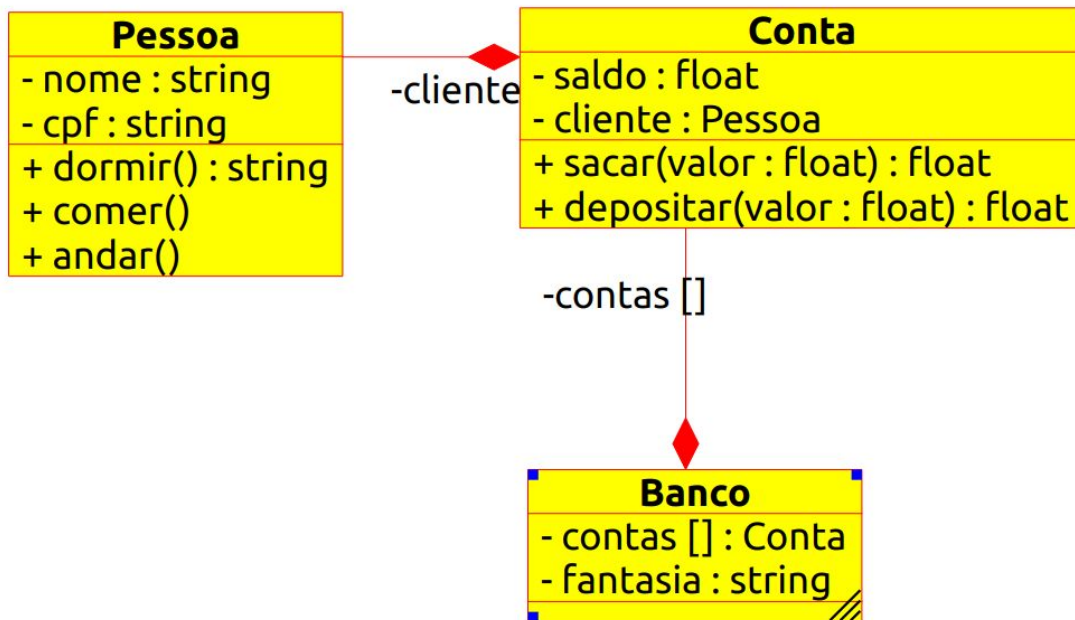
A execução dessas ações se dá através de mensagens, tendo como função o envio de uma solicitação ao objeto para que seja efetuada a rotina desejada.

Na Classe, são implementados como funções/procedimentos;

Objetos chamam métodos (enviam mensagens);

Orientação a Objetos

3 . Métodos (comportamento do objeto) notação UML



Orientação a Objetos

3 . Métodos (comportamento do objeto) Python

```
class Pessoa:

    #atributos
    nome=None
    cpf=None

    #métodos
    def dormir(self):
        return "zzzzZZZZZZ fiiuuuuuu"
    def andar(self, passos):
        passos = passos + 1
    def comer(self):
        return 2
```

Orientação a objetos

4. Construtor

O construtor de uma classe é um método especial, pois inicializa seus atributos toda vez que é instanciado (inicializado).

Constrói o objeto em si a partir da existência de uma classe;

São chamados no momento da construção de um objeto;

Orientação a objetos

4. Construtor (Python)

Em Python é o método `__init()` que vai inicializar o objeto chamando o método construtor `__new()`;

Seu primeiro parâmetro, assim como todo método de instância, é a própria instância `self`;

Em Python, gerlamente, os atributos são definidos no método `__init()`, garantindo assim a inicialização dos mesmos;

Orientação a objetos

4. Construtor (Python)

```
class Conta:  
    def __init__(self, numero, saldo, titular, limite):  
        self.numero = numero  
        self.saldo = saldo  
        self.titular = titular  
        self.limite = limite
```

Orientação a objetos

5 . **Objeto**

Espaço na memória referenciado por uma variável do mesmo tipo da classe;

Só pode ser criado após a existência de uma classe;

Criado pela chamada do construtor;

Orientação a objetos

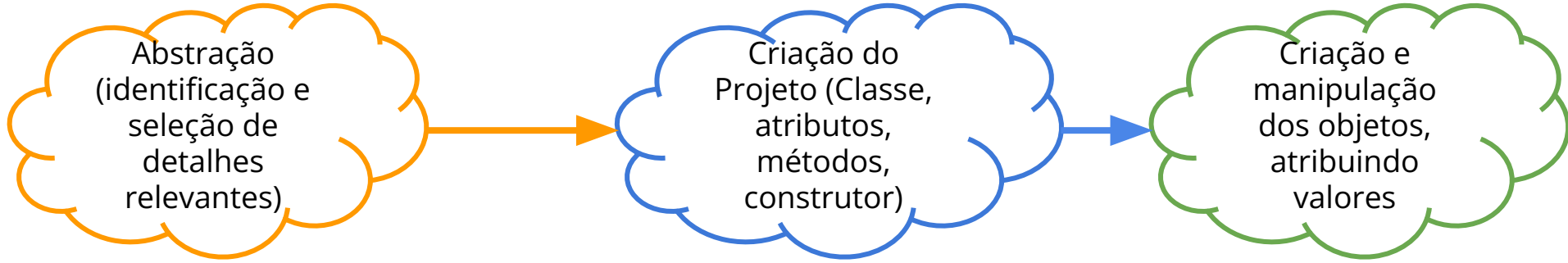
5 . Objeto

```
class Conta:
    def __init__(self, numero, saldo, titular, limite):
        self.numero = numero
        self.saldo = saldo
        self.titular = titular
        self.limite = limite
```

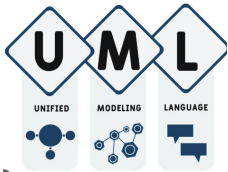
```
c1 = Conta(1234, 0.00, "José", 1000.00)
c2 = Conta(98776, 100.00, "Raimundo", 500.00)
```

```
print(c1)
print(c2)
```

Observação



Tarefa mental, rabiscos e
desenhos



Código Fonte (Programação)



Exercício

Imagine um sistema de controle bancário composto com informações sobre clientes, funcionários, contas(e seus diferentes tipos), movimentações financeiras, etc...

1. Quais seriam os objetos? E quais seriam as classes? E quais seriam os atributos e métodos de cada classe?
2. Construa o diagrama de classes e o código fonte dos elementos que você conseguiu levantar.

Exemplo (Abstração)

Identificar objetos,
atributos e
comportamentos

Funcionário

Agência

Diretor

Cliente

Caixa
eletrônico

Endereço

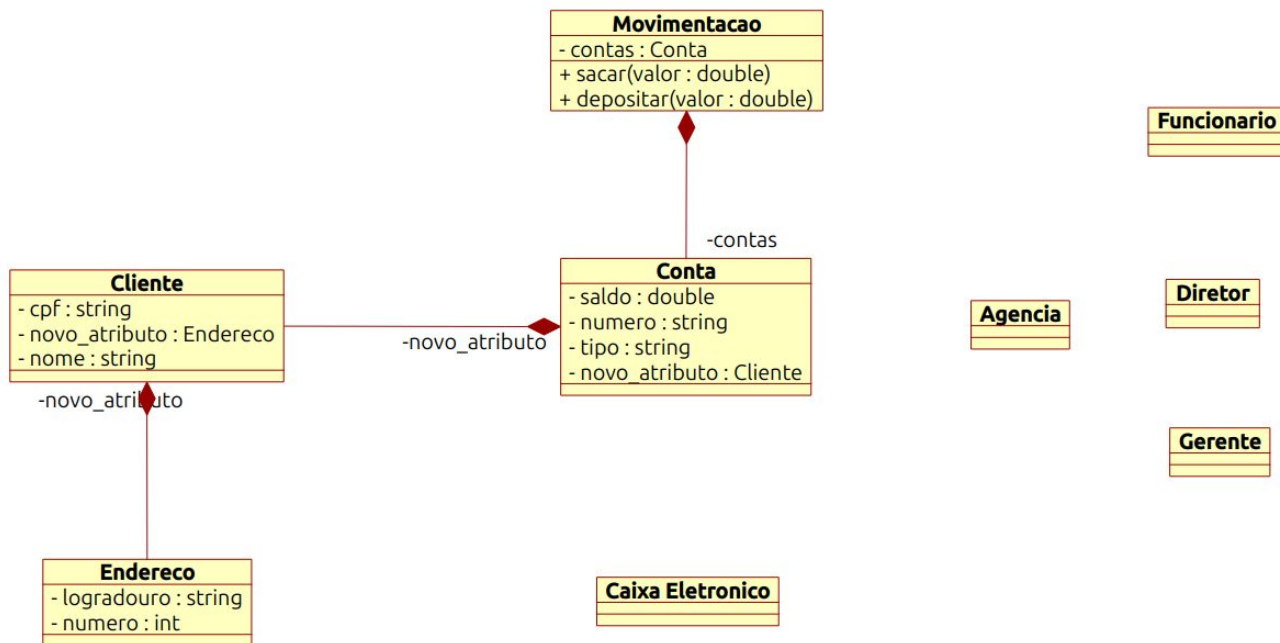
Conta

Movimentação

Gerente

Exemplo (Abstração)

Identificar objetos,
atributos e
comportamentos



Exemplo



Construção do
projeto

```
1 class Conta:
2     # Em Python, comumente definimos
3     # atributos no construtor da classe
4     def __init__(self, saldo, numero, tipo, cliente):
5         self.saldo = saldo
6         self.numero = numero
7         self.tipo = tipo
8         self.cliente = cliente
9
10    def exibir_informacoes(self):
11        print('-----')
12        print(f"Cliente:{self.cliente.nome}")
13        print(f"Saldo:{self.saldo}")
14        print(f"Número:{self.numero}")
15        print(f"Tipo: {self.tipo}")
16
```

Exemplo

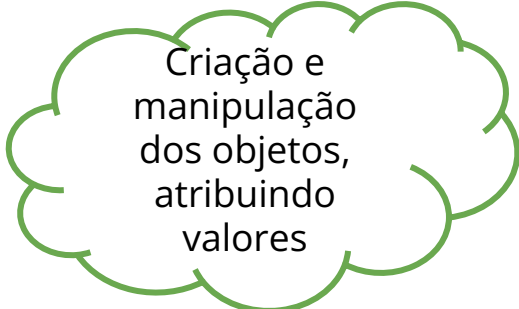


Construção do
projeto

```
1 class Cliente:
2
3     def __init__(self, cpf,nome,endereco):
4         self.cpf = cpf
5         self.nome = nome
6         self.endereco = endereco
7
```

```
1 class Endereco:
2     def __init__(self, logradouro,numero):
3         self.logradouro = logradouro
4         self.numero = numero
5
```

Exemplo



Criação e
manipulação
dos objetos,
atribuindo
valores

```
1  from Cliente import Cliente
2  from Conta import Conta
3  from Endereco import Endereco
4
5  p1 = Cliente("987.666.444-99", "Kauã", endereco = Endereco("Rua Irmãos Pereira", 554))
6
7  p2 = Cliente("222.654.987-88", "Francisco", endereco = Endereco("Avenida Flores", 23))
8
9  c1 = Conta(1000.00, "1234", "Poupança", p2)
10
11 # Criar conta c2 e tornar pessoa p1 dona da conta
12 c2 = Conta(300.0, "098-66", "Corrente", p1)
13 |
14 c1.exibir_informacoes()
15 c2.exibir_informacoes()
```

Exercício

Considerando os cenários que serão apresentados, realize a atividade de abstração, elencando os principais objetos, implementando suas classes, atributos e métodos, usando a linguagem de programação Python.

Sistema de informação de academia

Trata-se de um sistema que visa dar suporte ao funcionamento de academia de ginástica. O sistema é utilizado pela administração da academia, bem como pelos alunos (clientes) da mesma. Cada aluno é identificado através de um cartão magnético (CPF). O sistema inicia a atuar na porta da academia, onde o aluno passa o seu cartão e o sistema libera a sua entrada. Ao chegar à sala de musculação o aluno se dirige a um outro terminal e passa novamente o cartão, nesse caso o sistema exibe a lista de exercícios que ele deverá realizar, e quem será o professor a lhe acompanhar.

Sistema de informação de Clínica Veterinária

Os clientes primeiramente marcam consultas com a secretária, fornecendo suas informações pessoais e as dos animais que desejam tratar. A secretaria deverá criar ou atualizar o cadastro. Em cada sessão de tratamento (uma sessão equivale a uma consulta), o cliente deve informar os sintomas aparentes do animal, os quais devem ser registrados, dependendo do diagnóstico do médico-veterinário, o veterinário pode marcar exames para o animal.

Sistema de informação de uma biblioteca

Livros, revistas, periódicos e artigos estão disponíveis para locação na biblioteca do IFPI campus Pedro II. Sabendo que cada obra pode ser emprestada para um aluno ou servidor do campus, e que este pode ficar no máximo 1 semana antes da devolução, elabore um sistema que armazene dados sobre empréstimos desta obra para os atores descritos acima. Cada pessoa que tem direito a empréstimos deve possuir uma matrícula ativa no sistema. Ao completar uma semana de empréstimo o livro pode ser renovado(por mais uma semana) ou devolvido ao acervo. A multa por dia de atraso é de R\$1,00.

Sistema de informação de controle bancário

Imagine um sistema de controle bancário onde você pode cadastrar pessoas físicas e pessoas jurídicas. Neste sistema, cada pessoa pode ter uma ou várias Contas associadas a si. Os tipos de conta são: conta corrente, conta poupança e conta especial. A conta poupança rende juros de 1% ao mês para o cliente. A conta especial tem limite de R\$500,00 para aquele cliente que ficar no vermelho. A conta corrente é uma conta simples que tem uma tarifa de R\$ 50,00 para o cliente que opte por ela. O sistema, pode ainda, cadastrar Conta, efetuar saques, depósitos e transferências.

Sistema de informação de locação de veículos

Veículos estão disponíveis para locação na IFVeículos. Sabendo que clientes cadastrados podem alugar um veículo para transitar pela cidade, existem pacotes que o cliente pode escolher. Todo carro locado é entregue com o tanque cheio e um limite de 500km. A diária custa R\$300,00. Ao devolver o veículo, serão observados a diária e a quantidade de quilômetros rodados. Se ultrapassar 500km, cada km terá um custo adicional de R\$10,00. O cliente pode devolver o carro sem se preocupar com a quantidade de combustível.

Próxima aula

Início do Jogo

Encapsulamento

Atividade prática em laboratório

Considere que estamos prestes a iniciar o desenvolvimento do jogo Super Mario Brother.

1. Quais seriam os objetos? E quais seriam as classes? E quais seriam os atributos e métodos de cada classe?
2. Construa o diagrama de classes e o código fonte dos elementos que você conseguiu levantar.



Versão 0 - Estrutura inicial do jogo

Objetivos (versão 0.0)

Conceitos abordados:

Abstração: identificar os elementos essenciais do jogo (ex.: jogador, cenário).

Classe: criar a estrutura de um **jogador**.

Objeto: instanciar o **jogador** no programa.

Atributos: definir **características** (nome, posição, energia, etc.).

Métodos: definir **comportamentos** (mover, pular, perder energia).

Construtor: **inicializar** o estado do objeto.

Atividade

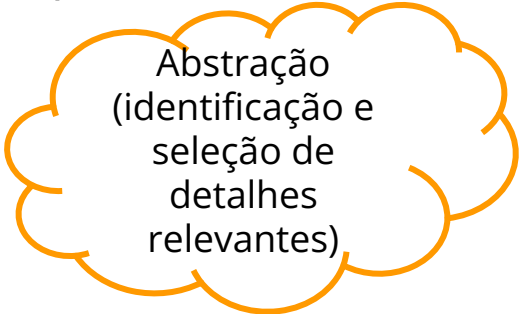
Crie um programa que simule um jogador em um jogo simples.

O jogador **tem** nome, energia e posição.

Ele **pode** mover-se e descansar.

Toda vez que se **move, gasta energia**.

Quando **descansa, recupera energia**.



Abstração
(identificação e
seleção de
detalhes
relevantes)

Atividade

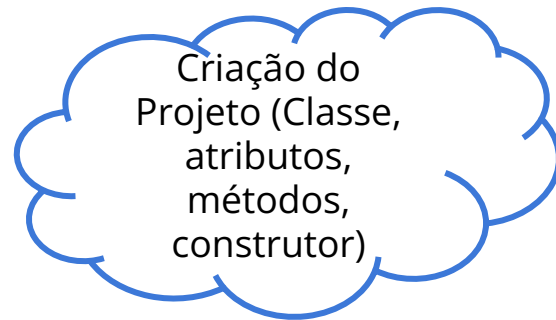
Crie um programa que simule um jogador em um jogo simples.

O jogador **tem** nome, energia e posição.

Ele **pode** mover-se e descansar.

Toda vez que se **move, gasta energia**.

Quando **descansa, recupera energia**.



Atividade

```
1  #Classe que representa o jogador do jogo.
2  class Jogador:
3      #Atributos nome,energia e posicao (declarados e inicializados no construtor)
4
5      # Construtor - inicializa o estado do objeto
6      def __init__(self, nome):
7          self.nome = nome
8          self.energia = 100
9          self.posicao = 0
10
11     # Método mover: Move o jogador para frente ou para trás (quantidade de passos)
12     # e gasta energia ao se mover
13     def mover(self, passos):
14         self.posicao += passos
15         self.energia -= abs(passos) * 2
16
17     # Método que exibe uma mensagem de status, exibindo informações sobre o jogador
18     def status(self):
19         """Exibe o estado atual do jogador"""
20         return f"Jogador: {self.nome} | Posição: {self.posicao} | Energia: {self.energia}"
21
22     # Método que representa repouso do jogador e consequentemente restabelecendo a energia.
23     def descansar(self):
24         """Recupera parte da energia"""
25         self.energia = min(self.energia + 10, 100)
```

Atividade

Crie dois objetos Jogador, com nomes diferentes.

Faça cada jogador se mover uma quantidade diferente de passos.

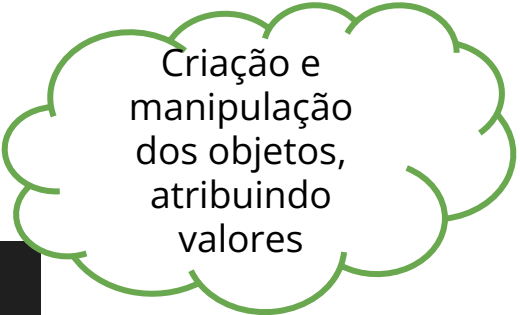
Mostre o status de cada jogador após as ações.

Crie um método novo chamado atacar(outro_jogador) que:

- Reduz a energia do outro jogador em 10.

- Exibe uma mensagem de ataque.

Atividade



Criação e
manipulação
dos objetos,
atribuindo
valores

```
1 from Jogador import Jogador
2
3 # Criação do objeto (instancia da classe)
4 j1 = Jogador("Anderson")
5
6 print("=== Versão 0 do Jogo ===")
7 print(j1.status())
8
9 # Simulação simples
10 j1.mover(5)
11 print(j1.status())
12
13 j1.descansar()
14 print(j1.status())
```

Exercícios

Resolva a lista de exercícios em anexo

Link:

[https://docs.google.com/document/d/1K1e_G-eWKB3jiZgGxMW7DONIOaSdCUUjIN1uTBaedQg/edit?usp=drive link](https://docs.google.com/document/d/1K1e_G-eWKB3jiZgGxMW7DONIOaSdCUUjIN1uTBaedQg/edit?usp=drive_link)



Versão 0.1: Movimento básico

Objetivos da versão 0.1

Continuar com a abstração e POO com movimento real na tela.

O jogador (classe Jogador) poderá:

- mover-se para esquerda/direita

- pular

- abaixar

O programa mostrará o personagem como um retângulo colorido (ainda sem sprite).

Verificações ...

Verificar se o pygame está instalado

```
$ pip show pygame
```

Se tudo estiver ok ...

- Criar um ambiente virtual dentro da pasta do jogo

- Instalar o pygame dentro da pasta

- Rodar a versão 0.1 do jogo

No terminal, rode:

```
python3 -m venv .venv  
source .venv/bin/activate  
pip install pygame
```

Depois volte para a pasta jogo e rode o arquivo main.py

Acompanhe

Conceitos trabalhados até agora

Classe: **class Jogador** define o modelo do personagem

Objeto: **jogador = Jogador(100, 300)** cria uma instância

Atributos **x, y, velocidade, pulando, altura**, etc.

Métodos **mover() e desenhar()** controlam o comportamento

Construtor **__init__()** inicializa o objeto e seu estado

Abstração **o jogador é uma simplificação visual (retângulo)**

Próxima aula . . .

Encapsulamento

Outras versões

Código da aula de hoje disponível no github poo.

