ATIVIDADES POO - LIVE 10

ABSTRAÇÃO

🅐 Atividade A — Classe `Animal` (Clínica Veterinária)

📌 Objetivo

Uma clínica veterinária precisa de um sistema simples para **registrar os animais atendidos**.
Cada animal possui informações básicas que ajudam na identificação, no acompanhamento e no atendimento veterinário.

Seu objetivo é **abstrair um Animal do mundo real** e representá-lo por meio de uma **classe em Node.js**, definindo apenas **atributos e métodos que façam sentido para esse contexto**.

Você **não deve se preocupar com banco de dados, validações ou regras complexas**.
O foco é pensar: **o que é um animal para uma clínica veterinária?**

❓ Perguntas guia

1. Quais **informações básicas** são necessárias para identificar um animal em uma clínica?
2. O **tipo do animal** (cachorro, gato, etc.) deve ser representado? Como?
3. Um animal possui **idade, peso ou raça**? Quais desses dados são relevantes?
4. Quais **ações** um animal pode realizar dentro do sistema?
    - Ser cadastrado?
    - Ter seus dados exibidos?
    - Atualizar alguma informação?
5. Existem outros métodos fazem sentido para o sistema, relacionado ao animal?Se sim, quais?

🅑 Atividade B — Classe `Filme` (Plataforma de Conteúdo Digital)

📌 Objetivo

Uma plataforma de streaming deseja organizar seu catálogo de filmes.
Cada filme precisa conter informações que permitam **exibição ao usuário**, **classificação do conteúdo** e **interação básica com a plataforma**.

Sua tarefa é **abstrair o conceito de um Filme** e criar uma classe que represente esse objeto do mundo real.

Pense no filme **como um item de um catálogo digital**, não como um arquivo técnico.

❓ Perguntas guia

1. Quais dados são essenciais para identificar um filme em uma plataforma?
2. O filme precisa ter **título, gênero, duração ou ano de lançamento, premiações**?
3. Um filme pode ter **classificação indicativa**? Isso é relevante?
4. Quais ações fazem sentido para um filme dentro da plataforma?
    - Exibir informações?
    - Marcar como assistido?
    - Iniciar reprodução?

🅒 Atividade C — Classe `Produto` (Sistema de Estoque)

📌Objetivo

Uma empresa precisa de um sistema simples para **controlar produtos em estoque**.
Cada produto possui características que ajudam na **organização, controle e visualização** dentro do sistema.

Seu objetivo é **abstrair um Produto do mundo real** e criar uma classe que represente esse conceito em Node.js.

Considere apenas **um produto individual**, não o estoque completo.

❓ Perguntas guia

1. Quais informações são fundamentais para identificar um produto?
2. Um produto deve ter **nome, preço, categoria ou código**?
3. A **quantidade em estoque** deve ser um atributo do produto?
4. Quais ações fazem sentido para um produto?
    - Exibir informações?
    - Atualizar quantidade?
    - Aplicar desconto?
5. Quais outros métodos podem representar **comportamentos naturais de um produto em um sistema de estoque**?

🎯 Orientações

- Foquem em **pensar antes de codar** ✏️
- Listem os **atributos** primeiro
- Depois, pensem nos **métodos como ações do objeto**
- Não existe resposta única — o importante é **coerência com o problema**
- O código deve ser simples e legível

ENCAPSULAMENTO

🅓 Atividade D — Classe `Usuario` (Sistema de Usuários)

📌 Objetivo

Um sistema precisa gerenciar usuários cadastrados para permitir acesso a determinadas funcionalidades. Cada usuário possui **informações públicas** e **informações sensíveis**, que **não devem ser acessadas ou alteradas diretamente fora da classe**.

Seu objetivo é **abstrair o conceito de um usuário do mundo real** e representá-lo em uma **classe em Node.js**, aplicando os seguintes conceitos:

🔹 **Abstração -** Identificar quais **características (atributos)** e **comportamentos (métodos)** definem um usuário dentro de um sistema.

🔹 **Encapsulamento -** Proteger os **dados sensíveis**, permitindo que o acesso ou a modificação desses dados aconteça **apenas por meio de métodos controlados**, definidos dentro da própria classe.

💡 **Dica importante:**
Em Programação Orientada a Objetos, os métodos responsáveis por **acessar ou alterar atributos privados** costumam ser chamados de:

- **Getters (`get`)** → usados para **obter** o valor de um atributo privado
- **Setters (`set`)** → usados para **alterar** o valor de um atributo privado de forma controlada

Esses métodos ajudam a manter a **segurança e a organização do código**, garantindo que os dados não sejam manipulados diretamente.

🚫 Você **não deve se preocupar** com: Banco de dados, Autenticação real, Criptografia
🎯 O foco desta atividade é **a modelagem do objeto** e o **controle de acesso aos dados** usando **abstração e encapsulamento**.

❓ Perguntas guia — Abstração

1. Quais informações são **necessárias para identificar um usuário** em um sistema?
2. Um usuário precisa ter **nome, email, login ou tipo de perfil**?
3. Quais **ações um usuário pode realizar** dentro do sistema?
    - Visualizar seus próprios dados?
    - Atualizar informações?
    - Ativar ou desativar sua conta?
4. Quais desses comportamentos devem ser **métodos da classe**?

🔒 Perguntas guia — Encapsulamento

1. Quais dados de um usuário **não devem ser acessados diretamente**?
    - Senha?
    - Status da conta?
2. Como você pode **proteger esses dados** dentro da classe?
3. Quais métodos são necessários para:
    - Alterar a senha de forma controlada?
    - Verificar se o usuário está ativo?
4. Faz sentido permitir **leitura direta** de todos os atributos? Por quê?

🎯 Requisitos da atividade

- Criar uma classe `Usuario`
- Definir:
    - **Atributos públicos** (ex.: nome, email)
    - **Atributos privados** (ex.: senha, status)
- Criar **métodos públicos** para:
    - Exibir informações do usuário (sem mostrar dados sensíveis)
    - Alterar a senha
    - Ativar ou desativar o usuário
- **Não permitir** acesso direto aos atributos privados fora da classe