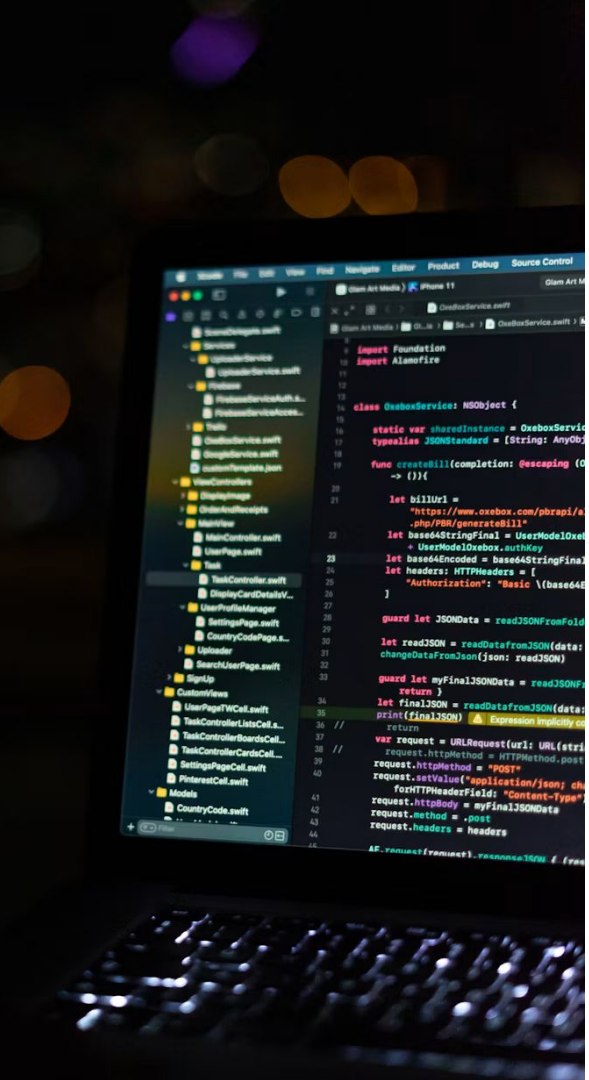


5954018 - Programação Orientada a Objetos

Aula 02 - Paradigmas de Programação e Conceitos Fundamentais de POO

Prof. Dr. Denis Martins

DCM | FFCLRP | USP

Objetivos de Aprendizagem

01 Entender o conceito de **paradigma de programação**

02 Reconhecer características da programação **não estruturada, procedural e modular**

03 Explicar as motivações para o surgimento da **orientação a objetos**

04 Conhecer os **conceitos fundamentais** de orientação a objetos: objetos, classes, métodos, atributos, herança e polimorfismo

Créditos: slides baseados no material de aula original do Prof. Clever Farias.

Paradigmas de Programação

As linguagens de programação evoluíram de acordo com diferentes paradigmas

01

Programação não-estruturada

02

Programação procedural

03

Programação modular

04

Programação orientada a objetos

> Um paradigma é uma **forma de pensar** e organizar programas. Ele orienta como dados e operações aparecem no código. Diferentes paradigmas resolvem **o mesmo problema** de formas distintas.

Exemplo: Conta Bancária

CONTA BANCÁRIA

Número: **59.123-4**

Saldo: **R$ 1.000,00**

Ações possíveis (simplificadas)

⊕ Creditar

Adicionar algum valor ao saldo de uma conta.

⊖ Debitar

Subtrair algum valor ao saldo de uma conta.

Exemplo: Conta Bancária

Exemplo: Conta Bancária

Programação Não-Estruturada

Paradigma caracterizado pela **ausência de elementos estruturantes (loop, funções, etc.)** para a construção de um programa computacional.

Visão Geral:

- **Programa:** sequência direta de declarações ou comandos executados sequencialmente.
- Cada declaração ou comando modifica diretamente um conjunto de **dados globais** compartilhado por todo o sistema.
- Por exemplo, se a mesma seqüência é necessária em localizações diferentes ela **deve ser copiada**.

O fluxo pode **saltar** para diferentes partes do código.
Como uma receita que manda voltar várias páginas.

Programação Procedural

Paradigma caracterizado pela **divisão** de um programa em "pedaços" menores: **procedimentos** (ou funções).

Visão Geral:

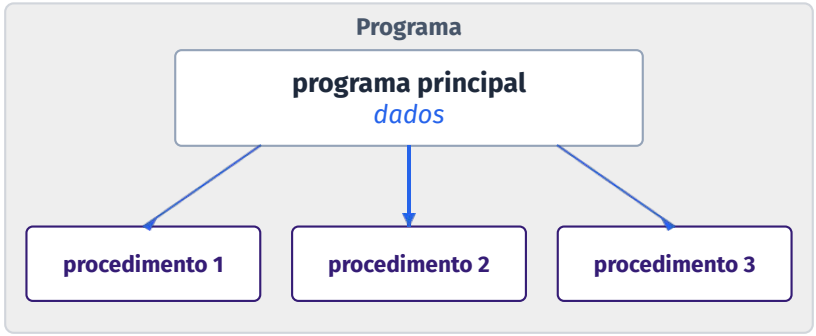
- **Programa:** a execução baseia-se fortemente em uma sequência lógica e ordenada de operações.
- Cada função ou procedimento executa uma única operação bem definida.
- O código é estruturado de forma limpa e modularizada através de funções.

Funções e dados ficam separados. Em sistemas grandes, muitas funções podem alterar os mesmos dados.

Fluxo de Controle

Ativação e Retorno:

Cada chamada ativa o procedimento para executar sua tarefa. Após a execução, **o controle retorna** exatamente para o ponto seguinte.



Exemplos: C, Pascal, Fortran

Programação Modular

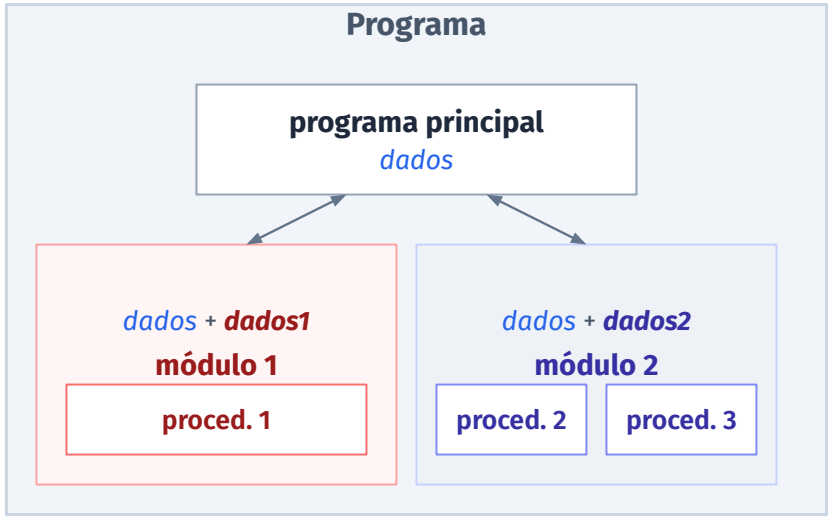
Paradigma caracterizado pelo agrupamento de **procedimentos relacionados** em módulos separados

Visão Geral:

- **Programa:** conjunto de partes menores (módulos) que interagem por meio de chamadas de procedimentos.
- Cada módulo reúne funcionalidades relacionadas.
- O objetivo é separar responsabilidades.
- Módulos podem ser reutilizados por diferentes programas.

Cada módulo conta com seus **próprios dados**, que definem o estado interno de cada um.

Módulos ajudam a organizar o sistema. Mas ainda pode faltar uma representação direta das entidades.



Exemplos: Modula 2, Ada

Comparação entre Paradigmas

Paradigma	Organização	Foco Principal
Não-estruturado	Instruções lineares e saltos incondicionais (GOTO)	Fluxo de execução direto e linear
Procedural	Estruturas de controle (laços, condicionais) e funções	Operações e procedimentos sobre dados
Modular	Agrupamento de procedimentos e dados relacionados em módulos	Separação de responsabilidades e reutilização

Apesar da evolução, a falta de uma representação direta das entidades do mundo real e a separação rígida entre dados e comportamentos nos paradigmas anteriores motivam a transição para a **Orientação a Objetos**.

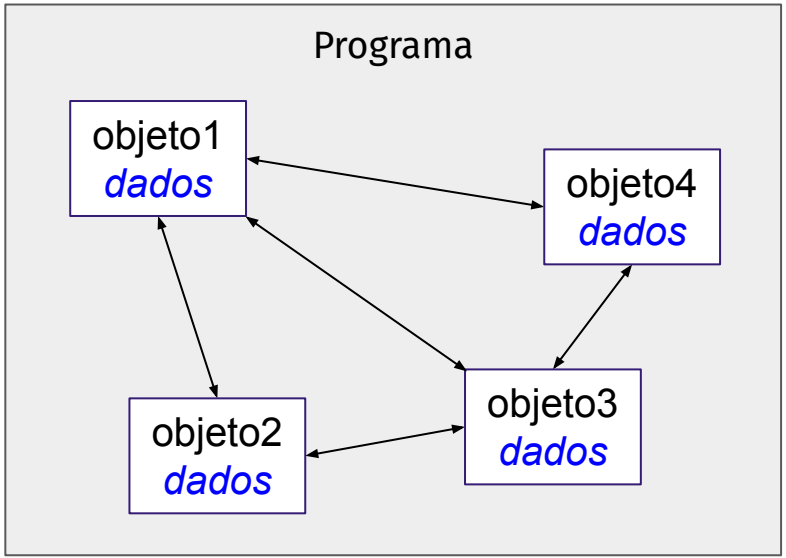
Programação Orientada a Objetos

Paradigma caracterizado pelo uso de um conjunto de **objetos interagentes**, cada qual responsável pelo gerenciamento de seu estado interno

Visão Geral:

- **Programa:** conjunto de objetos que interagem uns com os outros por meio da troca de mensagens.
- Objetos possuem **estado** e **comportamento**.
- Objetos interagem por meio de chamadas de **métodos**.

> Cada objeto é responsável pela **inicialização** e **destruição** de seus **dados internos.**



Exemplos: Eiffel, SmallTalk, C++, Java, C#

Conceitos Fundamentais: Objeto

Definição

Um objeto representa uma **entidade específica** e **individual** dentro do sistema.

No nosso exemplo

*Um objeto do tipo **Conta Bancária** encapsula todas os dados e operações que o competem.*

Exemplos

Entidades do mundo real mapeadas:

- Aluno
- Produto
- Pedido
- **Conta Bancária**

Composição

Cada objeto **encapsula** seus próprios dados (**atributos**) e as ações (**métodos**) que ele pode executar no sistema.

Conceitos Fundamentais: Classe

Definição

Uma classe é o **molde (template)** usado para criar objetos. Ela especifica atributos (dados) e métodos (operações) que esses objetos suportam.

Analogia

A **planta de uma casa** é a classe; a casa construída e habitável é o objeto.

No nosso exemplo: **Conta Bancária** é uma classe. A *conta bancária do usuário X* é um objeto.

CONTA BANCÁRIA

Atributos:

- **Número** (*string*)
- **Saldo** (*float*)

Métodos :

- **Creditar**
- **Debitar**

Relação entre Classe e Objeto

Classe

Funciona como o **molde**, planta ou **definição** a partir da qual os objetos são concebidos.

Objeto

Representa uma **instância** concreta de uma classe, com seus próprios dados e comportamentos.

Importante: *A criação de um objeto a partir de uma classe é chamada de* ***instanciação****.*

Instâncias da Classe Conta Bancária

Conta Bancária do Usuário A

Atributos (Estado)

```
numero = "12345-6"
saldo = 1500.50
```

Métodos (Comportamento)

```
+ creditar(valor)
+ debitar(valor)
```

Conta Bancária do Usuário B

Atributos (Estado)

```
numero = "98765-4"
saldo = -50.00
```

Métodos (Comportamento)

```
+ creditar(valor)
+ debitar(valor)
```

Conceitos Fundamentais: Atributos e Métodos

Atributos & Métodos

Atributos: representam as **características** ou **dados** do objeto. São armazenados no **segmento de dados**.

Métodos: representam as **ações** ou **comportamentos** que o objeto pode executar. São armazenados no **segmento de código**.

Exemplo: *Uma conta corrente possui* **saldo** *(atributo) e pode* **exibi-lo** *(método) ao usuário.*

Estado & Comportamento

Estado: é a situação ou configuração **atual** dos dados do objeto.

Comportamento: é o que o objeto sabe **fazer** (definido pelos seus métodos).

Importante: *Dois objetos da mesma classe podem possuir* **estados diferentes**.

Notação UML: Classe

O que é a notação UML?

A **UML (Unified Modeling Language)** é o padrão da indústria para visualizar e documentar a estrutura de sistemas orientados a objetos.

Uma classe é representada graficamente por um retângulo dividido em **três compartimentos**:

- **Superior:** Nome da classe
- **Intermediário:** Atributos (dados)
- **Inferior:** Métodos (operações)

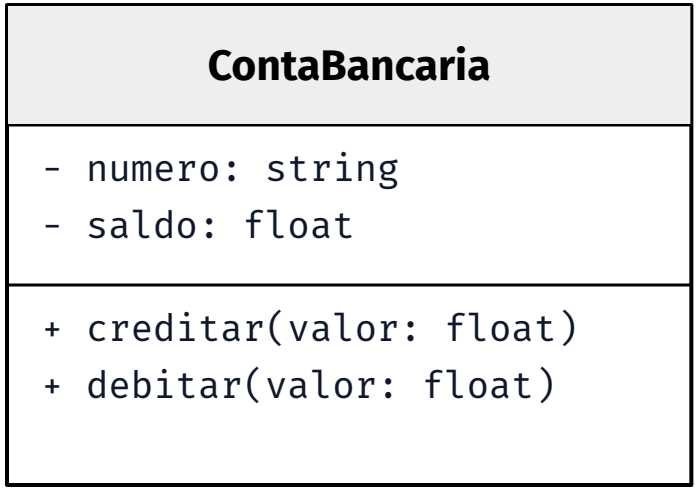


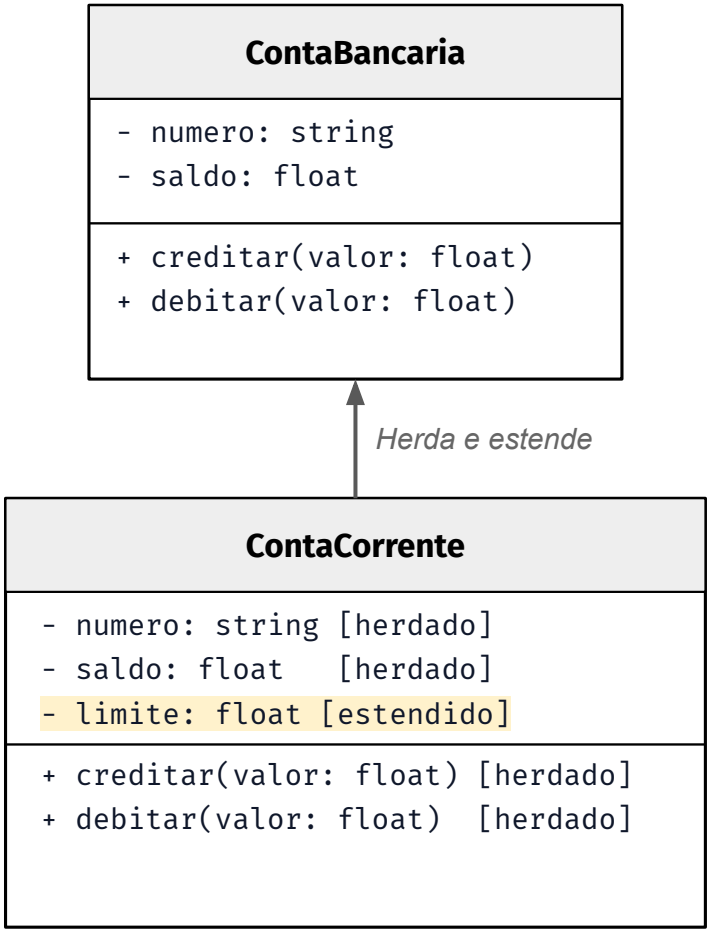
Diagrama de Classe UML para Conta Corrente

Conceitos Fundamentais: Herança

Herança: Mecanismo de Reuso de Código

Permite que uma classe especializada (**subclasse**) herde atributos e métodos de uma classe mais geral (**superclasse**).

- Relação "É-Um" (*is_a*): Uma Conta Corrente é uma Conta Bancária.
- A subclasse pode **estender** a superclasse adicionando **novos atributos** e **métodos** específicos, ou redefinindo comportamentos herdados.

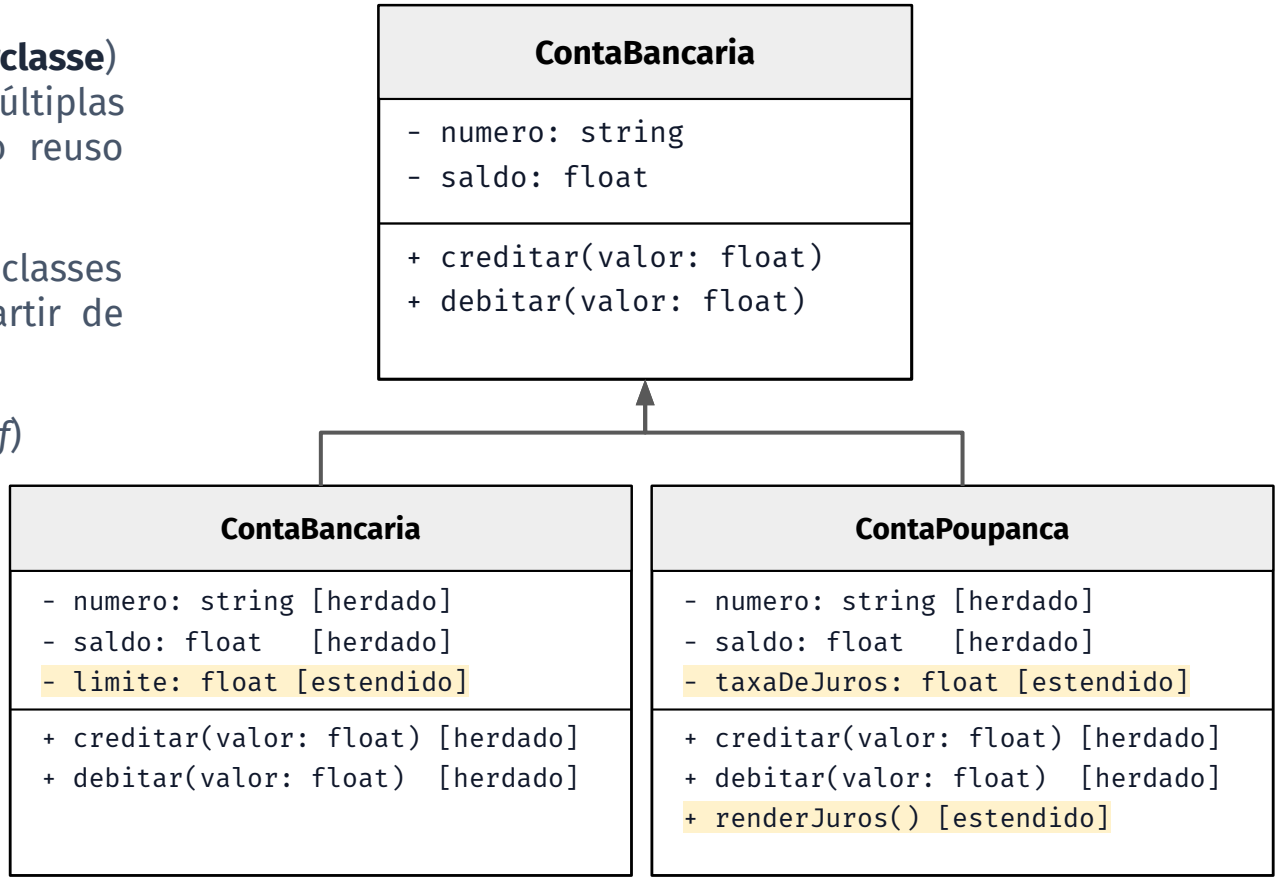


Conceitos Fundamentais: Herança

Uma mesma classe geral (**superclasse**) pode ser estendida por múltiplas subclasses distintas, permitindo reuso eficiente de código.

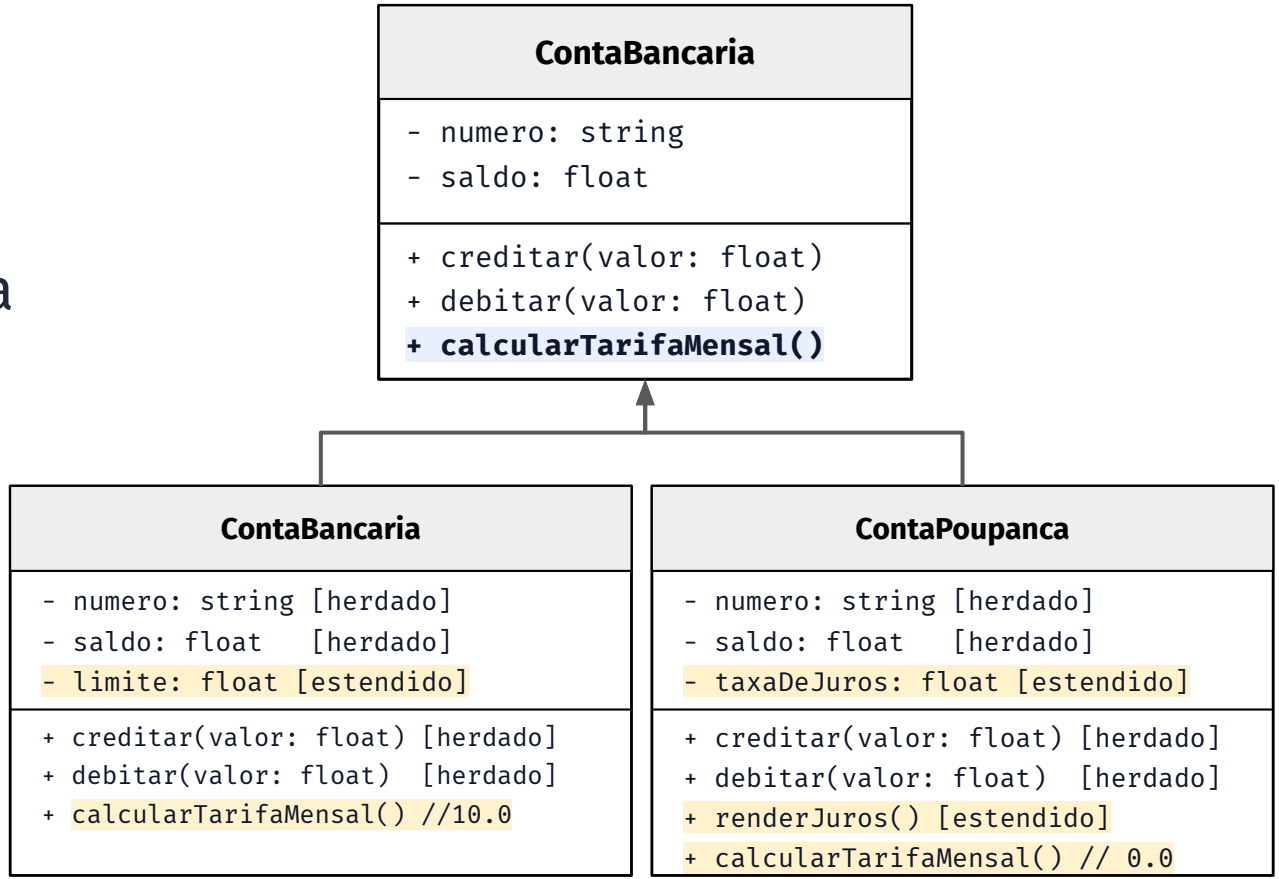
Reuso que permite a criação de classes especializadas (subclasses) a partir de uma classe base (superclasse)

Relação: é-um-tipo-de (*is_kind_of*)



Conceito Fundamental: Polimorfismo

Polimorfismo é a habilidade de uma **mesma operação** comportar-se **diferentemente** em diferentes classes.



Conclusão

Paradigma e Evolução

Paradigmas de programação definem formas de abordar problemas e organizar sistemas.

A programação evoluiu historicamente da organização por instruções sequenciais para a estruturação em torno de objetos.

Classes e Objetos

A Programação Orientada a Objetos (POO) organiza programas de forma lógica e coesa.

Classes funcionam como moldes para instanciar objetos. Cada objeto possui estado (atributos) e comportamento (métodos) próprios.

Próximos Passos

O entendimento da base conceitual abre espaço para tópicos mais complexos.

Os pilares avançados como Herança, Polimorfismo e a modelagem formal com UML serão aprofundados e detalhados nas próximas aulas.

Atividade

Contexto

Considere um aplicativo moderno de biblioteca digital da universidade (semelhante ao Kindle ou Spotify) que deve permitir:

- Criar perfis de usuários
- Explorar catálogo de e-books e audiobooks
- Registrar histórico de leitura e escuta

Sua tarefa (em grupos)

Modele as entidades principais do sistema em termos de Classes.

O que cada entidade "representa" ou "faz" dentro do sistema?

Identifique os atributos e métodos fundamentais.

Dúvidas e Discussão

Veja também o vídeo no Youtube (do Programador Primata):
https://www.youtube.com/watch?v=967tQaGKgck