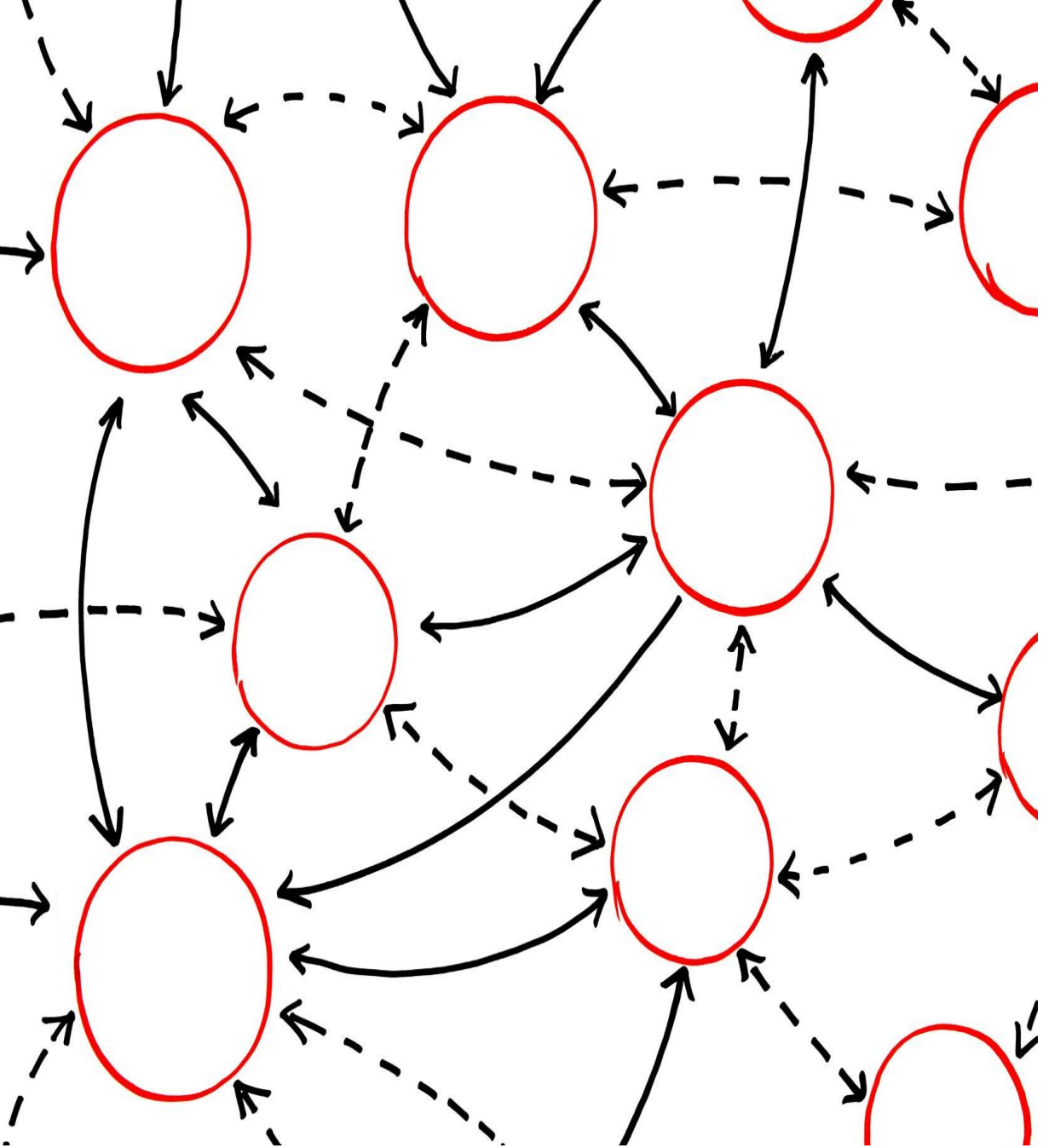
Entenda princípios essenciais usando exemplos claros em Python

CONCEITOS FUNDAMENTAIS DE ABSTRAÇÃO E HERANÇA NA ORIENTAÇÃO A OBJETOS COM EXEMPLOS PRÁTICOS EM PYTHON

PROGRAMAÇÃO DA APRESENTAÇÃO

- O conceito de abstração na orientação a objetos
- Exemplo prático de abstração: classe Pessoa em Python
- Herança na orientação a objetos: definição e aplicação
- Exemplo prático de herança em Python

O CONCEITO DE ABSTRAÇÃO NA ORIENTAÇÃO A OBJETOS



DEFININDO ABSTRAÇÃO:

Simplificando a complexidade dos Sistemas

Conceito de Abstração

Abstração representa entidades ***reais*** (classes concretas) ou ***abstradas*** (representação de conceitos) destacando somente os aspectos relevantes para o contexto específico.

Redução da Complexidade

A abstração diminui a complexidade dos sistemas, facilitando sua compreensão e gerenciamento.

VANTAGENS DA ABSTRAÇÃO: MODULARIDADE E REUTILIZAÇÃO DE CÓDIGO

Modularidade no Código

A abstração torna o código modular, facilitando a manutenção e permitindo expansão eficiente dos sistemas.

Reutilização de Código

Abstração promove reutilização, usando classes genéricas como base para outras, otimizando o desenvolvimento.

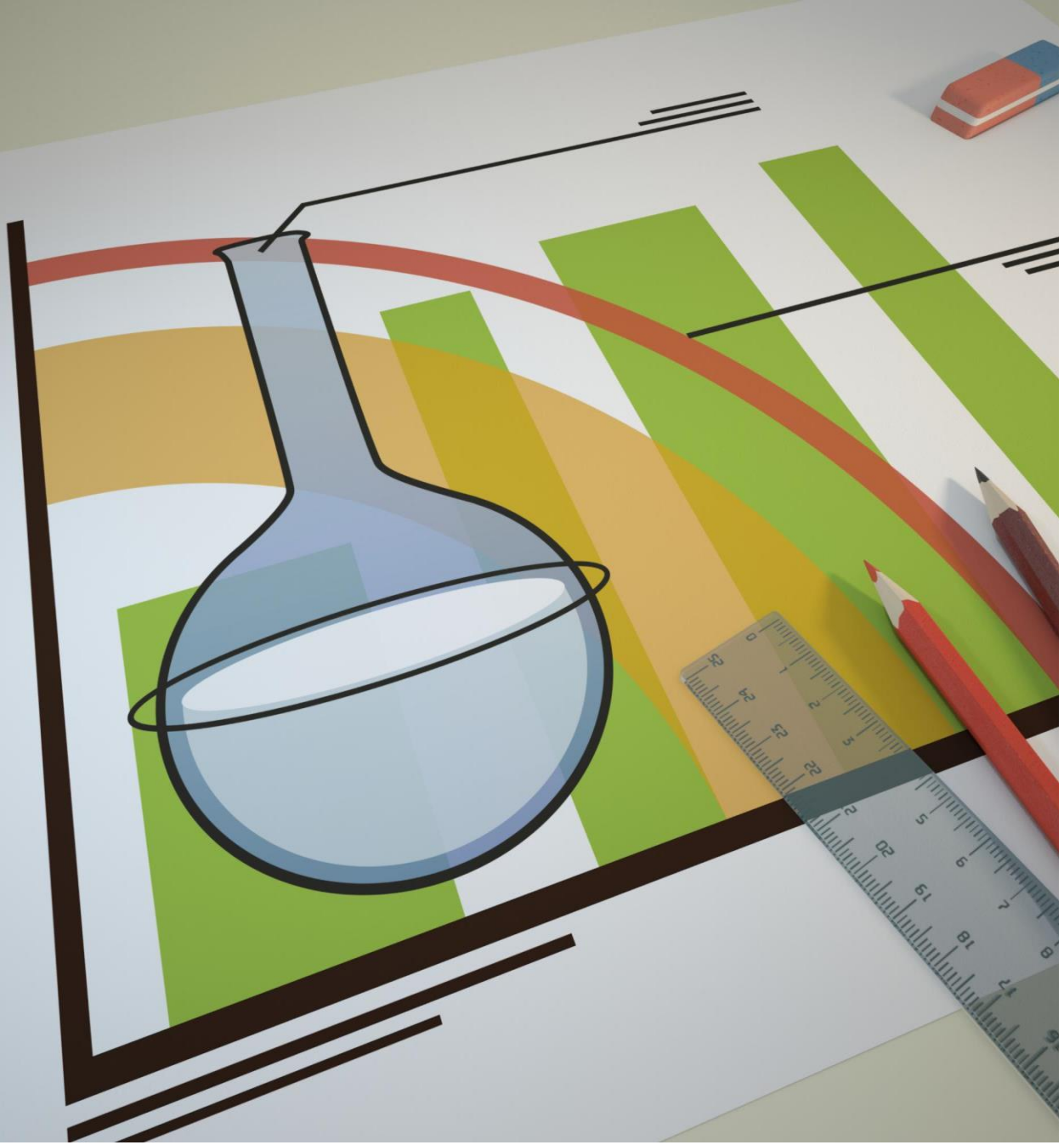


ILUSTRAÇÃO DIDÁTICA: REPRESENTANDO ENTIDADES REAIS POR MEIO DE CLASSES

Entidades do Mundo Real

Classes representam objetos reais, facilitando a organização e estruturação.

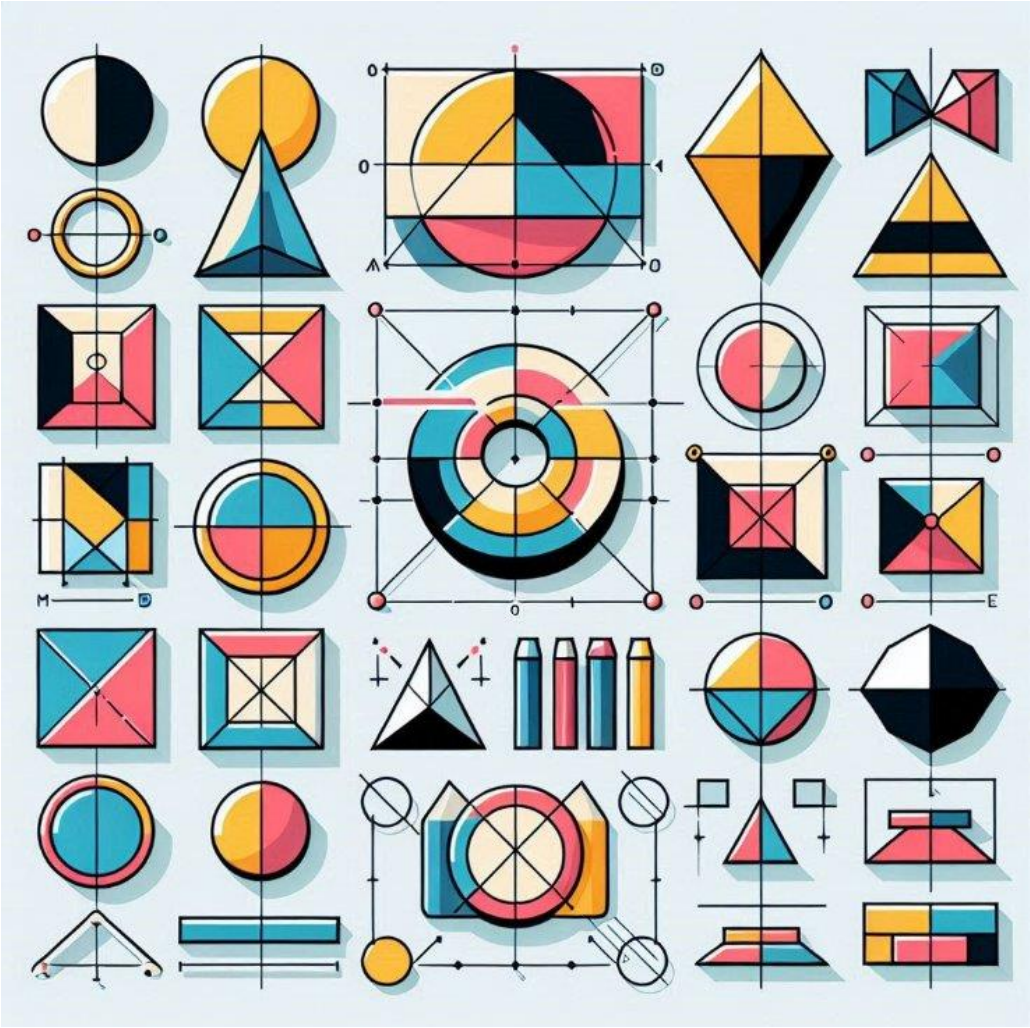
Abstração em Programação

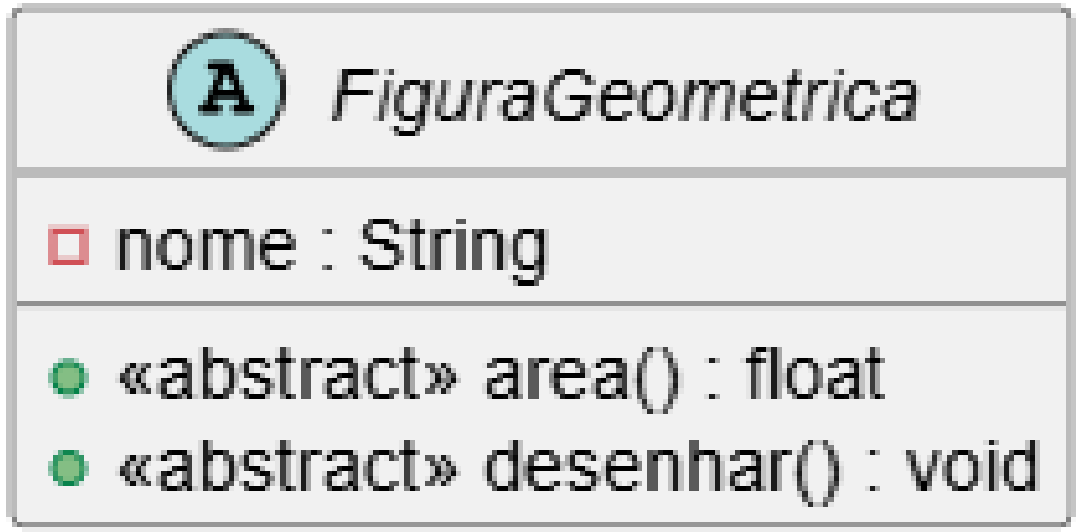
A abstração simplifica sistemas complexos modelando apenas características essenciais das entidades.

Modelagem Eficiente

Modelar entidades com classes promove reutilização e facilita manutenção de sistemas.

EXEMPLO: CLASSE ABSTRADA: UM CONCEITO



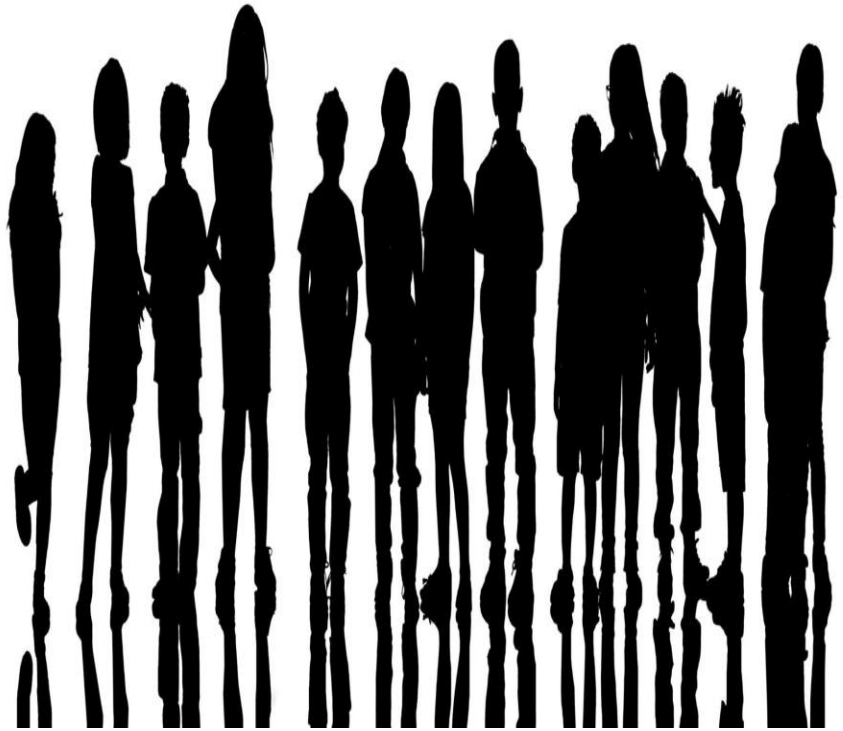


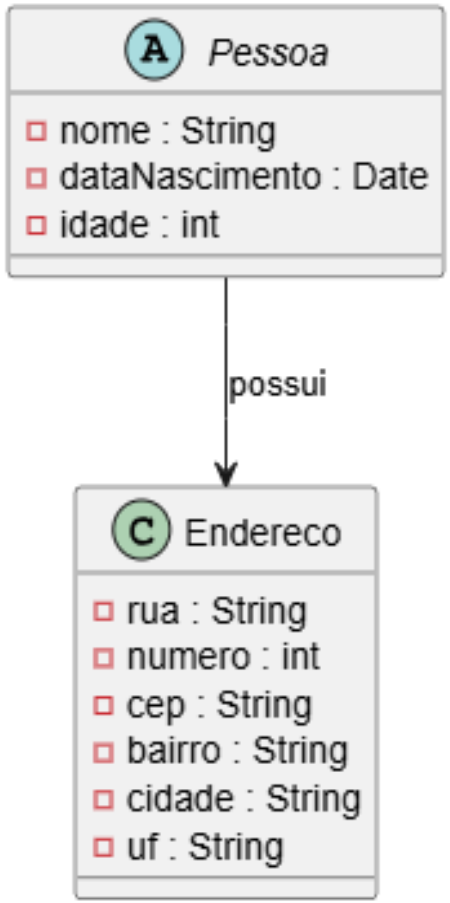
OBS:

1 – **A** : significa classe **A**bstrada

2 - Representa o **Conceito** de uma Figura Geométrica, por essa razão **não** pode ser instanciada.

EXEMPLO: CLASSE ABSTRADA PESSOA



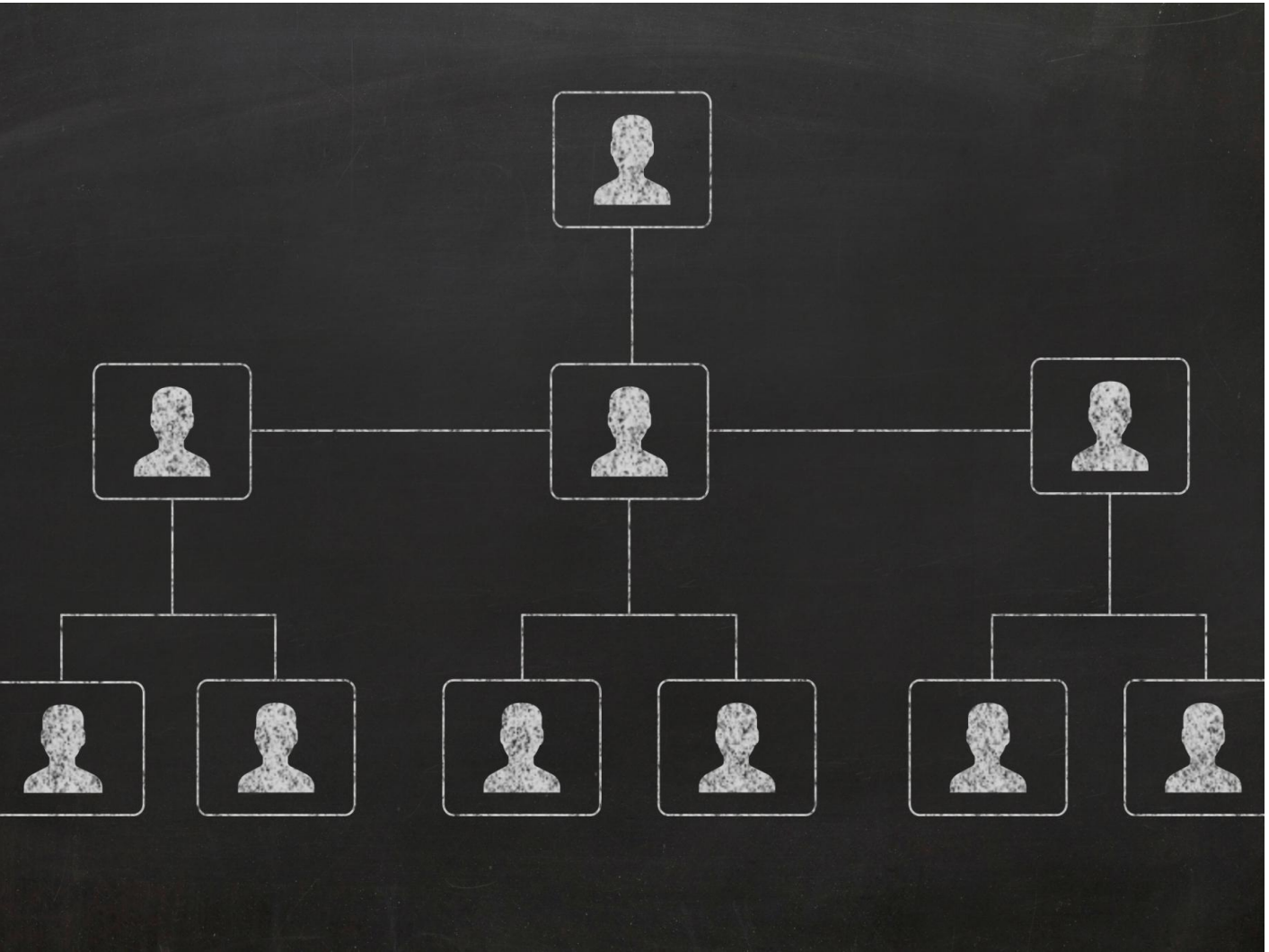


OBS:

1 – A : significa Classe **A**bstrada, **não** pode ser instanciada.

2 – C : significa Classe **C**oncreta, pode ser instanciada

ABSTRAÇÃO: EXEMPLO



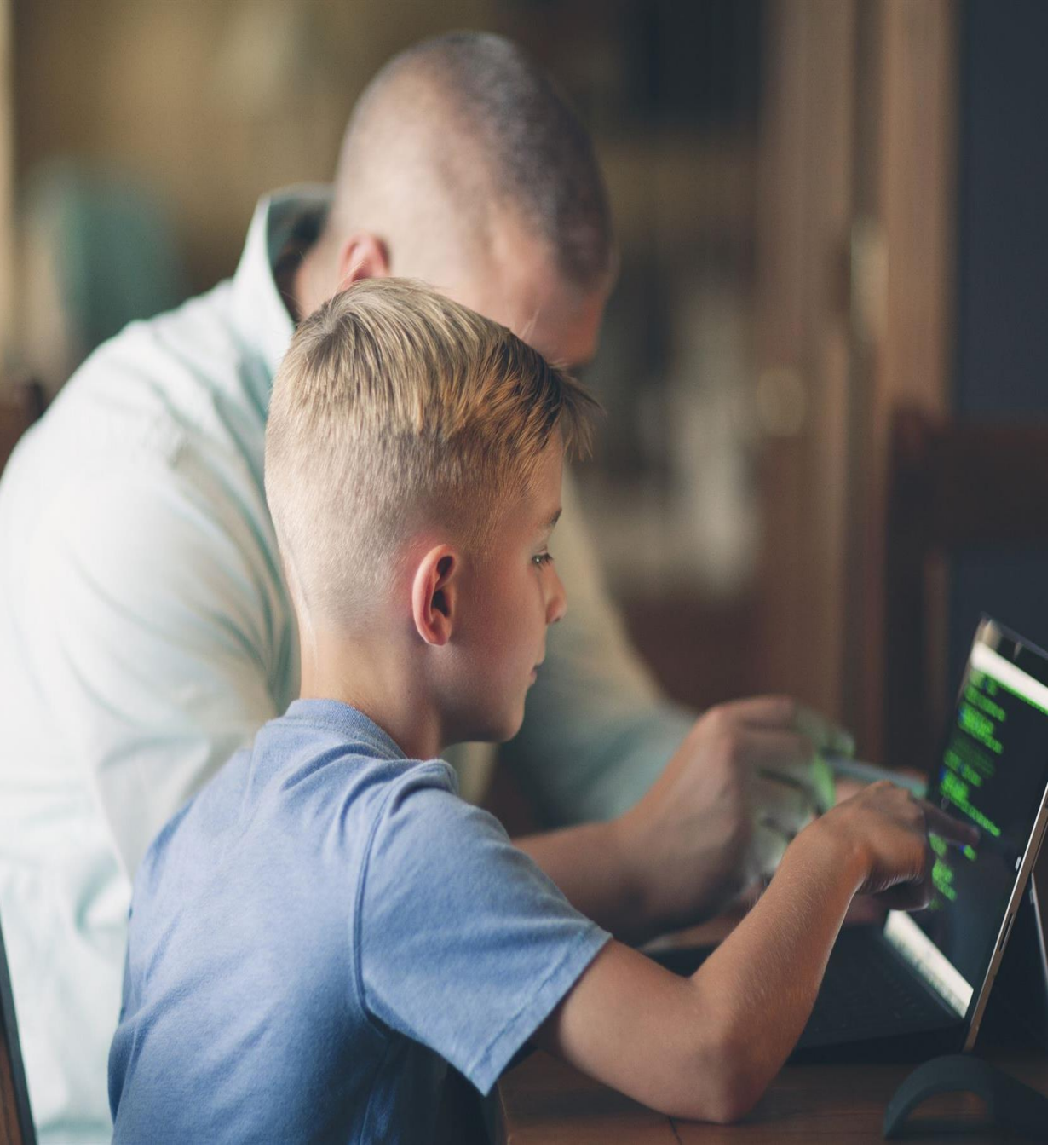
Atributos Fundamentais

A classe Pessoa inclui nome e data de nascimento para representar informações essenciais sobre uma pessoa.

Exemplo de Abstração

A estrutura foca nos aspectos relevantes, demonstrando o conceito de abstração em programação orientada a objetos.

HERANÇA NA ORIENTAÇÃO A OBJETOS: DEFINIÇÃO E APLICAÇÃO



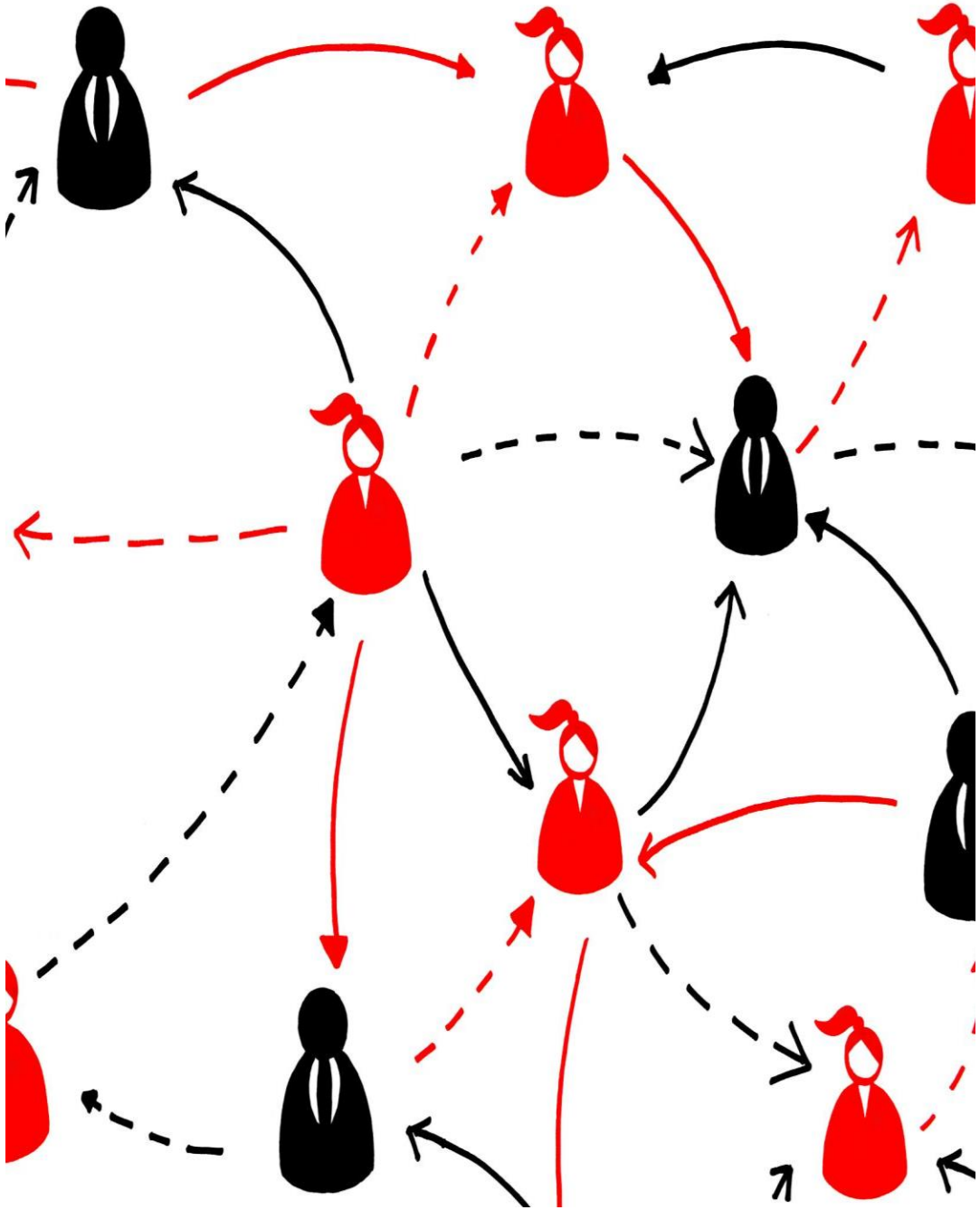
DEFINIÇÃO DE HERANÇA E HIERARQUIA DE CLASSES

Conceito de Herança

Herança permite que uma classe filha (***SubClasse***) adquira **atributos e métodos** de uma classe pai (***SuperClasse***).

Hierarquia de Classes

A hierarquia de classes organiza classes de forma estruturada para evitar duplicação de código.



BENEFÍCIOS DA HERANÇA: REUTILIZAÇÃO E EXTENSÃO DE FUNCIONALIDADES

Reutilização de Código

Herança permite reaproveitar código existente, reduzindo o esforço de desenvolvimento e mantendo consistência.

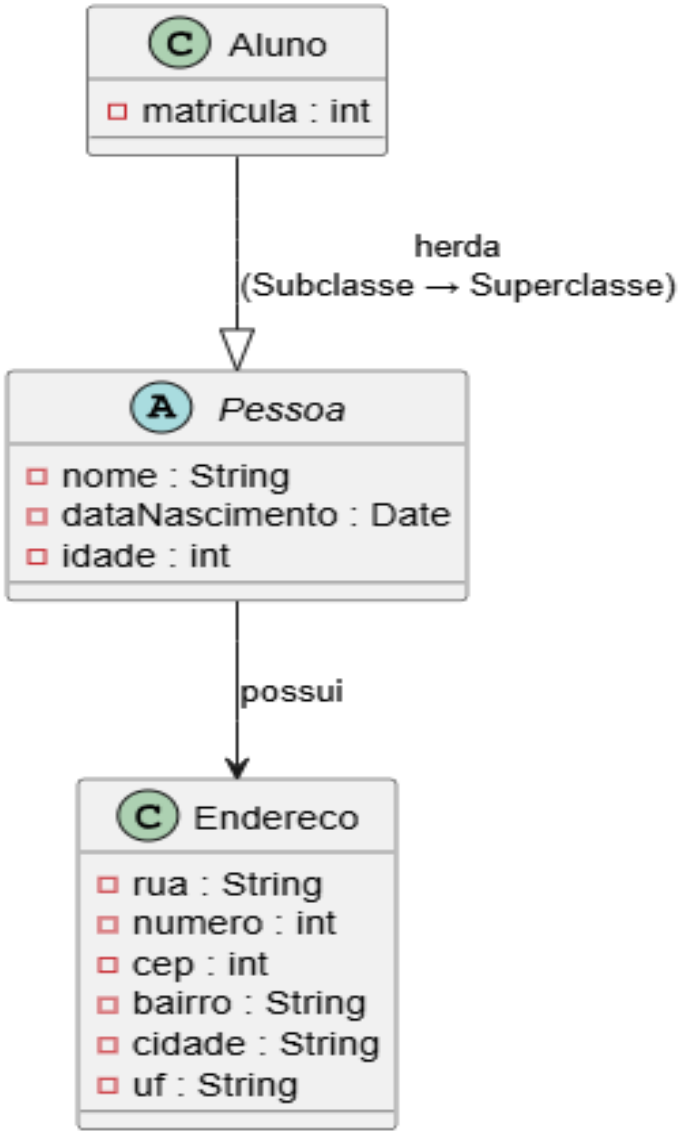
Especialização de Comportamentos

Subclasses podem especializar e modificar comportamentos para atender necessidades específicas.

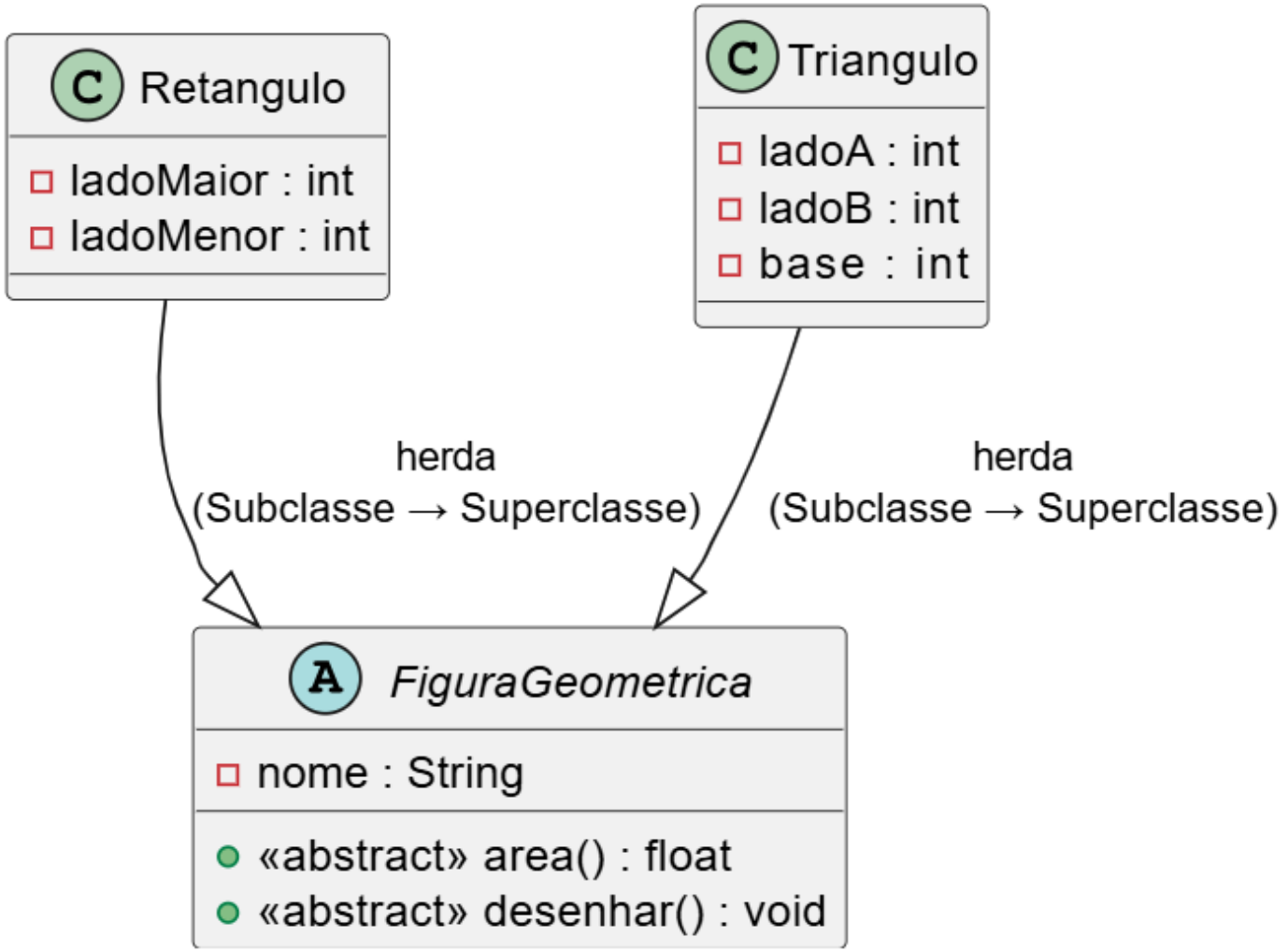
Programação Eficiente e Flexível

Herança torna o desenvolvimento mais eficiente e flexível, facilitando manutenção e expansão do software.

RELAÇÃO DE HERANÇA



HERANÇA (SIMPLES)



HERANÇA MÚLTIPLA

A Herança Múltipla é um tipo de herança onde uma subclasse pode herdar características e comportamentos de duas ou mais superclasses distintas. Em outras palavras, uma classe filha pode ter múltiplos pais.

CONCLUSÃO

Importância da Abstração

A abstração permite esconder detalhes complexos, facilitando o foco no que é importante no código.

Função da Herança

A herança promove reutilização do código ao permitir que classes derivem propriedades e métodos de outras.

Código Modular e Reutilizável

Esses conceitos ajudam a criar código organizado, modular e fácil de manter, aumentando a eficiência.

CONCLUSÃO

- **Abstração** e **Herança** são pilares essenciais da ***programação orientada a objetos*** que promovem código modular, reutilizável e organizado. Com exemplos práticos em Python, demonstramos como esses conceitos facilitam o desenvolvimento de sistemas eficientes e flexíveis. Além da herança tradicional, destacamos também a **herança múltipla**, que permite que uma classe herde características e comportamentos de duas ou mais superclasses distintas. Esse recurso, presente em linguagens como Python, amplia ainda mais as possibilidades de reutilização e combinação de funcionalidades, tornando o código mais versátil. No entanto, é importante utilizar a herança múltipla com atenção para evitar conflitos e garantir a clareza na estrutura do sistema. Assim, ao dominar abstração, herança simples e múltipla, o desenvolvedor está apto a criar soluções mais robustas, organizadas e adaptáveis às necessidades dos projetos.