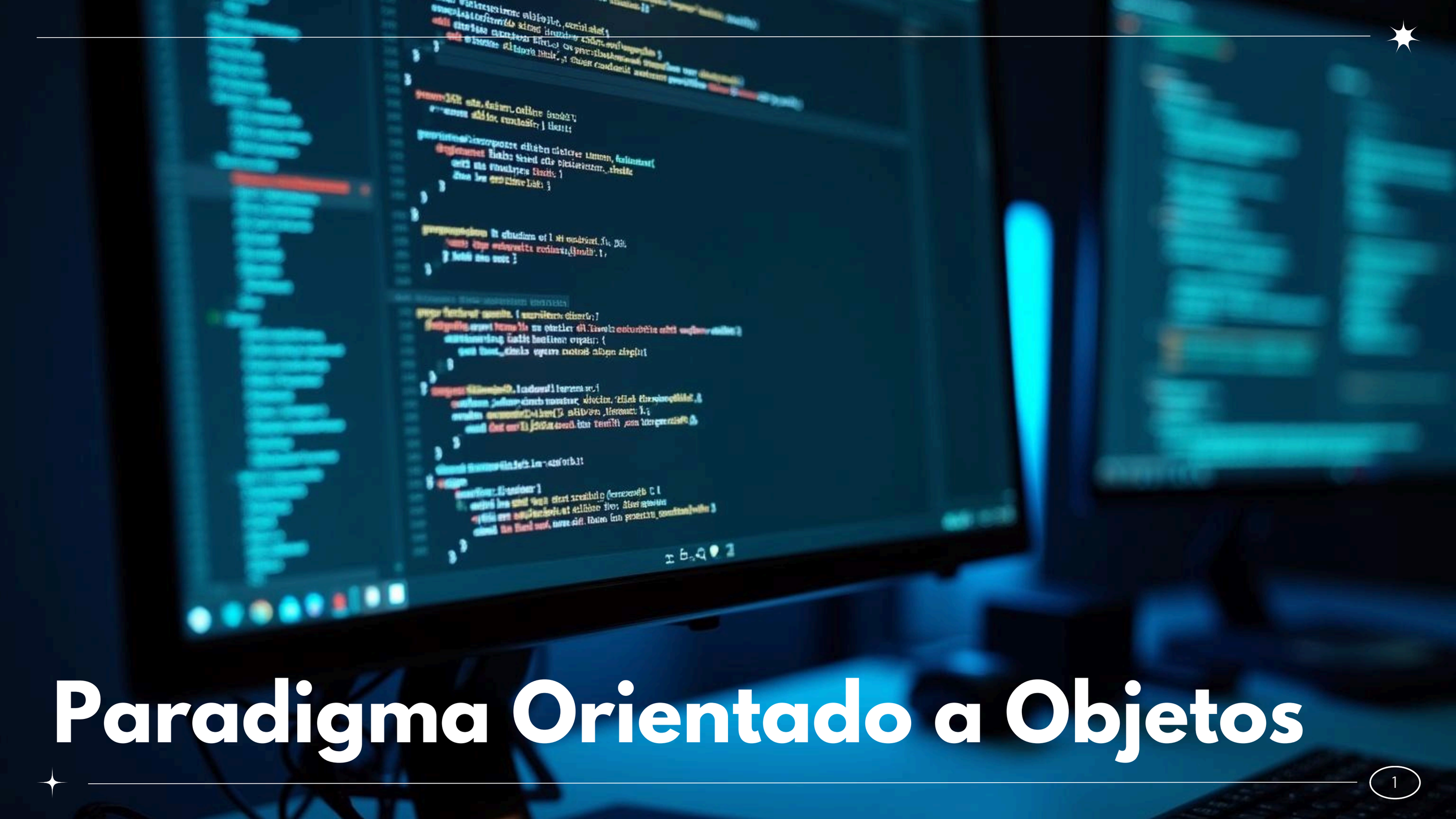


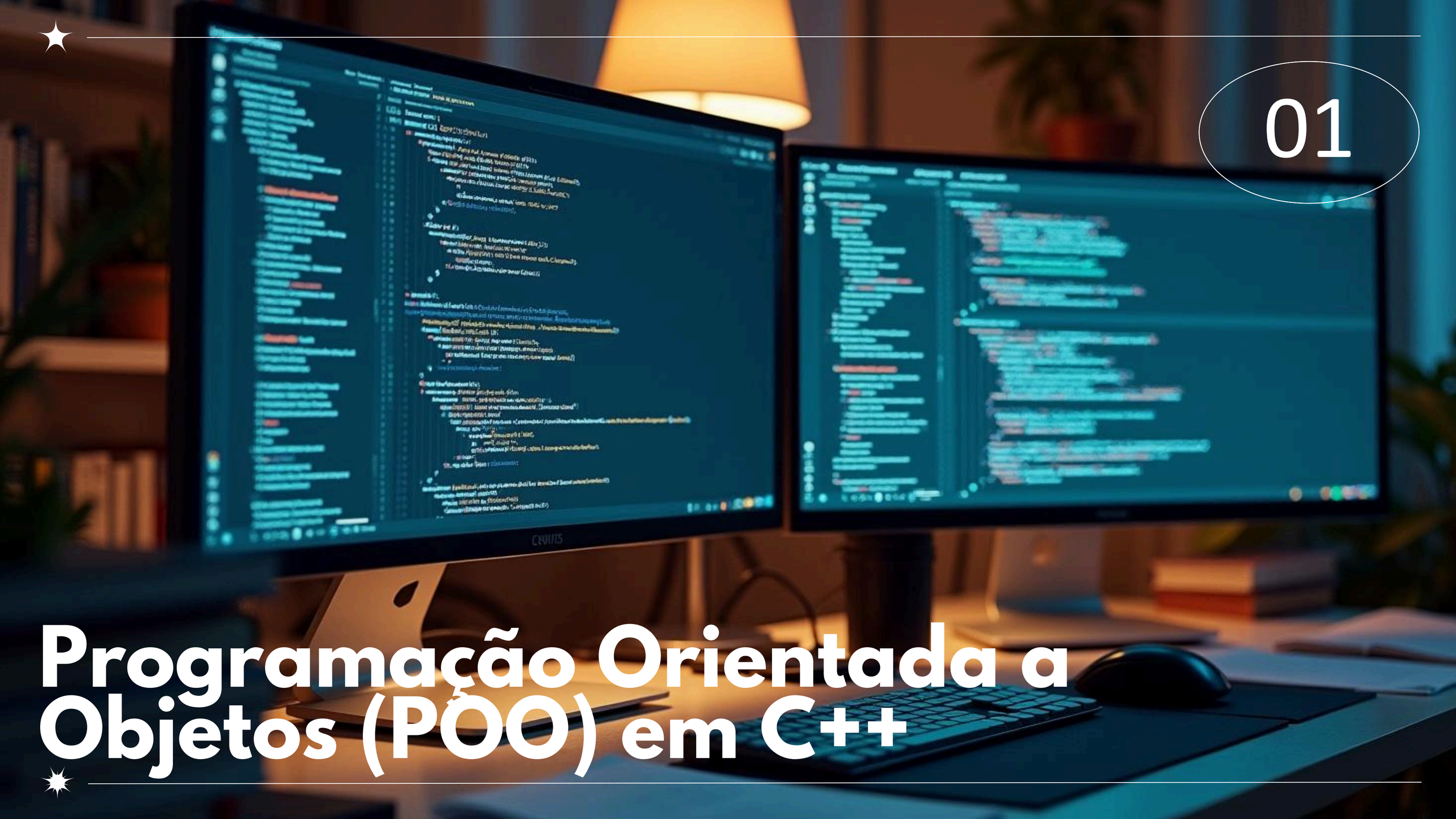
Paradigma Orientado a Objetos

Introdução

A *programação orientada a objetos (POO)* organiza código em torno de objetos reais, facilitando a modelagem de sistemas complexos. Podemos especificar as características e comportamentos desses objetos, além de estabelecer relações entre eles.

Exemplos de Objetos do mundo real que podemos abstrair para o código:

- Ser Humano, Animal, Eletrônicos e muitos outros.



01

Programação Orientada a Objetos (POO) em C++

Conceitos Básicos da POO

A POO cria **classes** como moldes para **objetos** que possuem atributos e métodos, aproximando a programação do mundo real.

Vantagens da POO em C++

A POO no **C++** permite melhor organização, reutilização de código e manutenção facilitada, promovendo sistemas altamente escaláveis e flexíveis.

Os 4 Pilares da POO

1. Herança
2. Polimorfismo
3. Encapsulamento
4. Composição

Extra: Abstração

Abstração

Abstraímos Entidades do mundo real para nosso código, dando a elas caraterísticas e comportamentos que representem a realidade.

·Exemplo:

- Identidade : Aluno.
- Propriedades: Nome, Idade, Matrícula.
- Métodos: Questionar, Estudar , Surtar.

Herança

Uma classe pode "herdar" características de outra. Isso evita repetir código, portanto atributos da classe “mãe” são passados para a classe filha.

·Exemplo

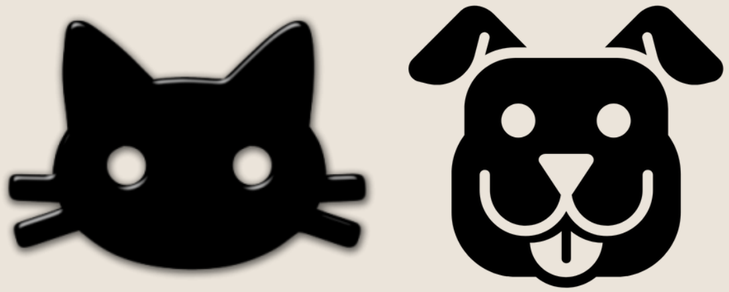
- Uma classe Professor pode herdar de Ser Humano. Com isso atributos (nome, idade) e métodos (Falar, Comer) da classe Ser Humano seriam passados para Professor.

Polimorfismo

É as características de dois ou mais objetos derivados de
uma mesma classe possuírem respostas diferentes para cada método.

Exemplo:

- Classes Animal possui um método para emitir som. Suas classes filhas, como cachorro e gato, teriam esse método diferente, pois emitem diferentes sons.



Encapsulamento

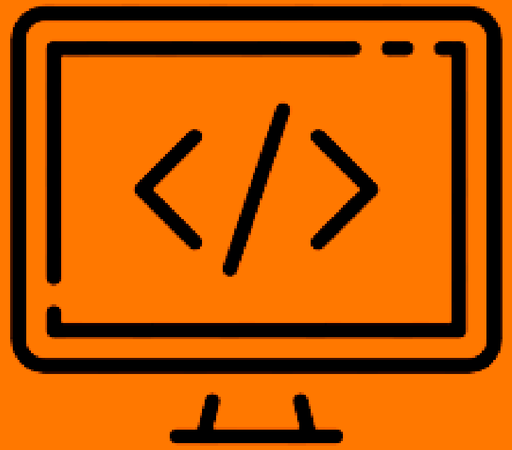
Ao definirmos os atributos de um objeto, devemos garantir que eles sejam "trancados", ou seja, o objeto pode ter suas propriedades consultadas, mas não modificadas por outros objetos.

02

C++: Características e Aplicações

Áreas e Exemplos de Uso do C++

Utilizado em jogos AAA, sistemas embarcados, softwares financeiros e bancos de dados, destacando-se pela eficiência e baixa latência.



KEEP CALM

AND

PROGRAME ORIENTADO A OBJETOS

Exercício 1

Exercício 1:

Cenário: Em um RPG, cada personagem tem um nome, um nível, dano de sua arma e uma quantidade de pontos de vida.

Tarefa:

1. Crie uma classe chamada Personagem.
2. Adicione os atributos nome (string), nivel (int), dano(int) e vida (int).
3. Crie um método chamado atacar() que recebe como parâmetro um objeto da classe Personagem exiba na tela uma mensagem de ataque e faça o personagem atacado perder N pontos de vida.
4. No main, crie 2 objeto da classe Personagem, atribua valores aos seus atributos e chame o método atacar(). Escolhendo quem irá atacar quem. Por fim mostre as informações de cada um após o combate.

Exercício 2

Exercício 3: Classes de Personagens

Cenário: Em Persona, cada personagem, seja o protagonista ou um aliado, tem características pessoais que devem ser protegidas e alteradas de forma controlada.

Tarefa:

1. Crie uma classe base chamada Pessoa.
2. Adicione os atributos nome (string) e idade(int) como privados.
3. Crie uma classe Protagonista e outra Personagem, ambas herdeiras de Pessoa. Protagonista deve ter um atributo de nível(int) e Personagem deve ter um atributo de rank(int) (obs: o rank dos Personagens/social link varia de 0 a 10). Ambos devem ser privados.
4. Na main, crie um objeto da classe Protagonista e atribua um nome e nível. Faça o mesmo para um objeto da classe Personagem. Imprima os valores e demonstre que os atributos estão sendo acessados através de um método e não diretamente.

Exercício 3

Exercício 4: Classes Professor e Aluno

Cenário: Em um sistema de gestão do Inatel, tanto professores quanto alunos são considerados pessoas da comunidade acadêmica, mas realizam ações diferentes.

Tarefa:

1. Crie uma classe base chamada Pessoa com um atributo nome (string) e um método virtual chamado apresentar(). O método deve exibir "Olá, meu nome é [nome] e eu sou uma pessoa.".
2. Crie as classes Professor e Aluno que herdem de Pessoa. E adicione os atributos disciplina em professor e curso/matrícula em aluno.
3. Sobrescreva o método apresentar() em ambas as classes.
    - No Professor, ele deve exibir "Olá, meu nome é [nome] e eu sou um professor de [disciplina].".
    - No Aluno, ele deve exibir "Olá, meu nome é [nome] e eu sou um aluno de [curso]".
4. Na main, crie ponteiros da classe base Pessoa e aponte-os para objetos do tipo Professor e Aluno. Chame o método apresentar() através desses ponteiros para demonstrar o polimorfismo.

Exercício 4

Exercício 2: Classes de Seres Vivos

Cenário: Em um mundo de fantasia, existem diferentes raças como Elfos, Humanos e Fada. Todos eles são considerados "Seres Vivos", mas cada raça tem sua própria forma de se apresentar. O seu desafio é criar um sistema que possa armazenar todos eles juntos e fazer com que cada um se apresente de maneira única.

Tarefa:

1. Crie uma classe base chamada SerVivo. Nela, adicione um atributo nome (string) e um método virtual apresentar(). Dentro do método, imprima uma mensagem genérica.
2. Crie três classes filhas que herdem publicamente de SerVivo: Humano, Elfo e Fada
3. Em cada uma das classes filhas, sobrescreva o método apresentar() para que ele exiba uma mensagem específica de sua raça.
4. Na main, crie uma única lista para armazenar os seres vivos.
5. Crie uma instância de cada raça (Humano, Elfo, fada) e adicione-as à sua lista.
6. Percorra a lista. Para cada elemento, chame o método apresentar()

Fim. Obrigado!

Repositório com os códigos vistos durante as aulas de cada Relatório:
https://github.com/zSh3rl0cK/S01-Monitoria

Até a próxima aula!