alura

01000101 01110011 01100011 01101111 01101100 01100001
00100000 01100100 01100101 00100000 01000100 01100001
01100100 01101111 01110011 00001010

CLASSES, OBJETOS E ABSTRAÇÃO

Praticando C# e OO

Instrutora: Iasmin Araújo

ESCOLA_ **PROGRAMAÇÃO**

Nome	Descrição	Preço	Estoque
Teclado	Modelo compacto e silencioso, perfeito para produtividade diária.	R$80,00	15
Cadeira gamer	Conforto ergonômico com design moderno para longas sessões de jogo.	R$600,00	5
Notebook	Potência e portabilidade em um notebook ideal para trabalho e entretenimento.	R$3500,00	8
Teclado	Teclado mecânico com teclas suaves e iluminação RGB para máxima performance.	R$150,00	4
Mouse	Precisão e velocidade com sensor avançado e design ergonômico.	R$100,00	10

Como representar todos esses produtos no código?

Nome	Descrição	Preço	Estoque
Teclado	Modelo compacto e silencioso, perfeito para produtividade diária.	R$80,00	15
Cadeira gamer	Conforto ergonômico com design moderno para longas sessões de jogo.	R$600,00	5
Notebook	Potência e portabilidade em um notebook ideal para trabalho e entretenimento.	R$3500,00	8
Teclado	Teclado mecânico com teclas suaves e iluminação RGB para máxima performance.	R$150,00	4
Mouse	Precisão e velocidade com sensor avançado e design ergonômico.	R$100,00	10

Todos os produtos têm informações e comportamentos em comum.

Para representá-los, você pode criar um tipo personalizado `Produto`.

Fazemos isso utilizando **classes**.

```
class Produto
{
    public string nome;
    public string descricao;
    public decimal preco;
    public int estoque;
}
```

CRIANDO UMA CLASSE

- ✔ Para criar uma classe no Visual Studio, adicione um novo **arquivo de código** com o nome do tipo.
- ✔ Nesse arquivo, declare uma classe usando a palavra chave **class** seguida do nome do tipo criado.
- ✔ Dentro das chaves coloque todas as características do tipo. Elas devem ser declaradas com um **public** antes para o código compilar.

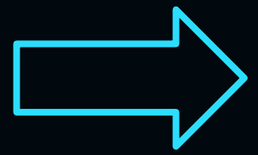
CLASSES E OBJETOS

- Uma classe é um **modelo**, algo que **descreve** o tipo que estamos criando.
- As **variações de cada característica** são representadas por objetos ou instâncias.

alura

EXEMPLO DE CLASSE E OBJETOS

Produto
Nome
Descrição
Preço
Estoque



Item 1	Item 2	Item 3
“Teclado”	“Mouse”	“Teclado”
“Ergonômico”	“Rápido”	“Prático”
80.00	150.00	100.00
15	10	4

CLASSE

Definições das características de um produto

OBJETOS

Produtos com valores variados de características

```
Produto item1 = new Produto();
item1.nome = "Teclado";
item1.descricao = "Modelo compacto e
silencioso,perfeito para produtividade
diária.";
item1.preco = 80.00m;
item1.estoque = 15;
```

CRIANDO UM OBJETO

- ✔ Para criar um objeto, use a estrutura **NomeClasse nomeVariavel = new NomeClasse().**
- ✔ Para atribuir valores a cada característica da classe, pode fazer **nomeVariavel.caracteristica**.

```
string nome1 = "Teclado";
string descricao1 = "Modelo compacto e
silencioso, perfeito para produtividade
diária.";
decimal preco1 = 80.00m;
int estoque1 = 15;

Produto item1 = new Produto();
item1.nome = "Teclado";
item1.descricao = "Modelo compacto e silencioso,
perfeito para produtividade diária.";
item1.preco = 80.00m;
item1.estoque = 15;
```

VANTAGENS DE USAR CLASSES

Repare a diferença entre o primeiro e o segundo bloco de código.

Usando classes, conseguimos ter certeza de que o nome “Teclado” está associado ao preço 80.00m. Isso não acontecia antes!

Com classes, conseguimos **agrupar dados**.

COMPORTAMENTOS E FUNÇÕES

Podemos representar os comportamentos de uma classe como “funções”. Os nomes dessas funções, por convenção, irá ter o formato PascalCase.

A função pode ou não retornar algum valor. Caso retorne, usamos a palavra `return` .

Caso contrário, o tipo de retorno é declarado como `void`.

```
public bool EstaDisponivel()
{
    return estoque > 0;
}

public void AlterarPrecoComDesconto
(decimal desconto)
{
    preco = preco *(1 - desconto/100);
}
```

Os parâmetros não são obrigatórios.

```
Produto item1 = new Produto();
item1.preco = 100.00m;
item1.estoque = 10;

item1.AlterarPrecoComDesconto(0.10m);
Console.WriteLine($"Preço atual:
{item1.preco}");
var disponivel = item1.EstaDisponivel();
Console.WriteLine($"Disponível:
{disponivel}");
```

SAÍDA

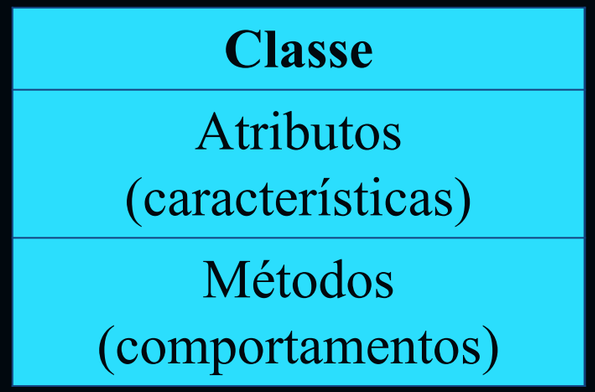
```
Preço atual: 90.000
Disponível: True
```

CHAMANDO AS FUNÇÕES

- ✔ Assim como as características da classe, podemos chamar os comportamentos usando o "." também:
  nomeVariável.Comportamento()

Formalmente, as funções que representam **comportamentos** de classes em comum são chamadas de **métodos**.

E as **características** que descrevem a classe/o tipo são chamadas de **atributos** ou **campos**.



PILARES DA POO

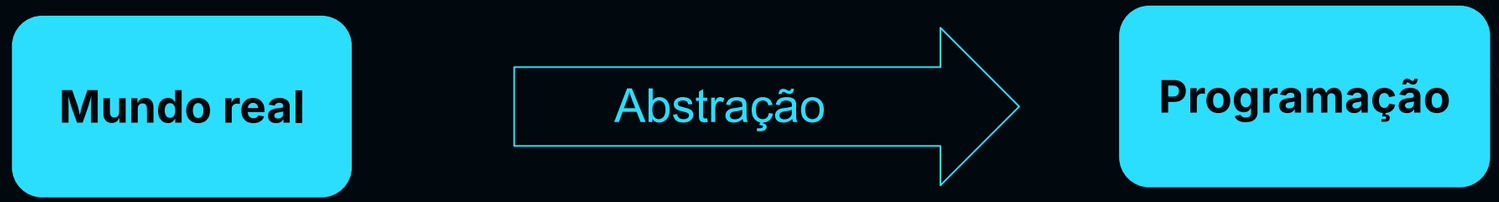
Quando trabalhamos com classes e objetos, estamos usando **programação orientada a objetos, a POO.** Ela é baseada em 4 pilares:

- ✔ Abstração
- ✔ Encapsulamento
- ✔ Herança
- ✔ Polimorfismo

ABSTRAÇÃO

O grande desafio de trabalhar com orientação a objetos é saber **quais atributos e quais métodos escolher** para representar as classes corretamente, de acordo com o nosso **contexto**.

Abstração é olhar pro mundo real e saber como representá-lo por meio da programação.



alura

Compartilhe um resumo de seus novos conhecimentos em suas redes sociais.

#aprendizadoalura