Willian Menezes

Treinamento de C# e .NET

Básico/Intermediário

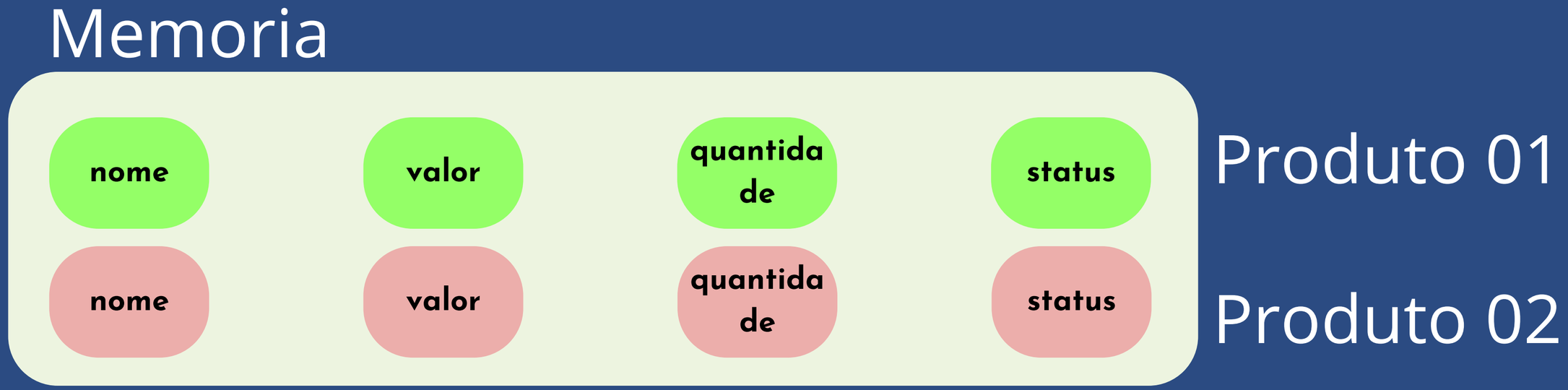
Resolvendo sem POO

Cadastro de produtos

Exemplo no github

Resolvendo sem POO

Várias variáveis para representar um produto



Classe

POO foi feito para aproximar nossos programas do mundo real

Uma classe eh um modelo, um molde, que representa alguma entidade que queremos (Produto, ContaCorrente, Carrinho, Pagamento e etc)

Uma classe contem:

- Atributos e métodos
- Construtores
- Sobrecarga
- Encapsulamento
- Heranca
- Polimorfismo

Abstracao

Abstração nada mais eh do que transformar um problema que nos temos em uma solução de software.

Podemos abstrair objetos, regras de negócio, processos complexos de instituições.

Tudo isso utilizando trazendo problemas do mundo real para dentro de nossos programas de computador.

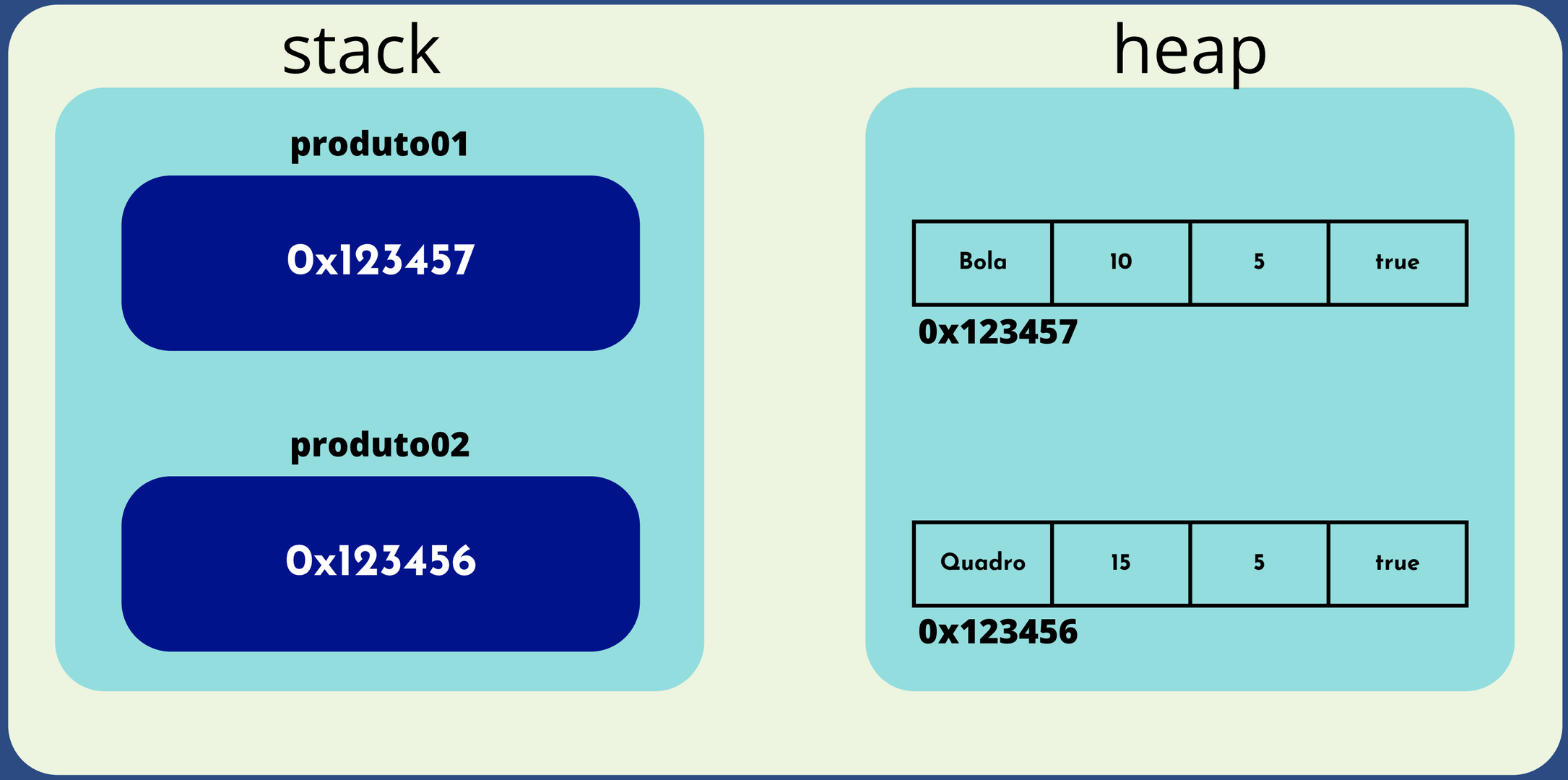
Resolvendo com POO

Cadastro de produtos

Exemplo no github

Classe

Memoria



Classe

Exercicios

Crie uma classe que modele uma pessoa com os atributos: nome, idade, peso e altura.
Mostrar os dados da pessoa no terminal.

Implemente a classe Funcionário. Um empregado tem um nome (um string) e um salário(um double). Imprima os dados do funcionário, aumente seu salário em 10% e imprima novamente.

Resolvendo com POO

Cadastro de produtos
Utilizando métodos de classe

Exemplo no github

Vantagens:
Reaproveitamento, sem repetição.

Atribuir a responsabilidade para o dono da informação.

Apenas um ponto de falha e manutenção.

Resolvendo com POO

Criar uma classe produto, com as propriedades nome, valor, quantidade em estoque e status.

Exibir informações do produto e alterar quantidade em estoque.

Construtores

Executada quando instanciamos um objeto

Utilizado geralmente para iniciar valores da nossa classe (Por enquanto).

Combinar construtores com encapsulamento deixa as nossas classes mais segurar e coesas. (Veremos mais exemplos a frente)

Construtores

Usando o nosso exemplo do produto, quando utilizamos um new Produto(), estamos iniciando um produto com todas as usas propriedades com valores padrao.

Pensando no ponto de vista de negocio, faz sentido ter um produto assim? E se o programador esquecer de colocar os dados no produto? Temos varias formas de resolver isso, utilizaremos o construtor desta vez.

Construtores

Posso declarar apenas um construtor? Entendendo sobrecarga.

Posso colocar quantos parâmetros quiser?

Tem a possibilidade de ter valores padrão para o construtor?

Posso reaproveitar construtores dentro da minha classe?

Encapsulamento

Precisamos sempre mostrar tudo que nossa classe faz para quem instancia a classe?

Encapsulamento ajuda a manter o estado da nossa classe consistente.

Sempre devemos garantir que nossas classes tenham o estado correto, e a própria classe tem que ter essa responsabilidade.

Entendendo get; set; em nossas propriedades

Modificadores de acesso

Em C# possuímos vários modificadores de acesso, alguns deles são:

- Private
  - Mantém a informação visível somente para a classe em que ela foi declarava, somente essa classe pode visualizar e alterar.
- Public
  - Toda a informação fica disponível para quem utilizar a classe, podendo ler e alterar a informação.
- Protected
  - Toda a informação fica disponível para a classe e para quem herdar desta mesma classe, podendo ler e alterar a informação.
- Internal
  - Toda a informação fica disponível para quem utilizar a classe dentro do mesmo assembly/projeto, podendo ler e alterar a informação.

Classe

Exercicios

Classe Conta Corrente: Crie uma classe para implementar uma conta-corrente. A classe deve possuir os seguintes atributos: número da conta, nome do correntista e saldo. Os métodos são os seguintes: alterarNome, depósito e saque;

Classe Bomba de Combustível: Faça um programa completo utilizando classes e métodos que:
Possua uma classe chamada **bombaCombustível**, com no mínimo esses atributos: **tipoCombustivel**, **valorLitro**, **quantidadeCombustivel**.
Possua no mínimo esses métodos: **abastecerPorValor**( ) – método onde é informado o valor a ser abastecido e mostra a quantidade de litros que foi colocada no veículo, **abastecerPorLitro**( ) – método onde é informado a quantidade em litros de combustível e mostra o valor a ser pago pelo cliente, **alterarValor**( ) – altera o valor do litro do combustível, **alterarCombustivel**( ) – altera o tipo do combustível, **alterarQuantidadeCombustivel**( ) – altera a quantidade de combustível restante na bomba.

OBS: Sempre que acontecer um abastecimento é necessário atualizar a quantidade de combustível total na bomba.

Herança

Tipo de associação que permite que uma classe herde dados e comportamentos de outra classe.

Com herança podemos fazer o uso de polimorfismo (Veremos depois) e reutilizar classes.

Herança

```
6 references
public class Pessoa
{
    1 reference
    public string Nome { get; set; }
    1 reference
    public int Idade { get; set; }
    1 reference
    public double Peso { get; set; }
    1 reference
    public int AlturaEmCentimetros { get; set; }

    0 references
    public Pessoa() { }

    2 references
    public Pessoa(string nome)
    {
        Nome = nome;
    }

    0 references
    public void Envelhecer()
    {
        Idade += 1;
    }

    0 references
    public void Engordar()
    {
        Peso += 2;
    }

    0 references
    public void Crescer(int altura = 5)
    {
        AlturaEmCentimetros += altura;
    }
}
```

```
1 reference
public class PessoaFisica : Pessoa
{
    2 references
    public string Cpf { get; set; }

    0 references
    public PessoaFisica(string nome, string cpf) : base(nome)
    {
        Cpf = cpf;
    }

    0 references
    public void AlterarCpf(string cpf)
    {
        Cpf = cpf;
    }
}
```

```
1 reference
public class PessoaJuridica : Pessoa
{
    2 references
    public string Cnpj { get; set; }

    0 references
    public PessoaJuridica(string nome, string cnpj) : base(nome)
    {
        Cnpj = cnpj;
    }

    0 references
    public void AlterarCnpj(string cnpj)
    {
        Cnpj = cnpj;
    }
}
```

- Generalização/Especialização
- Pessoa física eh uma pessoa
- Classe base (Pessoa), subclasse(PF e PJ)
- Herança/extensão - PF estende a classe pessoa.
- Herança eh uma associação entre classe, ou seja, a subclasse tem todas as propriedades e métodos da classe base.

Herança

Implementando nosso exemplo

Herança

Sobreposicao ou sobrescrita

Implementação de um método da classe base na subclasse

Para permitir que um método seja sobreposto, precisamos utilizar a palavra virtual

Quando formos sobrescrever um método precisamos utilizar a palavra override.

Sobreposicao/Sobrescrita

Implementando nosso exemplo

Polimorfismo

Recurso que permite que variaveis de tipos mais genericos

Mesmo a classe sendo genérica, o objeto instanciado eh de um tipo específico, assim temos uma execução diferente em cada tipo.

O compilador nao sabe para qual tipo a chamada ta sendo feita.

Classes e métodos abstratos

Classes abstratas não podem ser instanciadas.

Métodos abstratos não tem implementação, cabendo a quem herda implementá-lo.

Quando uma classe tem pelo menos um método abstrato, essa classe também tem que ser abstrata.

Exercicio

Criar juntos um carrinho de compras.

Criar um jogo da velha.

Duvidas???