UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
DEPARTAMENTO DE PROCESSAMENTO DE ENERGIA ELÉTRICA

Definição e Sintaxe de Classes

Curso de Engenharia de Controle e Automação

DPEE1090 - Programação orientada a objetos para automação

Prof. Renan Duarte

1º semestre de 2024

Sumário

Definição e Sintaxe de Classes

- Conceitos fundamentais de classes
- Estrutura e sintaxe de declaração de classes
- Atributos, métodos e construtores

Introdução

https://bit.ly/3vrwpse

Exercício 1 – Resolvendo um problema sem orientação à objetos

Fazer um programa para ler as medidas dos lados de dois triângulos X e Y (suponha medidas válidas). Em seguida, mostrar o valor das áreas dos dois triângulos e dizer qual dos dois triângulos possui a maior área.

Fórmula de Herão:

$$Area = \sqrt{p(p-a)(p-b)(p-c)}$$

$$p = \frac{a+b+c}{2}$$

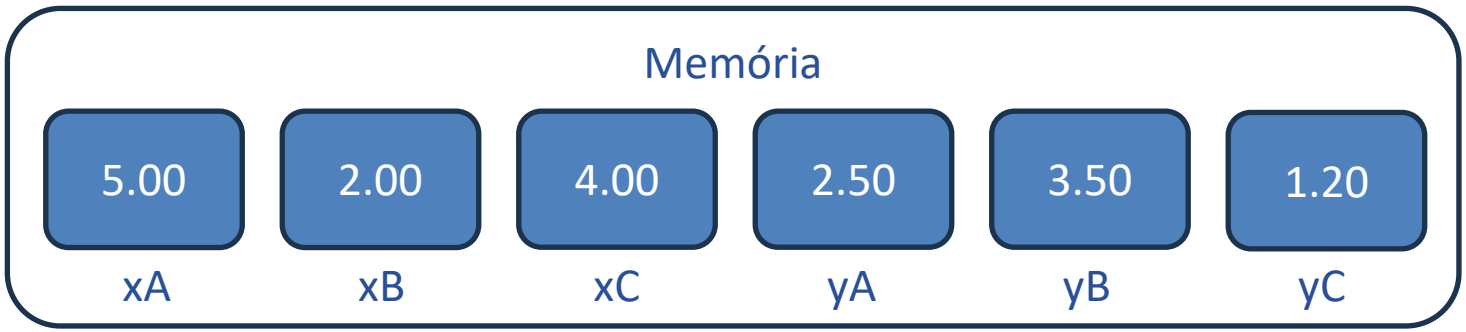
Herão de Alexandria
10 d.C. - 80 d.C.

Introdução

Quais os problemas da solução desenvolvida?

- Triângulo é uma entidade com três atributos: a, b, c
- Estamos usando três variáveis distintas para representar cada triângulo:

```
// Variaveis para os lados dos triangulos
double xA, xB, xC, yA, yB, yC;
```



Classe

Definição

É um tipo estruturado que pode conter (membros):

- Atributos (dados/campos)
- Métodos (funções/operações)

A classe também pode prover muitos outros recursos, tais como:

- Construtores
- Sobrecarga
- Encapsulamento
- Herança
- Polimorfismo

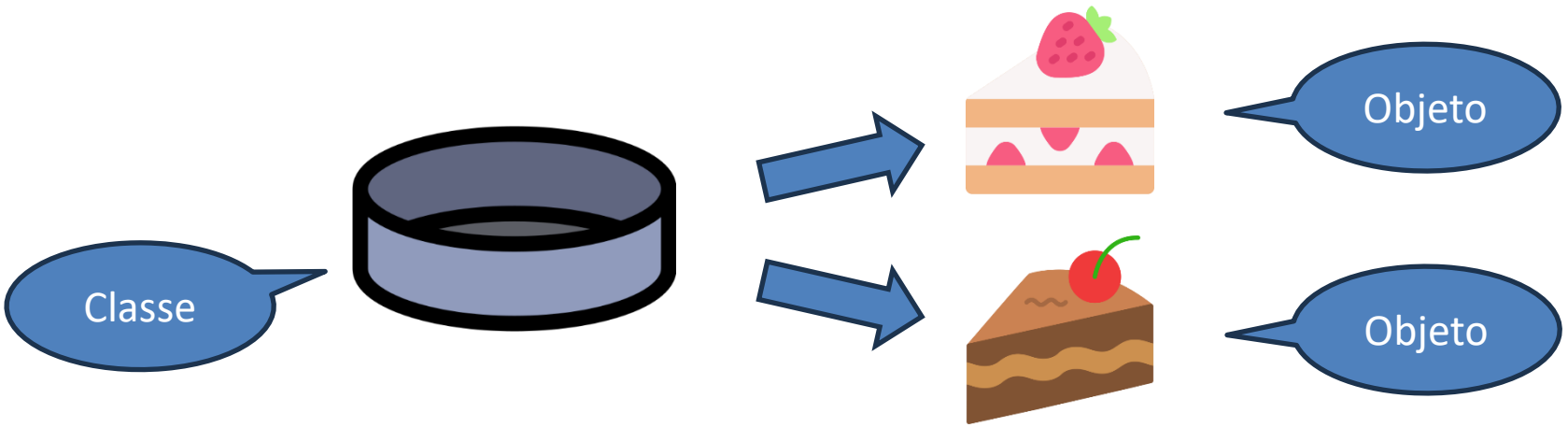
É o "elemento base" da orientação à objetos

Classe

Definição

Classes podem representar inúmeras “coisas”:

- Entidades: Produto, Cliente, Triangulo
- Serviços: ClienteService, EmailService, StorageService
- Controladores: ProdutoController, ClienteController
- Utilitários: Calculadora, Compactador
- Outros (views, repositórios, gerenciadores, etc.)

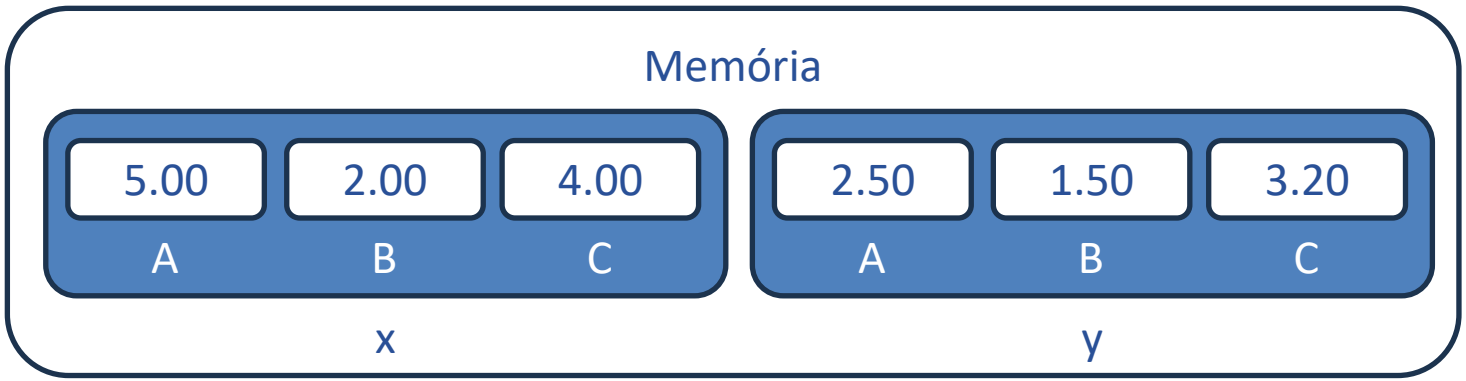


Classe

https://bit.ly/4aCfaTL

Exercício 2 – Melhorando a solução com atributos de uma classe

```
namespace Course
{
    internal class Triangulo
    {
        public double A;
        public double B;
        public double C;
    }
}
```



Classe

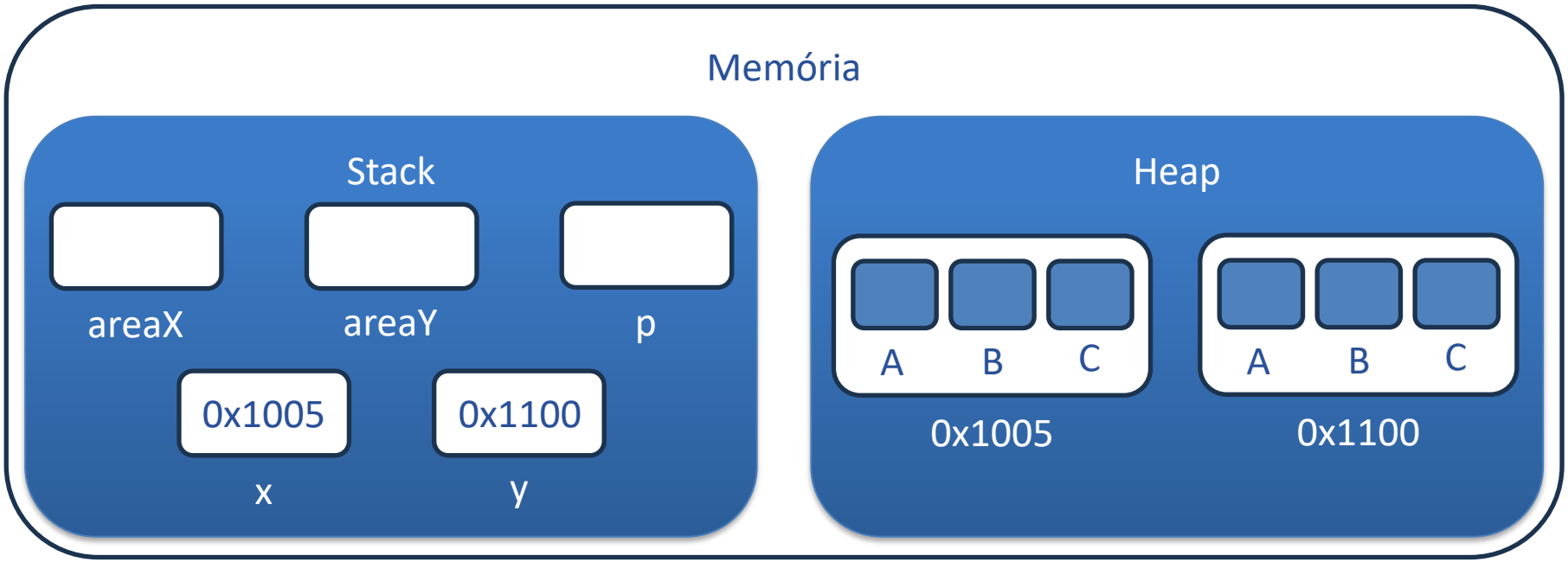
Instanciação - Alocação dinâmica de memória

```
double areaX, areaY, p;
Triangulo x, y;
```

} Declaração → Alocado espaço na stack

```
x = new Triangulo();
y = new Triangulo();
```

} Instanciação → Alocado espaço na heap



Classe

Vantagens da nova solução

Agora, atributos ligados a entidade triângulo ficam agrupados dentro de uma única variável → Objeto composto que contém as variáveis A, B e C.

Dessa forma, nosso código fica mais coeso, facilitando a leitura e interpretação, especialmente à medida que os problemas se tornam mais complexos.

Além disso, aumentamos a modularidade do código, visto que o uso de uma classe permite reaproveitar unidades de código sem a necessidade de grandes modificações na solução como um todo.

Classe

Exercício 3

Fazer um programa para ler os dados de duas pessoas, depois mostrar o nome da pessoa mais velha.

Exemplo:

```
Dados da primeira pessoa:
Nome: Maria
Idade: 17

Dados da segunda pessoa:
Nome: Joao
Idade: 16

Pessoa mais velha: Maria
```

Classe

https://bit.ly/4aft3rr

Exercício 4 – Melhorando o exercício 2 com métodos de uma classe

Na solução anterior para o problema dos triângulos, melhoramos o código com o uso de atributos de uma classe “Triangulo”. Contudo, ainda temos repetição de código no programa:

```
// Calculo da area de X
double p = (x.A + x.B + x.C)/2;
double areaX = Math.Sqrt(p * (p - x.A) * (p - x.B) * (p - x.C));

// Calculo da area de Y
p = (y.A + y.B + y.C) / 2;
double areaY = Math.Sqrt(p * (p - y.A) * (p - y.B) * (p - y.C));
```

Vamos utilizar métodos de uma classe para obter os benefícios de reaproveitamento e delegação

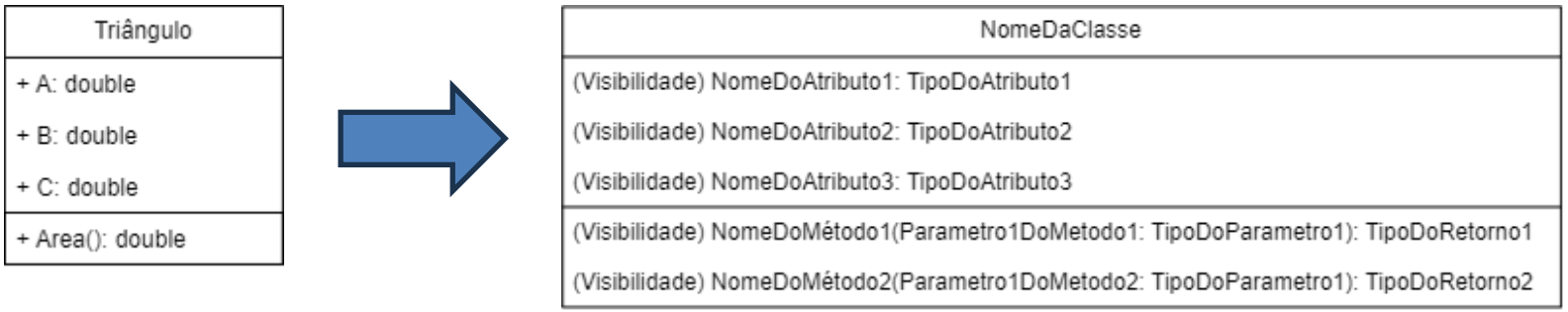
Classe

Projeto de uma classe

UML: Linguagem de modelagem unificada (do inglês *Unified Modeling Language*)

- É uma linguagem-padrão para a elaboração da estrutura de projetos de software

Representação da nossa classe triângulo:



https://www.youtube.com/watch?v=rDidOn6KN9k
https://www.youtube.com/watch?v=WnMQ8HlmeXc

Classe

Construtor

É uma operação especial da classe, que executa no momento da instanciação do objeto → `Triangulo x = new Triangulo();`

Usos comuns:

- Iniciar valores dos atributos
- Permitir ou obrigar que o objeto receba dados / dependências no momento de sua instanciação (injeção de dependência)

Se um construtor customizado não for especificado, a classe disponibiliza o construtor padrão:

- Produto p = new Produto();

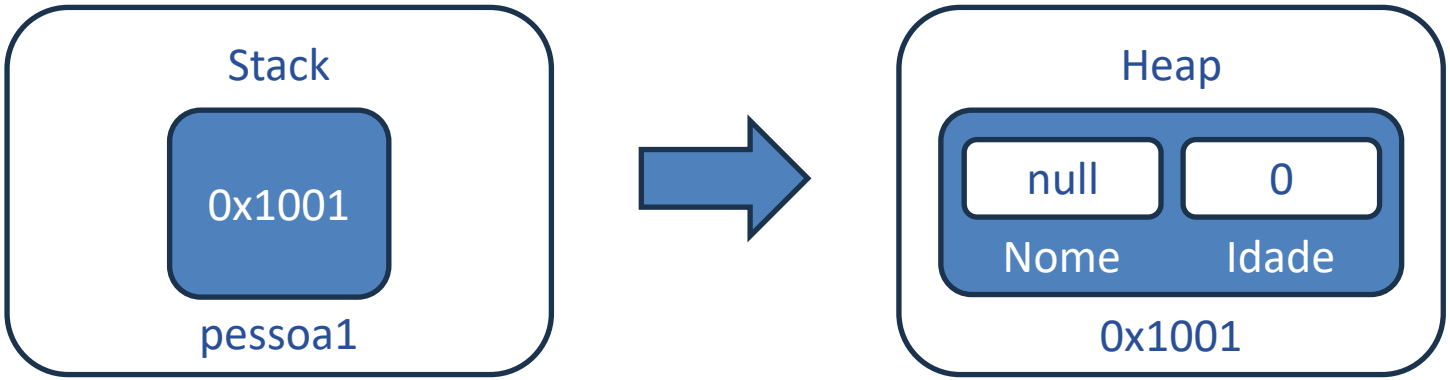
É possível especificar mais de um construtor na mesma classe

Classe

Construtor

Usando o Exercício 3 como exemplo:

```
Pessoa pessoa1 = new Pessoa();
```



Faz sentido uma pessoa que não tenha nome ou “idade = 0” no programa?

Como evitar essa situação?

Classe

Exercício 5 – Criando um construtor para o exercício 3

```
// Construtor – Na classe
public Pessoa(string nome, int idade)
{
    Nome = nome;
    Idade = idade;
}

// Instanciação – No programa principal
Pessoa pessoa1 = new Pessoa(); → Inválido: Parâmetros nulos

Pessoa pessoa1 = new Pessoa("Maria", 25); → Ok
```

Programador é obrigado a passar os parâmetros `nome` e `idade` na instanciação de `pessoa1`

Exercício 6 – Construtor com sobrecarga

Podemos declarar mais de um construtor para a mesma classe com diferentes parâmetos em cada um → **Sobrecarga**

Dessa forma podemos fazer com que um ou mais atributos sejam opcionais na instanciação da classe

Exemplo: Modificar o Exercício 5 de forma que um novo atributo `altura` seja opcional na instanciação da classe `Pessoa`

```
// Construtor 2
public Pessoa(string nome, int idade, double altura)
{
    Nome = nome;
    Idade = idade;
    Altura = altura;
}
```

Classe

A palavra *this*

No exemplo abaixo, como diferenciar os artibutos da classe dos parâmetros do construtor?

```
// Atributos da pessoa
public string nome;
public int idade;

// Construtor
public Pessoa(string nome, int idade)
{
    nome = nome;
    idade = idade;
}
```

Classe

A palavra *this*

Nestes casos (que podem ser evitados se a convenção de nomeclatura do C# for seguida), deve-se usar a palavra **this** para se referir aos atributos internos da classe:

```
// Atributos da pessoa
public string nome;
public int idade;

// Construtor
public Pessoa(string nome, int idade)
{
    this.nome = nome;
    this.idade = idade;
}
```

Cores iguais representam variáveis iguais

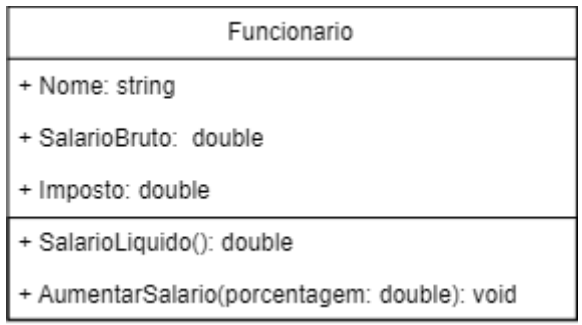
Classe

https://bit.ly/3vA2QEK

Exercício 7

Fazer um programa para ler os dados de um funcionário (nome, salário bruto e imposto). Em seguida, mostrar os dados do funcionário (nome e salário líquido). Em seguida, aumentar o salário do funcionário com base em uma porcentagem dada (somente o salário bruto é afetado pela porcentagem) e mostrar novamente os dados do funcionário.

Use a classe projetada abaixo:



Exercício 7 - Continuação

Exemplo:

```
Nome: Renan Duarte
Salário bruto: 6000.00
Imposto: 1000.00

Funcionário: Renan Duarte, $5000.00

Digite a porcentagem para aumentar o salário: 10.0

Dados atualizados: Renan Duarte, $5600.00
```

Revisão

Próxima aula

Objetos

- Compreensão dos conceitos de objetos e instância de classes
- Declaração e utilização de objetos
- Exemplos práticos

Dúvidas?

renan.duarte@gedre.ufsm.br

GEDRE – Prédio 10 – CTLAB