

# Programação Orientada a Objetos

Turma 3

Carlos Henrique Bughi, MSc



Onde estamos?  
(e para onde vamos)

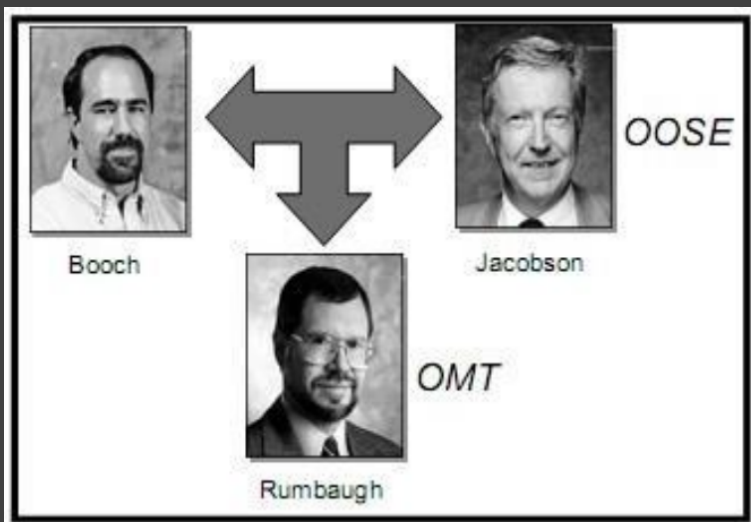
- Pilares da POO
- UML
- Mais UML



# Aula 4

## UML - Unified Modeling Language

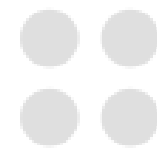
# UML



- A UML (Unified Modeling Language) é uma notação para descrição de sistemas orientados a objetos
- Baseia-se na experiência dos principais autores dos 3 principais métodos OO, padronizada pela OMG (Object Management Group) em 1997




# O que é UML?



- Linguagem visual para especificação de sistemas orientados a objetos
  - Fornece representação gráfica para os elementos essenciais do paradigma de objetos:
    - Classes,
    - Atributos,
    - Objetos,
    - Troca de mensagens

# O que NÃO é UML?

- UML NÃO DEFINE UMA ETAPA DO DESENVOLVIMENTO
  - Análise, projeto, implementação, testes, ...
- UML NÃO DEFINE UM PROCESSO
  - Ciclo de vida em cascata, incremental, etc.
- UML NÃO DEFINE UMA LINGUAGEM DE PROGRAMAÇÃO
  - Java, C++, C#, etc...



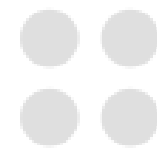
# O que é a UML

- Privilegia a descrição de um sistema segundo três perspectivas:

- Dados (estrutural)
  - Diagrama de classes
- Operações (funcional)
  - Diagrama de casos de uso
- Eventos (temporal)
  - Diagrama de sequência, diagrama de atividades, diagrama de estados



# Diagrama de Classes

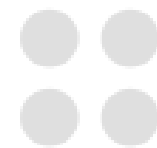


- Um diagrama de classes captura a estrutura lógica de um sistema, é um modelo estático, descrevendo o que existe e quais atributos e comportamentos possui;
- Representação da estrutura das classes e suas relações, servindo de modelo para os objetos;
- Diagrama de Classes é a base para outros diagramas, como sequência e estados.

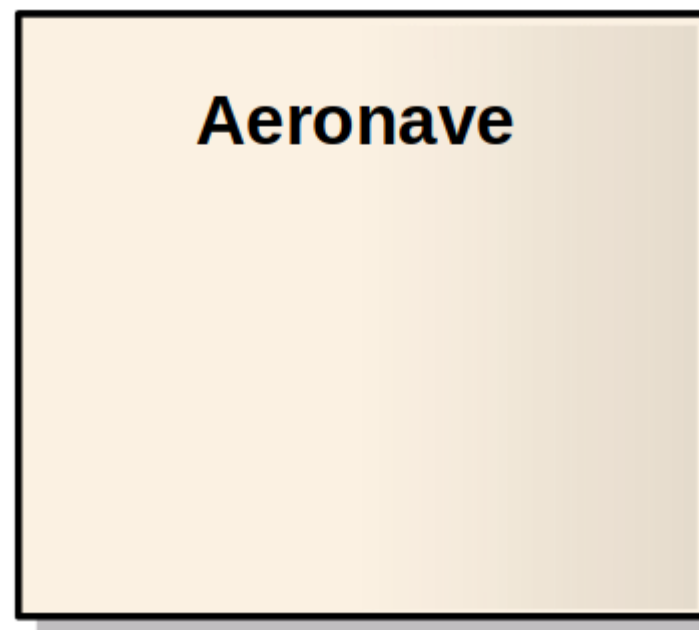




# Classes

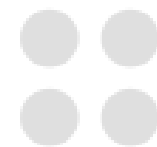


- Cada classe deve possuir um nome único, preciso e conciso que representa o tipo de objeto representado por ela;
- Nome da classe no primeiro quadro.





# Atributos



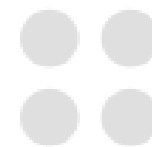
- Responsáveis por armazenar as informações que um \*objeto possui
- Os valores dos atributos variam para cada objeto da classe
- Localizados no segundo quadro da classe

Aeronave
- capacidade: int - nome: String - velocidadeMaxima: double

[visibilidade] [/] **nome** [: tipo] [multiplicidade] [= valor padrão]



# Métodos



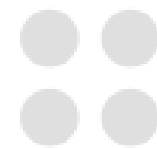
- Responsáveis pelas ações/comportamentos que o \*objeto possui
- Os métodos podem realizar ações e/ou alterar o estado do objeto
- Localizados no terceiro quadro da classe

[visibilidade] **nome** ([lista de parâmetros]):tipo do retorno

Aeronave	
-	capacidade: int
-	nome: String
-	velocidadeMaxima: double
-	decolar() : void
-	pousar() : void
-	abastecer() : void



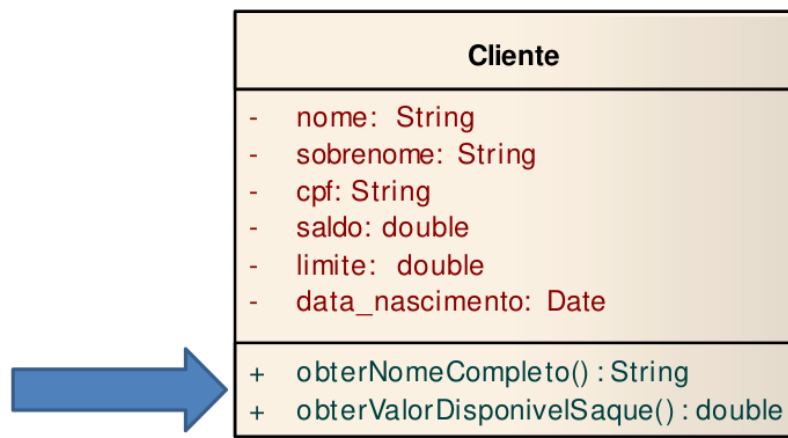
# Visibilidade (encapsulamento)



- Recurso na OO que visa a integridade dos valores dos atributos e acesso às operações.

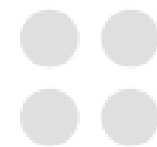
- **Público (public +)**

- Permite o acesso de atributos e operações para qualquer objeto do sistema



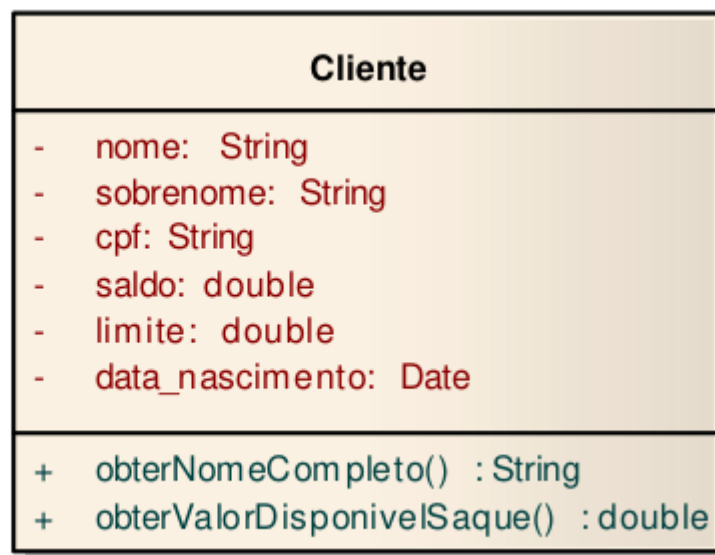


# Visibilidade (encapsulamento)



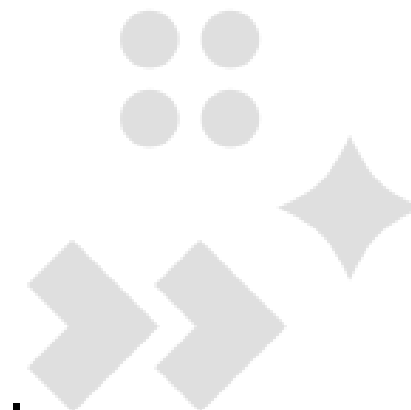
- **Privado (private -)**

- Limita o acesso aos atributos e operações somente para as instâncias da classe.



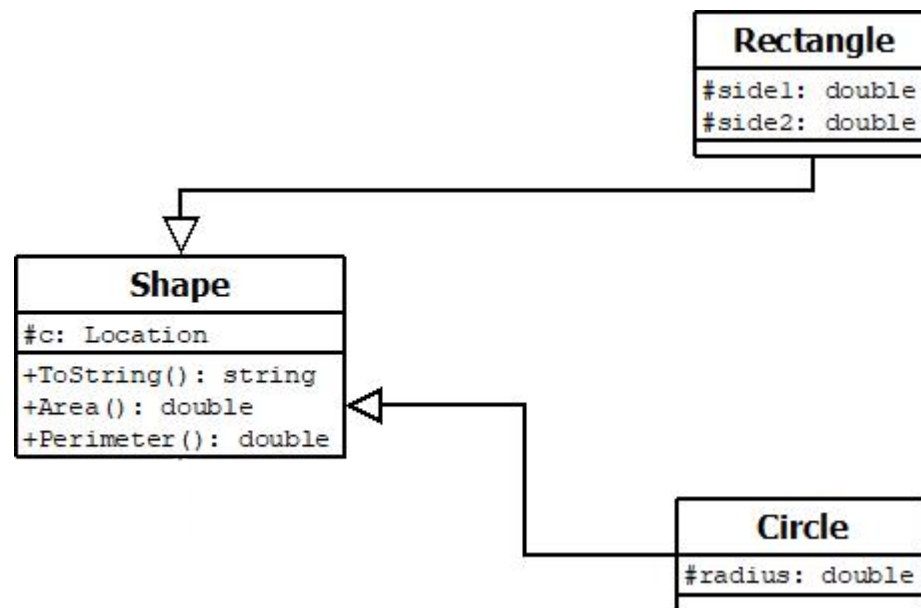


# Visibilidade (encapsulamento)



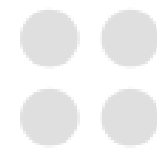
- **Protegido (protected #)**

- Limita o acesso aos atributos e operações somente pelas subclasses (classes filhas) ou por classes do mesmo pacote.





# Relacionamentos



- Como já dito, um sistema orientado a objetos é composto por um conjunto de classes que interagem entre si.
- Essa interação pode ser através de troca de mensagem (chamada de métodos) ou por relacionamento.
- Os relacionamentos especificam os vínculos entre as classes.



# Relacionamentos

- Identificando os relacionamentos

- Carro
- Pneu
- Motor







# Relacionamentos

- Identificando os relacionamentos

- Árvore
- Galhos
- Folhas



# Relacionamentos

- Identificando os relacionamentos

- Nota fiscal
- Prestador de serviço
- Favorecido
- Produtos

 PREFEITURA DO SALVADOR SECRETARIA MUNICIPAL DA FAZENDA	<b>NOTA FISCAL DE PRESTAÇÃO DE SERVIÇOS</b>		
	SÉRIE ELETRÔNICA: A Nº: 126		
	EMITIDA EM SUBSTITUIÇÃO À NOTA FISCAL ELETRÔNICA "A-105"		
	CÓDIGO DE SEGURANÇA: 4131.3671.3D74.62DE.3526.34B4.6ED0.37AF		
	PRESTADOR DE SERVIÇOS: [REDACTED]		
INSCRIÇÃO MUNICIPAL: [REDACTED]		INSCRIÇÃO ESTADUAL: [REDACTED]	
CNPJ: [REDACTED]		CIDADE: SALVADOR	UF: BA
ENDEREÇO: [REDACTED]			
TOMADOR DO SERVIÇO: [REDACTED]			
INSCRIÇÃO MUNICIPAL: [REDACTED]		INSCRIÇÃO ESTADUAL: [REDACTED]	
CNPJ: [REDACTED]		CIDADE: SALVADOR	UF: BA
ENDEREÇO: [REDACTED]			
DATA DE EMISSÃO: 19/07/2006		COMPETÊNCIA: 6/2006	
PROTOCOLO: 544			
SERVIÇOS: [REDACTED] [REDACTED] [REDACTED] [REDACTED]			
BASE DE CÁLCULO: R\$ 78,00		VALOR TOTAL DA NOTA FISCAL: R\$ 78,00	
ALÍQUOTA: 2%		VALOR DO ISS: R\$ 1,56	



# Tipos de Relacionamentos



• Herança



• É um

• Associação



• Usa

• Agregação



• Tem

• Composição



• Possui (é dono)

•



# Multiplicidade



(1-1): cliente tem uma e somente uma conta



(0-1): cliente pode ter uma conta



(1-N): cliente tem no mínimo um conta, mas pode ter mais



(0-N): cliente pode ter várias contas





# Navegabilidade



- Um relacionamento sem navegabilidade implica que ele pode ser lido de duas formas, isto é, em suas duas direções.



Uma empresa possui um trabalhador, como também um trabalhador trabalha em uma empresa.

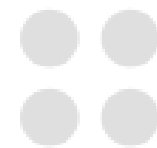
- Utilizando a propriedade de navegabilidade, podemos restringir a forma de ler um relacionamento. Isto é, em vez de termos duas direções, teremos apenas uma direção (de acordo com a direção da navegação)



Uma empresa possui um trabalhador

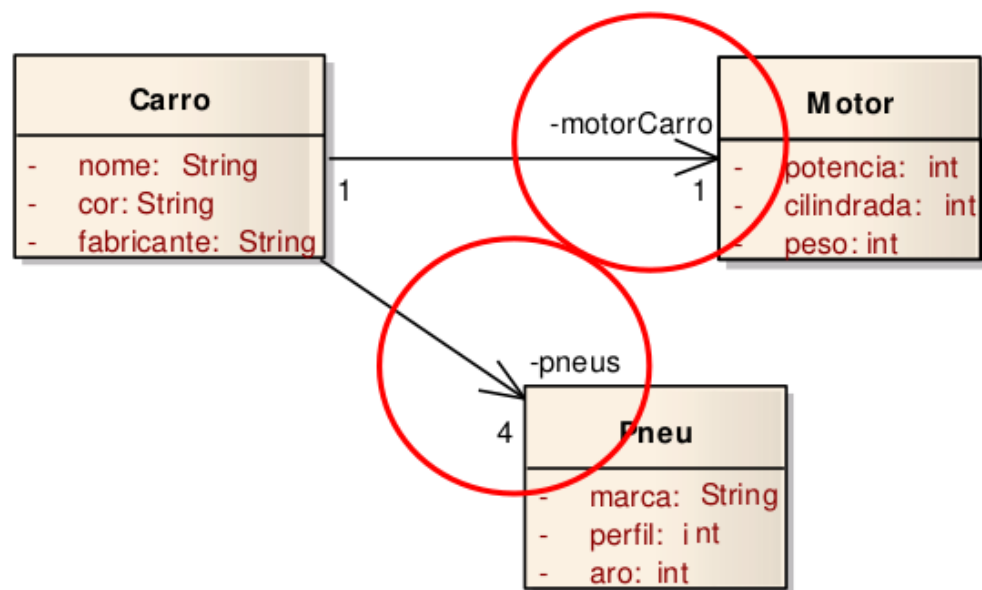


# Associação simples



- Classes suficientemente independentes porém possuem relação:
  - Um carro é dirigido por um motorista
  - Uma disciplina é cursada por um ou mais alunos
  - Um morador habita uma casa

# Associação simples



- Um carro possui 1 motor
- O atributo que armazena o motor é denominado “motorCarro”
- O atributo “motorCarro” é privado
- A partir do objeto “carro” é possível acessar o objeto “motor”

- Um carro possui 4 pneus
- O atributo que armazena os pneus é denominado “pneus”
- O atributo “pneus” é privado
- A partir do objeto “carro” é possível acessar os objetos “pneu”

# Agregação

- Agregação é uma relação em que um objeto é parte de outro, de tal forma que a parte pode existir sem o todo.
- Em mais baixo nível, uma agregação consiste de um objeto contendo referências para outros objetos, de tal forma que o primeiro seja o todo, e que os objetos referenciados sejam as partes do todo.
- A diferença entre os relacionamentos de associação e agregação ainda é algo de bastante discussão entre os gurus. De forma geral, utiliza-se para dar mais “força” a relação.



# Agregação



```
public class A {
    private B b;
    public A() {
    }
    public void setB( B b ) {
        this.b = b;
    }
    public B getB() {
        return b;
    }
}

public class B {
    public B() {
    }
}
```

## Regras:

- O objeto agregado (todo) pode potencialmente existir sem os seus objetos constituintes;
- A qualquer hora, cada objeto pode ser um constituinte (parte) com mais de um agregado;



# Composição

- Pode-se dizer que composição é uma variação da agregação. Uma composição tenta representar também uma relação todo - parte.
- No entanto, na composição o objeto-pai (todo) é responsável por criar e destruir suas partes.
- Em uma composição um mesmo objeto-parte não pode se associar a mais de um objeto-pai.

# Composição



```
public class A {
    private B b;
    public A() {
        b = new B();
    }
}
public class B {
    public B() {
    }
}
```

## Regras:

- O objeto composto (todo) não existe sem os seus componentes (parte);
- A qualquer hora, cada dado objeto componente pode ser parte de somente um objeto composto;
- Uma associação entre o objeto composto e cada um dos objetos componentes aparece no diagrama sob a forma de uma linha de associação, com um pequeno diamante preto junto ao objeto composto;



# Agregação

x

# Composição

