



UNIVERSIDADE
FEDERAL DO CEARÁ

TÓPICO
05



PROGRAMAÇÃO ORIENTADA A OBJETOS

Prof. Marcos Vinicius de Andrade Lima
E-mail: marcos.vinicius@ufc.br



Olá!

Sou Marcos Vinicius

Nos tópicos passados nós aprendemos a definir as características e os comportamentos dos nossos objetos...

Neste tópico veremos qual a relação existe entre **POO** e **UML**! Uma breve introdução!

“

Nós só podemos ver um pouco do futuro, mas o suficiente para perceber que há muito a fazer
(Alan Turing)



Introdução à *Unified Modeling Language* (UML)

INTRODUÇÃO

A **Linguagem de Modelagem Unificada** (do inglês, ***Unified Modeling Language*** - UML) foi criada por Grady Booch, James Rumbaugh (OMT) e Ivar Jacobson no final dos anos 90.

É uma linguagem **gráfica** que padronizou as notações para os componentes do **paradigma de orientação a objetos**.

A versão atual é a **UML 2.5** (mar/2015)



Prof. Marcos Vinicius - UFC/Russas - POO

5/45

A **UML não é uma metodologia** de desenvolvimento!

Ela é um **auxílio** para **visualizar** seu **projeto** e a comunicação entre os objetos.



Os 7 OBJETIVOS DA UML

1. Prover aos usuários uma **linguagem de modelagem visual** expressiva e pronta para uso, de forma que eles possam desenvolver e intercambiar modelos significativos;
2. Prover **mecanismos de extensibilidade e especialização** para ampliar os conceitos centrais;
3. Ser **independente de linguagens de programação** e processos de desenvolvimento particulares;
4. Prover uma **base formal** para entendimento da linguagem de modelagem;

Prof. Marcos Vinicius – UFC/Russas - POO

7 / 45

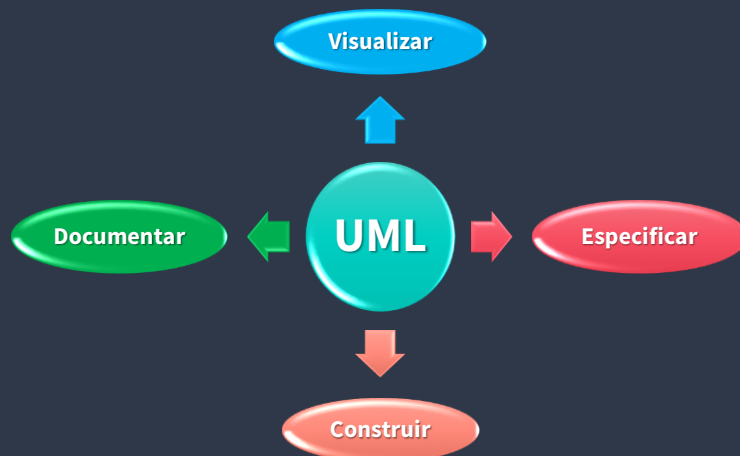
Os 7 OBJETIVOS DA UML

4. Estimular o crescimento do mercado de **ferramentas OO**;
5. Suportar conceitos de **desenvolvimento de nível mais alto**, tais como colaborações, estruturas, modelos e componentes;
6. Integrar as **melhores práticas**.

Prof. Marcos Vinicius – UFC/Russas - POO

8 / 45

UML é para...

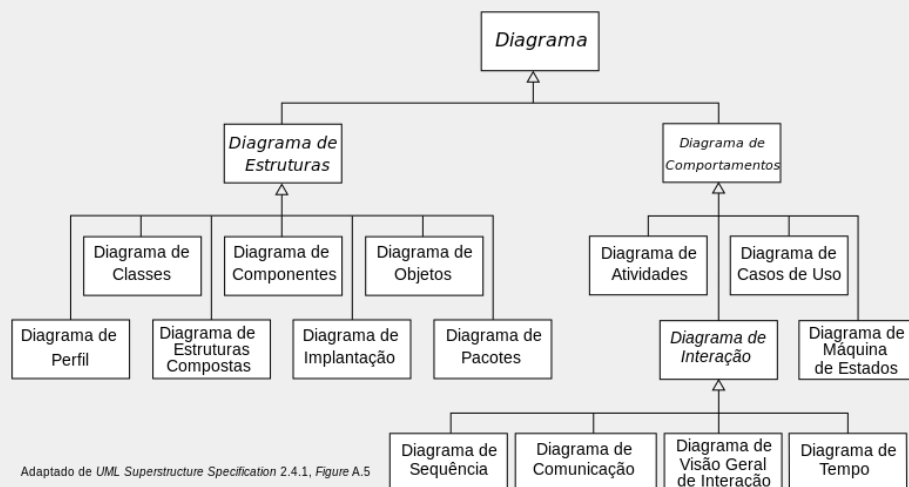


... os artefatos de um **sistema complexo**
de *software*.

DIAGRAMAS DA UML



DIAGRAMAS DA UML



Prof. Marcos Vinícius – UFC/Russas - POO

11/45

Diagramas da UML

Diagrama de Classes

MODELO DE CLASSES

- É utilizado para representar o **aspecto estrutural** estático do sistema.
- O **modelo de classes** é composto desse **diagrama** e da **descrição** textual associada.



Prof. Marcos Vinicius - UFC/Russas - POO

MUITO CUIDADO NESTA HORA!

- O modelo de classes **evolui** durante o desenvolvimento do sistema.
- À medida que o sistema é desenvolvido, o modelo de classes é **incrementado** com **novos detalhes**!

Abstração
na veia!



Prof. Marcos Vinicius - UFC/Russas - POO

14/45

**Depois não diga
que eu não avisei!**

O diagrama de classes mostra **quais classes “conhecem” quais classes ou quais classes “são parte” de outras classes**, mas não mostram a troca de mensagens entre elas.

NOTAÇÃO UML PARA CLASSE

- Representada através de uma **retângulo** com no **máximo três subdivisões** visíveis.

Nome da Classe

(a)

Nome da Classe

lista de atributos

(b)

Nome da Classe

lista de operações

(c)

Nome da Classe

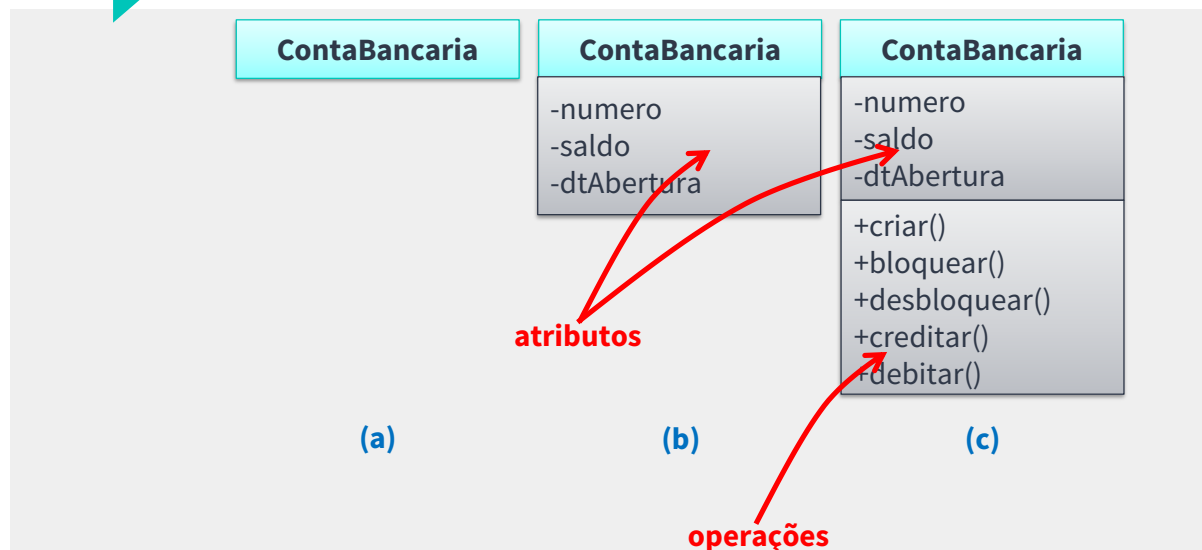
lista de atributos

lista de operações

(d)

- A **notação** utilizada depende do nível de **abstração** desejado:

EXEMPLO DE CLASSE NA UML



Prof. Marcos Vinicius – UFC/Russas - POO

17/45

ATRIBUTOS DE UMA CLASSE

- Na UML, **atributos** são mostrados com pelo menos seu **nome**, e podem também mostrar seu **tipo**, **valor inicial** e **outras propriedades**.
- Atributos podem também ser exibidos com sua **visibilidade**. A visibilidade tem impacto direto ao nível de **encapsulamento** que o atributo possui.

Visibilidade de atributos das classes:

Símbolo	Descrição
+	indica atributos públicos (menor nível de encapsulamento).
#	indica atributos protegidos.
~	indica atributos de pacote.
-	indica atributos privados (maior nível de encapsulamento).

Prof. Marcos Vinicius – UFC/Russas - POO

18/45

OPERAÇÕES DE UMA CLASSE

- As operações (na codificação serão chamadas de métodos) também são exibidas com pelo menos seu nome, e podem também mostrar seus parâmetros e valores de retorno.
- As operações podem, assim como os atributos, mostrar sua visibilidade.

Visibilidade de operações das classes:

Símbolo	Descrição
+	indica operações públicas (menor nível de encapsulamento).
#	indica operações protegidas.
~	indica operações de pacote.
-	indica operações privadas (maior nível de encapsulamento).

Prof. Marcos Vinicius – UFC/Russas - POO

19/45

RELACIONAMENTOS

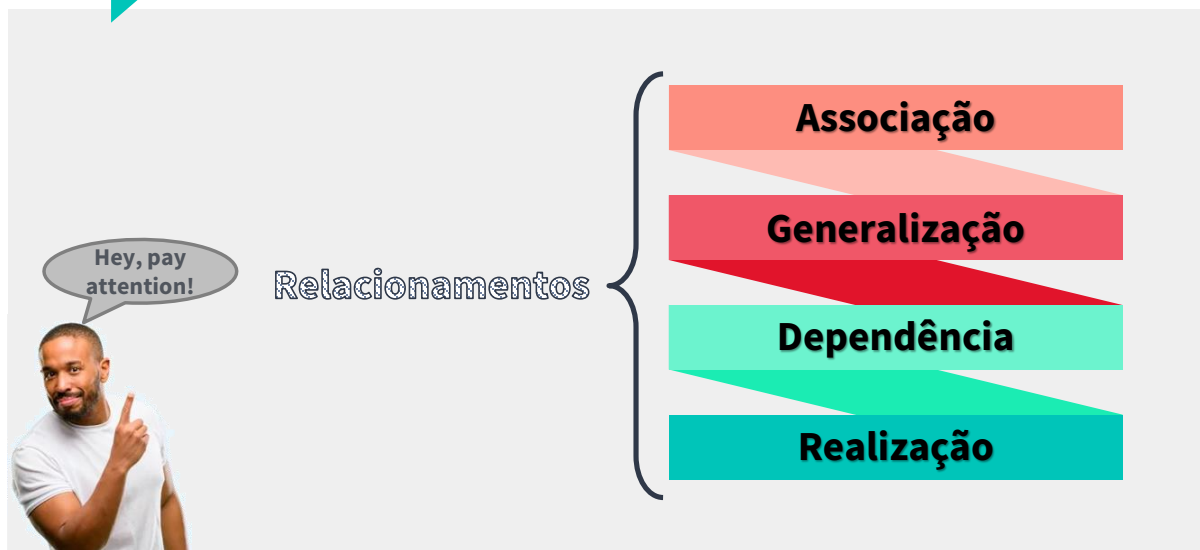
- Os **sistemas** OO possuem **classes** e **objetos**;
- A **colaboração** entre os **objetos** determina o comportamento do sistema;
- Os objetos interagem via **troca de mensagens**;
- Os relacionamentos fornecem o meio (duto) para a interação entre os objetos.



Prof. Marcos Vinicius – UFC/Russas - POO

20/45

ALGUNS TIPOS DE RELACIONAMENTOS DO DIAGRAMA DE CLASSES



Prof. Marcos Vinicius - UFC/Russas - POO

21/45

TIPOS DE RELACIONAMENTOS: ASSOCIAÇÃO

- Para representar o fato de que **objetos** podem (e devem) se **relacionar** uns com os outros, utiliza-se a associação.
- Uma associação representa relacionamentos (ligações) que são **formados entre objetos durante a execução do sistema**.
- Embora as associações sejam representadas entre classes do diagrama, tais associações representam ligações possíveis entre **objetos** das classes envolvidas.

Prof. Marcos Vinicius - UFC/Russas - POO

22/45

NOTAÇÃO DA ASSOCIAÇÃO

- Representada por meio de um **segmento de reta** ligando as classes cujos objetos se relacionam.



Prof. Marcos Vinicius – UFC/Russas - POO

23/45

CLASSE ASSOCIATIVA

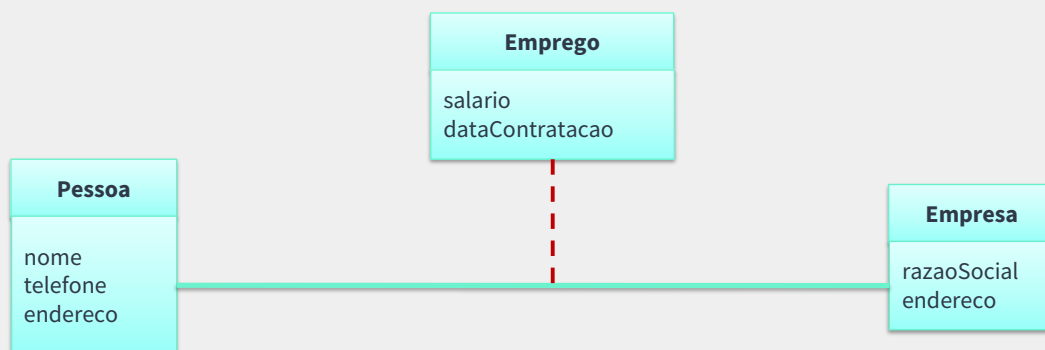
- É uma **classe** que está **ligada a uma associação**, ao invés de estar ligada a outras classes.
- É normalmente utilizada quando **duas ou mais classes estão associadas** e é necessário **manter informações sobre esta associação**.

Prof. Marcos Vinicius – UFC/Russas - POO

24/45

NOTAÇÃO DA CLASSE ASSOCIATIVA

- Representada pela notação utilizada para uma classe. A diferença é que esta classe é ligada a uma associação.



Prof. Marcos Vinicius – UFC/Russas - POO

25/45

ASSOCIAÇÃO REFLEXIVA

- Associa **objetos** da **mesma classe**.
- Cada objeto tem um papel distinto na associação.



Prof. Marcos Vinicius – UFC/Russas - POO

26/45

TIPOS DE RELACIONAMENTOS: DEPENDÊNCIA

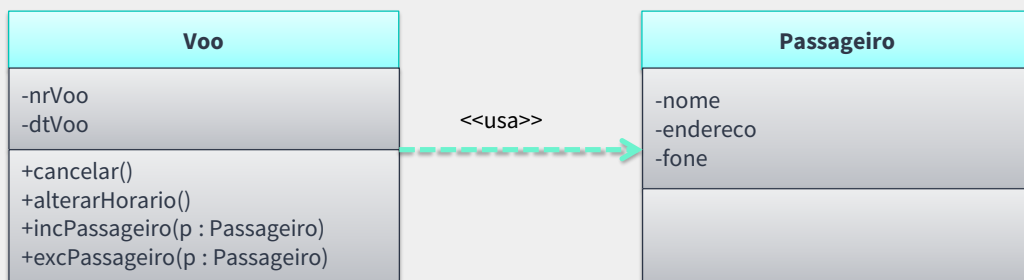
- Uma classe **usa a outra**, depende da outra.
- Dependência **não** gera atributo de conexão.
- Quando uma classe é **passada como parâmetro** ou como **tipo de retorno** da outra.
- **Uma classe não acessa atributos da outra.**

Prof. Marcos Vinicius – UFC/Russas - POO

27 / 45

NOTAÇÃO DA DEPENDÊNCIA

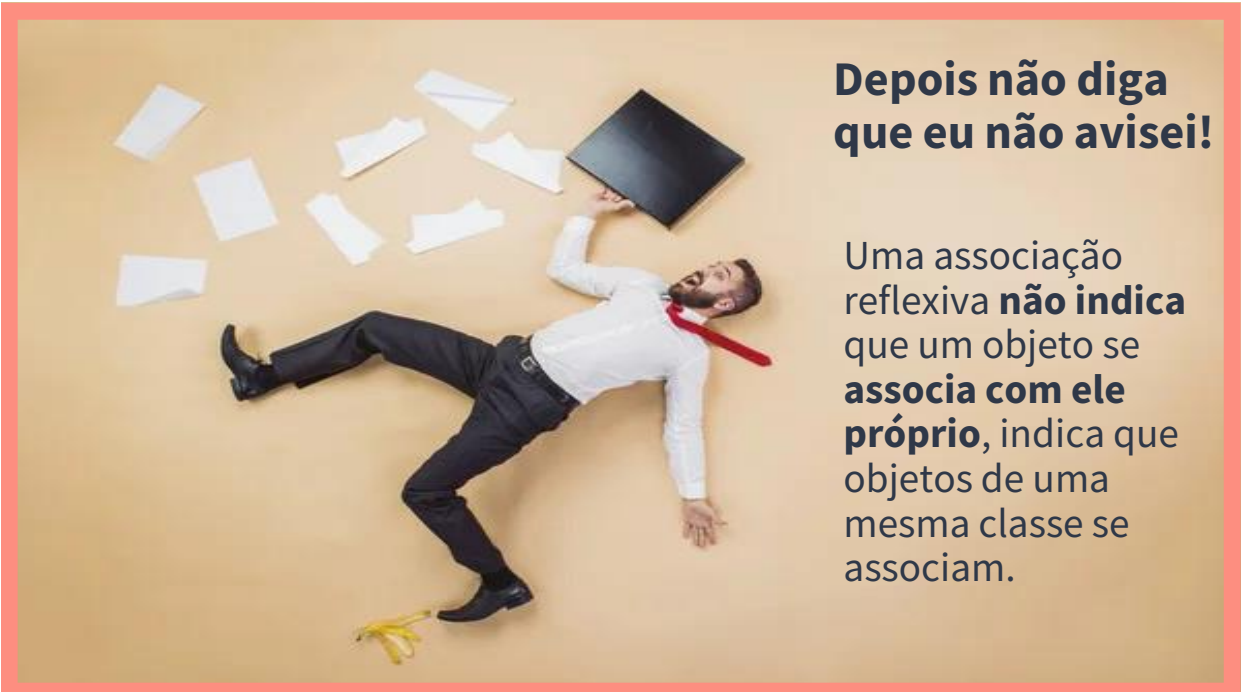
- Unidirecional com reta pontilhada com seta aberta em uma das extremidades.



- A classe Voo usa a classe Passageiro via argumento de operação do tipo Passageiro.

Prof. Marcos Vinicius – UFC/Russas - POO

28 / 45






Depois não diga
que eu não avisei!

Uma associação reflexiva **não indica** que um objeto se **associa com ele próprio**, indica que objetos de uma mesma classe se associam.

IDENTIFICANDO CLASSES

- Um **sistema** de *software* **orientado a objetos** é composto de **objetos em colaboração** para realizar as tarefas do sistema.
- Por outro lado, **todo objeto pertence a uma classe**.
- Portanto, quando se fala na **identificação das classes**, o objetivo na verdade é saber **quais objetos irão compor o sistema**.

RESPONSABILIDADES

- Costuma-se categorizar os objetos de um sistema de acordo com o tipo de responsabilidade a ele atribuída:
 - ✓ **objetos de entidade** (*entity object*) 
 - ✓ **objetos de controle** (*control object*) 
 - ✓ **objetos de fronteira** (*boundary object*) 
- Esta categorização foi proposta por Ivar Jacobson (Jacobson *et al.*, 1992) em uma técnica denominada **Análise de Robustez**.

Prof. Marcos Vinicius – UFC/Russas - POO

31/45

OBJETOS DE ENTIDADE

- Um **objeto de entidade** é um **repositório** para alguma informação manipulada pelo sistema.
- Esses objetos representam conceitos do **domínio do negócio**.
- Normalmente** esses objetos **armazenam informações persistentes**.
- Há várias instâncias** de uma **mesma classe** de entidade **coexistindo no sistema**.

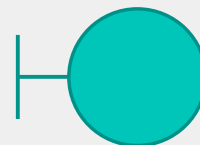


Prof. Marcos Vinicius – UFC/Russas - POO

32/45

OBJETOS DE FRONTEIRA

- Esses objetos **traduzem os eventos gerados por um ator em eventos relevantes ao sistema.**
- Também são **responsáveis por apresentar os resultados** de uma interação dos objetos em algo inteligível pelo ator.
- Um objeto de fronteira **existe para que o sistema se comunique com o mundo exterior.**



Prof. Marcos Vinicius – UFC/Russas - POO

33/45

OBJETOS DE CONTROLE

- São a “**ponte de comunicação**” entre objetos de **fronteira** e objetos de **entidade**.
- Responsáveis por **controlar a lógica de execução** correspondente a um caso de uso.
- **Decidem** o que o sistema **deve fazer quando um evento externo relevante ocorre.**
 - ✓ Eles realizam o controle do processamento
 - ✓ Agem como gerentes (coordenadores, controladores) dos outros objetos para a realização de um ou mais caso de uso.



Prof. Marcos Vinicius – UFC/Russas - POO

34/45

Diagramas da UML

Diagrama de Objeto

DIAGRAMA DE OBJETO: INTRODUÇÃO

- Com base nas **relações definidas no diagrama de classe**, o **diagrama de objeto** representa uma **foto instantânea do estado do sistema**.
- Permite representar **objetos concretos** que aparecem em um **sistema** em um **ponto específico no tempo**.



NOTAÇÃO DO DIAGRAMA DE OBJETO

- Um objeto é mostrado como um **retângulo** que pode ser **subdividido em múltiplos compartimentos**, assim como ocorreu com as classes.
- O **primeiro** compartimento contém a identificação do objeto, normalmente composto por **nome:Classe** (centralizado e sublinhado).

meuProf:Pessoa

meuProf

:Pessoa

meuMedico:Pessoa

nome = "Marcos"
sobrenome = "Lima"
profissao = "Professor"

nome = "Jonathan"
sobrenome = "Lopes"
profissao = "Médico"

lista de operações

(a)

(b)

(c)

(d)

Prof. Marcos Vinicius - UFC/Russas - POO

37/45

**Se o nome do objeto for omitido,
ele será chamado de
objeto anônimo!**

Diagramas da UML

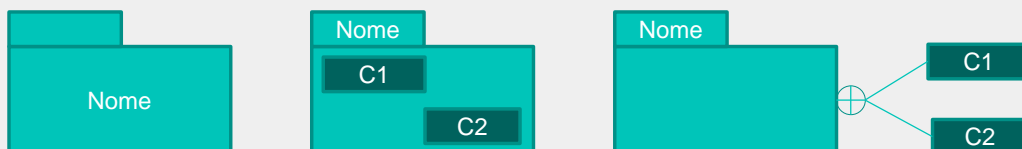
Diagrama de Pacote

DIAGRAMA DE PACOTE: INTRODUÇÃO

- Em linguagens de programação observamos o conceito de ***namespace***, utilizado para organizar melhor o código.
- Em **Java**, por exemplo, isso é feito por meio de pacotes (***packages***).
- A **UML** tem suporte para essa **organização dos sistemas** por meio do **diagrama de pacote**.
- Um **pacote** habilita você a **agrupar elementos**, como classes, tipos de dados, atividades, estados, entre outros. Inclusive outros pacotes!

NOTAÇÃO DO DIAGRAMA DE PACOTE

- É representado por um retângulo com um pequeno retângulo no topo do lado esquerdo, lembrando pasta compactada.



Prof. Marcos Vinícius – UFC/Russas - POO

41/45

**Depois não diga
que eu não avisei!**

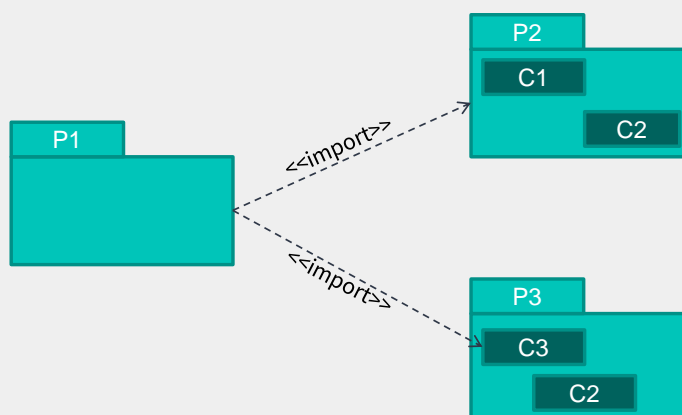
Um **elemento** só
pode **pertencer** a no
máximo **um pacote**.

Elementos podem ter
o **mesmo nome**,
desde que estejam
em **pacotes**
diferentes!



OLHA QUE LEGAL!

- Um dado pacote pode importar um outro pacote ou elemento contido em outro pacote.



Prof. Marcos Vinicius – UFC/Russas - POO

43 / 45

FERRAMENTAS CASE PARA UML

- Existem várias ferramentas CASE que visam auxiliar o processo de modelagem de um sistema.
- Entre elas temos:

astah professional

StarUML™

ENTERPRISE
ARCHITECT

Umbrello
UML Modeller

Visio

Qual
ferramenta
utilizar?



Prof. Marcos Vinicius – UFC/Russas - POO

44 / 45



Obrigado!

Mais alguma dúvida?

Acesse o **AME** para mais informações e treinamento do **NERDS!**

<http://ame2.russas.ufc.br>

