

Diagrama de Classes

UML e Relacionamentos entre Classes

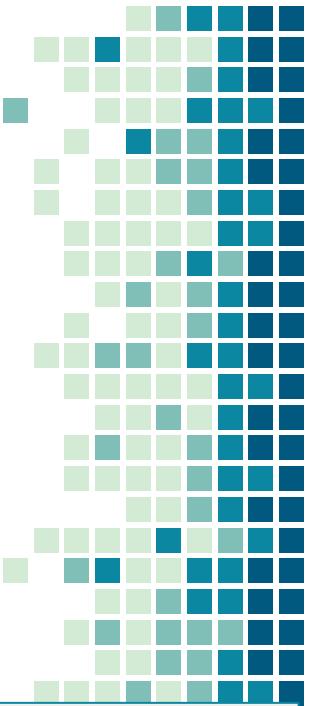
Prof. Jônata Tyska Carvalho



Diagrama de Classe: Agenda

1. UML: definição, função e diagramas
2. Diagrama de Classes
 - a. Classes e Objetos
 - b. Relacionamento entre classes

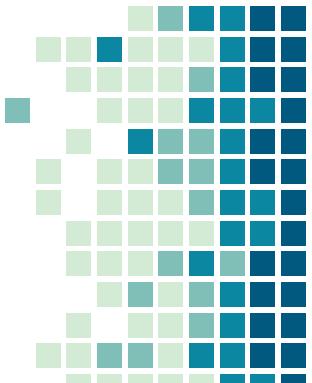
Diagrama de Classes: UML



- *Unified Modeling Language (UML)*: **Linguagem de Modelagem Unificada** criada em 97 pelo *Object Management Group (OMG)* que unificou as diferentes notações da época: **melhores práticas**
- **Notação gráfica** baseada em **diversos diagramas** para a modelagem de um sistema.
- Independente de linguagem de programação



Diagrama de Classes: UML



- Como UML pode ajudar?
 - a. Desenvolver software: **tarefa complexa**
 - i. Especificação de requisitos
 - ii. Implementação
 - iii. Manutenção
 - iv. Adaptação/Evolução
 - v. Times com múltiplos dev's
 - b. Criação da 'planta' do sistema
planejamento

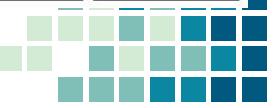
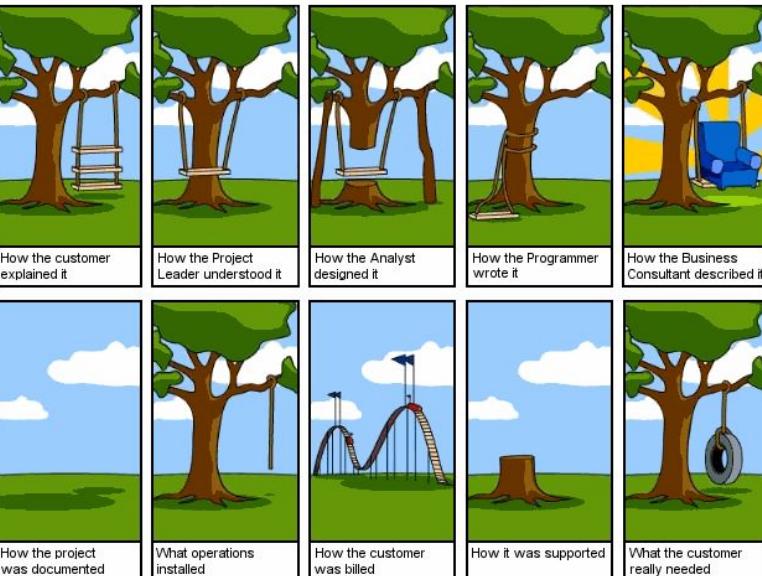


Diagrama de Classes: UML

- UML
 - a. Possibilita construir modelos destacando aspectos importantes
 - b. Esconde detalhes irrelevantes na modelagem
 - c. Divide os diferentes aspectos, em diferentes diagramas

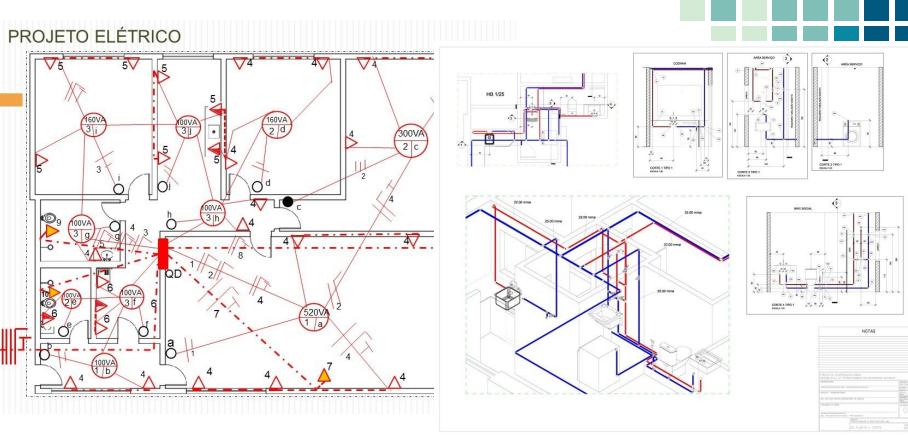
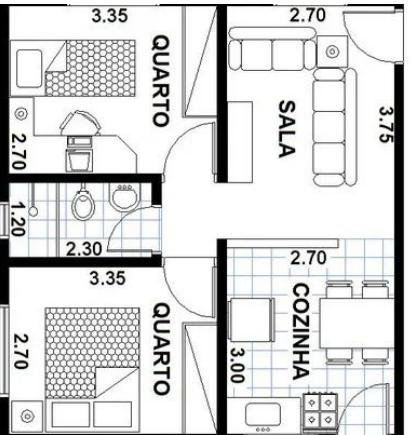
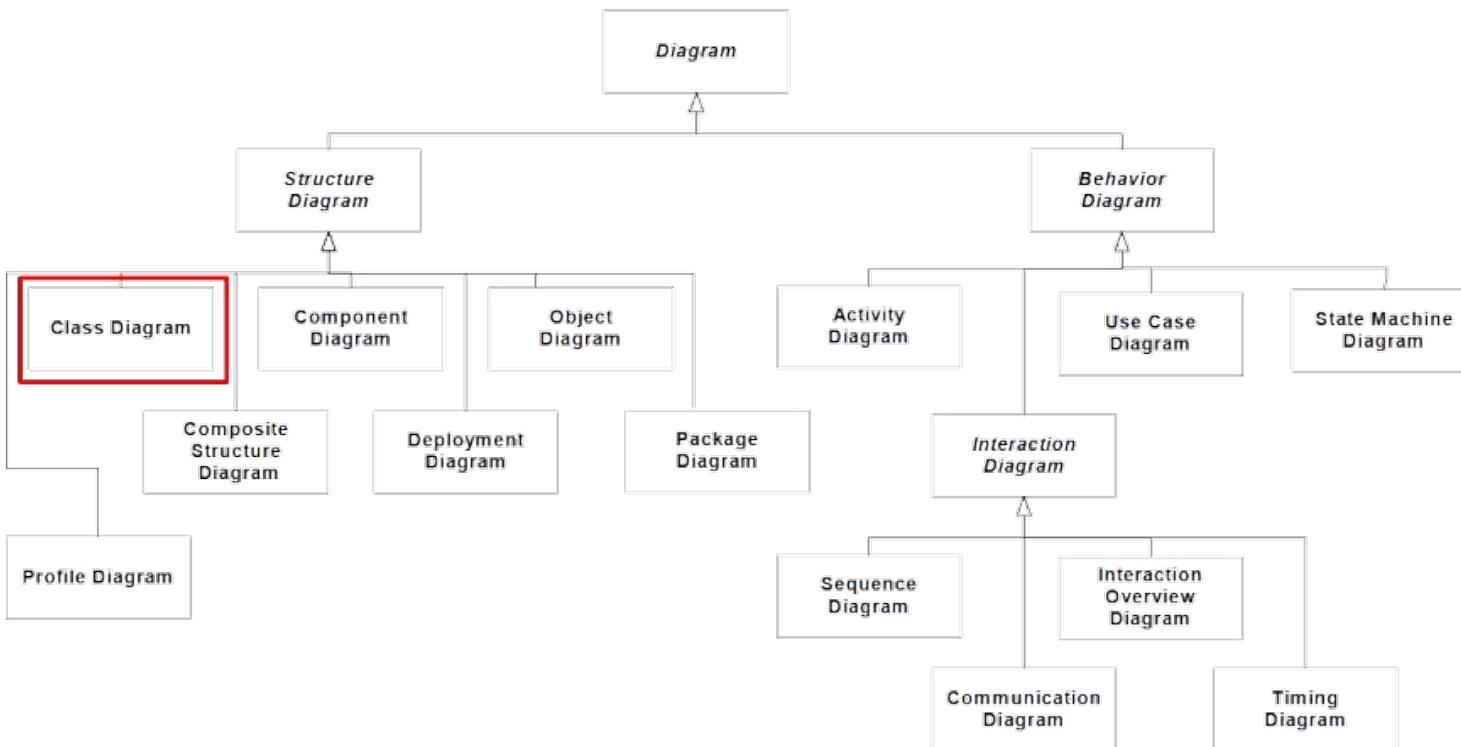


Diagrama de Classes: UML



[OMG, 2015]

Diagrama de Classes: UML

- Por quê modelar um sistemas de software?
 - Escolher quais abstrações utilizar em um sistema é **fundamental**

Boas abstrações tornam a **programação simples**

- i. Legibilidade
- ii. Manutenibilidade
- iii. Extensibilidade

Máis abstrações = Spaghetti Code

- i. Rigidez
- ii. Fragilidade
- iii. Complexidade
- iv. Redundância

Diagrama de Classes: UML

- Por quê modelar um sistemas de software?
 - Escolher quais abstrações utilizar em sistema é **fundamental**

Boas abstrações tornam
programação **simples**

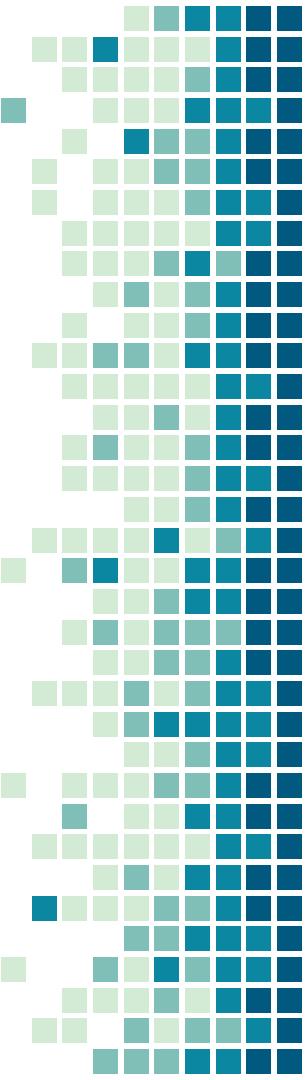
- i. Legibilidade
- ii. Reutilização
- iii. Encapsulamento

E como modelar ajuda?

Malas abstrações = Spaghetti Code

- i. Rigidez
- ii. Fragilidade
- iii. Complexidade
- iv. Redundância

Diagrama de Classes: UML



Modelos podem ser usados de três formas:

1. **Esboço:** servem para comunicar e promover discussão sobre certos aspectos - **pensar**
2. **Planta/diagrama (blueprint):** **completude**, permitem construir software sem tomar decisões de projeto - normalmente não especificam tudo
3. **Programas executáveis:** código sendo gerado a partir da especificação de modelos

Diagrama de Classes: Agenda

1. UML: definição, função e diagramas
2. **Diagrama de Classe**
 - a. Classes e Objetos
 - b. Relacionamento entre classes

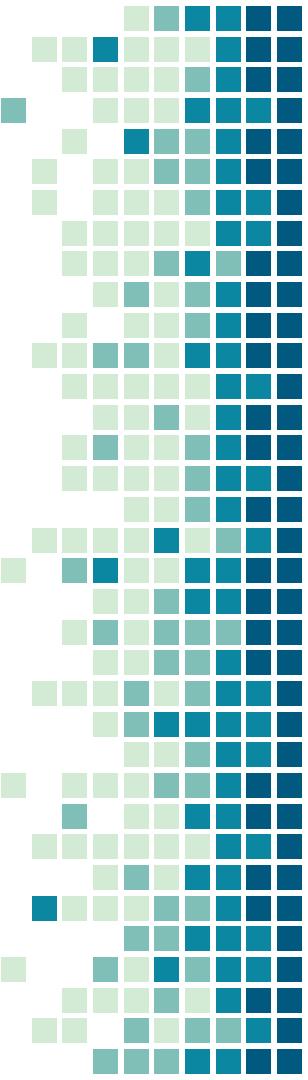


Diagrama de Classes

- Objetivo: modelar a estrutura estática

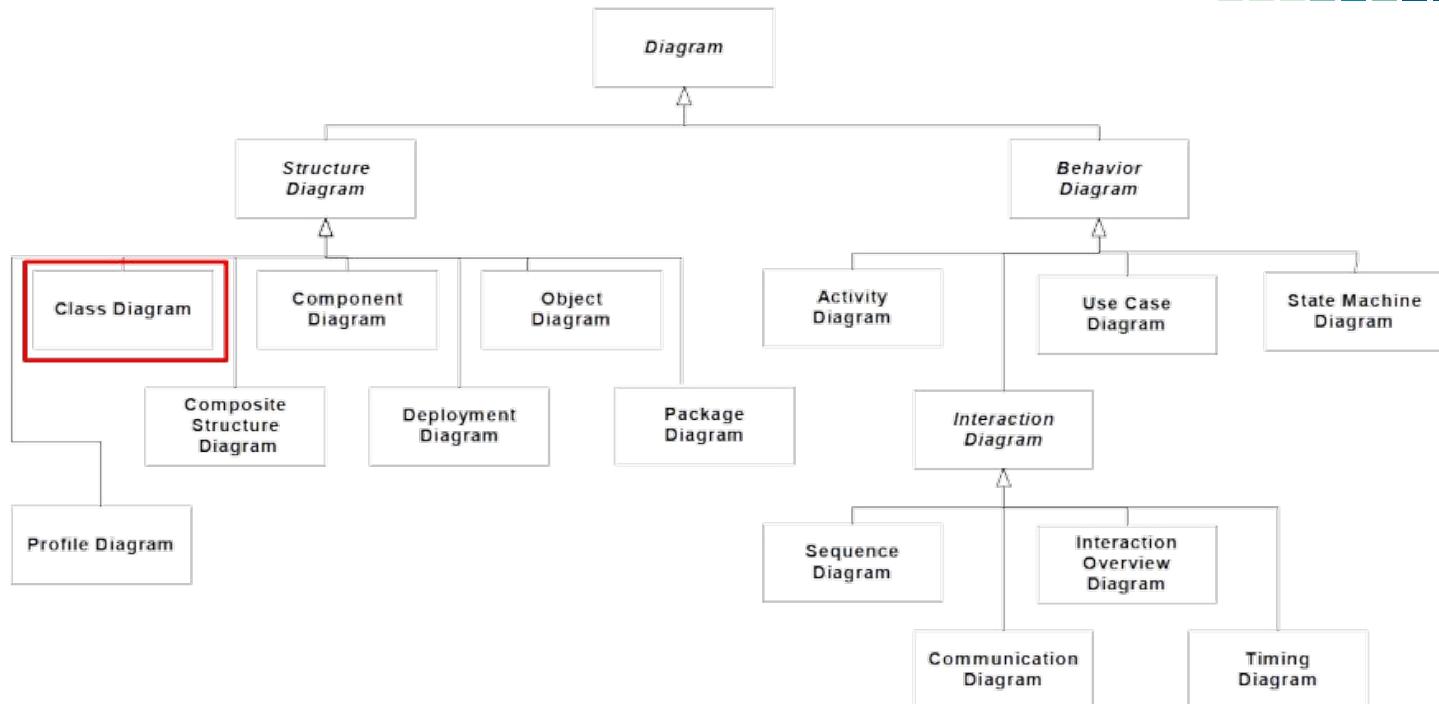


Diagrama de Classes

- Modela a estrutura estática
 - Descreve:
 - elementos (classes) e suas características
 - relacionamentos entre as classes
- Diagrama UML mais usado
- Usado em várias fases: diferentes níveis de abstração

Diagrama de Classes

- Nas fases iniciais: visão conceitual e definição de vocabulário comum
- Simples e popular: muito útil para esboços rápidos
- Pode ser usado para gerar código automaticamente

Diagrama de Classes: Agenda

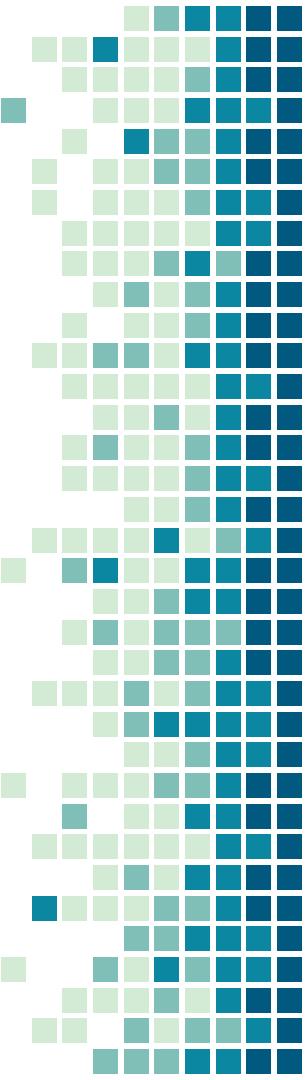
- 
1. UML: definição, função e diagramas
 2. Diagrama de Classe
 - a. **Classes e Objetos**
 - b. Relacionamento entre classes

Diagrama de Classes: Classes e Objetos

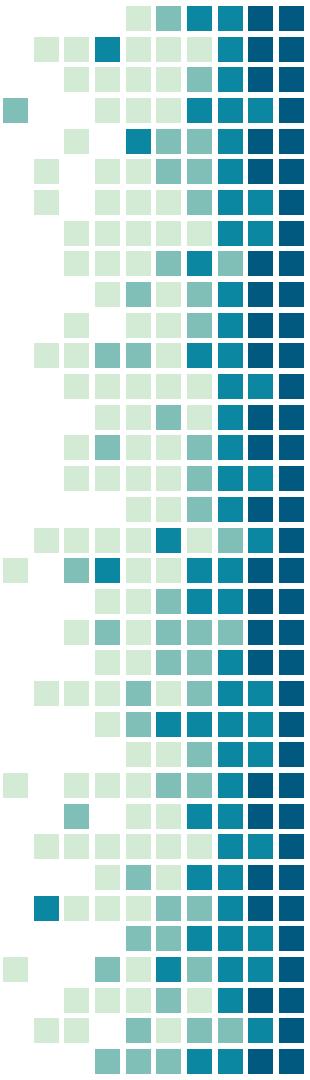


Figure 4.1

Example of an object diagram

Diagrama de Objetos

valor de objetos em um dado instante
"snapshot"

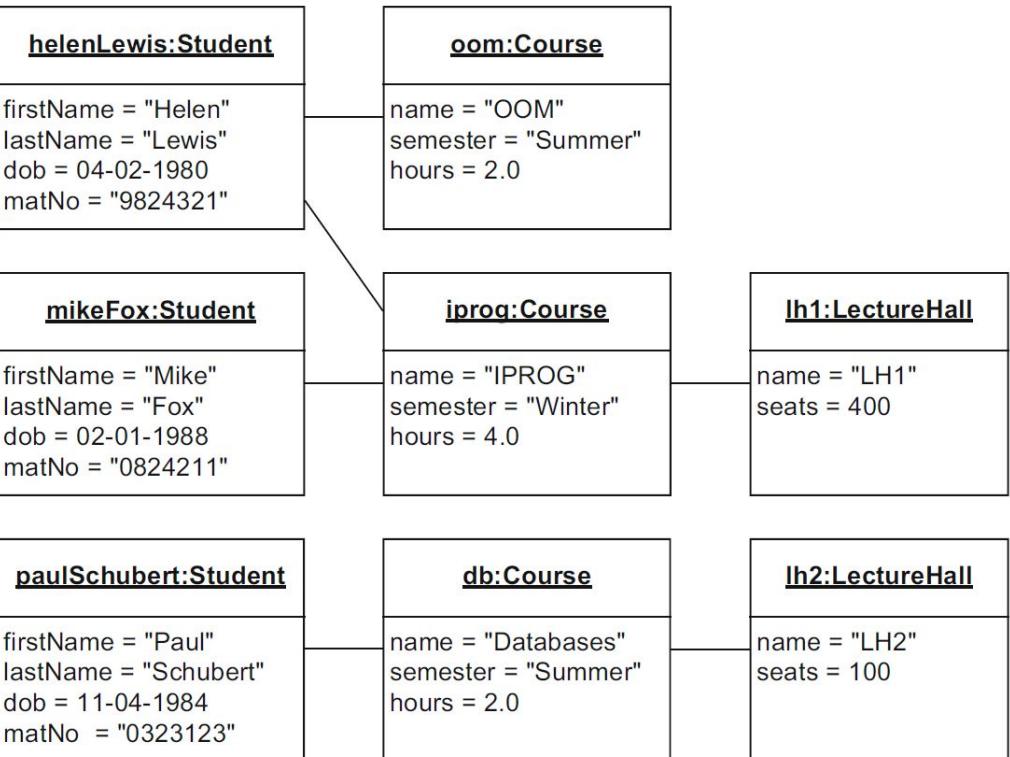


Diagrama de Classes: Classes e Objetos

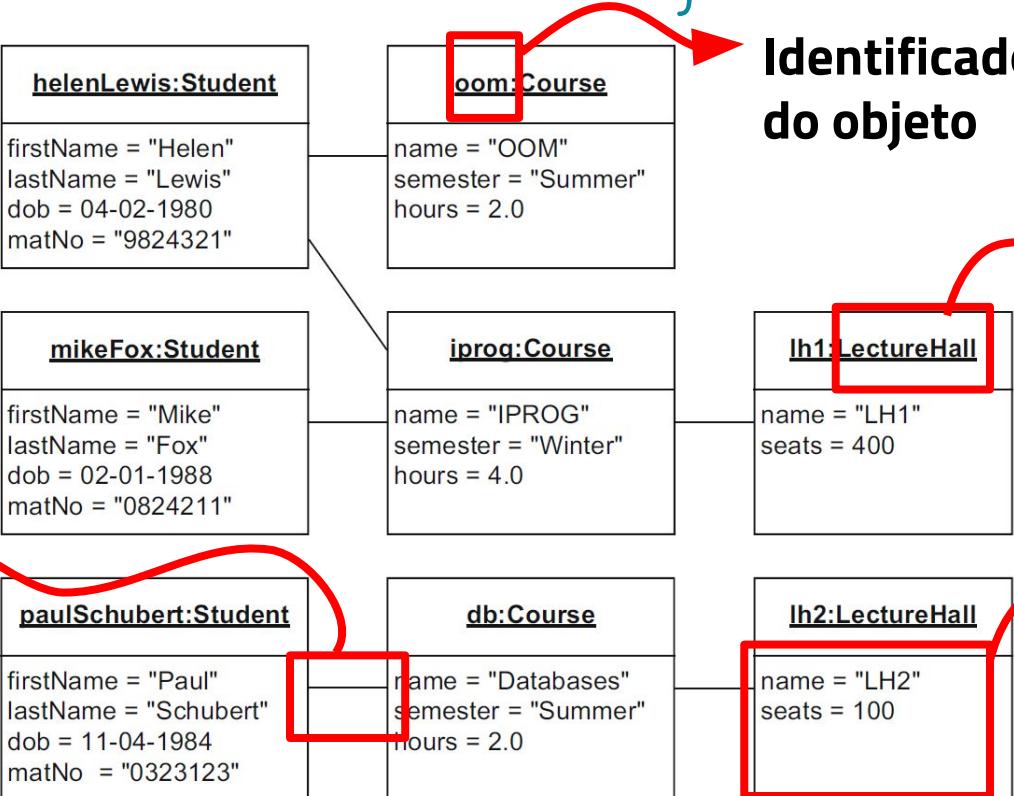


Figure 4.1
Example of an object
diagram

Diagrama de Objetos

valor de objetos em um
dado instante
"snapshot"

relacionamentos



**Identificador
do objeto**

**Nome da
classe**

**Atributos e
valores**

Diagrama de Classes: Classes e Objetos

Diagrama de Objetos

Notações alternativas

maxMiller

maxMiller:Person

:Person

maxMiller

```
firstName = "Max"
lastName = "Miller"
dob = 03-05-1973
```

maxMiller:Person

```
firstName = "Max"
lastName = "Miller"
dob = 03-05-1973
```

:Person

```
firstName = "Max"
lastName = "Miller"
dob = 03-05-1973
```

Figure 4.2
Notation alternatives for
objects

Diagrama de Classes: Classes e Objetos

Diagrama de Classe

Figure 4.3

Definition of a class in UML and Java

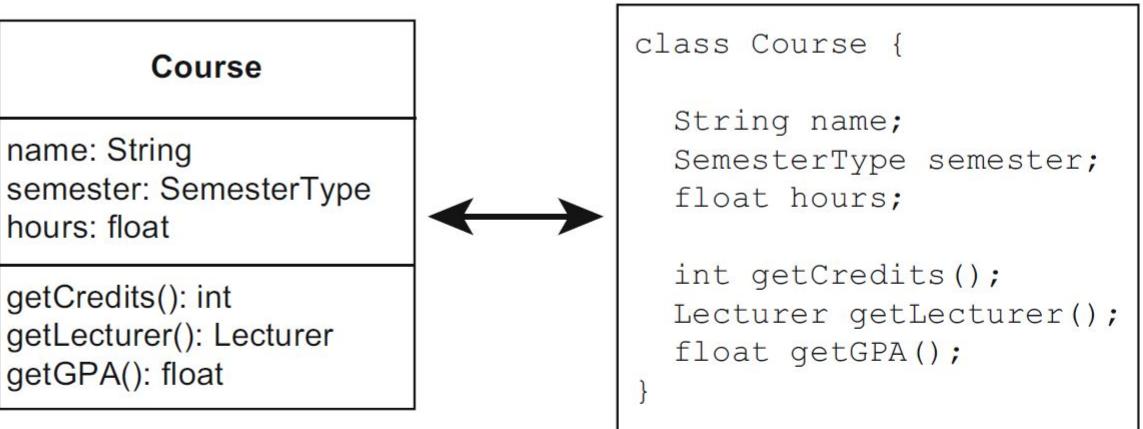


Diagrama de Classes: Classes e Objetos

Diagrama de Classe



(a)



name
semester
hours

getCredits()
getLecturer()
getGPA()

(b)



+ name: String
+ semester: SemesterType
- hours: float
- /credits: int

+ getCredits(): int
+ getLecturer(): Lecturer
+ getGPA(): float
+ getHours(): float
+ setHours(hours: float): void

(c)

Figure 4.4

Representation of a class and its characteristics

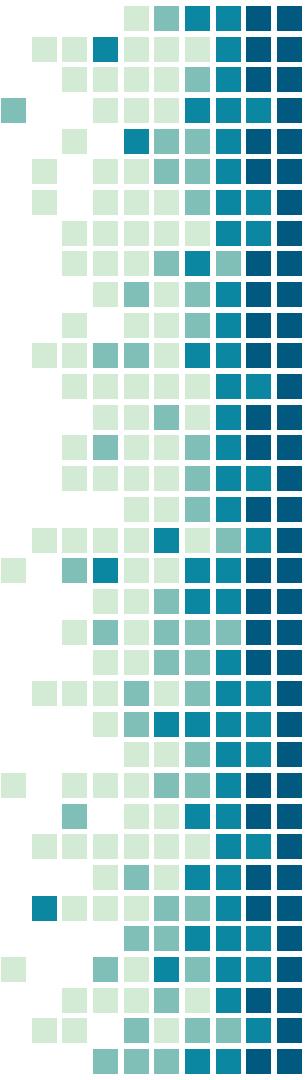


Diagrama de Classes: Classes e Objetos

Diagrama de Classe: atributos

Person
firstName: String
lastName: String
dob: Date
address: String [1..*] {unique, ordered}
ssNo: String {readOnly}
/age: int
password: String = "pw123"
<u>personsCounter: int</u>
getName(out fn: String, out ln: String): void
updateLastName(newName: String): boolean
<u>getPersonsCounter(): int</u>

```
age = now.getYear() -  
dob.getYear()
```

Figure 4.5
Syntax of the attribute specification

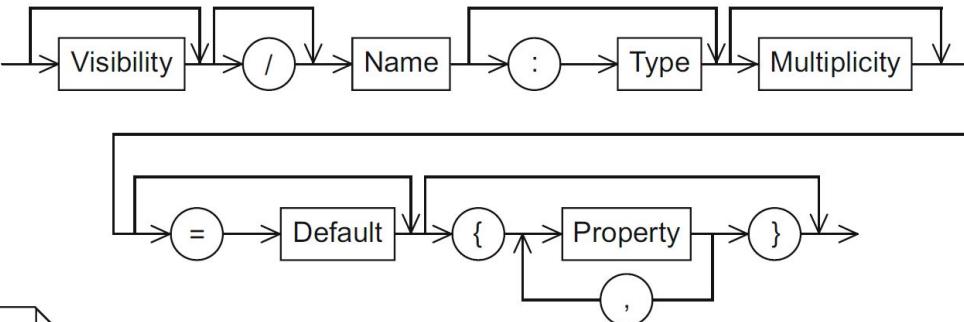


Diagrama de Classes: Classes e Objetos

Diagrama de Classe: métodos

Figure 4.10

Translation of a class from UML to Java

visibilidade

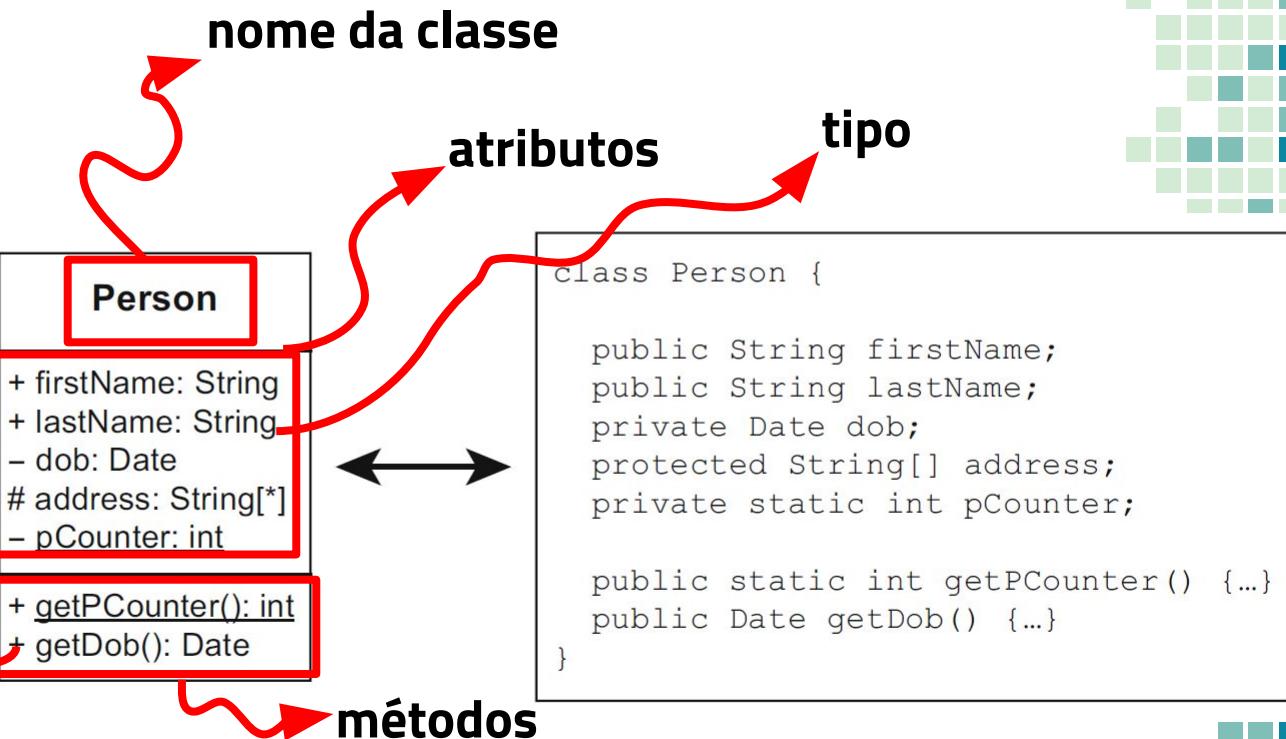


Diagrama de Classes: Agenda

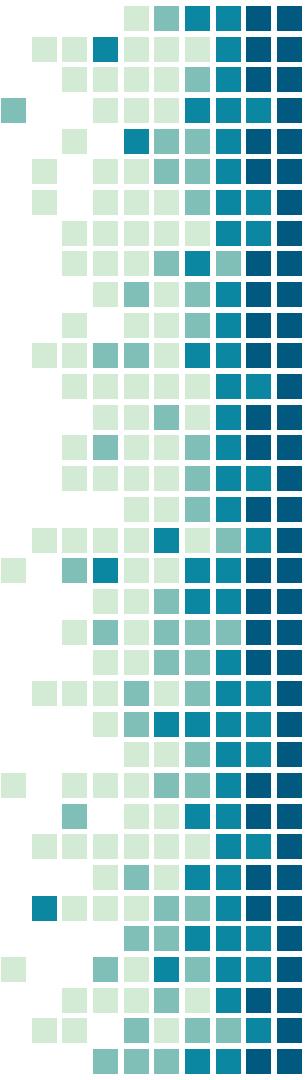
- 
1. UML: definição, função e diagramas
 2. Diagrama de Classe
 - a. Classes e Objetos
 - b. Relacionamento entre classes**

Diagrama de Classes: Relacionamentos

- Principais tipos de relacionamento
 - **Associação**
 - Agregação e Composição
 - Dependência
 - Generalização

Relacionamentos: Associação

- **Associação**

- representam relações entre objetos

Associação binária

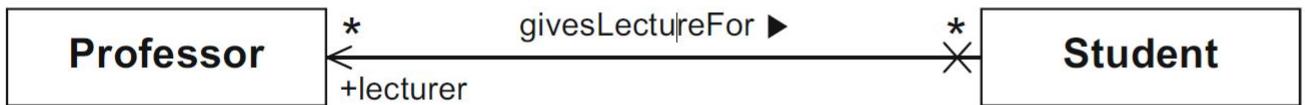


Figure 4.11
Example of a binary association in a class diagram
and a valid object diagram

Relacionamentos: Associação

- **Associação**

- representam relações entre objetos

Associação binária



Figure 4.11
Example of a binary association in a class diagram
and a valid object diagram

Relacionamentos: Associação

- **Associação**

- representam relações entre objetos



Figure 4.11
Example of a binary association in a class diagram
and a valid object diagram

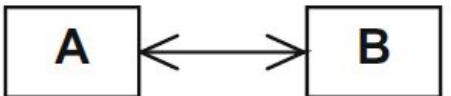
Relacionamentos: Associação

- **Associação**

- representam relações entre objetos



Navigability



Non-navigability

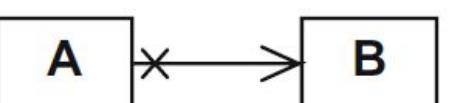


Figure 4.11
Example of a binary association in a class diagram and a valid object diagram

Relacionamentos: Associação

- **Associação**

- representam relações entre objetos

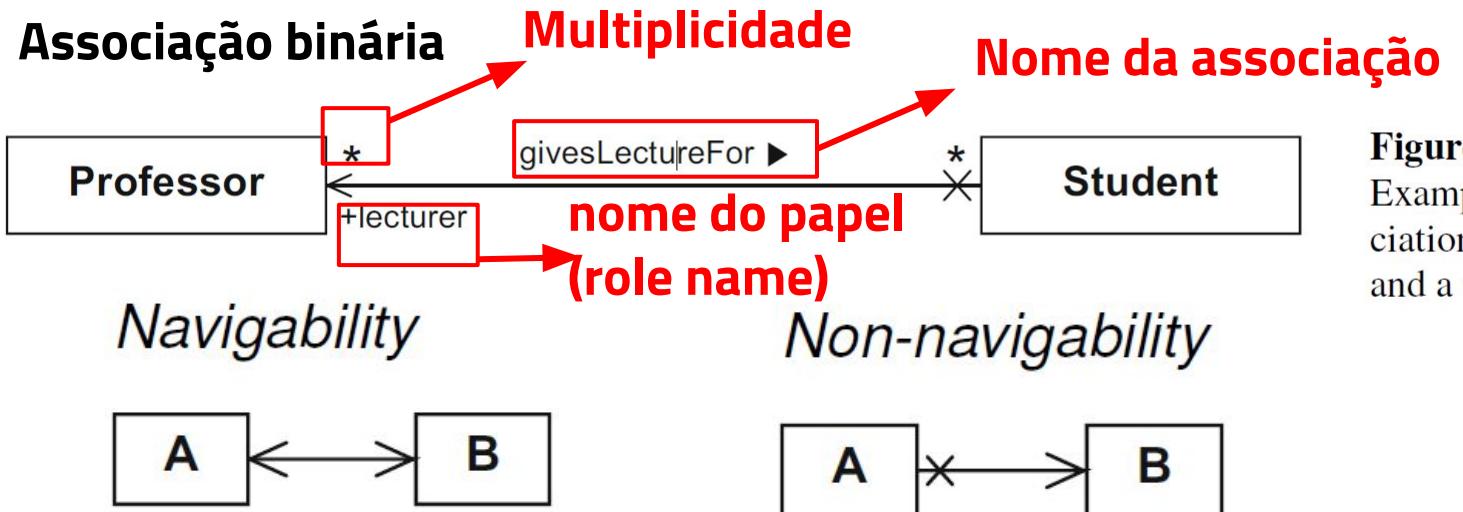
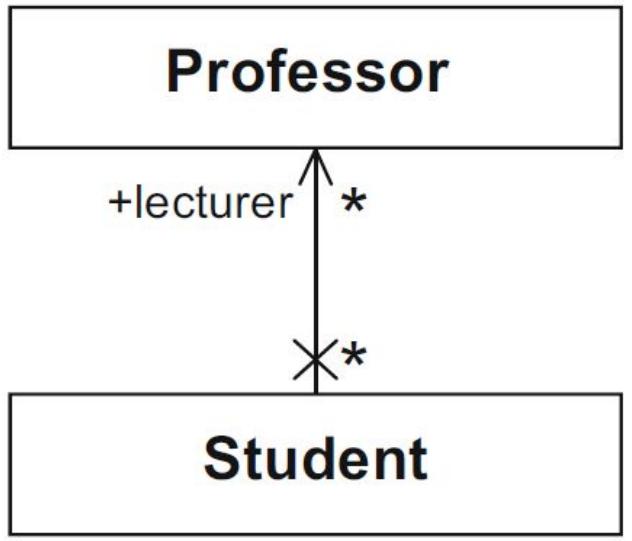


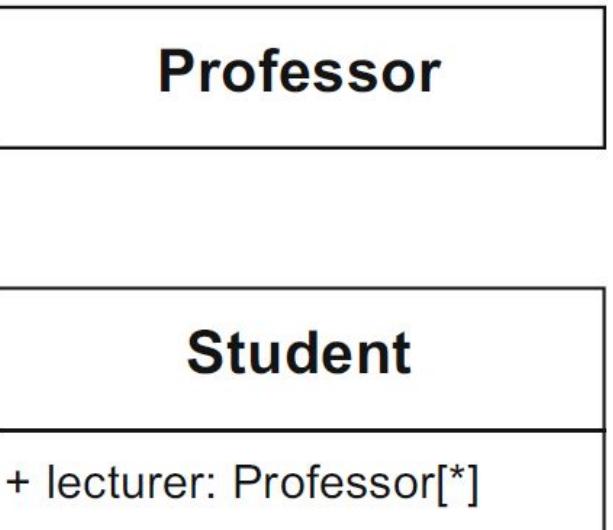
Figure 4.11
Example of a binary association in a class diagram
and a valid object diagram

Relacionamentos: Associação

- **Associação**

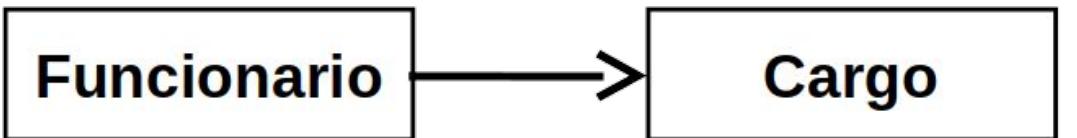


(a)

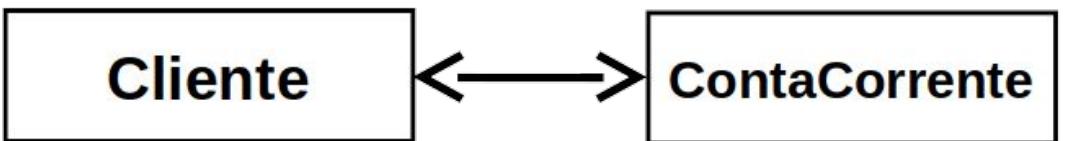


(b)

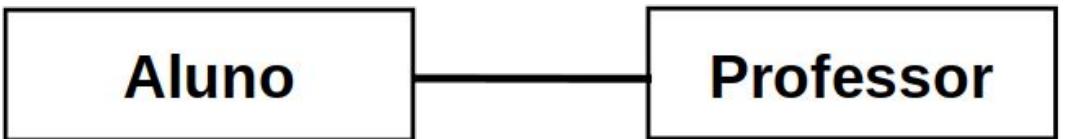
Associação: Navegabilidade



unidirecional



bidirecional



bidirecional
ou
não determinada

Associação: Multiplicidade

Muitos/Vários/Zero, um ou mais *

Muitos/Vários/Zero, um ou mais 0...*

 Um ou mais 1...*

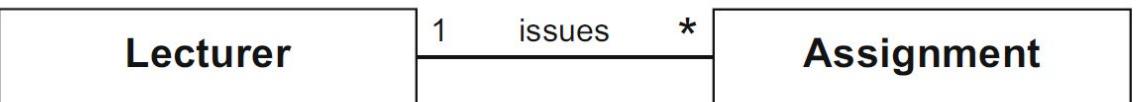
 Zero ou um 0..1

 Exatamente um 1

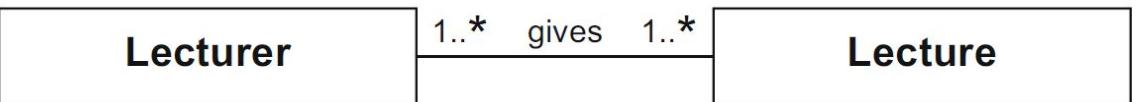
 Faixa especificada 2..4, 6..8



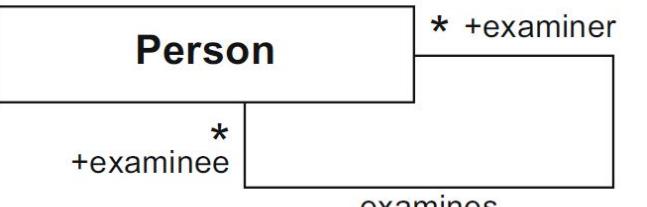
Associação: Multiplicidade



(a)



(b)



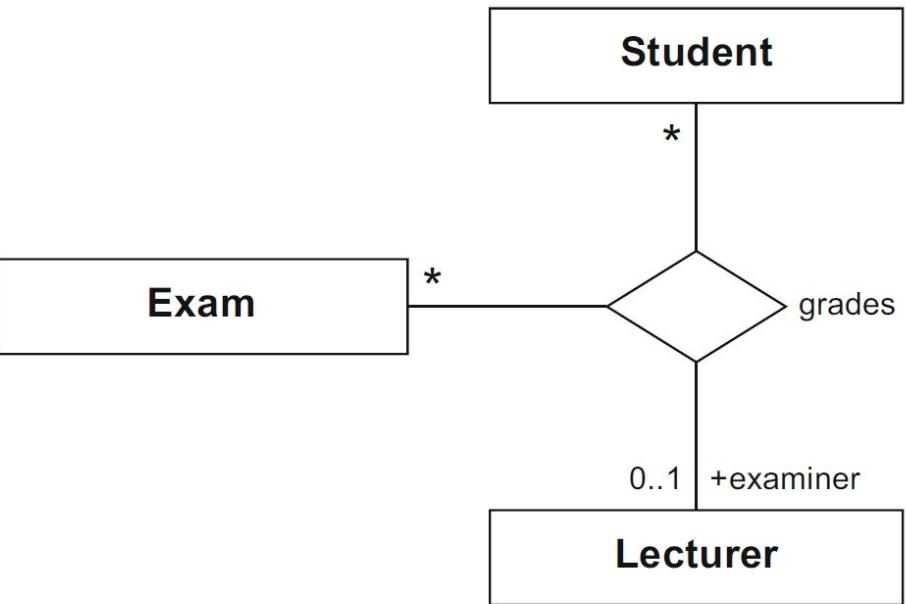
(c)

Associação: N-ária (ternária, quaternária...)

- Ligações não direcionais entre todas as partes

Figure 4.15

Example of n-ary (here ternary) association ...

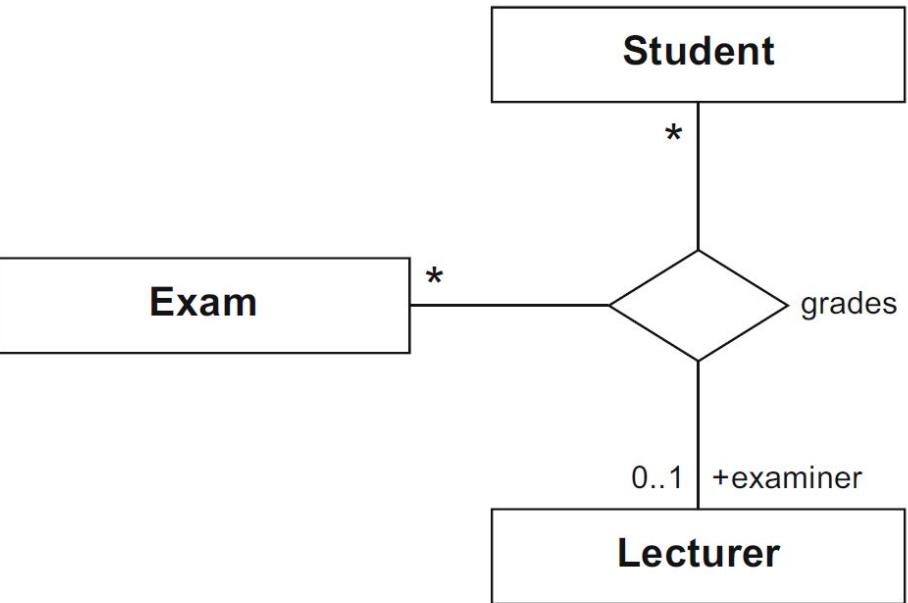


Associação: N-ária (ternária, quaternária...)

- Ligações não direcionais entre todas as partes

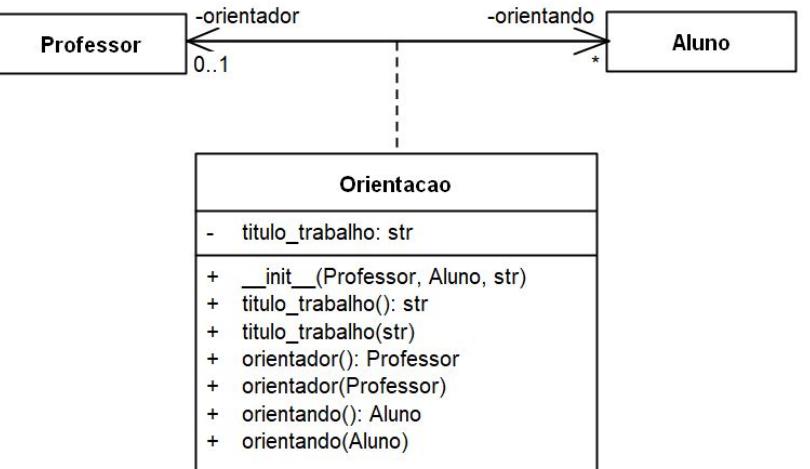
Figure 4.15

Example of n-ary (here ternary) association ...



Associação: Classes Associativas

- Permitem adicionar atributos e operações a relacionamento entre classes
- Útil para implementar relações bidirecionais (**consistência**)



Associação: Consegue ler o diagrama?

Figure 4.18

Examples of association classes

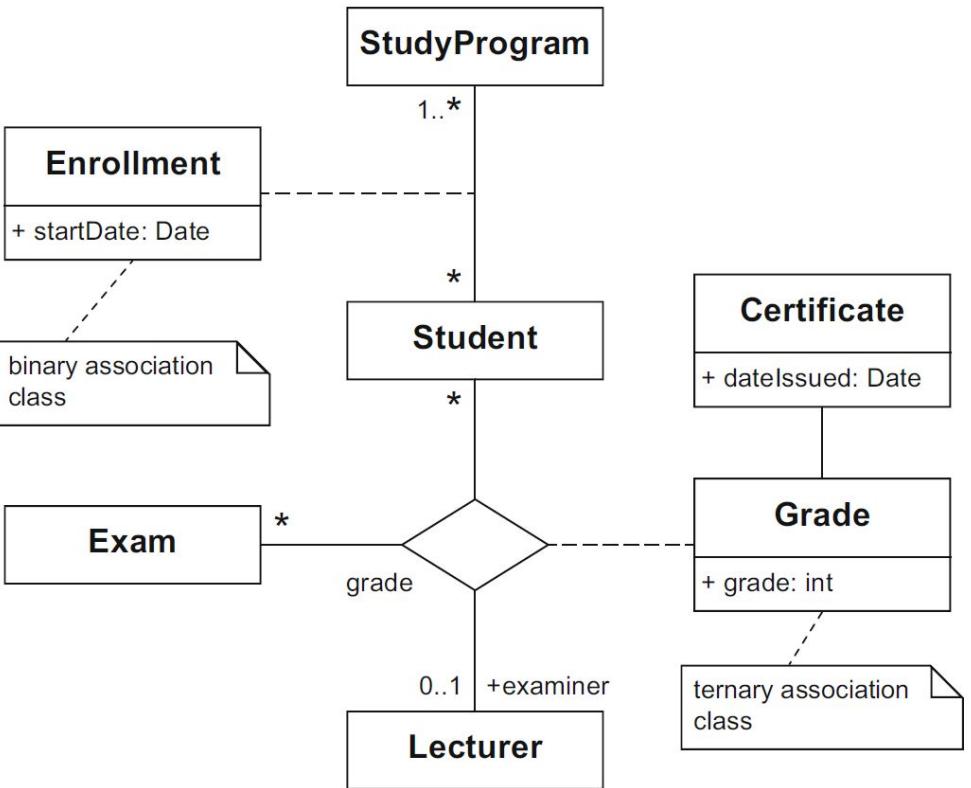
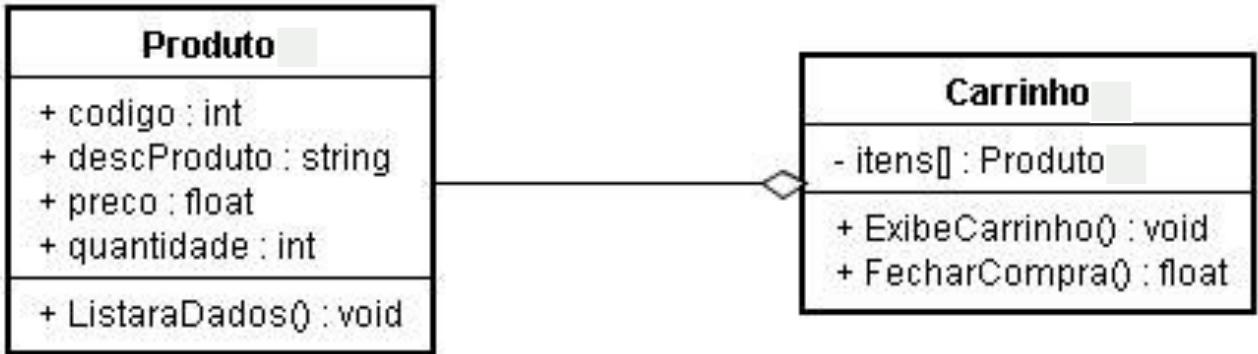


Diagrama de Classes: Relacionamentos

- Principais tipos de relacionamento
 - Associação
 - **Agregação e Composição**
 - Generalização
 - Dependência

Associação: Agregação

- Tipo específico de associação
- Objetos da classe B são partes da classe A
- Partes (classe B) podem existir sem o todo (classe A) - (**compartilhados**)
- Representado por losango **não-preenchido** na conexão



Associação: Composição

- Tipo específico de associação
- Objetos da classe B são partes da classe A
- Partes (classe B) **NÃO EXISTEM** sem o todo (classe A)
- Representado por losango **preenchido** na conexão



Diagrama de Classes: Relacionamentos

- Principais tipos de relacionamento
 - Associação
 - Agregação
 - Composição
 - **Generalização**
 - Dependência

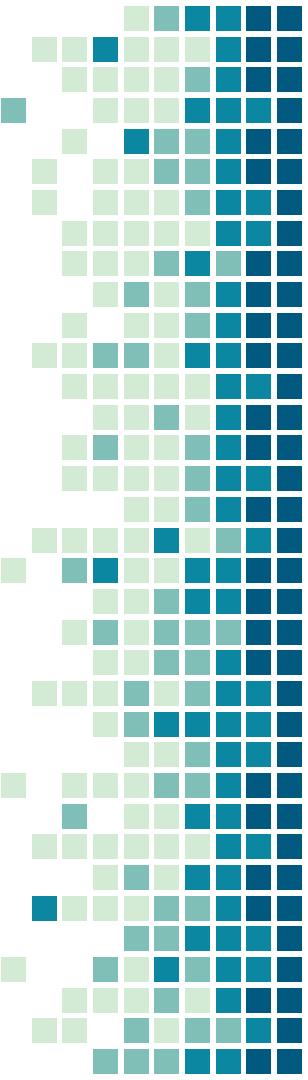
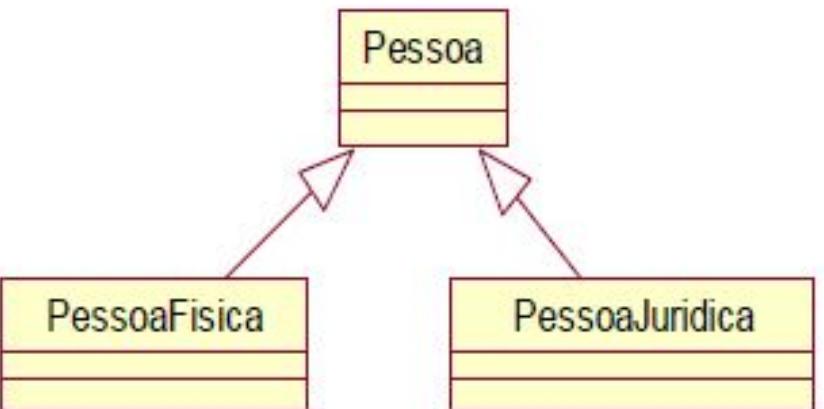
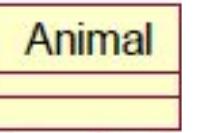


Diagrama de Classes: Generalização (herança)

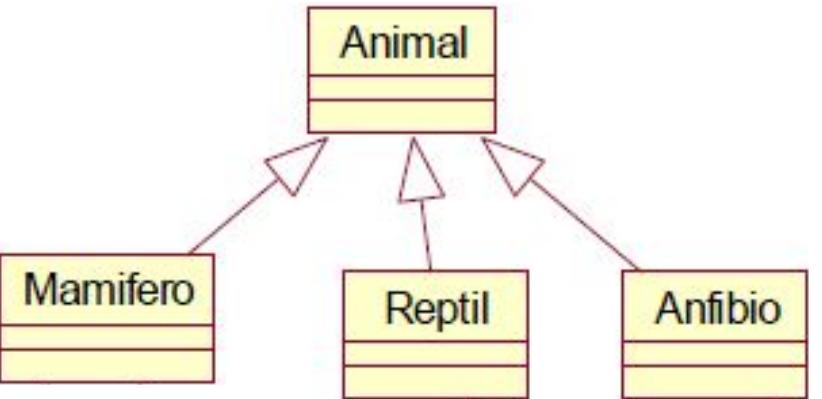
- Expressa características e associações comuns para classes gerais
- Classe B é um tipo de Classe A
 - Classe B é uma especialização da classe A
 - Classe A é uma generalização da Classe B



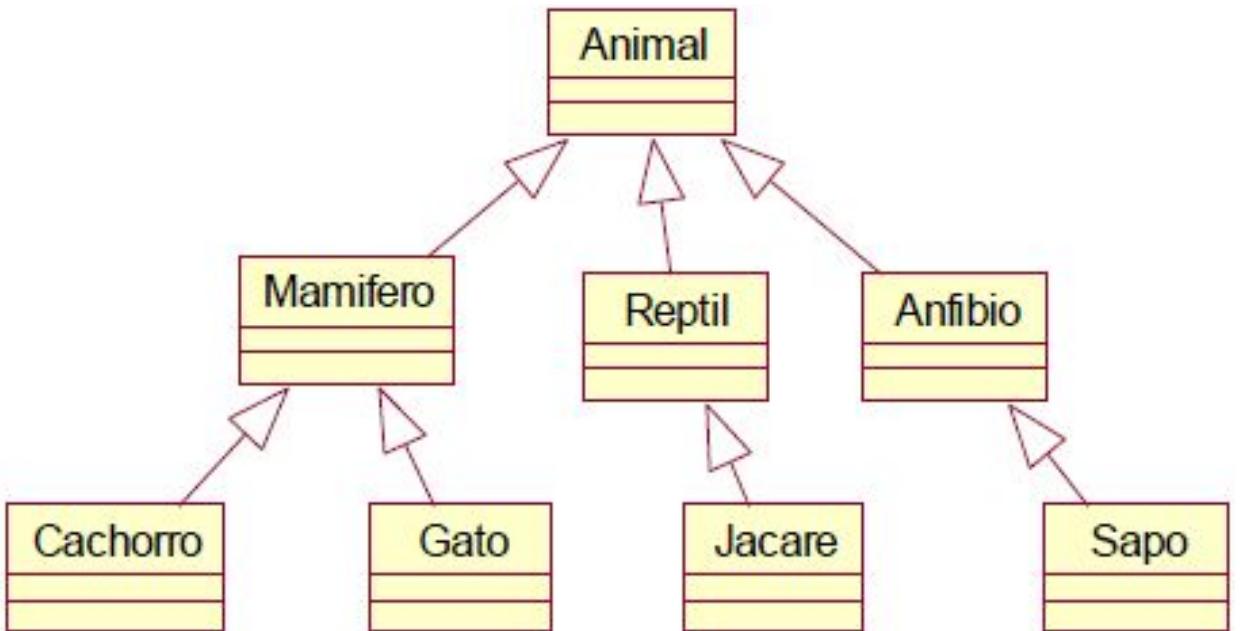
Generalização (herança): Exemplo



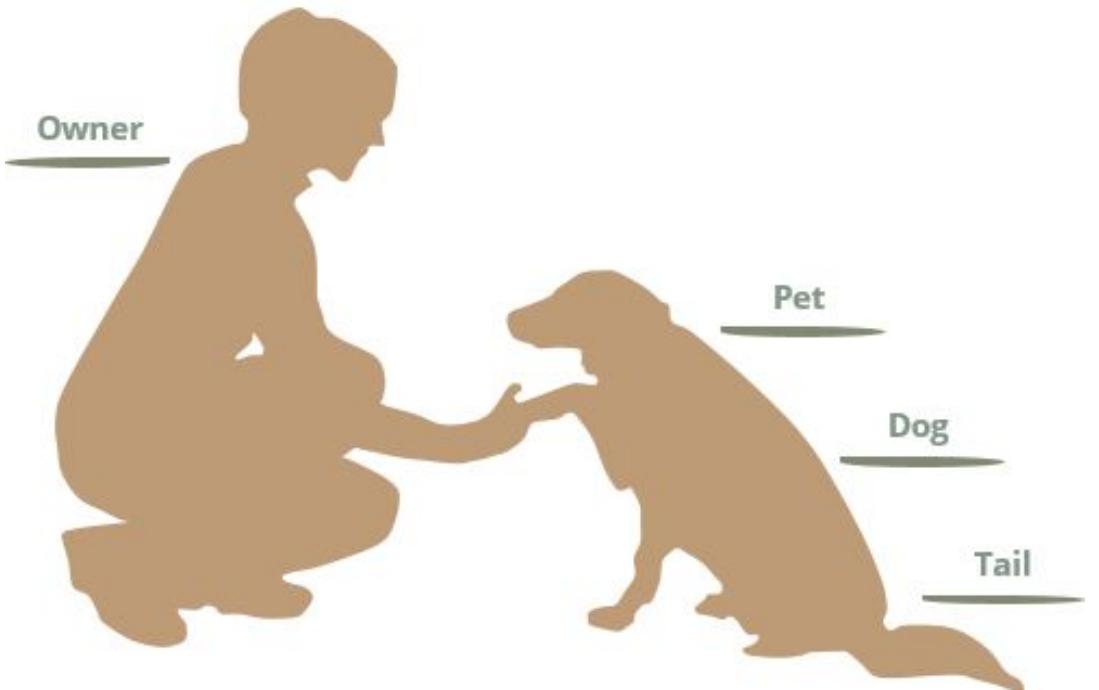
Generalização (herança): Exemplo



Generalização (herança): Exemplo

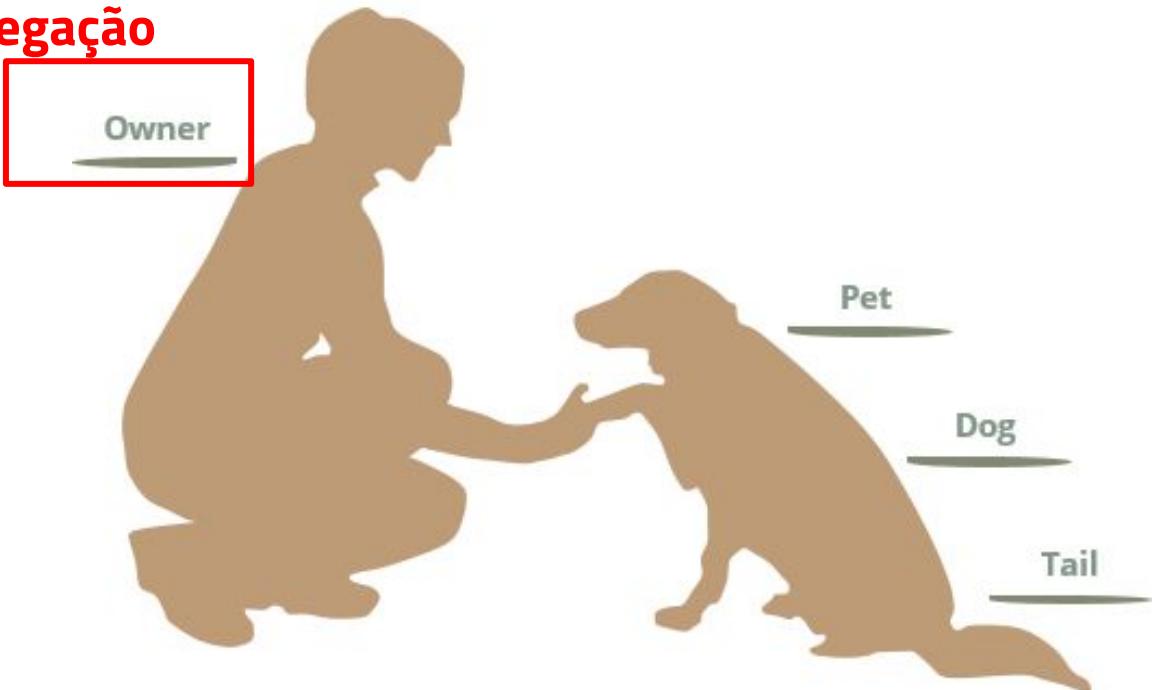


Agregação x Composição x Generalização

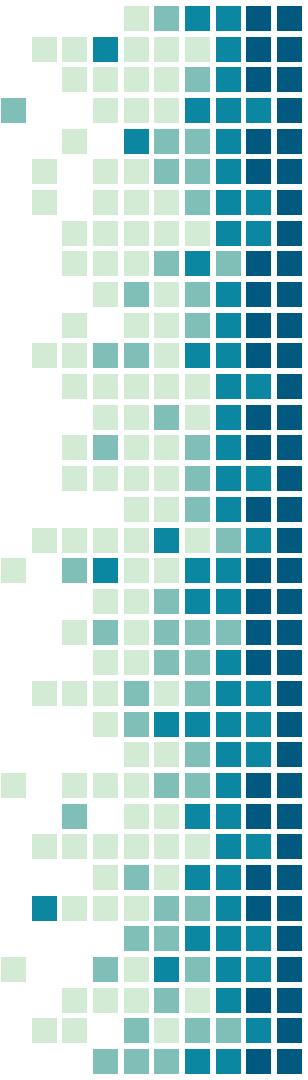
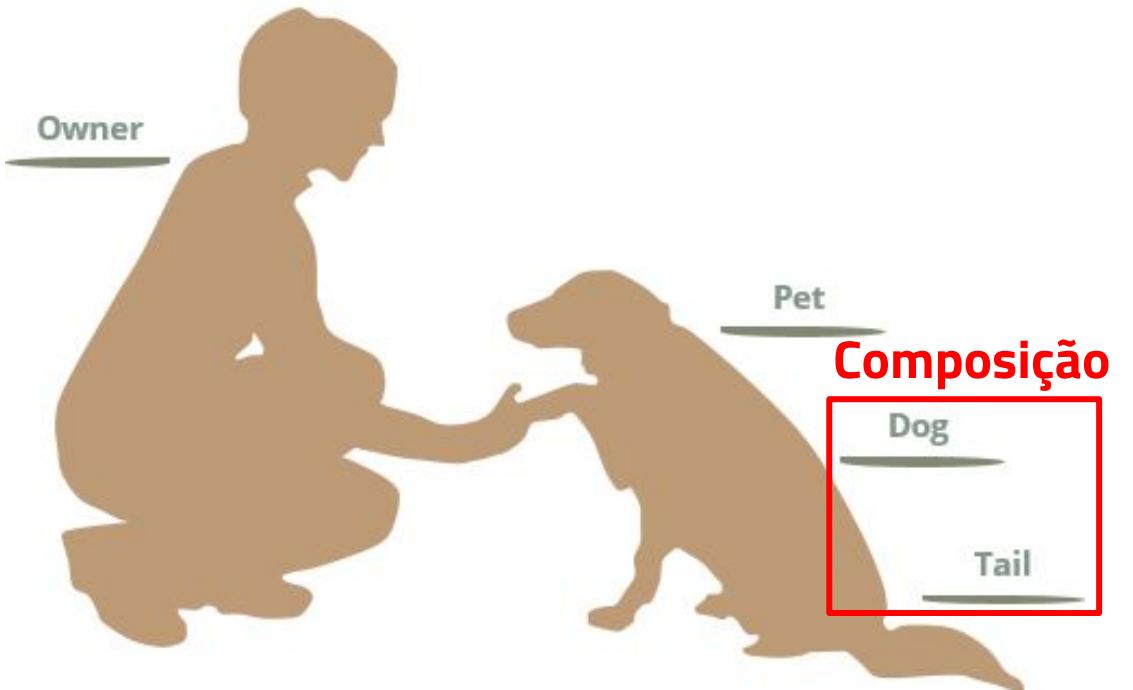


Agregação x Composição x Generalização

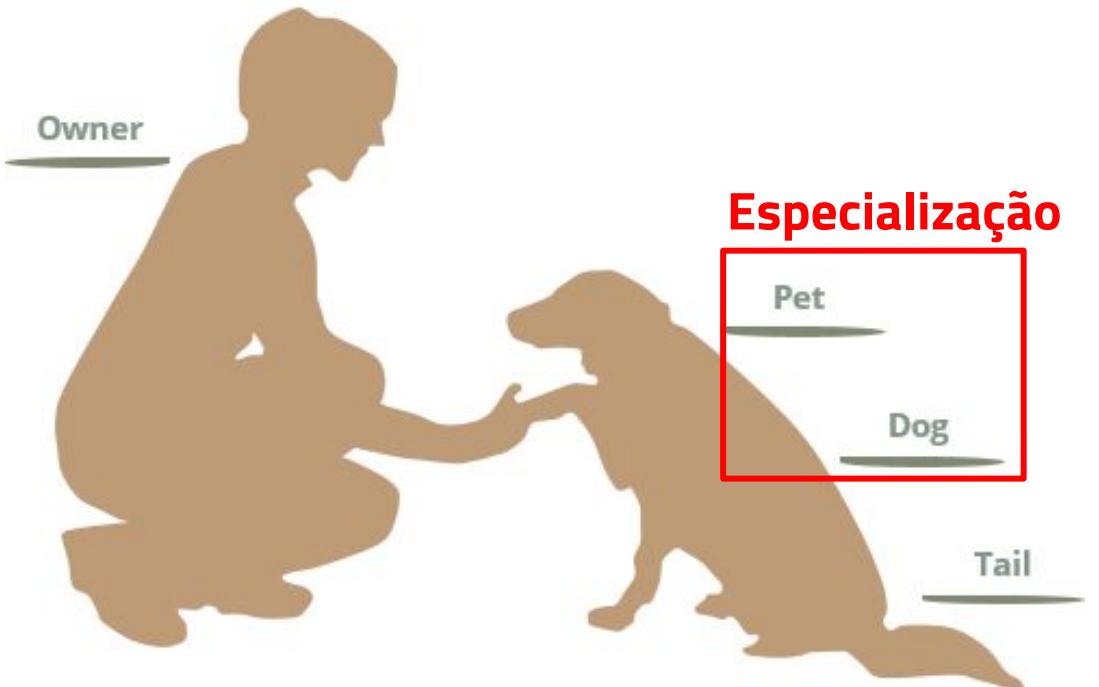
Agregação

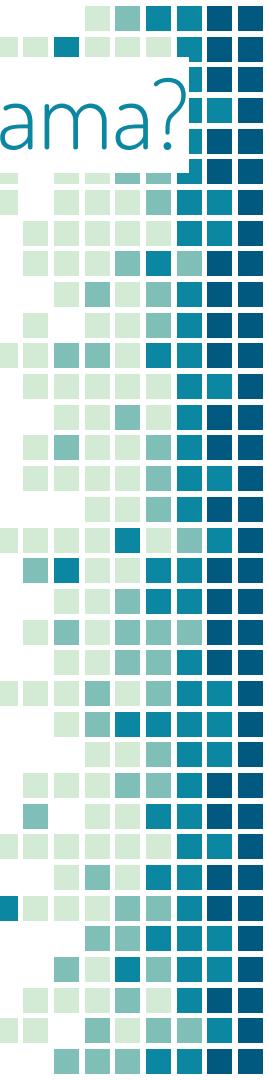


Agregação x Composição x Generalização



Agregação x Composição x Generalização





Relacionamentos: Consegue entender o diagrama?

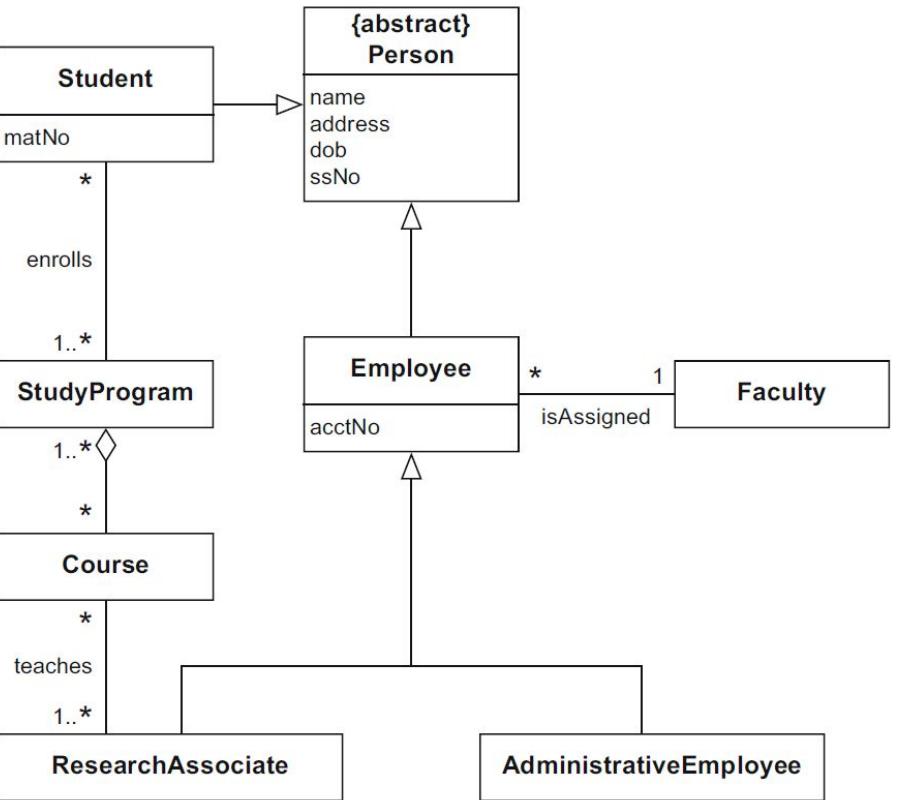


Diagrama de Classes: Relacionamentos

- Principais tipos de relacionamento
 - Associação
 - Agregação
 - Composição
 - Generalização
 - **Dependência**

Relacionamentos: Dependência

- Um objeto **depende** de outro (utilização)
 - representada por linha pontilhada com seta simples

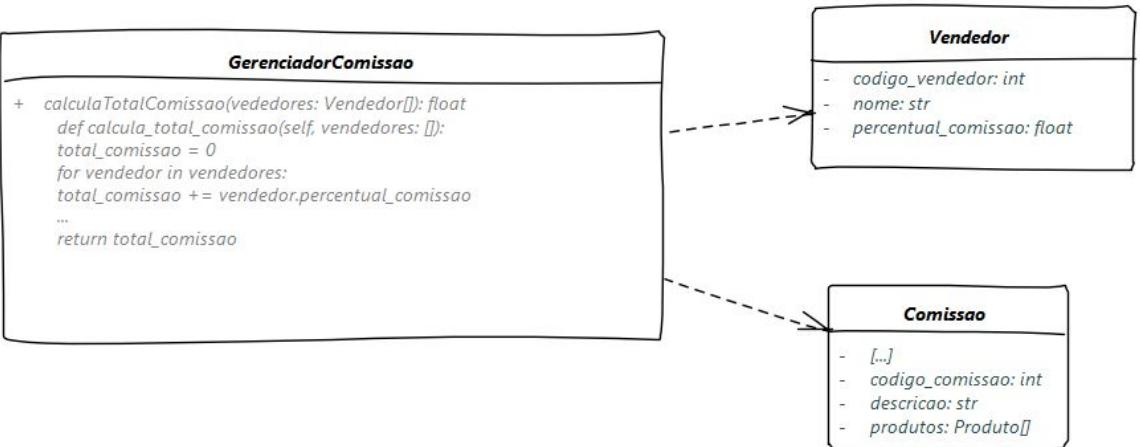


Diagrama de Classe: Relações entre Classes

Grau de acoplamento

Generalização:



Composição:



Agregação:



Grau de Acoplamento

Associação:



Dependência:

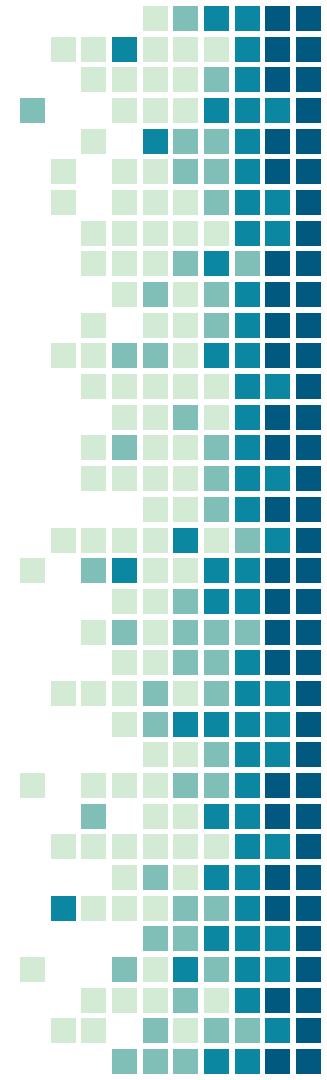


Diagrama de Classe: Relações entre Classes

Resumo

Generalização:



A é um tipo de B

Composição:



A contém B, A é composto por B
B é propriedade exclusiva de A
A instancia e destrói B

Aggregação:



A contém B, A é formado por B
B é compartilhável

Associação:



A tem atributo
do tipo B

Dependência:



Dependência de A para B
não é permanente

Referências

SEIDL, M. et al. **UML@ Classroom: An Introduction to Object-Oriented Modeling**. Cham: Springer, 2015.