Introdução à UML e Diagrama de Classes

1

O que é UML ?

- A UML (*Unified Modeling Language*) é uma linguagem para **especificação**, **documentação**, **visualização** e **desenvolvimento** de sistemas orientados a objetos;
- É possível representar sistemas de softwares sob **diversas perspectivas** de visualização;
- Facilita a comunicação de todas as pessoas envolvidas no processo de desenvolvimento de um sistema.

2

2

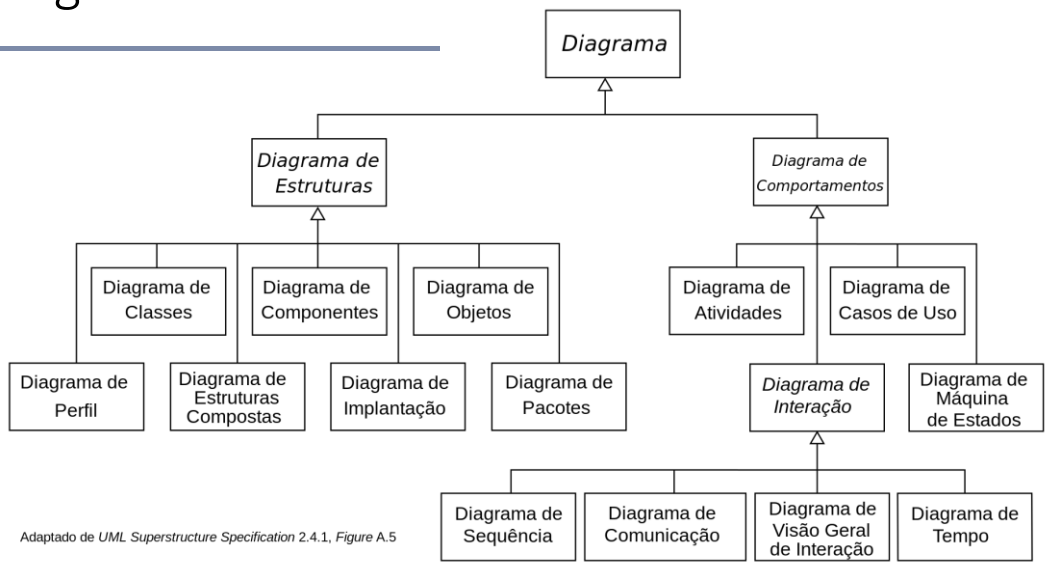
O que é UML?

- Foi criada pala unificação dos modelos de Booch, Rumbaugh e Jacobson;
- O trabalho de unificação começou em 1994;
- A UML 1.0 foi padronizado pela OMG em 1997;
- A versão UML 2.0 foi disponibilizada no início de 2005 (também padronizada pela OMG);
- Versão atual é a 2.4.1 de 2011.

Propósito de um Modelo

- Ajuda o desenvolvedor a visualizar como o sistema é ou como ele deverá ser;
- Permite que o desenvolvedor especifique a estrutura e comportamento do sistema;
- Dá ao desenvolvedor um *template* que o guia na construção do sistema;
- Os modelos servem como documentação das decisões tomadas.

Diagramas da UML 2.0



Adaptado de *UML Superstructure Specification* 2.4.1, *Figure* A.5

5

5

Diagrama de Classes

- São diagramas mais comuns encontrados em modelagem de sistemas orientados a objetos;
- Eles mostra uma visão estrutural do sistema, contendo suas **classes**, **interfaces e pacotes,** incluindo os **relacionamentos** entre eles;
- **Seu principal uso** é modelar a visão **estática** do **projeto** de um sistema.

6

6

Conceitos de Diagramas de Classes

- Uma classe descrição um molde para criação de objetos que compartilham os mesmos: atributos, métodos e relacionamentos.
- Graficamente, uma classe é desenhada como um **retângulo** composto de três partes:
  - Nome;
  - Atributos;
  - Métodos.

7

7

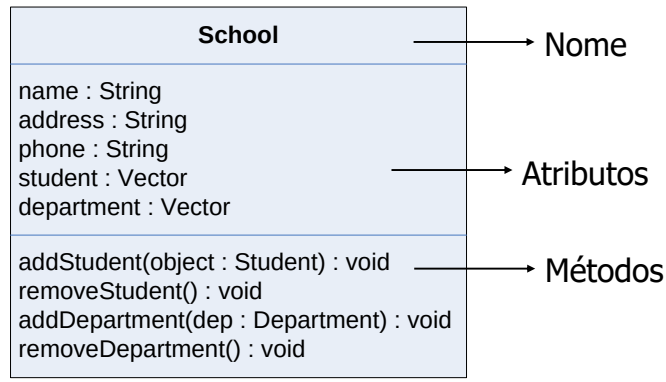
Nome, Atributo e Método

- **Nome**: Toda classe deve ter um nome único dentro do sistema;
- **Atributo**: cada atributo armazena o valor de uma das **propriedades** que descrevem o **estado** do objeto;
- **Método**: é a implementação de um **comportamento** provido pelo objeto que pode ser utilizado para alterar seu **estado** e/ou possibilitar a **comunicação** entre os objetos.

8

8

Exemplos de Classe em UML 2.0



9

9

Notações de Encapsulamento em UML 2.0

- Encapsulamento:

Público (+): Visível para qualquer elemento que possa ver a classe;

Protegido (#): Visível a outros elementos dentro da classe e de subclasses;

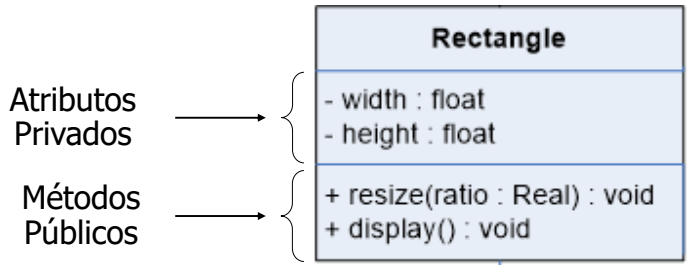
Privado (-): Visível a outros elementos dentro da classe;

Pacote Protegido (~): Visível a elementos do mesmo pacote.

10

10

Exemplo de Encapsulamento



11

11

Relacionamento

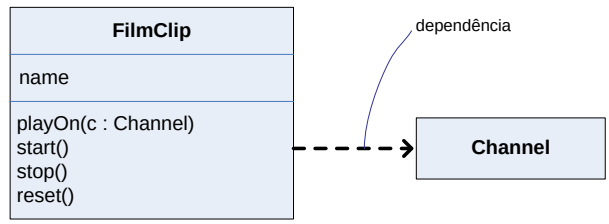
- Relacionamentos representam as formas com as quais as entidades se conectam umas com as outras;
- Em Orientação a Objetos, há três tipos de relacionamentos:
  - Dependências
  - Generalizações
  - Realizações
  - Associações
    - Associação de Navegação
    - Agregação
    - Composição

12

12

Relacionamento: Dependência

- **Dependência** é um relacionamento que mostra que uma classe **usa informações ou operações de outra classe,** acessando-a por meio de uma **variável ou parâmetro**.
- Por exemplo, a classe *FilmClip* usa informações ou operações da classe *Channel* como parâmetro do método *playOn(c: Channel)*.

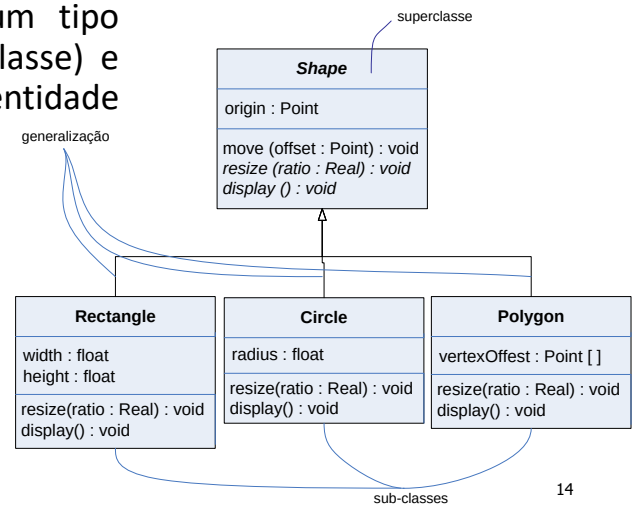


13

13

Relacionamento: Generalização

- É um relacionamento entre um tipo geral de uma entidade (superclasse) e um tipo mais específico desta entidade (subclasse).

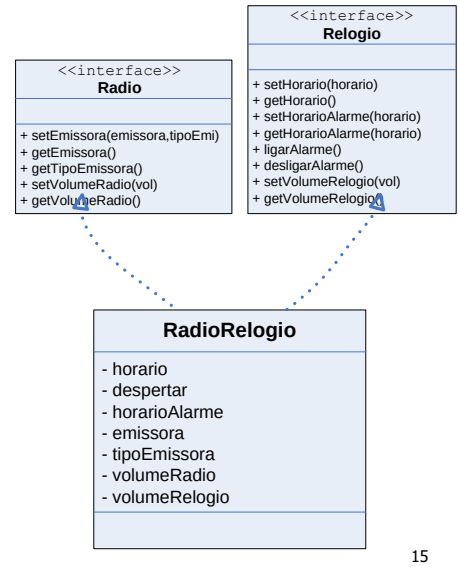


14

14

Relacionamento: Realização

- É um relacionamento utilizado para padronizar comportamentos comuns entre classes diferentes, mesmo que elas não tenham uma mesma superclasse comum.

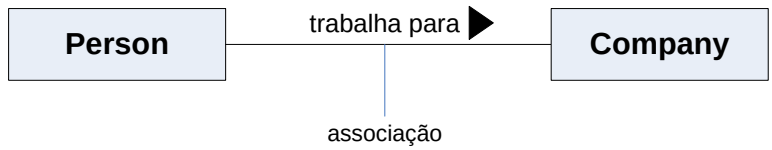


15

15

Relacionamento: Associação

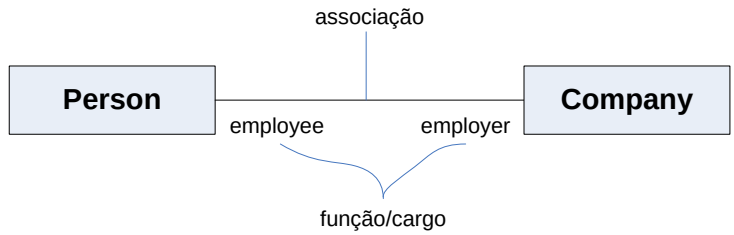
- É um relacionamento que especifica que objetos de uma classe se comunicam (trocam mensagens) com objetos de outra classe.
- Dada uma associação entre duas classes, é possível que objetos de uma classe interajam com objetos da outra classe.



16

16

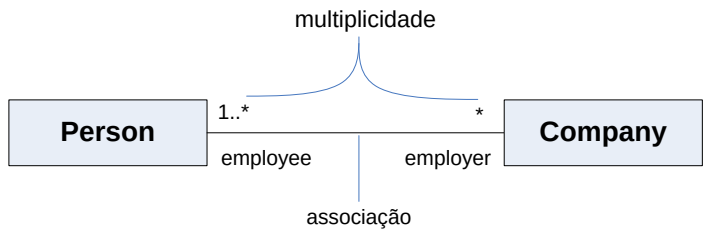
Relacionamento: Associação (função)



17

17

Relacionamento: Associação (cardinalidade)



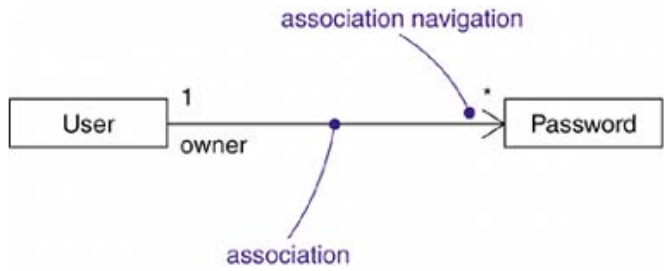
* significa significa **0..***

18

18

Relacionamento: Associação de Navegação

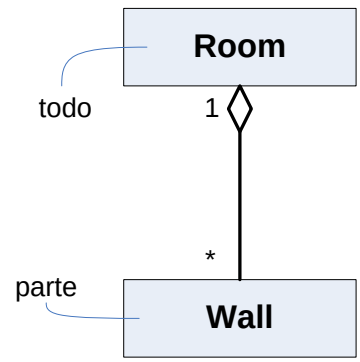
- É implementada quando objetos de um tipo podem interagir com objetos de outro tipo, mas não o contrário.



19

Relacionamento: Agregação

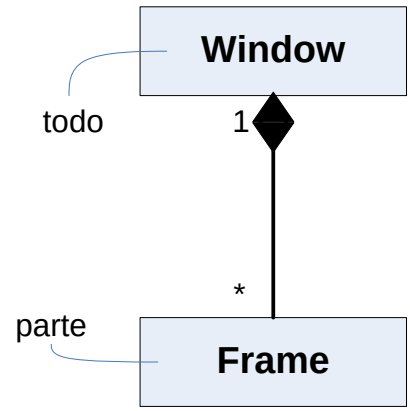
- Agregação é um **tipo especial** de associação.
- Modela um relacionamento do tipo "todo/parte", em que uma classe representa uma entidade completa (todo), composta de outras entidades (partes).



20

Relacionamento: Composição

- Composição é uma **forma de agregação**, contudo dá o sentido de forte posse e tempo de vida entre **parte** e o **todo**.
- O **todo** é responsável pela **disposição** de suas partes, ou seja, que o todo deve gerenciar a **criação** e **destruição** de suas partes.
- Um objeto pode ser uma parte de **somente** uma composição **por vez**.

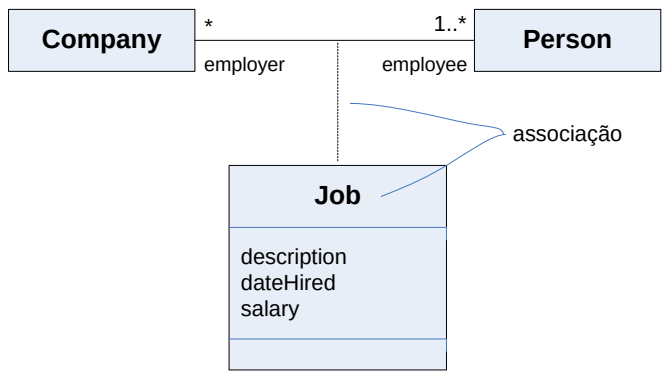


21

21

Relacionamento: Associação com propriedades

- Em uma associação entre classes onde a própria associação pode ter propriedades.



22

22

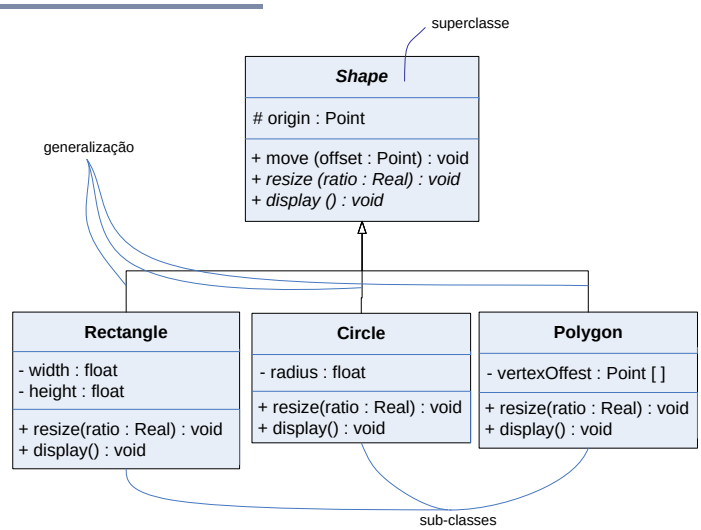
Classes e Métodos Abstratos

- Como expressar **classes abstratas**:
  - O nome da classe deve ser escrito no **estilo itálico**.

- Como expressar **métodos abstratos**:
  - O nome do método deve ser escrito no **estilo itálico**.

23

Exemplo de classe e método abstrato



24