# 5408 Sistemas de informação – conceção

Sessão 3

21/02/2024

Margarida Fachada

# Diagrama de Classes

- Um diagrama de classes serve para modelar o vocabulário de um sistema
- Construído e refinado ao longo das várias fases do desenvolvimento do software, por analistas, projetistas (designers) e implementadores
- Está entre os tipos mais úteis de diagramas UML pois mostra de forma clara a estrutura de um determinado sistema.
- Também servem para:
  - Especificar colaborações (no âmbito de um caso de uso ou mecanismo)
  - Especificar esquemas lógicos de bases de dados
  - Especificar vistas (estrutura de dados de formulários, relatórios, etc.)

# Diagrama de Classes - Objeto

- Um objeto é uma entidade ou conceito existente no contexto de modelação sendo relevante para ser incorporado no modelo de informação. É caracterizado por um conjunto de propriedades, um comportamento e uma identidade.
- Propriedades: características que definem o objeto, transpostas para um conjunto de atributos, cujos valores estabelecem o Estado do objeto.
- Comportamento: operações que o objeto pode efetuar.
- Identidade: permite identificar um objeto em particular como único num conjunto de objetos semelhantes.

# Diagrama de Classes - Objetos

- Por exemplo, podemos ter os seguintes objetos:


- O carro A é diferente dos carros B e C, contudo, todos eles possuem um conjunto de atributos (Nº serie, cor, data do fabrico, etc) que os definem (Estado) e realizam operações como Iniciar Marcha ou Acelerar (comportamento).
- Para além de algumas semelhanças, possuem uma identificação própria que os torna únicos.

# Diagrama de Classes - Objetos

- A mesma caracterização pode ser aplicada para os clientes de uma loja, por exemplo:


- Os diversos objetos cliente também partilham as mesmas propriedades como o Nome, Morada, Data Nascimento, Nº CC, Sexo, etc.
- O mesmo para o comportamento, pois todos têm que se registar como clientes (registar) ou podem atualizar os seus dados pessoais (alterarDados).

# Diagrama de Classes - Classes

- Representa uma abstração sobre um conjunto de objetos que partilham a mesma estrutura e comportamento.
- Na prática, um objeto é um caso particular de uma classe, também referido como uma instância da classe.
- No ex. anterior poderíamos resumir os diferentes objetos num conjunto de propriedades e operações comuns

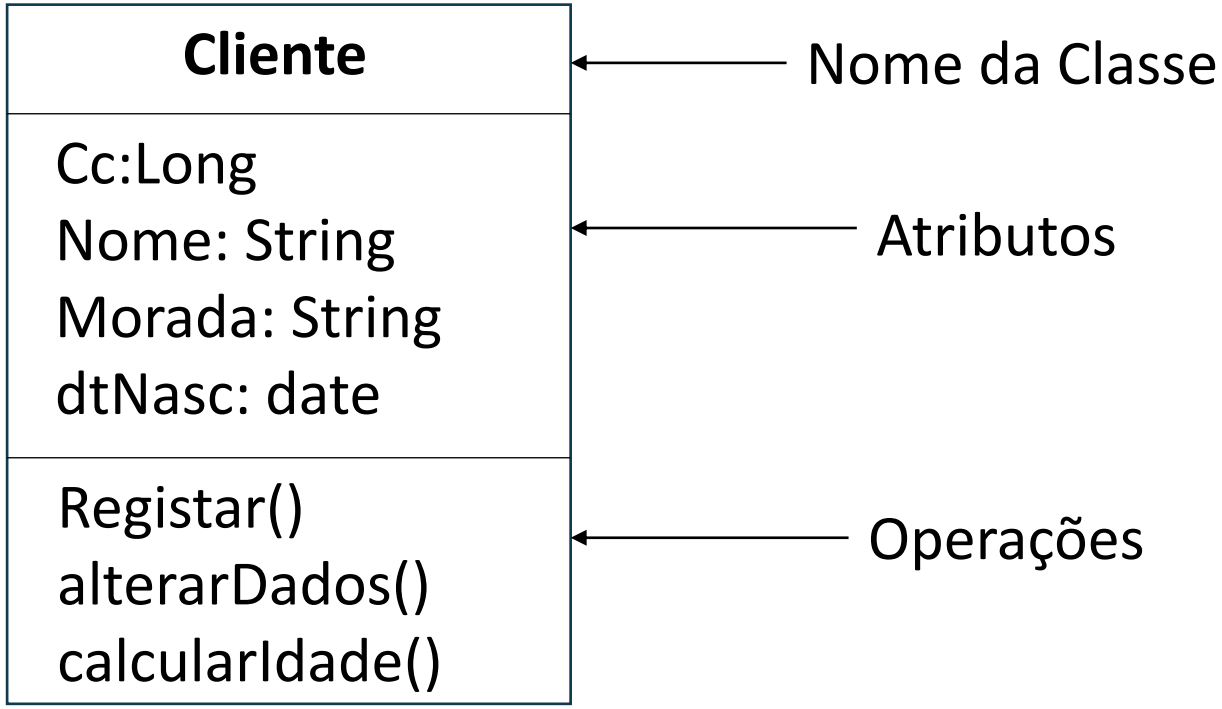
| Cliente X: Cliente |
| --- |
| CC:1242454<br>Nome=João<br>Morada=Rua X<br>datNasc=03/08/74 |

| Cliente Y: Cliente |
| --- |
| CC:1233424<br>Nome=Maria<br>Morada=Rua Y<br>datNasc=05/06/95 |

| Cliente Z: Cliente |
| --- |
| CC:15225457<br>Nome=Marco<br>Morada=Rua X<br>datNasc=26/09/99 |

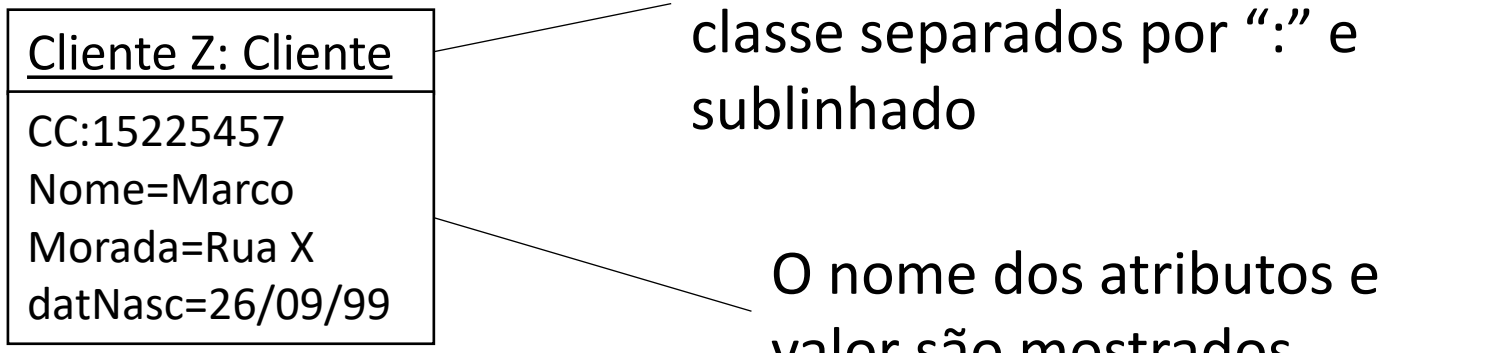
# Diagrama de Classes - Classes

- Sendo descritos na seguinte classe Cliente:



# Diagrama de Classes - Objetos

- As instâncias da classe (objetos) podem ser representadas da seguinte forma:



# Diagrama de Classes – Atributos e Operações

- Um atributo é uma característica que os objetos possuem e que é representada por um valor de dados.
- Por exemplo: o atributo Cor poderá ser igual a “Vermelho” ou “Azul”.
- Os nomes dos atributos são únicos numa classe, não podendo existir na mesma classe 2 atributos com o mesmo nome.
- Em classes diferentes podem existir atributos com nomes iguais.
- Os objetos apenas comunicam entre si por **mensagens** que na prática resulta na invocação de operações.

# Diagrama de Classes – Atributos

- As operações são a representação logica do comportamento de um objeto, consistindo em ações efetuadas por ou sobre um objeto.
- Por ex: na classe Cliente pode-se definir a operação calcularIdade ()
- A UML utiliza () para simbolizar a existência ou não de parâmetros.
- Para a operação anterior poderia ser necessário fornecer a data atual, o que implicaria definir a operação como calcularIdade(dtAtual).
- Uma operação também pode possuir um valor de retorno, que no ex anterior corresponderia ao retorno da idade do cliente.

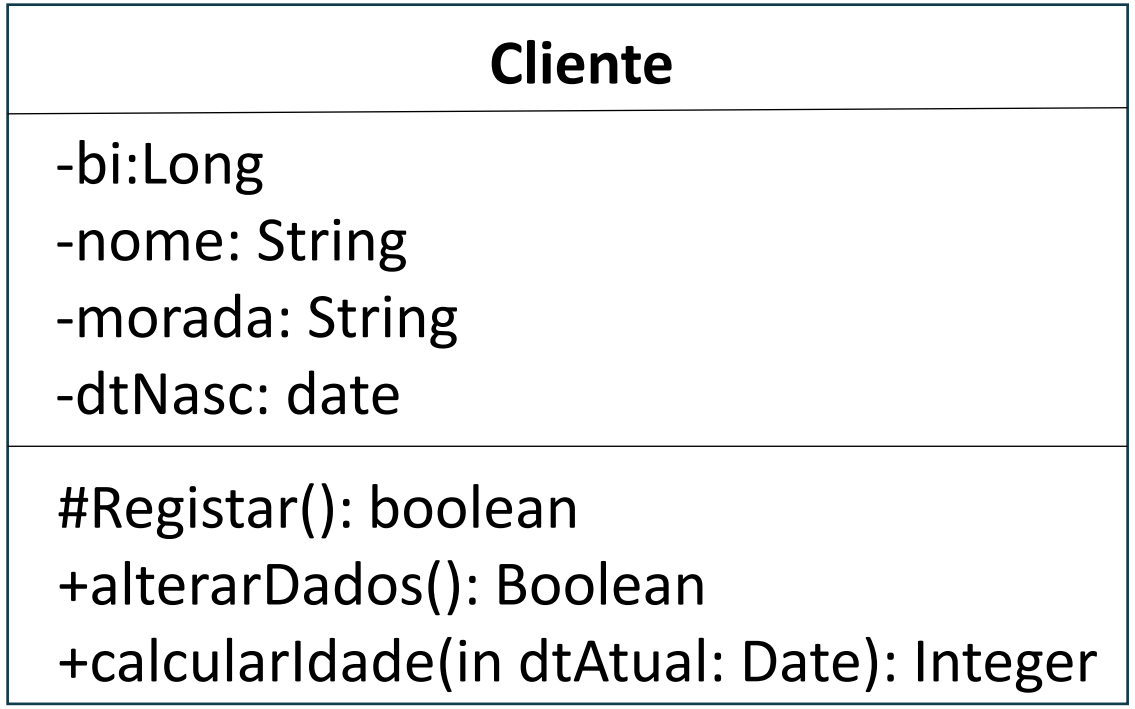
# Diagrama de Classes – Tipos de dados básicos

- Para cada atributo também pode ser identificado o seu tipo de dados, que caracteriza a informação que o atributo irá conter.
- Os tipos de dados disponíveis dependem diretamente da linguagem de programação em que o sistema será desenvolvido.
- Contudo, é possível restringirmo-nos ao seguinte conjunto de tipos básicos:

| Tipo | Aplicação |
|---|---|
| Integer | Representa um nº inteiro |
| Long | Representa um nº inteiro mas de maior dimensão |
| Double | Para números reais |
| String | Representa texto |
| Date | Para datas |
| Boolean | Valor lógico que representa Verdade ou Falso. |

# Diagrama de Classes – Tipos de dados básicos

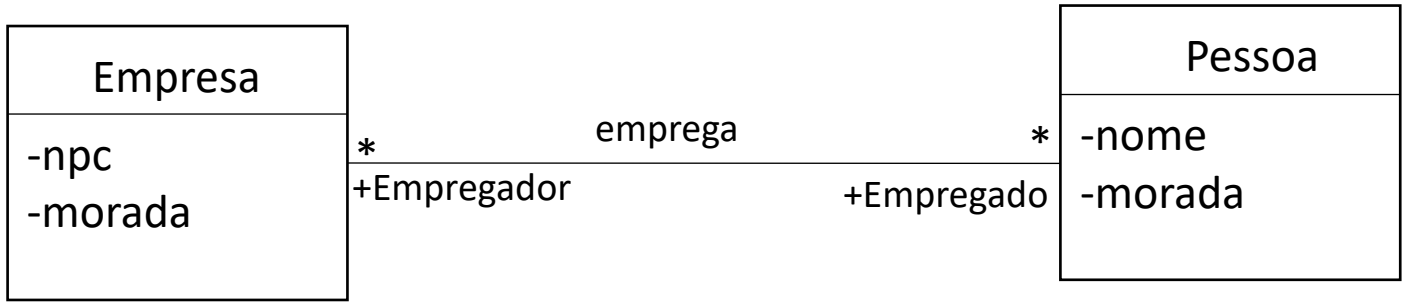
- As operações também podem possuir um tipo de dados para os seus argumentos e para o resultado da operação.




Para a operação calcularIdade é necessário fornecer a data atual (parâmetro dtAtual), logo o seu tipo de dados é Date.
A operação devolver a idade do cliente, logo o tipo de dados é um nº inteiro (Integer).

# Diagrama de Classes – Associações

- No diagrama de classes, as associações representam as relações entre os objetos.
- No prox. ex. temos um objeto da classe Pessoa que pode trabalhar em muitas empresas e, por sua vez, uma empresa pode empregar muitas pessoas.
- As associações são caracterizadas por possuir um nome e quando necessário podem incluir o papel que os objetos têm na relação.
- Por ex:

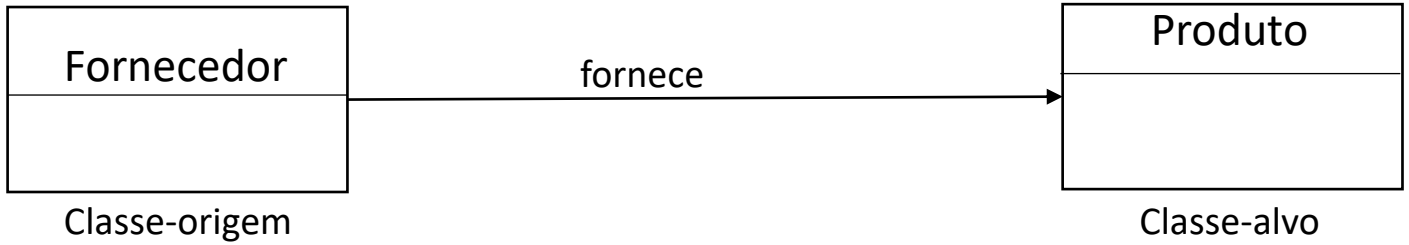



O papel de uma pessoa é ser o Empregado, enquanto que o papel de uma empresa é ser o Empregador.

**O nome das associações corresponde normalmente a verbos e possui um tamanho reduzido.**

# Diagrama de Classes – Multiplicidade

- As associações são também caracterizadas por possuir uma multiplicidade, que indica quantos objetos participam na relação.
- A multiplicidade diz respeito não apenas à quantidade de ocorrências de uma entidade em relação a outra, mas também à questão de obrigatoriedade da participação.
- Consideremos por exemplo, as entidades Fornecedor e Produto, bem como a relação entre elas traduzido por "fornece":

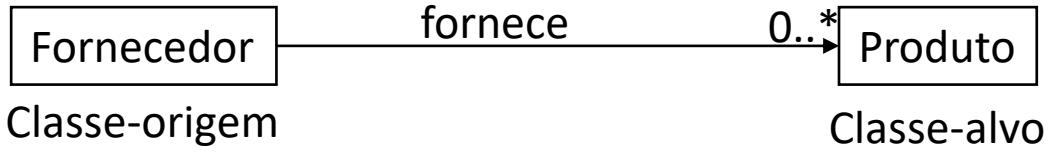


# Diagrama de Classes – Multiplicidade

- Coloca-se em saber quais as quantidades mínima e máxima de ocorrências da classe-alvo em relação à classe origem.
- Quanto à quantidade mínima, a resposta pode ser 0 ou 1.
- Quanto à quantidade máxima, a resposta pode ser 1 ou vários.
- No nosso ex, a questão coloca-se assim: Qual o mínimo e o máximo de produtos (classe-alvo) que um fornecedor (classe-origem) pode fornecer?
    1. quanto à quantidade mínima – um fornecedor por fornecer 0 ou terá de fornecer no mínimo, um produto?
    2. Quanto à quantidade máxima – um fornecedor só pode fornecer, no máximo, um produto ou pode fornecer vários?

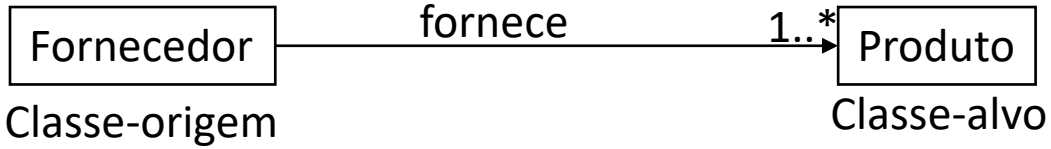
# Diagrama de Classes – Multiplicidade

- O indicativo de multiplicidade de um relacionamento coloca-se junto à classe-alvo e tem sempre dois símbolos:
  - 0..1 (zero ou um) – participação opcional
  - 1..1(um e só um) – participação obrigatória
  - 0..* (zero ou vários) - participação opcional
  - 1..*(um ou vários) - participação obrigatória
- Podemos ter diferentes situações consoante o que se passa na empresa.
- Suponhamos que cada fornecedor pode fornecer **0** ou **vários** produtos.



# Diagrama de Classes – Multiplicidade

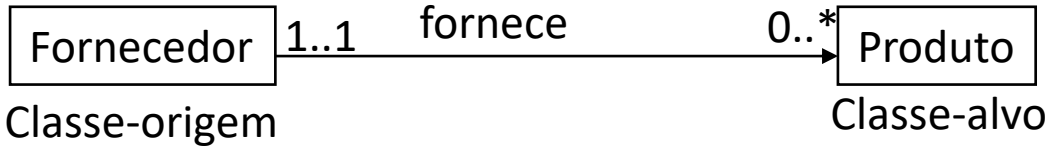
- Mas também podemos admitir que cada fornecedor tem obrigatoriamente de fornecer, pelo menos, um produto, podendo fornecer vários.




- Agora coloquemos a questão inversa: para cada produto, quantos fornecedores pode haver?
- Neste caso, podem ocorrer as 4 situações:
  - 0..1 – cada produto pode ser fornecido por 0 ou no máximo 1 fornecedor;
  - 1..1 – cada produto é fornecido por um e um só fornecedor;
  - 0..* - cada produto pode ser fornecido por 0 ou vários fornecedores;
  - 1..* - cada produto pode ser fornecido por um ou vários fornecedores.

# Diagrama de Classes – Multiplicidade

- Juntando os indicativos de multiplicidade de um lado e de outro, temos muitas situações possíveis.
- Considerando que:
  - Cada fornecedor pode fornecer 0 ou mais produtos;
  - Cada produto pode ser fornecido por um e só um fornecedor.



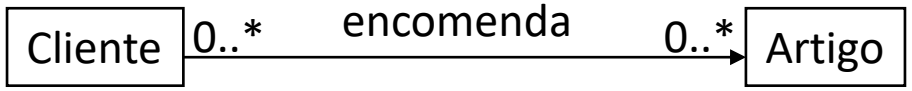
# Tarefa 2

# 1º Diagrama de classes:

- Um cinéfilo, detentor de uma coleção significativa de filmes, pretende uma aplicação que lhe permita armazenar e consulta os filmes.
- Pretende que os filmes possam ser consultados por titulo, género, país de origem, ano de realização, realizador ou atores intervenientes.
- Para além da informação referida, é também necessário saber a duração de cada filme e, caso exista, o endereço de internet dos realizadores e atores.

# Relações representadas por classes associativas

- Para além das **classes/entidades** envolvidas num relacionamento e da respetiva **cardinalidade** ou **multiplicidade**, há ainda a considerar outras situações típicas a ter em conta na elaboração de um **diagrama de classes**.
- Consideremos a parte de um SI de uma empresa que representa a relação entre cliente e produtos ou artigos através das encomendas efetuadas.



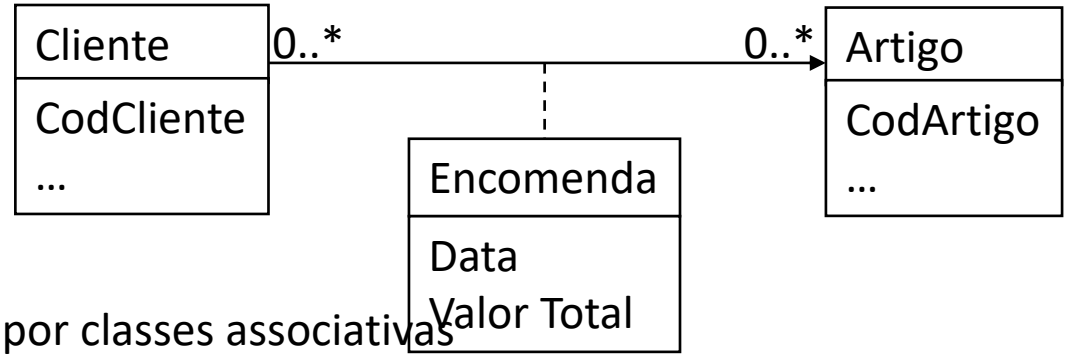

**Este diagrama de classes tem um relacionamento de M:M (0..* para 0..*)**

# Relações representadas por classes associativas

- Cada cliente pode encomendar 0 ou vários artigos (0..*); cada artigo pode ser encomendado por 0 ou vários clientes (0..*).
- Mas neste exemplo, interessa-nos registar dados acerca de cada encomenda efetuada, como por ex, a **data da encomenda** e o seu **valor global**, sem que o possamos fazer em nenhuma das classes originais do diagrama (Cliente e Artigo).
- Há aqui atributos que pertencem a uma classe que não estava considerada à partida e que resulta da relação ou associação entre as duas classes originais e que será a classe Encomenda.
- Em situações como esta, é habitual representar o relacionamento entre as classes originais por uma **classe** chamada **associativa**.

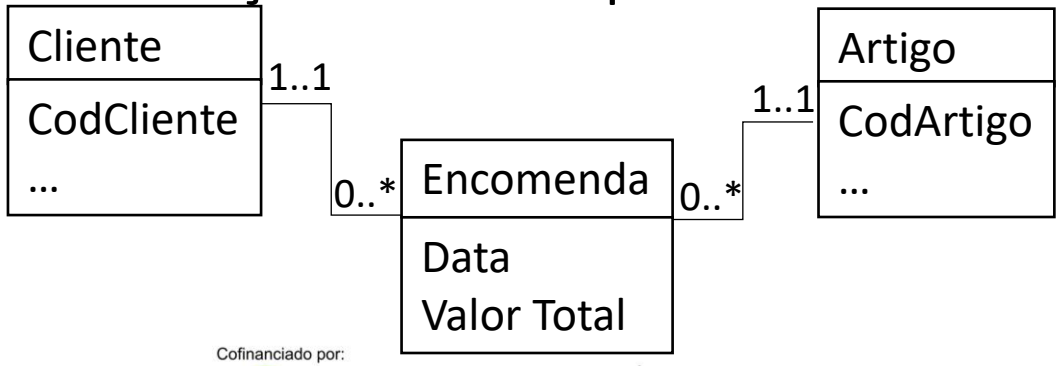
# Relações representadas por classes associativas

- Nova versão da relação entre Cliente e Artigo, em que foi acrescentada uma classe associativa com o nome Encomenda



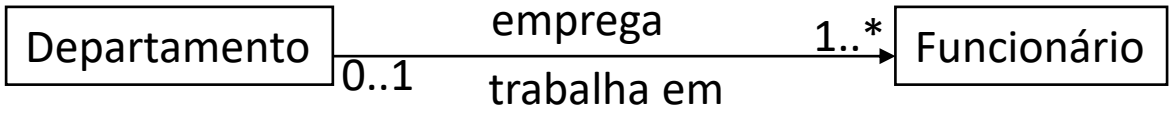

Relações representadas por classes associativas

- Em casos como este, também poderíamos logo à partida, ter considerado Encomenda como uma classe ou entidade, com os seus atributos. E então o diagrama de classes desta situação também poderia ser desenhado da seguinte forma:



# Relações representadas por classes associativas

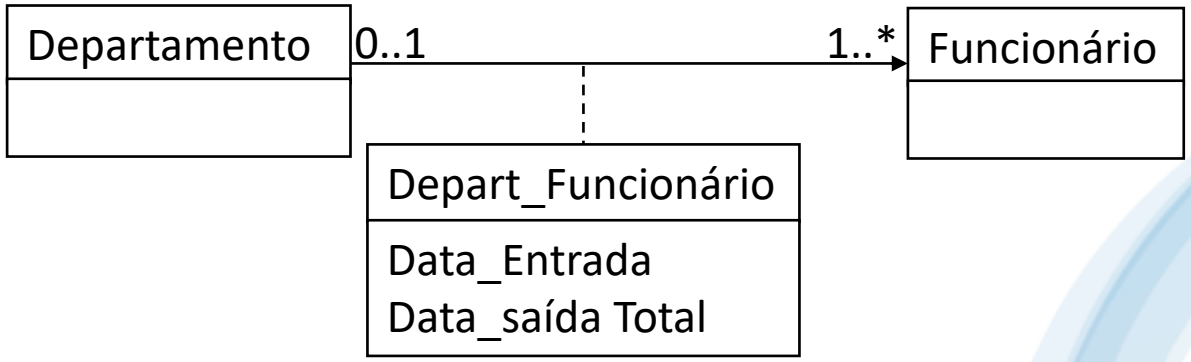
- Há no entanto, situações em que temos uma relação entre duas entidades/classes que requer uma classe associativa, sem que essa classe tivesse razão de ser por si própria logo à partida.
- Consideremos, por ex, a parte do SI da empresa em análise que representa os funcionários e os diferentes departamentos em que eles trabalham.




- Cada departamento emprega 1 ou vários funcionários (1..*)
- Cada funcionário trabalha em 0 ou 1 departamento (0..1)

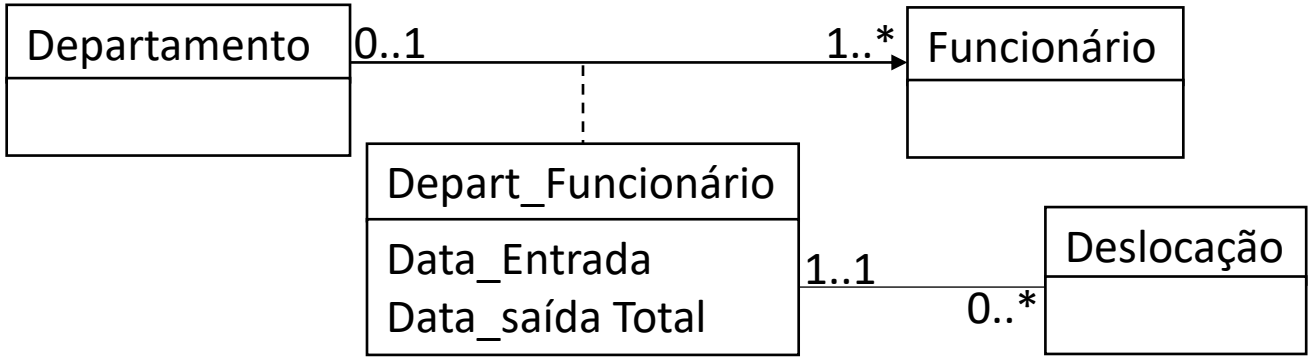
# Relações representadas por classes associativas

- Interessa registar as datas de entrada e de saída de cada funcionário em relação a cada departamento por onde passa.
- O nosso diagrama requer mesmo a representação de uma classe associativa que não existia à partida e que resulta apenas da relação/associação entre as classes Departamento e Funcionário.
- O seu nome costuma ser criado a partir da junção dos nomes das classes em associação que ele representa. Neste caso, poderíamos chamar a esta classe por exemplo: Depart_Funcionário.



# Relações representadas por classes associativas

- Ex: Os funcionários da nossa empresa em analise, para além de estarem ligados a departamentos, podem ser destacados para efetuarem deslocações a certos locais de interesse para a empresa. Interessa registar e saber, por exemplo, que empregados de que departamentos efetuaram a deslocação x ou y.



# 2º Diagrama de Classes

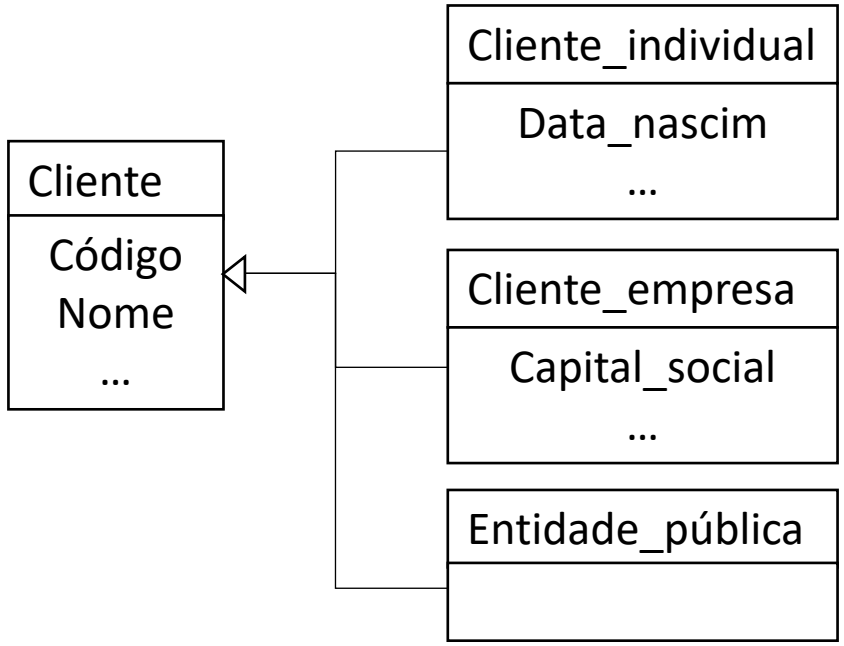
- Uma estação de televisão pretende um sistema de informação simples que a auxilie a armazenar e divulgar a sua programação diária.
- Quem consultar a programação por exemplo através de um browser, deverá poder visualizar, para cada dia, a sequência dos programas, com a indicação da hora de início e da duração de cada programa.
- Também se pretende que o espectador possa visualizar o tipo de programação (Noticias, Filme, etc.) e a correspondente classificação etária.

# Generalização e especializações

- Um outro tipo de relacionamento que é representado de uma forma especial, requerendo uma interpretação diferente, é o que corresponde a uma **generalização**.
- Uma **generalização** ocorre quando temos **uma classe mais genérica**, **que tem associadas a si duas ou mais classes mais especificas** – as quais herdam os atributos da classe genérica e acrescentam mais alguns atributos próprios.
- Consideremos, por exemplo, a seguinte situação: no registo dos clientes da nossa empresa, os analistas verificaram que há diferentes tipos de clientes: clientes individuais, outras empresas e entidades publicas (autarquias locais, etc.)

# Generalização e especializações

- Então, a representação da classe Cliente deve ser feita mediante uma classe geral (Cliente) e três classe específicas ou subclasses: Cliente_individual, Cliente_empresa e Entidade_pública




- A **classe genérica** - **Cliente**, contém os atributos que qualquer tipo de cliente terá tais como: Código, Nome, Morada, etc.
- **As 3 classes de especialização** – Cliente_individual, Cliente_empresa e Entidade_Pública. Cada uma destas classes herda os atributos da classe genérica e acrescenta os atributos que lhe são específicos. Por ex: a classe Cliente_individual poderá ter atributos como Data_Nascimento, sexo, profissão, que não existem nas outras classes específicas.

As empresas, por sua vez, terão atributos que os clientes individuais não têm como por exemplo: Capital_Social ou Ramo_Atividade.

- O **relacionamento entre as classes** – que neste caso, é representado por linhas que partem de cada uma das classes especificas e convergem para a classe genérica, com uma seta fechada em branco

# Generalização e especializações

- O relacionamento entre as classes – que neste caso, é representado por linhas que partem de cada uma das classes especificas e convergem para a classe genérica, com uma seta fechada em branco.

# 3º Diagrama de Classes

- Uma organização pretende uma aplicação de recursos humanos que armazene os resultados das avaliações periódicas que são efetuadas aos seus funcionários.
- Cada funcionário é avaliado por vários avaliadores, sendo que cada avaliador atribui uma nota para cada objetivo previamente atribuído ao funcionário avaliado.
- Sobre cada funcionário, para além do seu nome e nº de contribuinte, é necessário saber qual o seu superior hierárquico.
- Essa informação é relevante para que, caso o responsável pela avaliação assim o deseje, o sistema possa impedir que um funcionário possa avaliar o seu hierárquico direto.

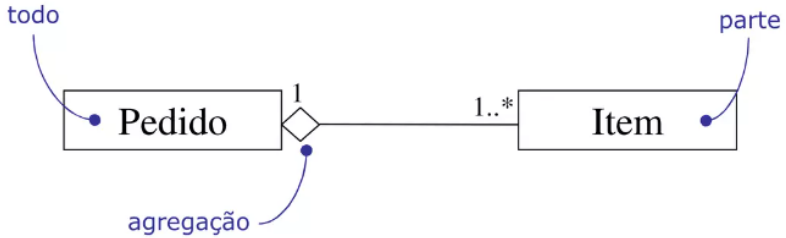
# Agregações e composições

- Uma outra situação especial que surge nos relacionamentos entre duas classes é aquela em que uma classe corresponde a uma parte de um todo que é a outra classe.
- A estas situações dá-se o nome genérico de agregação, mas distingue-se dois tipos:
  - Agregação
  - Composição

# Agregação

- Um objeto de uma classe pertence a um objeto da outra classe, mas este ultimo não tem a exclusividade da posse sobre o primeiro.
- Os objetos que são parte podem sê-lo de vários objetos da classe que os possui.
- Por exemplo: um sistema de uma empresa que regista os dados ou atributos da classe Empregado. Entre esses dados temos: Codigo, Nome, Data_Nascimento, Endereço.
- Os analistas entretanto, verificaram que há endereços que são partilhados por mais do que um funcionário (caso de cônjuges, pais, filhos, etc.)

# Agregação




- Então, Endereço pode ser considerado uma classe de agregação em relação à classe Empregado.
- E o diagrama poderá ficar assim:


- Exemplo de uma agregação: a classe Endereço é pertença da classe Empregado, mas sem exclusividade pois há endereços que podem ser partilhados por mais que um empregado.
- Cada empregado possui um e um só endereço, cada endereço pode pertencer a um ou vários empregados.

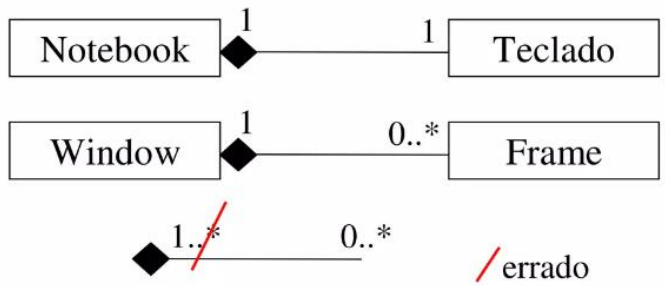
# Agregação

- Neste casos de agregação, o relacionamento vai da classe que é parte para a classe que é o todo.
- A agregação é representada por um losango em branco junto da classe que representa o todo.
- Este caso representa uma agregação porque:
  1. A classe Endereço pode ser vista como parte do todo que é a classe Empregado;
  2. Um endereço não é pertença exclusiva de um empregado, pois há endereços que são partilhados por mais do que um empregado.

# Composição

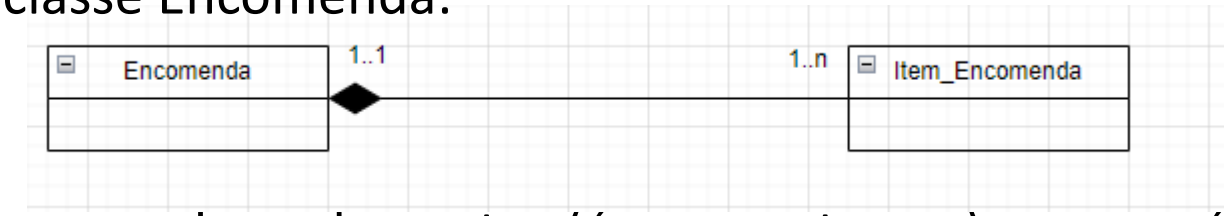
- Temos a situação em que um objeto de uma classe pertence, com exclusividade, a um objeto de outra classe.
- Isso implica que, se o objeto que representa o todo for removido, os objetos que lhe pertencem na outra classe terão também de ser removidos, pois não fazem sentido sem o seu "dono".
- Ex: consideramos a classe Encomenda – relativa às encomendas que a empresa recebe dos clientes.
- Uma encomenda pode conter vários pedidos de produtos.
- Entre encomenda e produtos teríamos uma relação de M..M , pois uma encomenda pode ter vários produtos e um produto pode ser encomendado várias vezes.

# Composição




- Quando o "todo" *morre* todas as suas "partes" também *morrem*

- Nestes casos, é habitual criar-se uma classe de composição, por exemplo Item_Encomenda, que fica agregada à classe Encomenda:

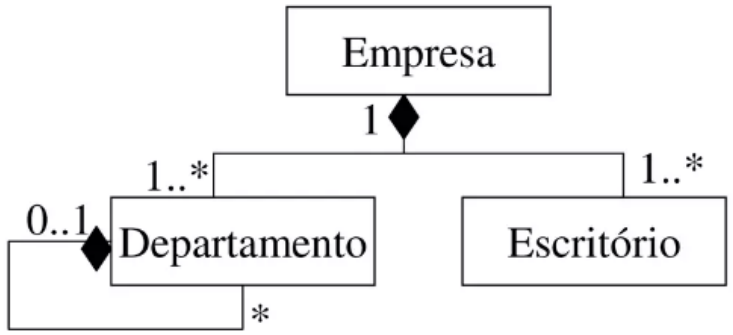


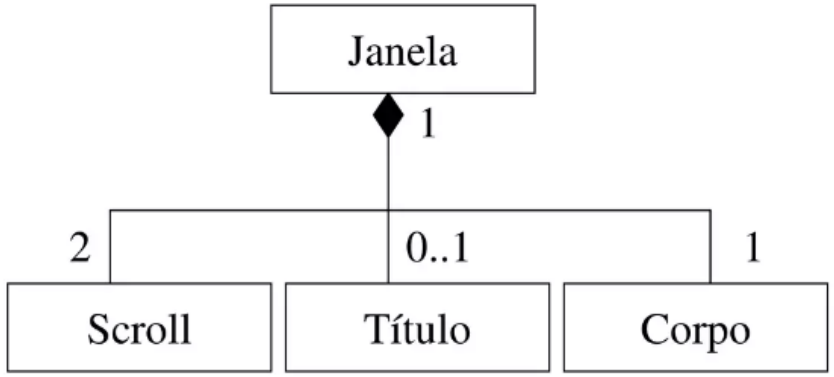

- Neste caso, uma encomenda pode conter (é composta por) um ou vários itens (produtos); cada item pertence a uma só encomenda.
- Isto é uma composição porque cada objeto da classe Item_Encomenda é pertença exclusiva de uma determinada encomenda.
- Se esta for removida, todos os itens que lhe dizem respeito também deverão ser removidos.
- Neste caso, o relacionamento vai da classe que é parte para a classe que é o todo e é representada por um losango preto junto da classe que representa o todo.

# Composição

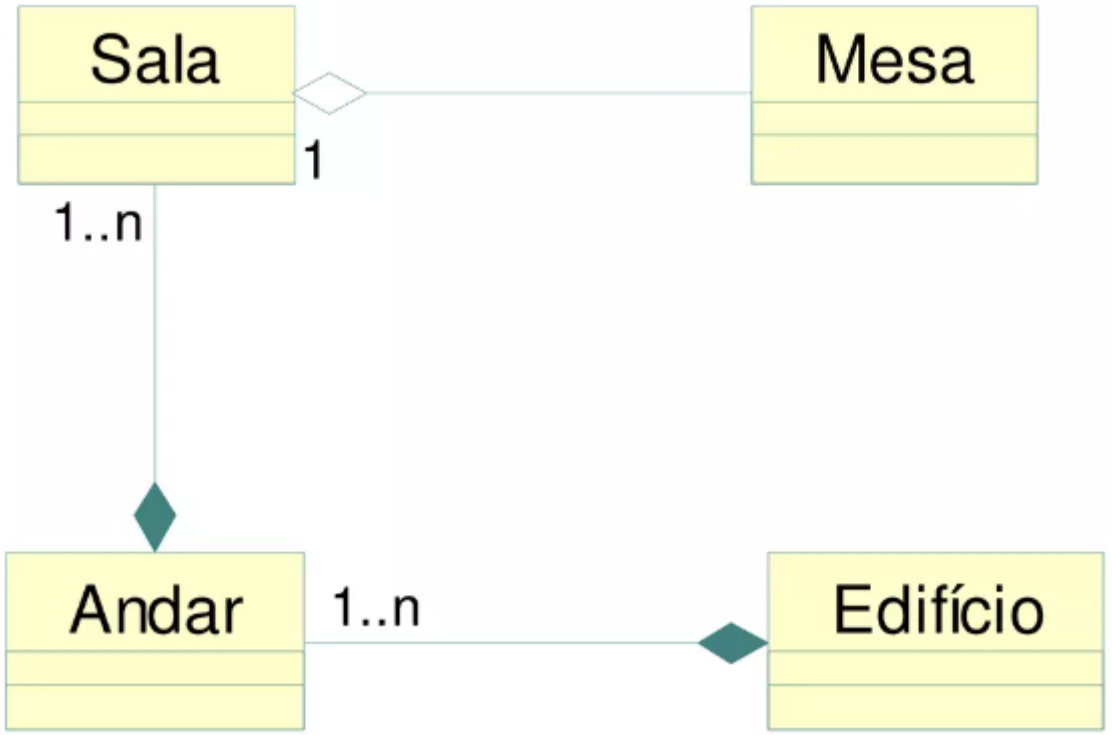
- Relacionamento: Composição

  Ex:



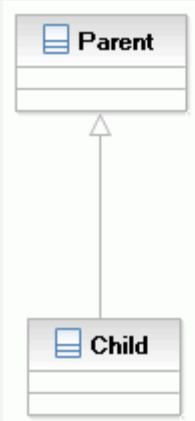



# Agregação x Composição

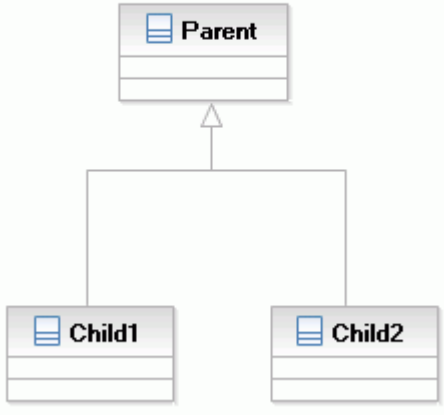


# Generalização

Único pai com um único filho



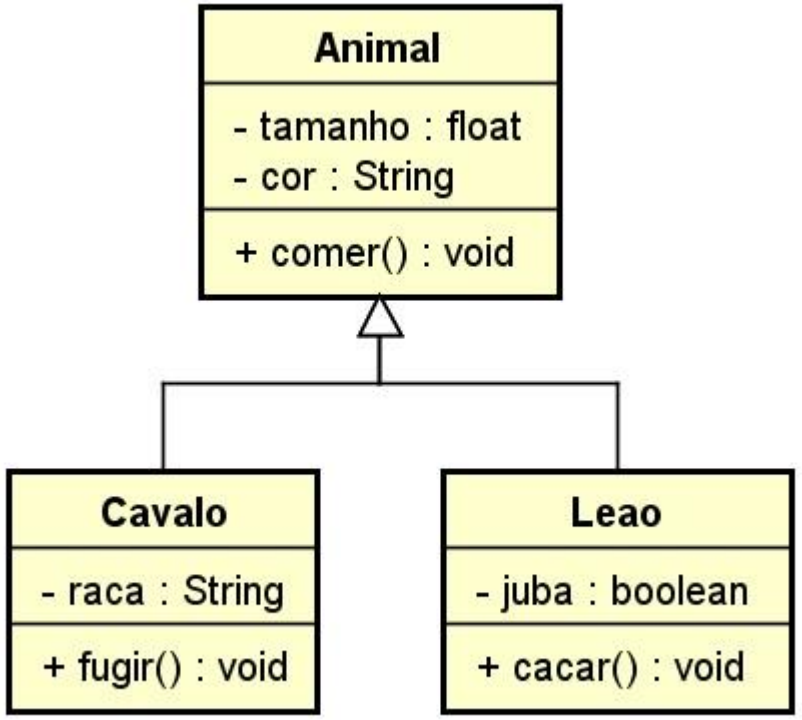

Único pai com vários filhos



# Generalização

- Um relacionamento de generalização é aquele no qual um elemento de modelo (o filho) tem como base outro elemento de modelo (o pai).
- Os relacionamentos de generalização são utilizados em diagramas de classe, componente, implementação e caso de uso para indicar que o filho recebe todos os atributos, operações e relacionamentos definidos no pai.
- Os elementos de modelo em um relacionamento de generalização devem ser do mesmo tipo
- Como os elementos de modelo filho nas generalizações herdam atributos, operações e relacionamentos do pai, é necessário definir para o filho apenas os atributos, as operações ou os relacionamentos que sejam distintos do pai.
- O elemento de modelo pai pode ter um ou mais filhos e qualquer elemento de modelo filho pode ter um ou mais pais. É mais comum ter um único elemento de modelo pai e vários elementos de modelo filho.

# Generalização



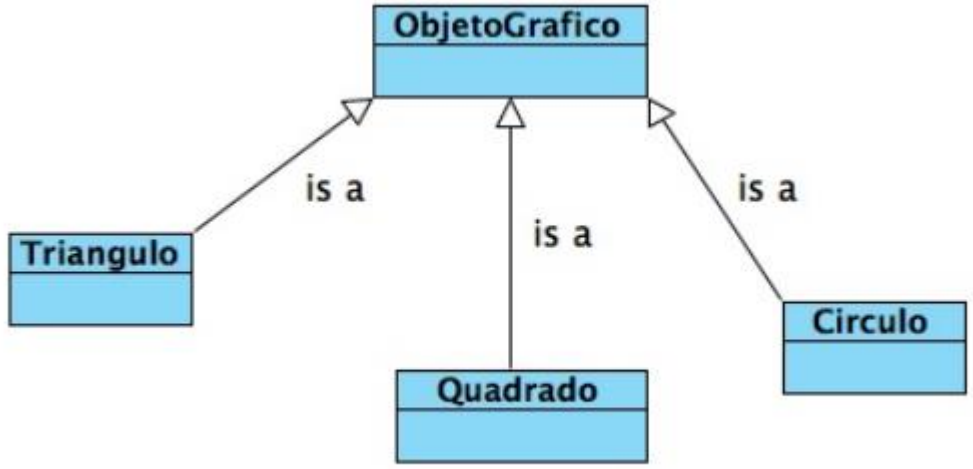

- Nesse exemplo houve a necessidade de criar uma classe mais geral, porque todos os animais possuem tamanho e cor e se alimentam de alguma forma.
- Por outro lado, houve a necessidade de especializar alguns animais por possuírem características únicas.
- Animal é uma generalização de Leão e Cavalo e Leão e Cavalo são especializações de Animal.
- Portanto, generalização e especialização são a mesma coisa, vistas de perspetivas diferentes.

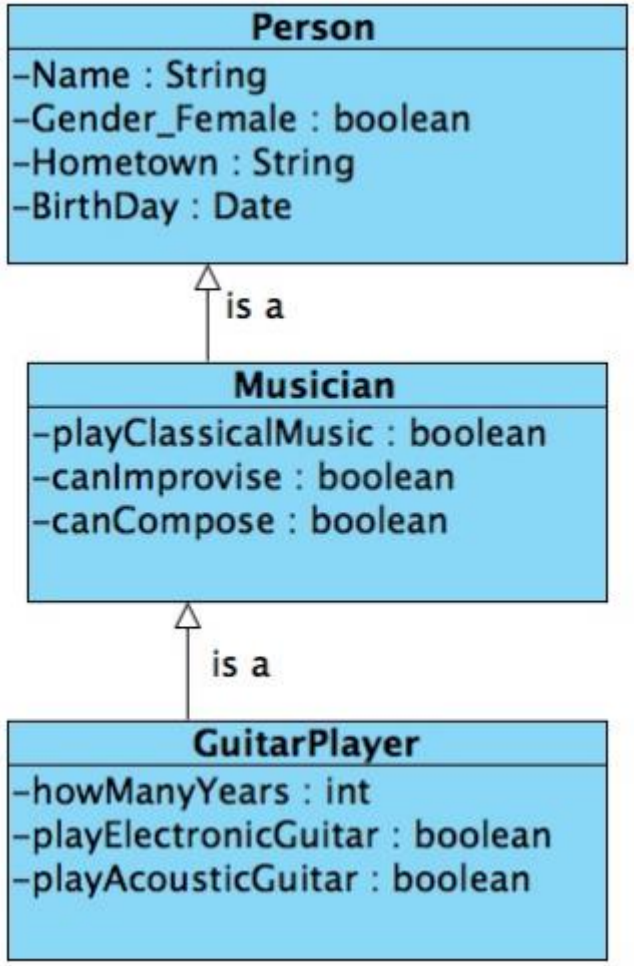
# Herança

- Herança é a capacidade que instâncias de uma classe filha ou Subclasse de aceder dados e procedimentos ou métodos associados com uma Classe Parente ou Superclasse.
- Ou seja, uma Subclasse herdará todos os métodos e dados da Superclasse.
- Além disso, a Subclasse poderá definir métodos e dados, ou mesmo em alguns casos, redefinir alguns métodos da Superclasse
- A Herança é sempre transitiva. Ou seja, uma classe pode herdar aspetos de Superclasses que estão a muitos níveis de distância.
- Ex: uma classe Filha pode herdar métodos da classe Pai, classe Avô, classe Bisavô, classe Tetravô e assim sucessivamente

# Herança

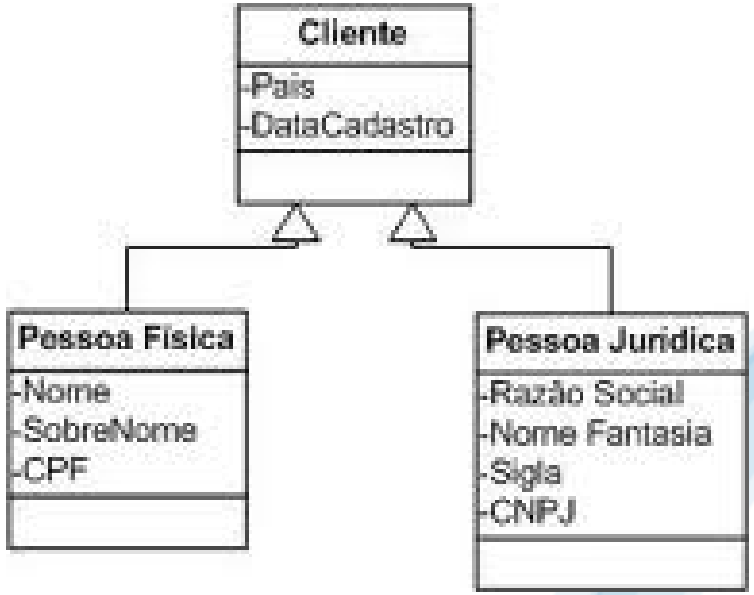
- A Representação no diagrama UML da relação de Herança entre classe é feita por meio de uma linha ligando as duas classes e um símbolo triângulo apontando para Superclasse.



# Herança

- De modo simplificado, herança é "uma classe (classe filha) que tem os mesmos atributos de outra (classe mãe), mais alguns atributos distintos". Alguns exemplos podem ser vistos abaixo:
- Exemplo 1: Uma classe Cliente que é a classe principal e duas classes filhas da mesma, "Pessoa Física" e "Pessoa jurídica". Neste exemplo, as duas classes podem ser entendidas como um cliente, mas cada uma com alguns atributos a mais.



# Tarefa 3