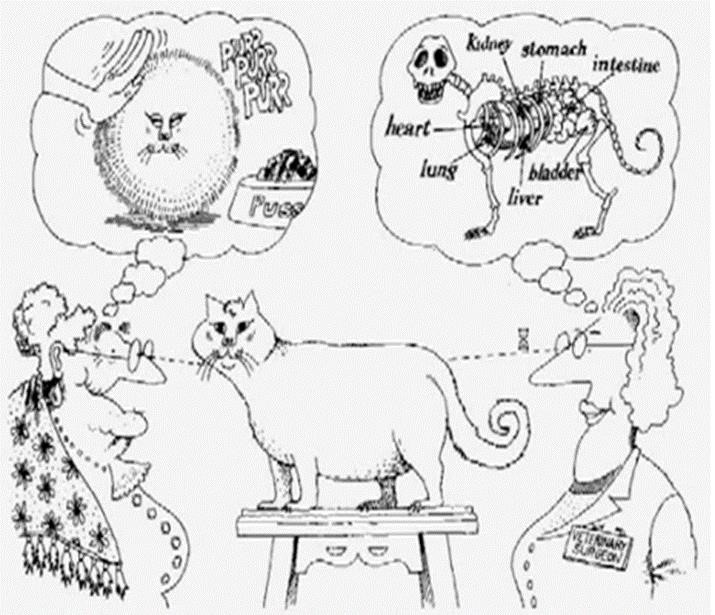
Revisão de OO, UML e Exercícios

# Paradigmas

|  |  |  |
| --- | --- | --- |
|  | |  | | --- | | Linguagens modernas são multiparadigmáticas | |

* As partes das quais o modelo é construído são os objetos que aparecem no domínio do problema.
* Objeto é uma entidade que combina estrutura de dados e comportamento funcional
* Os sistemas são modelados como um número de classes que ao serem instanciadas interagem, trocando informações.
* Envolve todas as atividades de desenvolvimento:
* Análise
* Projeto
* Programação
* Testes
* Pilares



* Uma visão mais detalhada
* Buscar trazer uma forma mais natural de ver o mundo.

Entrada

Programa

Saída

Estado

Diagrama 1

...

.

.

.

.

.

.

.

.

.

Estado

Entrada

Program

a

~~Saída~~

Estado

Entrad

a

Program

a

~~Saída~~

Estado

Entrada

Program

a

~~Saída~~

Estado

Entrada

Program

a

~~Saída~~

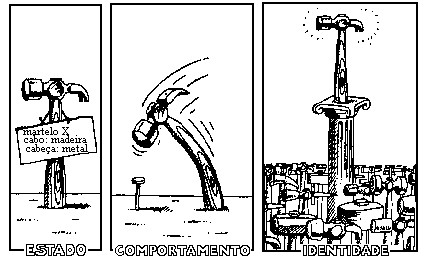
As ações e estados encapsulados

pelos objetos que se comunicam

Diagrama 2

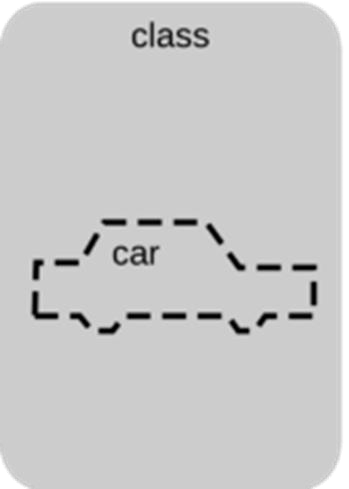
# Características de um Objeto

• Representa entidades do domínio do problema



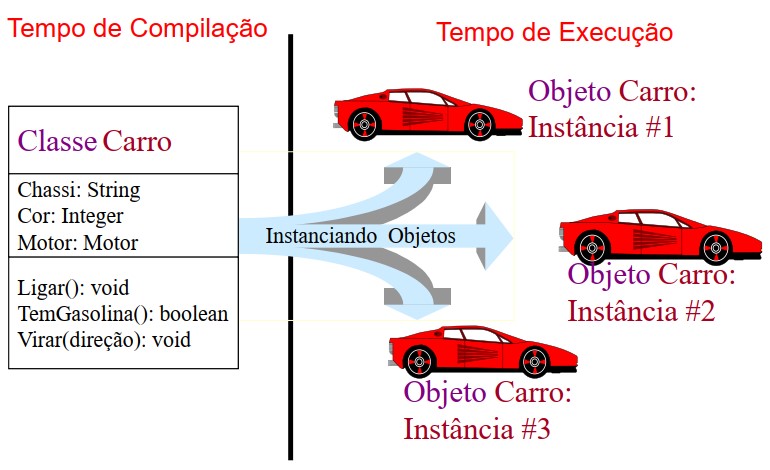
# Características de uma Classe

• Classifica objetos que tenham propriedades semelhantes



# Objetos x Classes

• Importante lembrar: Uma classe define um tipo, a partir do qual todos os objetos são criados.



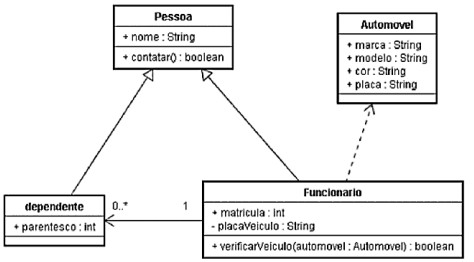
# Análise e Projeto OO

* Programar -> Como faremos? Ligar o computador e começara digitar ou pedir para a IA?
* E se fôssemos contratados para criar um software que gerencia os caixas eletrônicos de um banco?
* E se fôssemos trabalhar em uma equipe em que o trabalho é dividido entre 100 pessoas?

# Análise e Projeto OO

* Antes de escrever o código, é necessário **analisar** os requisitos (*o quê*) de seu projeto e **projetar** uma solução (*como*) satisfatória
* Pode poupar muitas horas de trabalho e dinheiro
* Quando esta análise envolve um ponto de vista de OO, chamamos de **análise e projeto orientados a objetos**
* Uma linguagem gráfica utilizada para comunicação de qualquer processo de análise e projeto OO é a **Unified Modeling Language (UML)**

# UML

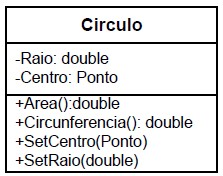
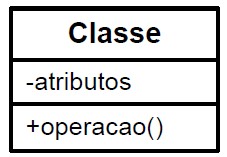
* A UML oferece padrões de diagramas estruturais e comportamentais (de interação)
* São vários tipos de diagramas diferentes.
* Estamos inicialmente interessados nos **diagramas estruturais**
* Mais especificamente, os **Diagramas de Classes**
* São uma representação da estrutura e relações das classes;
* Servem de base para criação de outros diagramas.
* Classes e associações em diagramas de classes UML são semelhantes a tipos de entidades e tipos de relacionamentos nos diagramas entidade-relacionamento, mas têm um maior poder descritivo

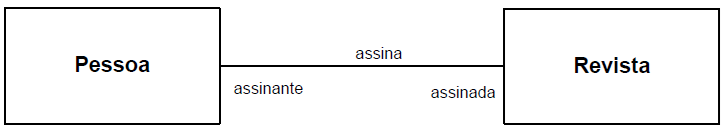
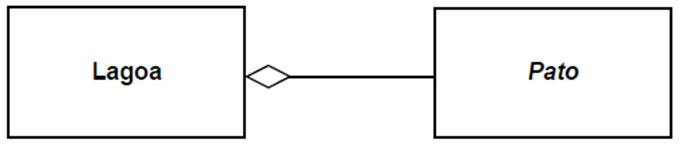
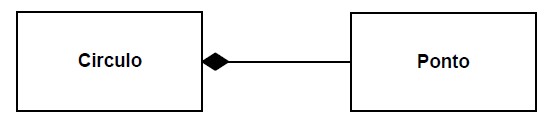
# Elementos do Diagrama

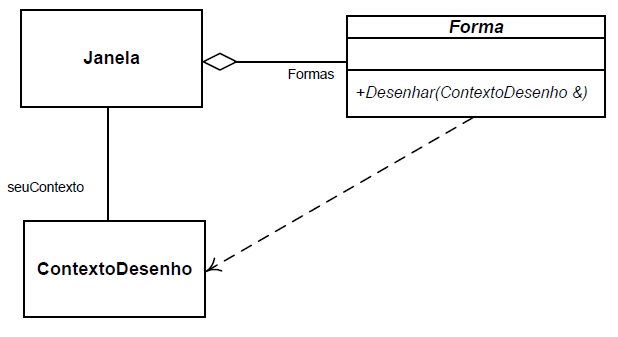
* De forma geral, vamos encontrar os seguintes elementos no diagrama:
* Classe
* Relações
* Multiplicidade

# Elementos do Diagrama : Classes

* Classe: representada por um retângulo separado em seções (compartimentos).
* Seção superior: apresenta atributos da classe
* Seção inferior: apresenta as operações (métodos) que podem ser executadas com as instâncias da classe



* São representadas por linhas e descrevem as potencias relações entre classes, como: • Associação
* Composição
* O **diamante preto** fica no lado da classe que contém uma instância da outra
* Agregação 
* O **diamante branco** fica no lado da classe que contém uma instância da outra
* São representadas por linhas e descrevem as potencias relações entre classes, como:
* Dependência
* Às vezes o relacionamento entre duas classes é muito fraco
* Não são implementados por atributos que as una;
* Ao invés disto, pode ser implementado apenas através de parâmetros de métodos.



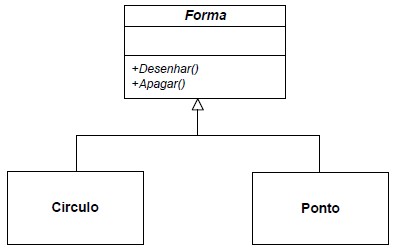
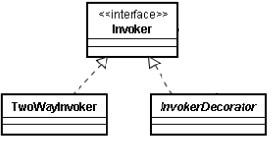
Uma funcionalidade de

*Forma*

depende de

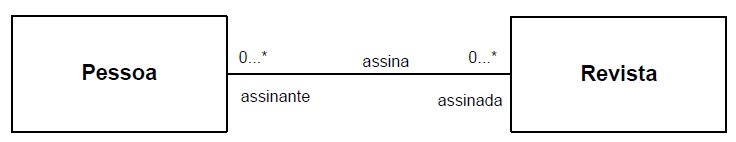
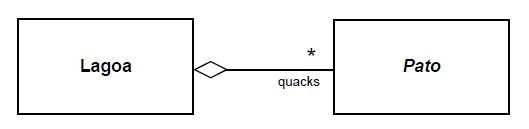
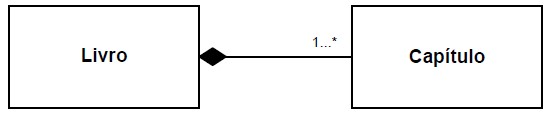
*contextoDesenho*

.

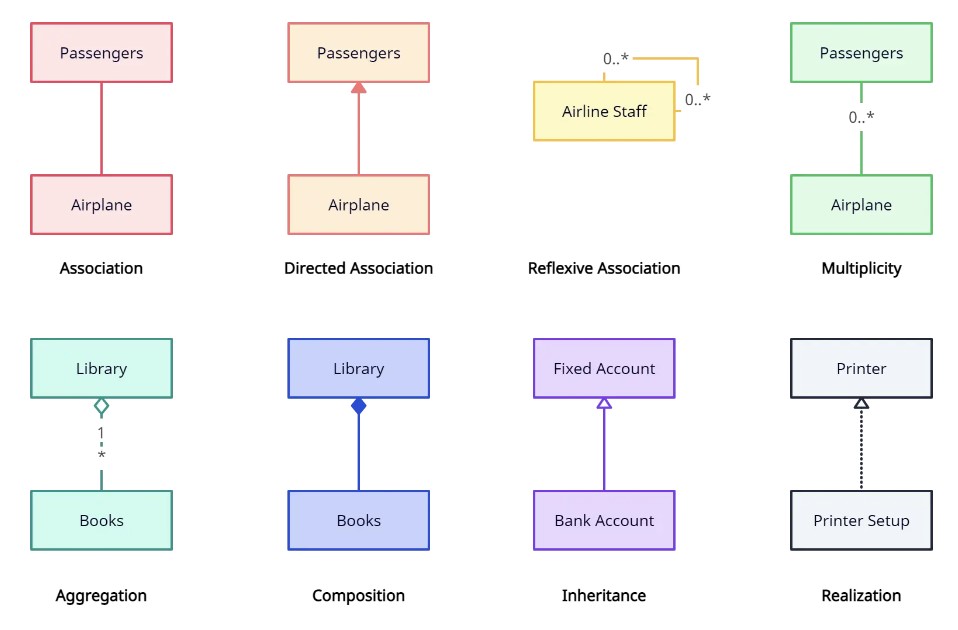
* São representadas por linhas e descrevem as potencias relações entre classes, como:
* Herança
* É representada por uma linha contendo uma seta triangular que aponta para a superclasse (ou classe base);
* Interfaces.
* É representada por uma linha pontilhada contendo uma seta triangular 

# Elementos do Diagrama : Multiplicidade

• Multiplicidade: representam quantas instâncias de uma classe podem ser associadas de uma maneira específica com um número de instâncias de outra classe.



# Elementos do Diagrama : Resumo



# Atividades Juntos

* Dados os digramas apresentados a seguir, identifique qual o tipo de relação que está sendo representada:
* Por exemplo:

◦ Associação Binária

◦ Associação Direta

◦ Associação Reflexiva

◦ Agregação

◦ Composição

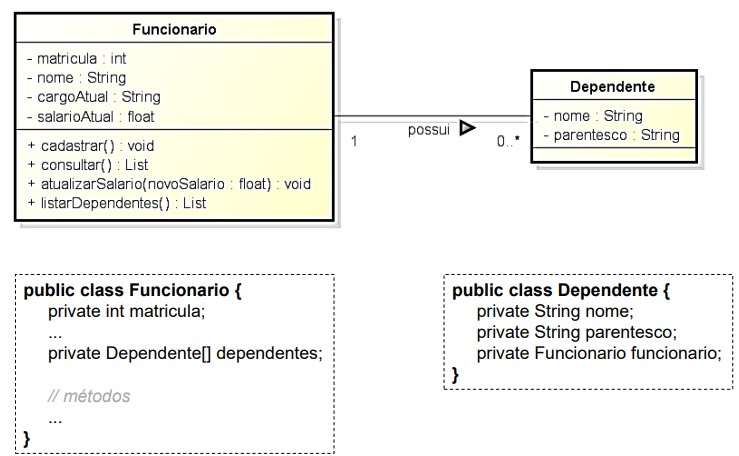
◦ Dependência

◦ Herança

◦ Realização / Implementação (Interface)

* Pode chutar que a canela aguenta.. ☺... Então.... Que comecem os jogos.....

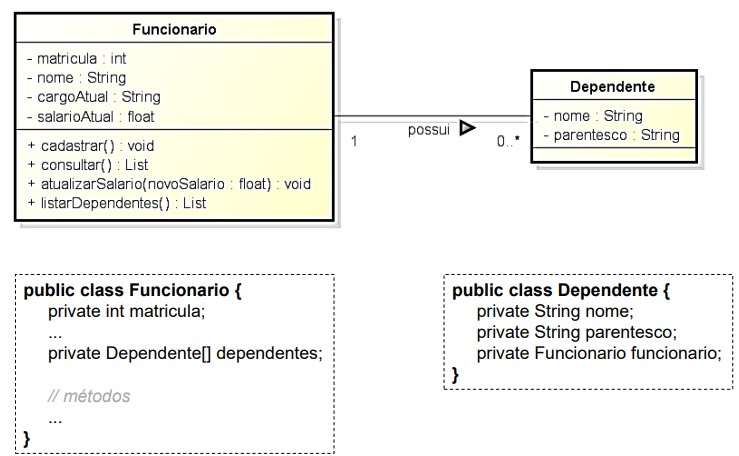
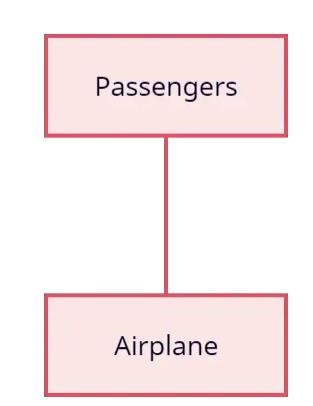
# #1



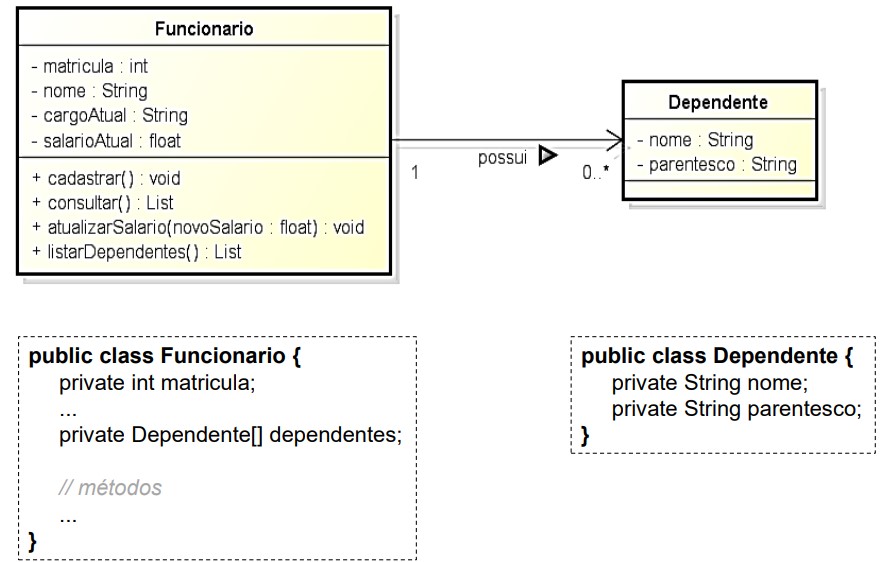
# #1 Associação Binária

* Quando há duas classes envolvidas na associação de forma direta de uma para outra. Por exemplo, passageiros e companhia aérea podem estar vinculadas.
* Podem possuir títulos para determinar o tipo de vínculo, como no exemplo

Funcionario-Dependente.

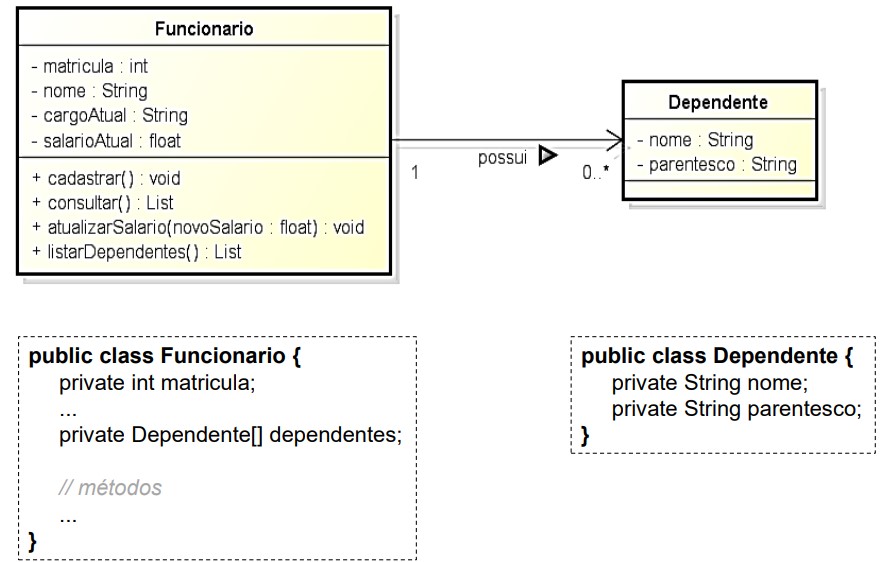
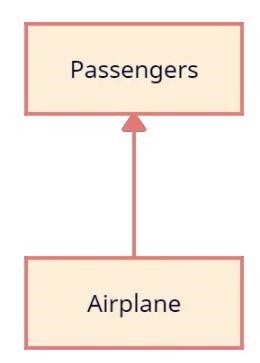


# #2

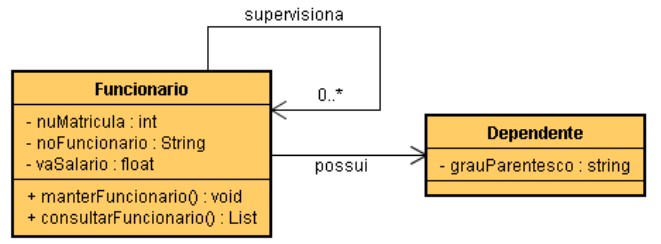


# #2 Associação Direta

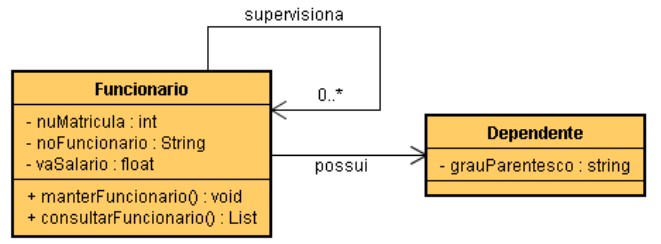
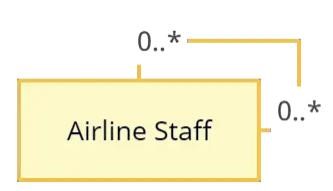
• Refere-se a uma relação direcional representada por uma linha com uma ponta de seta. A ponta da seta mostra um fluxo direcional contido em um “*contêiner*”.



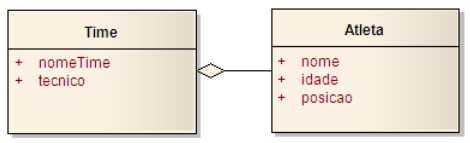
# #3



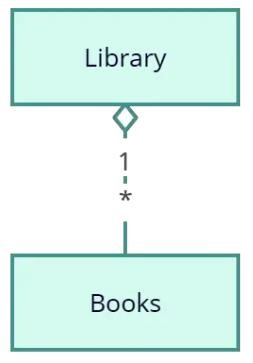
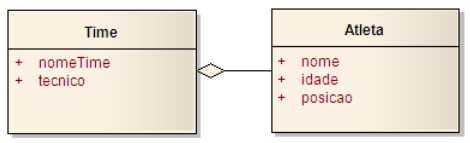
# #3 Associação Reflexiva ou Unária

* Ocorre quando há um relacionamento de um objeto de uma classe com objetos da mesma classe;
* No exemplo, um objeto da classe Funcionário pode (ou não) supervisionar outros objetos dessa mesma classe;
* Para o relacionamento ficar mais claro, pode-se informar a sua multiplicidade.

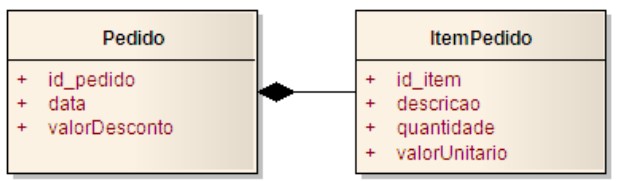
# #4



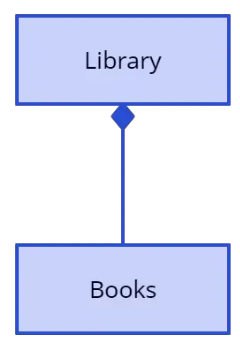
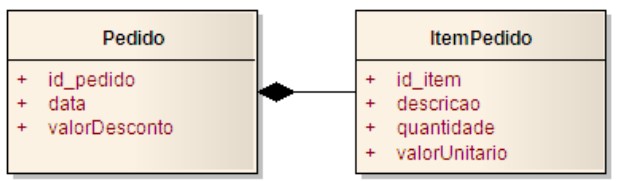
# #4 Agregação

* Tipo especial de associação que tenta demonstrar que as informações de um objetotodo precisam ser complementadas pelas informações contidas em um (ou mais) objetos-parte.
* Um time é formado por atletas, ou seja, os atletas são parte integrante de um time, mas os atletas existem independentemente de um time existir.

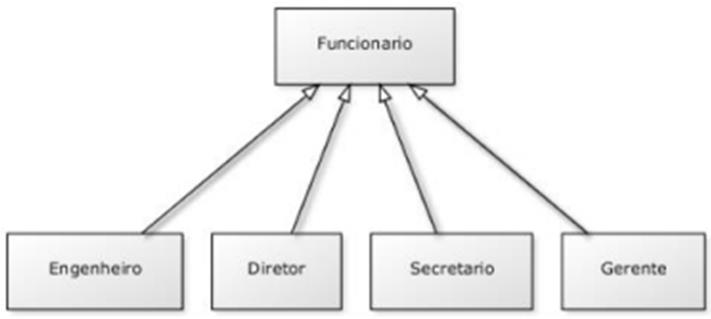
# #5



# #5 Composição

* É uma variação da agregação e considerada mais “forte”. O objeto-parte não pode existir sem o objeto-todo. Se o objeto-todo for destruído, o objeto-parte também será.
* Um pedido é composto por um ou vários itens, mas um item NÃO é item de um pedido se não existe pedido

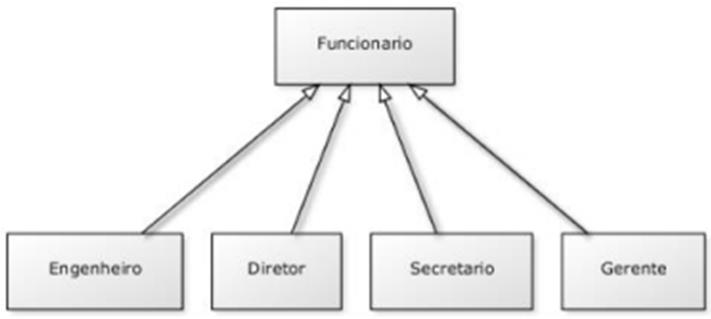
# #6



#6 – Herança (Especialização / Generalização)

• Refere-se a um tipo de relacionamento em que uma classe associada é filha de outra em virtude de assumir as mesmas funcionalidades da classe pai

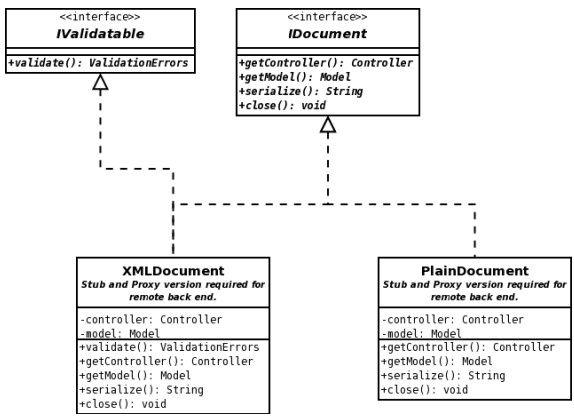
Classes filhas



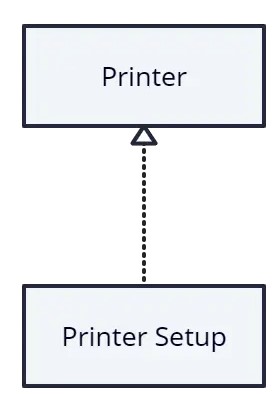
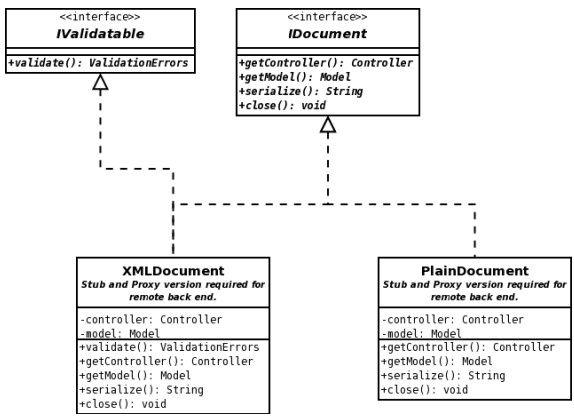
Classe pai

“é um tipo de”

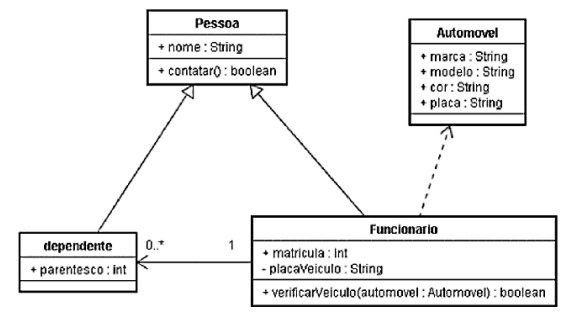
# #7



# #7 - Realização / Implementação (Interface)

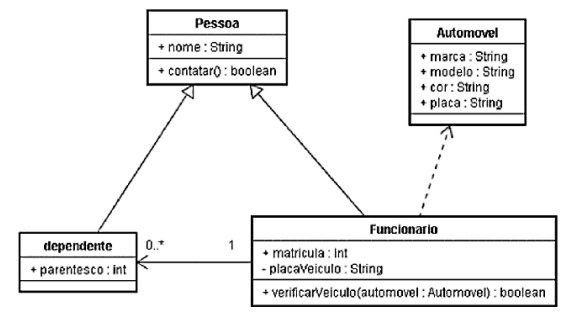
• Tipo especial “*de classe*” a qual não pode ser instanciada. Utilizada, por exemplo, para especificar operações externamente visíveis para uma outra classe implementar. 

# #8



# #8 - Dependência

* Indica um grau de dependência entre uma classe e outra
* No exemplo, Funcionário não instancia um Automóvel, apenas usa-o como parâmetro de um método.



# Passos para criar um Diagrama de Classes

1. Identificar os nomes das classes
   1. Identifique os objetos do sistema
2. Definir relações
   1. Determine como as classes estão relacionadas entre si
3. Criar a estrutura
   1. Dê nomes às classes
   2. Associe as classes com os conectores apropriados
   3. Inclua atributos e operações

# Comentários Finais

* A POO nos provê uma melhor organização do código e também contribui para o reaproveitamento do mesmo
* No entanto, o desempenho de execução pode ser inferior quando comparado com a programação estruturada
* Talvez a grande dificuldade para compreender a POO é na verdade entender o paradigma de projeto orientado a objetos
* A POO se preocupa com os **objetos** e seus **relacionamentos**;
* A Programação Estruturada se preocupa com as **ações**.

# Exercícios

• Estudo dos assunto teóricos apresentados por meio das questões em:

## Atividades Sala de Aula - Revisão - questões.pdf

• Apresentar as soluções na sala de aula e enviar no post da atividades no blog .