



Click to add text
Click to add text

Engenharia de Software

UML

Sara Estrela

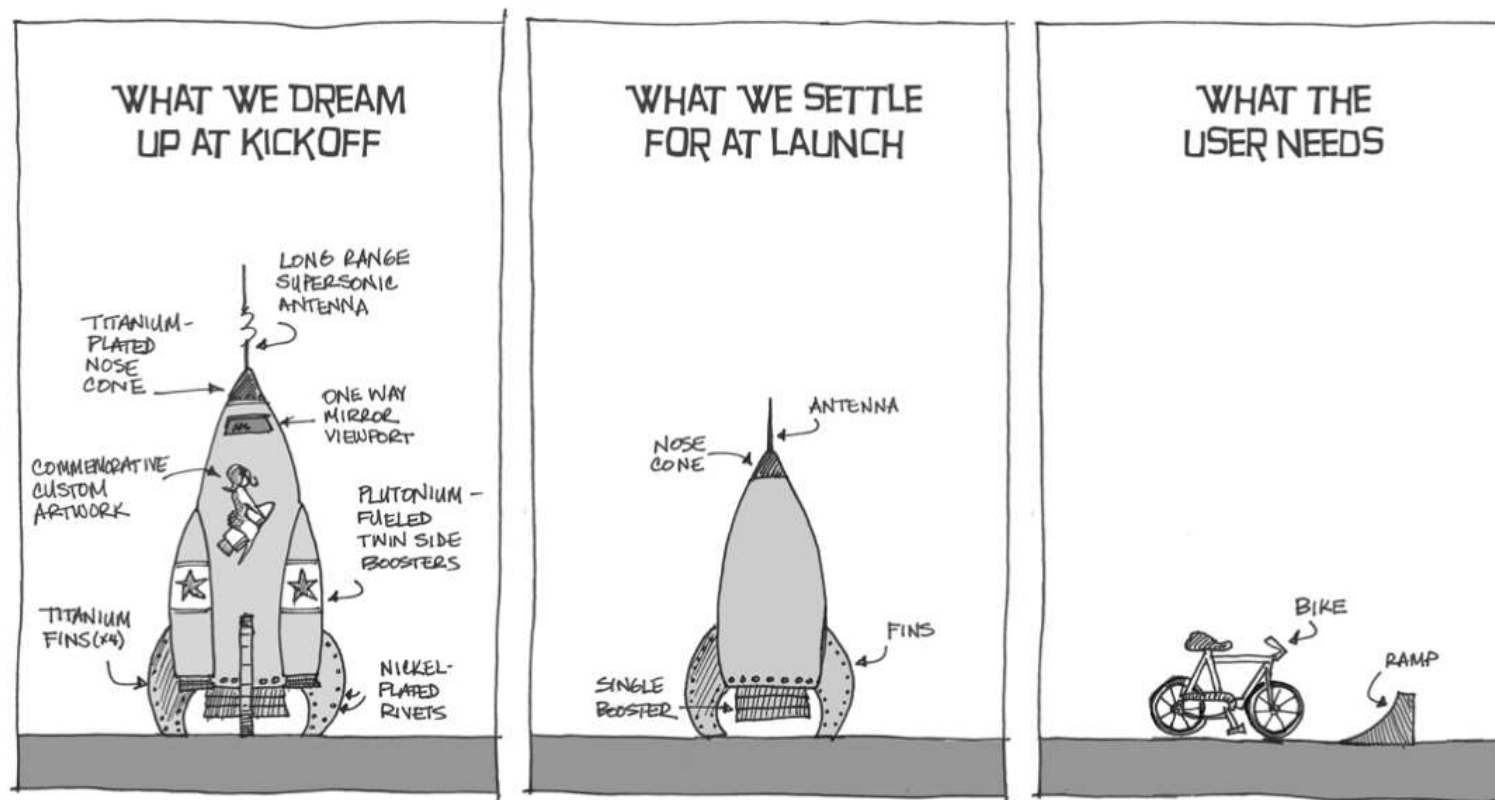
Engenharia de software

- “Engenharia de software é uma área da engenharia e da computação voltada à especificação, desenvolvimento, manutenção e criação de software, com a aplicação de tecnologias e práticas de gerência de projetos e outras disciplinas, visando organização, produtividade e qualidade.” – Segundo Wikipédia

Engenharia de software

- Dedicada ao processo de desenvolvimento de software com qualidade e eficiência.
- Recorre ao uso de diversas técnicas como levantamento de requisitos e diversos diagramas como:
 - Diagrama de casos de uso,
 - Diagrama de classes,
 - Diagrama de sequencia,
 - Entre outros;

Levantamento de requisitos



Levantamento de requisitos

- É um processo que deve incluir todas os intervenientes do projeto, e serve para registar todas as necessidades e especificações que o produto deve conter
 - É um processo de extrema importância para o correto desenvolvimento do projeto
- Sem um correto levantamento o produto final pode não atender às necessidades do cliente/utilizador final.
 - Um correto levantamento para além de auferir o objetivo do produto também deve caracterizar as nuances necessárias para o atingir.

not
✓
The customer is always right

Levantamento
de requisitos

Levantamento de requisitos

- Muitas vezes os clientes ou outros intervenientes, não têm conhecimento técnico suficiente para entender os passos todos que são necessários, por vezes acabam por omitir questões que são relevantes.

Levantamento de requisitos

- Consequências de um mau levantamento de requisitos:
 - Insatisfação cliente/utilizador final
 - Não cumprimento dos objetivos
 - Atrasos no lançamento
 - Inutilização do produto
 - Tempo perdido em redesenho e desenvolvimento
 - Custo
 - Insatisfação das equipas

Levantamento de requisitos

- Requisitos:
 - Identificam as necessidades e requisitos do produto
 - Identifica também constrangimentos e limitações que o produto deve ter em consideração
- Tipos de requisitos:
 - Requisitos funcionais
 - Requisitos não funcionais

Levantamento de requisitos

- Requisitos funcionais
 - Funções que o produto deve cumprir, como tarefas e processos que o sistema deve ser capaz de realizar.
 - Ex “deve permitir adicionar artigos ao carrinho de compras”
- Requisitos não funcionais
 - Não esta relacionado com tarefas, mas sim, atributos associados a aplicação. Como o desempenho, usabilidade, confiabilidade, segurança, disponibilidade, manutenção e tecnologias envolvidas.
 - Ex: Deve usar software open source
 - Ex: Deve ter uma indisponibilidade >1% do tempo

Levantamento de requisitos

- Processo
 - Identificar stakeholders e partes interessadas.
 - Entender o contexto e o problema.
 - Identificar requisitos.
 - Identificar requisitos funcionais
 - Identificar requisitos não funcionais (desempenho e qualidade)
 - Documentar os requisitos.
 - Validação de requisitos

Levantamento de requisitos

- Dificuldades no Levantamento de Requisitos
 - Mudanças frequentes nos requisitos
 - Compreensão inadequada das necessidades dos utilizadores
 - Falta de comunicação entre a equipa e os stakeholders
- Melhores Práticas
 - Envolver todos stakeholders desde o início
 - Utilizar ferramentas de modelagem e documentação adequadas
 - Manter um processo de revisão contínua dos requisitos

Levantamento de requisitos

- É um passo crucial para um bom desenvolvimento,
- O empenho deste tempo pode “colher frutos” ao longo de todo o desenvolvimento do software.
- É um esforço que permite o aumento da possibilidade de sucesso e qualidade do produto final

Exemplo

- Supondo um briefing com um cliente que pretende a criação de uma aplicação mobile de gestão de tickets de atendimento num Centro de reparações automóveis
- Levantamento de requisitos funcionais:
 - Cada utilizador deve poder fazer registo, login e logout
 - É permitido recuperar a password
 - O Utilizador pode fazer o pedido de um ticket para três tipos de atendimento:
 - Manutenção, Avaria, sinistro
 - É possível verificar, para cada serviço, em que estado se encontra, média de tempo de espera, a data e horas iniciou o atendimento, e data de término do respetivo atendimento

Exemplo

- Levantamento de requisitos não funcionais:
 - No caso de ticket para marcação, a aplicação acede à base de dados e (de acordo com o Número do cliente) verifica se tem consulta, apenas gerando um ticket se existir marcação.
 - Nos restante caso de urgência gera, mas quando ultrapassa o limite de atendimentos diários estabelecido, informa não ser possível o atendimento e o ticket não é gerado.
 - O sistema de tickets reinicia todos os dias às 23:59. A partir das 00:00 já é possível fazer um pedido de ticket para os serviços desse dia.
 - Não é possível pedir mais do que um ticket.
 - Se um utilizador fizer uso abusivo da aplicação será banido durante um determinado período de tempo

Exemplo

- Levantamento de requisitos não funcionais (de qualidade):
 - A Base de Dados deve ser desenvolvida em MySQL
 - A aplicação deve seguir uma arquitetura em 3 camadas (UI, BLL e DAL)
 - Deve usar-se POO
 - A aplicação deve permitir guardar apenas uma foto por utilizador

Exercícios

- Defina alguns dos requisitos **funcionais** e **não funcionais** dos seguintes Web Sites:
 - idealista - <https://www.idealista.pt/>
 - Wook - <https://www.wook.pt/>

Introdução UML

- *Unified Modeling Language* (em português Linguagem de Modelagem Unificada) é uma linguagem gráfica para visualizar, especificar, construir e documentar um Sistema (programa, aplicação, site, etc...)
- Esta apresentação tem como objetivo fornecer uma visão geral sobre os Diagramas de Casos de Uso, Diagrama de Classes e Diagrama de Sequência (existindo muitos mais para além destes)
- Estes diagramas são ferramentas que auxiliam na análise, no design e na documentação de sistemas de software

Diagrama de Casos de Uso

- O *Use Cases Diagram* é uma representação visual que descreve as interações entre um sistema e seus atores
- Mostra como os utilizadores interagem com o sistema em diferentes cenários
- Fornece uma visão geral das funcionalidades do sistema a partir de uma perspetiva externa

Elementos do Diagrama de Casos de Uso

- Ator: Uma entidade externa que interage com o sistema
 - Pode ser um utilizador, um administrador, uma API, um dispositivo, etc...



Elementos do Diagrama de Casos de Uso

- Caso de Uso: Uma função ou recurso específico que o sistema oferece ao utilizador
 - Como por exemplo pesquisar um livro, fazer login, editar notícia, ect...
 - É representado por uma elipse



Elementos do Diagrama de Casos de Uso

- Relações: Relações entre atores e/ou casos de uso
 - Associação: Liga um ator a um caso de uso e indica que o ator interage com o sistema por meio desse caso de uso

- Associação
- Generalização/Herança
- Inclusão
- Extensão

_____>

 <<includes>>
_____>

 <<extends>>
_____<

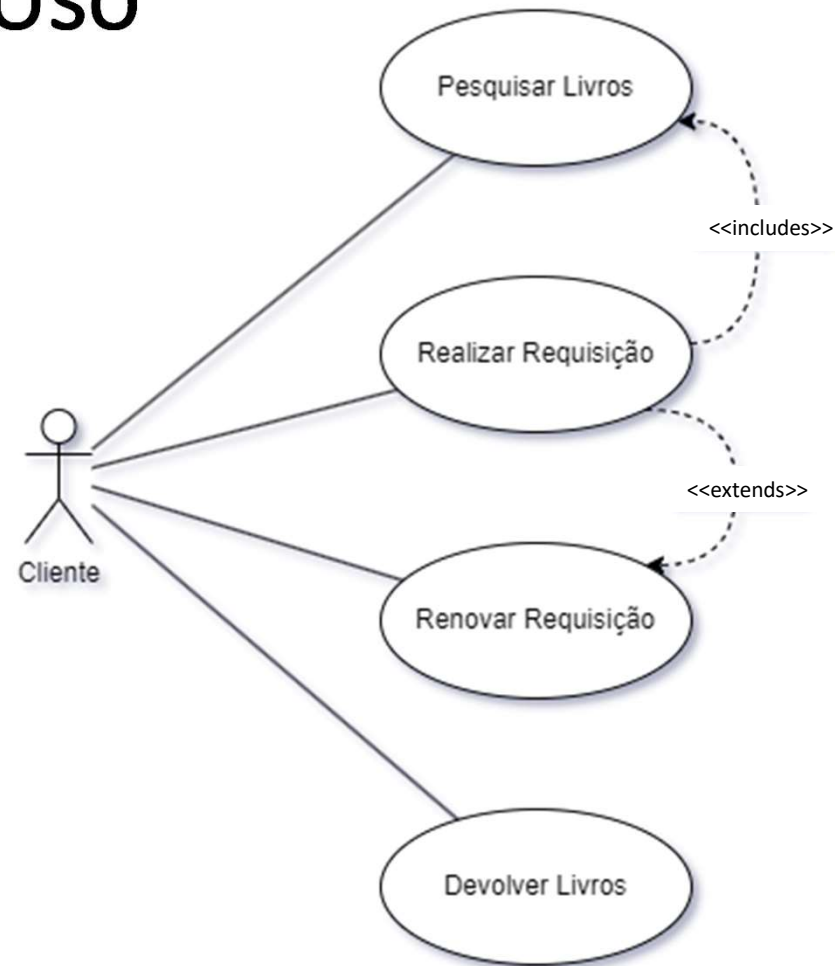
Elementos do Diagrama de Casos de Uso

- Relações (continuação...)
 - Inclusão:
Indica que um caso de uso pode ser incluído em outro caso de uso (base)
 - Extensão:
Indica que um caso de uso pode ser estendido por outro caso de uso (base) em circunstâncias específicas
 - Generalização/Herança:
indica que o ator ou caso de uso filho herda funcionalidades do pai

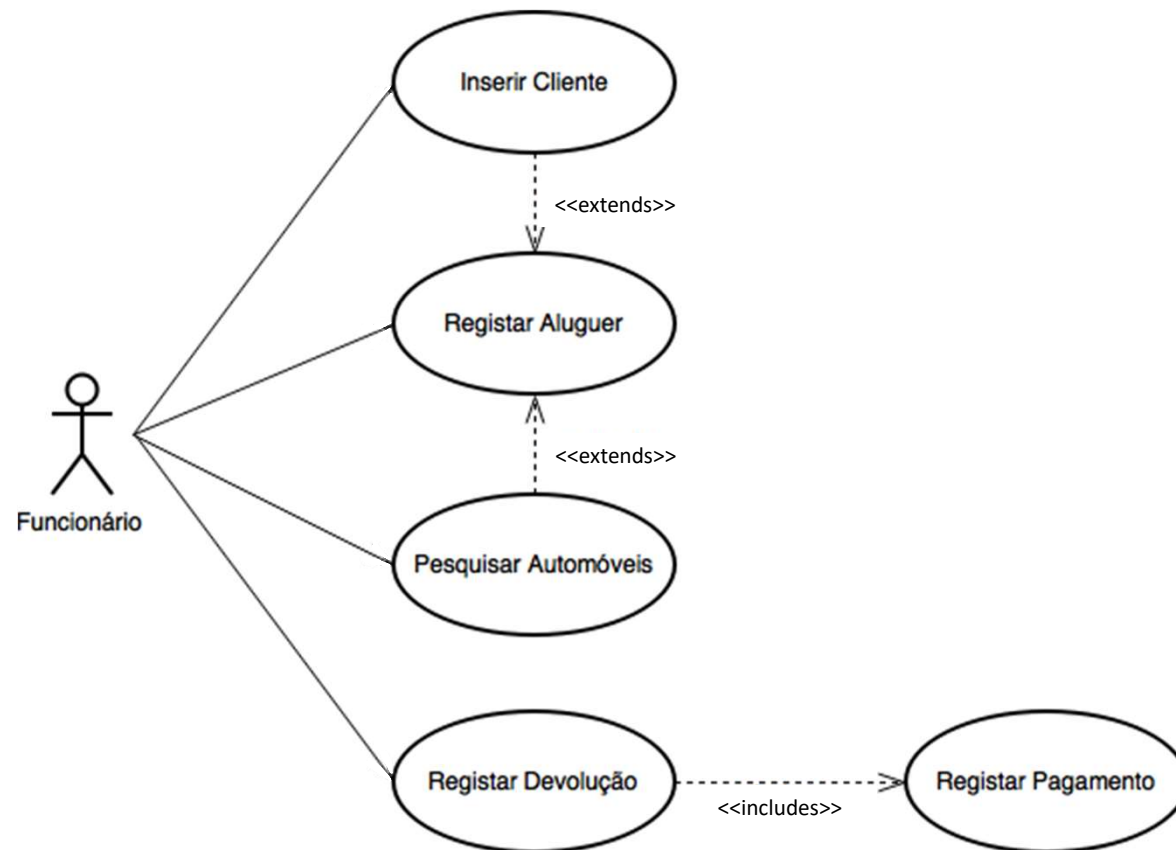
Como criar um Diagrama de Casos de Uso

- Identificar os atores envolvidos
 - Identificar todas as entidades externas que interagem com o sistema
- Identificar os casos de uso
 - Identificar as funcionalidades ou recursos que o sistema oferece aos atores
- Desenhar os diagramas
 - Usando símbolos gráficos para representar atores, casos de uso e suas relações
- Documentar os detalhes
 - Fornecer descrições claras para cada ator e caso de uso

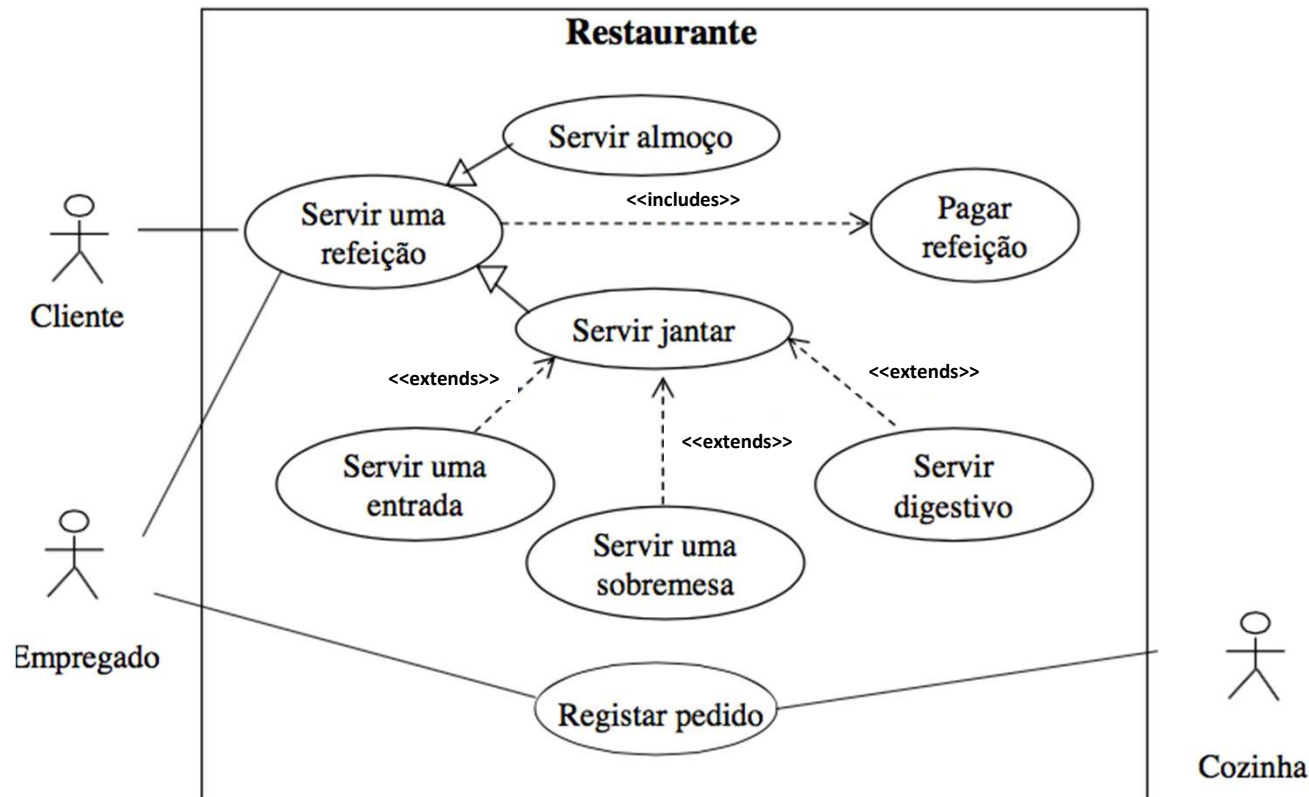
Exemplo de um Diagrama de Casos de Uso



Exemplo de um Diagrama de Casos de Uso



Exemplo de um Diagrama de Casos de Uso



Exercício prático em Grupo

- Mãos a obra...

Exercícios

- Analisar e desenhar um possível Diagrama de Casos de Uso dos seguintes Web Sites:
 - idealista - <https://www.idealista.pt/>
 - Wook - <https://www.wook.pt/>

Diagrama de Classes

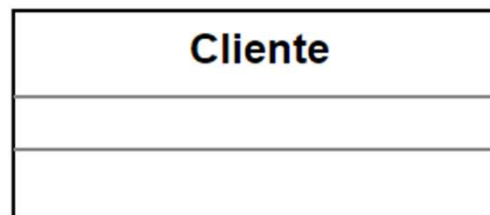
- O *Classes Diagram* é uma representação visual da estrutura e das relações das classes de um sistema
- Descreve as classes, seus atributos, métodos e as relações entre as classes
- Fornece uma visão detalhada da organização e estrutura do sistema

Elementos do Diagrama de Classes

- Classes
 - Atributos
 - Métodos
 - Visibilidades
- Relações
 - Dependência
 - Multiplicidade
 - Associação
 - Agregação
 - Composição
 - Generalização/Herança
 - Classe Associativa

Classes

- Uma classe é uma abstração que descreve um conjunto de objetos com características semelhantes
- As classes incluem no seu interior os atributos e os métodos
- É representada por um retângulo dividido em 3 partes, e parte superior é escrito o nome da classe
- Exemplo:



Atributos

- Atributos são características ou propriedades de uma classe
- São usados para armazenar os dados dos objetos de uma classe
- São representados na região central da classe
 - Syntax: Nome_atributo:tipo_de_dados
- Exemplo:

Cliente
- nome: string - idade: int

Métodos

- Métodos são as ações ou operações que uma instância de uma classe pode executar
- São listados no meio da classe e incluem o nome do método, parâmetros e tipo de dados do retorno
- Exemplo:

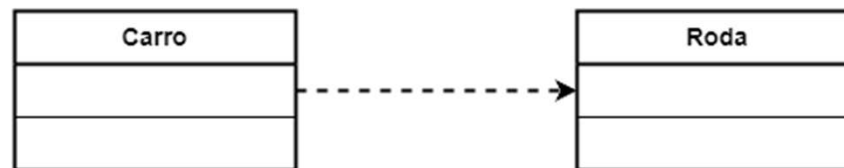
Cliente
- nome: string - idade: int
+ realizarCompra(produto: Produto): void

Visibilidades de atributos e métodos

- Representamos a visibilidade dos atributos e dos métodos usando os modificadores de acesso a seguir:
 - + Público
 - - Privado
 - # Protegido
 - ~ Pacote
 - / Derivado

Relação de Dependência

- Ilustra que uma classe usa ou precisa de informações ou serviços de outra classe em algum momento para que ela possa operar corretamente
- Do tipo: Classe A depende da Classe B
- A Classe A pode existir sem a Classe B e vice-versa
- Porém a Classe A precisa da Classe B para funcionar corretamente



Relações - Multiplicidade

- É usado para determinar o número mínimo e o número máximo de objetos envolvidos numa relação
- Pode também especificar o nível de dependência entre os objetos

Multiplicidade	Significado
0..1	No mínimo 0 e no máximo 1. Indica não obrigatoriedade de um relacionamento
1..1 <u>ou</u> 1 <u>ou</u> (nada)	1 e somente 1. Uma instancia da classe se relaciona com uma instancia de outra
0..*	Mínimo 0 (ou seja nenhum) e no máximo muito
1..*	Mínimo 1 e no máximo muitos
*	Muitos
x..y	No mínimo <u>x</u> e no máximo <u>y</u> . Ou seja, no mínimo <u>x</u> e no máximo <u>y</u> instancias envolvidas na relação. Ex.: 4..7

Relação de associação

- Indica que a classe mantém uma referência a outra classe ao longo do tempo
- As associações podem conectar mais de duas classes
- Do tipo: Classe A tem uma Classe B
- A Associação pode ter um nome (geralmente um verbo) e possui multiplicidade



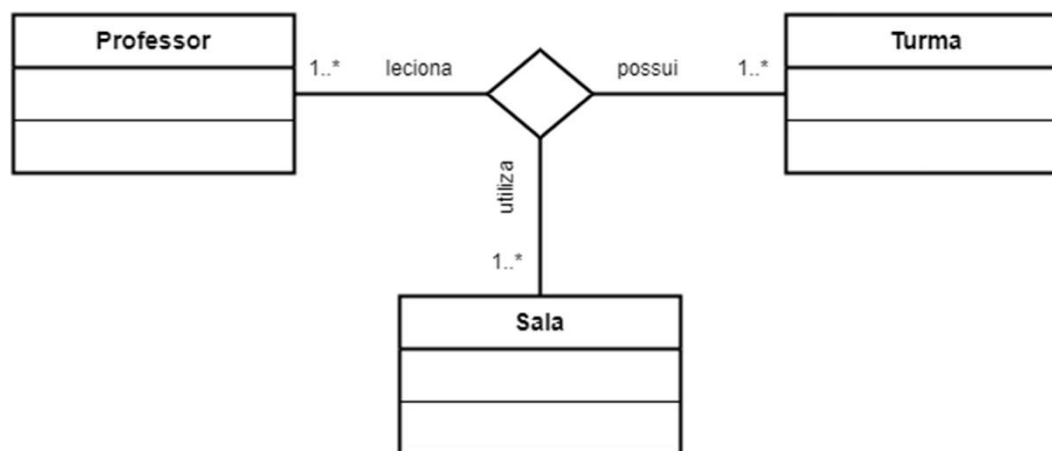
Relação de associação (unidirecional)

- Por vezes podemos ter a Navegabilidade
- É representada por uma seta, que identifica o sentido em que as informações são transmitidas entre os objetos das classes relacionadas



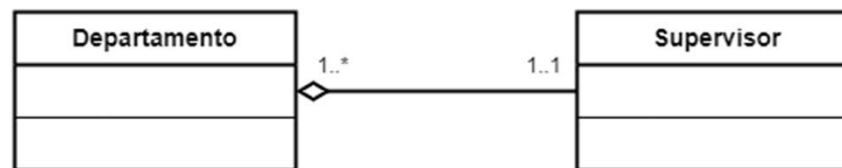
Relação de associação (ternária)

- Associação que conecta objetos de três classes
- Um losango indica o ponto de convergência/conexão das classes envolvidas
- Na prática, podemos ter associações n-árias, conectando n número de classes



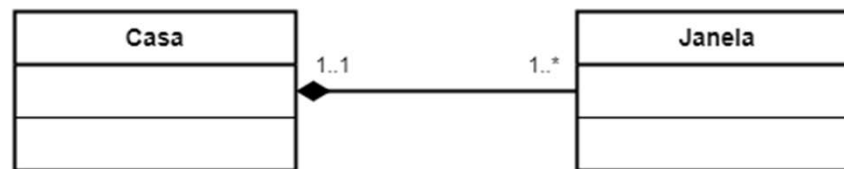
Relação de Agregação

- Uma relação mais específica que a de associação
- Indica que uma classe é um “container” de outra classe
- A classe contida não depende do “container” (então, se o “container” for destruído, a classe contida continua a existir)
- Do tipo: Classe A possui uma (ou mais) Classe B



Relação de Composição

- Variação mais específica da agregação
- Indica uma dependência de ciclo de vida forte entre as classes
- Então quando um “container” é destruído, a classe contida é também destruída
- Do tipo: Classe A é parte da classe B

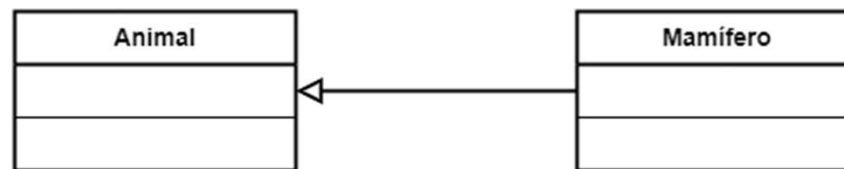


Relação de Generalização/Herança



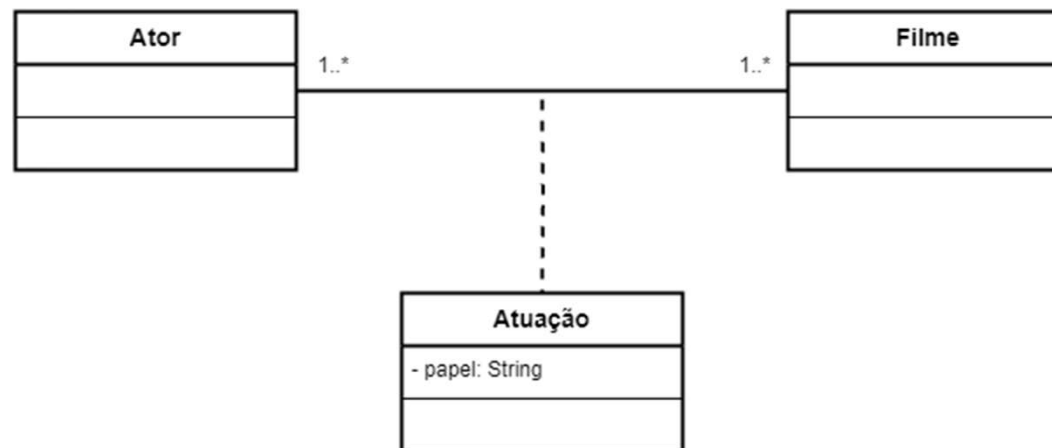
Centro para o Desenvolvimento
de Competências Digitais

- Relação entre um classe geral (superclasse/classe-mãe) e uma classe mais especifica dessa classe geral (subclasse/classe-filha)
- Indica a herança entre classes
- Do tipo: Classe A é do tipo classe B
- A classe filha herda atributos e métodos da classe mãe, alem de poder possuir os seus próprios atributos e métodos



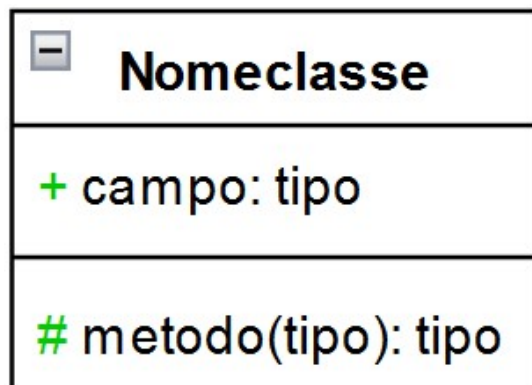
Classe Associativa

- São produzidas quando ocorrem associações de multiplicidade muitos (1..* ou *) em todas as extremidades
- No geral, existe atributos da associação que não podem ser armazenados em nenhuma das classes envolvidas

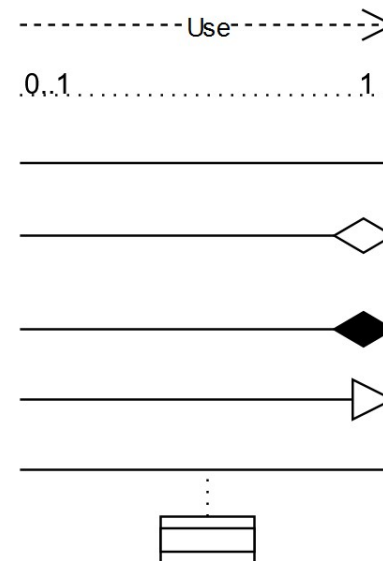


Elementos do Diagrama de Classes (Resumo)

- Classes
 - Atributos
 - Métodos
 - Visibilidades



- Relações



- Dependência
- Multiplicidade
- Associação
- Agregação
- Composição
- Generalização/Herança
- Classe Associativa

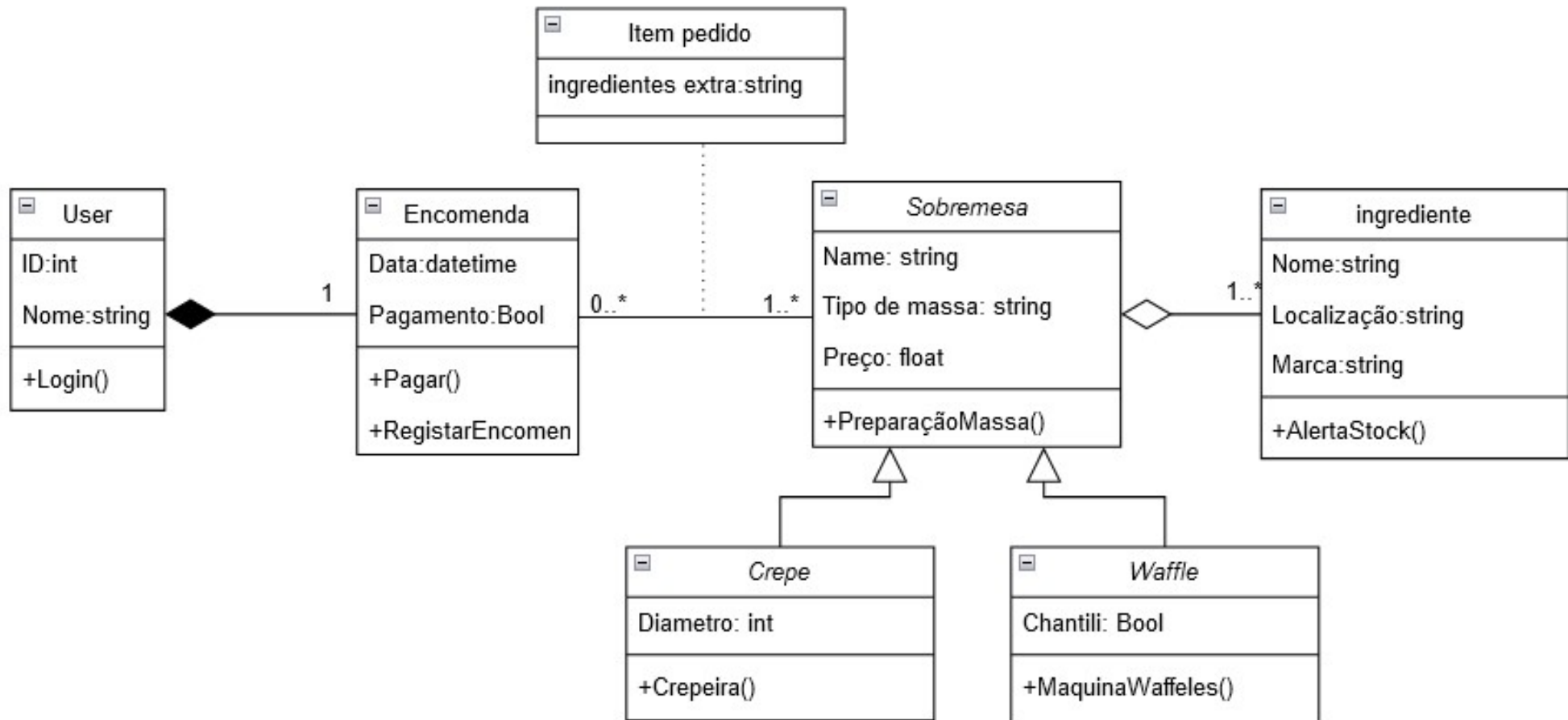
Como criar um Diagrama de Classes



Centro para o Desenvolvimento
de Competências Digitais

- Identificar as classes principais
 - Identificar as principais estruturas abstratas do sistema
- Identificar os atributos e métodos
 - Defina as características e ações de cada classe
- Identifique as relações
 - Identifique as associações, heranças e dependências entre as classes
- Desenhar o diagrama
 - Usando símbolos gráficos para representar as classes, atributos, métodos e relações
- Documentar os detalhes
 - Fornecer descrições claras para cada classe, atributo e método

Exemplo de um Diagrama de Classes



Exercício prático em Grupo

- Mãos a obra...