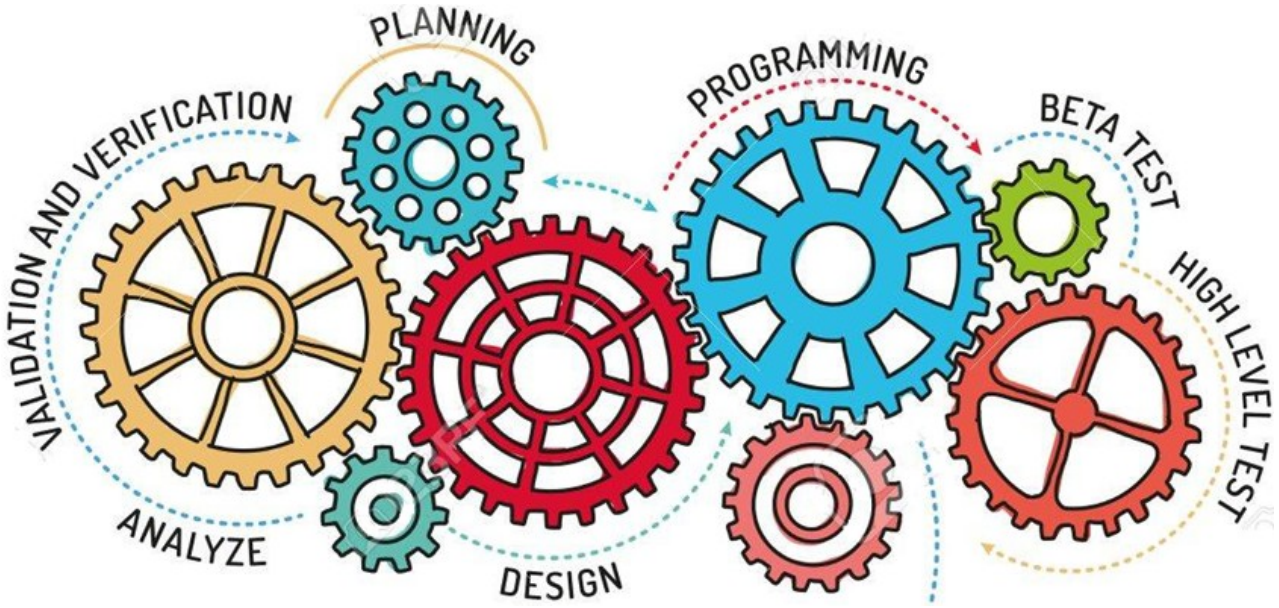
Engenharia de Software



Prof. Me. Clênio Silva
e-mail: clenio.silva@uniube.br

Uniube

UML Linguagem de Modelagem Unificada

- Desenvolvida por Grady Booch, James Rumbaugh e Ivar Jacobson;
- UML padronização de notações e desenvolvimento de novos conceitos para modelagem orientada a objetos.

Objetivos da UML

- Modelagem de sistemas (não apenas de software) usando os conceitos da orientação a objetos;
- Estabelecer uma união fazendo com que métodos conceituais sejam também executáveis;
- Criar uma linguagem de modelagem usável tanto pelo homem quanto pela máquina.

Fases do Desenvolvimento de um Sistema em UML

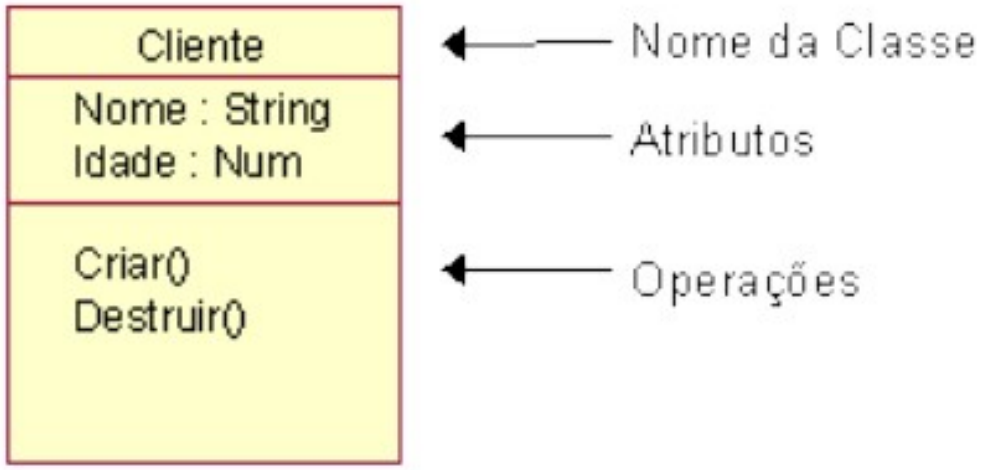
- **Análise de Requisitos:**
  - Captura as intenções e necessidades dos usuários do sistema a ser desenvolvido através do uso de funções chamadas “use-cases”. Através do desenvolvimento de “use-case”, as entidades externas ao sistema que interagem e possuem interesse no sistema são modelados entre as funções que eles requerem, funções estas chamadas “use-cases”.
- **Análise:**
  - Preocupada com as primeiras abstrações (classes e objetos) e mecanismos que estarão presentes no domínio do problema. As classessão modelados e ligadas através de relacionamentos com outras classes, e são descritas no Diagrama de classes.
- **Design (Projeto):**
  - O resultado da análise é expandido em soluções técnicas. Novas classes serão adicionadas para prover um infra-estrutura técnica: a interface do usuário e de periféricos, gerenciamento de banco de dados, comunicação com outros sistemas, dentre outros.

Fases do Desenvolvimento de um Sistema em UML

- **Programação:**
  - As classes provenientes do design são convertidas para o código da linguagem orientada a objetos.
- **Testes:**
  - Execução de testes de unidade, integração e aceitação. Testes de unidade são para classes ou grupos de classes e geralmente são testados pelo programador. Teste de integração são aplicados usando classes e componentes integrados para se confirmar se as classes estão cooperando uma com as outras como apresentado nos modelos. Testes de aceitação observam o sistema como um “caixa preta” e verificam se o sistema esta funcionando como o especificado nos primeiro digramas de “use-cases”.

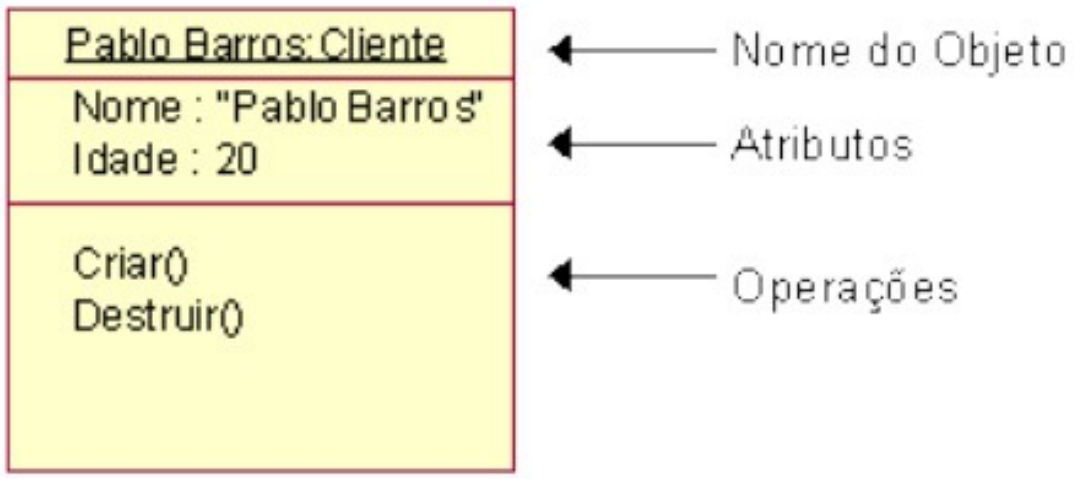
Modelos de Elementos

- **Classes:**



Modelos de Elementos

- **Objetos:**

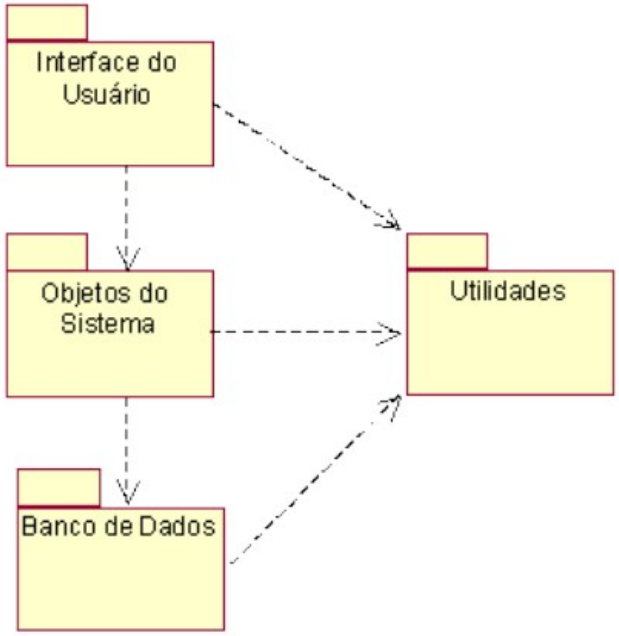


Modelos de Elementos

- **Estados:**
  - Significa o resultado de atividades executadas pelo objeto, e é normalmente determinada pelos valores de seus atributos e ligações com outros objetos.

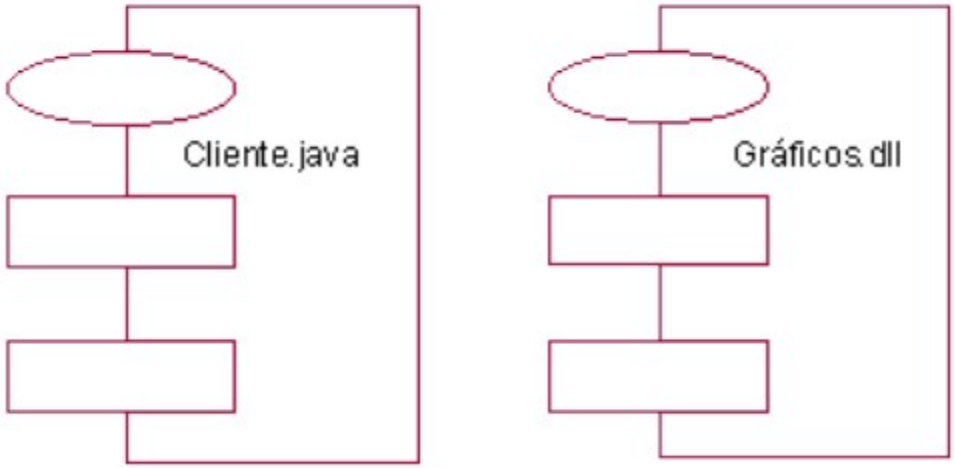
Modelos de Elementos

- **Pacotes:**
  - Mecanismo de agrupamento, onde todos os modelos de elementos podem ser agrupados. Ex. Um pacote que contem classes java para conexão com banco de dados.



Modelos de Elementos

- **Componentes:**
  - Um componente pode ser tanto um código em linguagem de programação como um código executável já compilado.

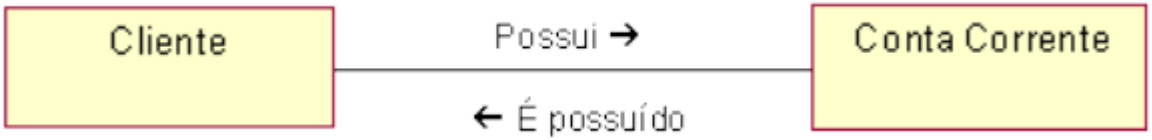


Relacionamentos

- **Os relacionamentos ligam as classes/objetos entre si criando relações lógicas entre estas entidades. Os relacionamentos podem ser dos seguintes tipos:**
  - **Associação:** É uma conexão entre classes, e também significa que é uma conexão entre objetos daquelas classes. Ex. *Uma classe cliente contendo um atributo endereco do tipo da classe Endereco.*
  - **Generalização:** É um relacionamento de um elemento mais geral e outro mais específico. O elemento mais específico pode conter apenas informações adicionais. Uma instância (um objeto é uma instância de uma classe) do elemento mais especifico pode ser usado onde o elemento mais geral seja permitido. Ex. *Herança em POO com Java*.
  - **Dependência e Refinamentos:** Dependência é um relacionamento entre elementos, um independente e outro dependente. Uma modificação em um elemento independente afetará diretamente elementos dependentes do anterior. Refinamento é um relacionamento entre duas descrições de uma mesma entidade, mas em níveis diferentes de abstração.

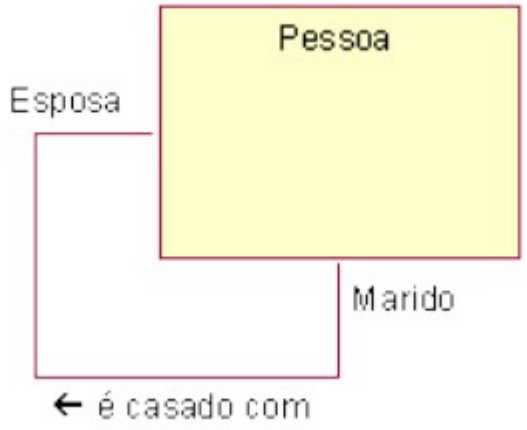
Associações

- **Associações Normais:**
  - É representada por uma linha sólida entre duas classes;
  - A associação possui um nome (junto à linha que representa a associação), normalmente um verbo, mas substantivos também são permitidos;
  - Para expressar a multiplicidade entre os relacionamentos, um intervalo indica quantos objetos estão relacionados no link. O intervalo pode ser zero para um (0..1), zero para vários (0..*) ou apenas *, um para vários (1..*), dois (2), cinco para 11 (5..11) e assim por diante. Se não for descrito nenhuma multiplicidade, então é considerado o padrão de um para um (1..1) ou apenas 1.



Associações

- **Associações Recursiva:**
  - É possível conectar uma classe a ela mesma através de uma associação e que ainda representa semanticamente a conexão entre dois objetos, mas os objetos conectados são da mesma classe. Uma associação deste tipo é chamada de associação recursiva.

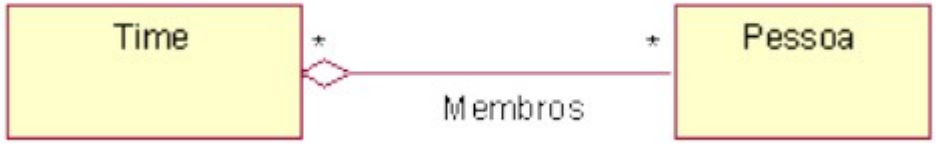


Associações

- **Agregação:**
  - A agregação é um caso particular da associação. A agregação indica que uma das classes do relacionamento é uma parte, ou esta contida em outra classe. A palavras chaves usadas para identificar uma agregação são: “consiste em”, “contém” e “é parte de”.

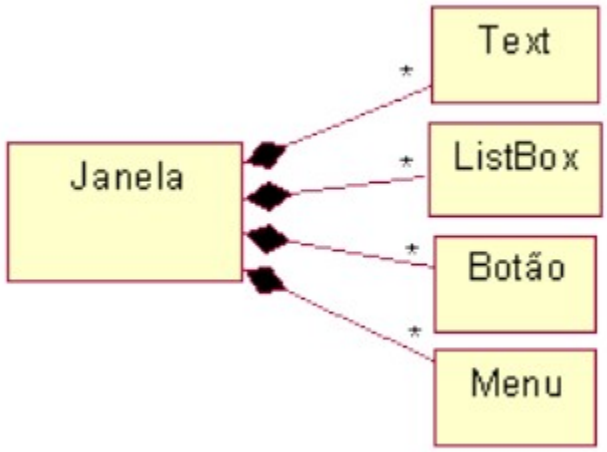
Associações

- **Agregação Compartilhada:**
  - É dita compartilhada quando uma das classes é uma parte, ou está contida na outra, mas esta parte pode fazer estar contida na outra várias vezes em um mesmo momento. No exemplo a seguir uma pessoa pode ser membro de um time ou vários times em um determinado momento.



Associações

- **Agregação de Composição:**
  - É uma agregação onde uma classe que está contida na outra “vive” e constitui a outra. Se o objeto da classe que contém for destruído, as classes da agregação de composição serão destruídas juntamente, já que as mesmas fazem parte da outra.

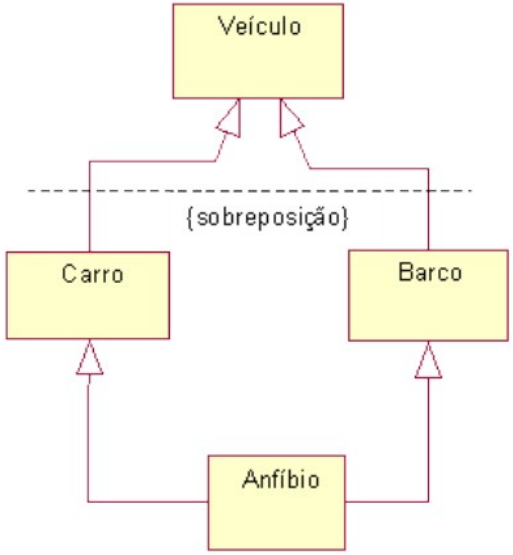


Generalizações

- **Generalização Normal:**
  - Na generalização normal a classe mais específica, chamada de subclasse, herda tudo da classe mais geral, chamada de superclasse. Os atributos, operações e todas as associações são herdadas.

Generalizações

- **Generalização Restrita:**
  - Uma restrição aplicada a uma generalização especifica informações mais precisas sobre como a generalização deve ser usada e estendida no futuro. As restrições a seguir definem as generalizações restritas com mais de uma subclasse:
    - Generalizações de sobreposição:



Generalizações

- **Generalização Restrita:**
  - Uma restrição aplicada a uma generalização especifica informações mais precisas sobre como a generalização deve ser usada e estendida no futuro. As restrições a seguir definem as generalizações restritas com mais de uma subclasse:
    - Generalizações Completa e Incompleta:

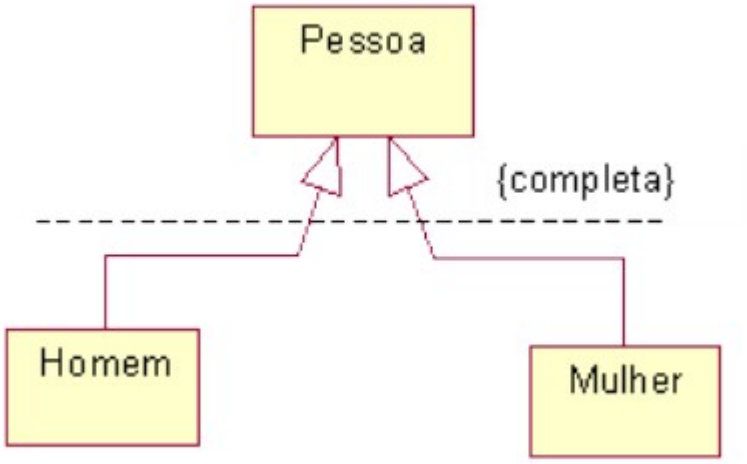


Diagrama de Classes