

PROGRAMAÇÃO ORIENTADA A OBJETOS

Prof. Bernardo Copstein

Profa. Isabel H. Manssour

UML E RELACIONAMENTO ENTRE CLASSES

Leitura recomendada:

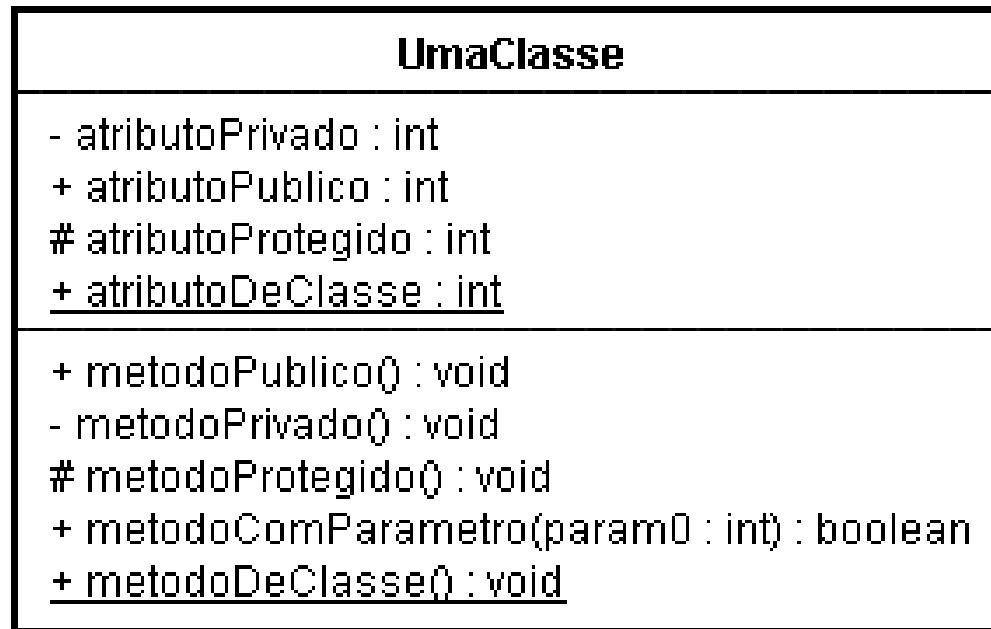
- Java for Everyone; Cay Horstmann; seção 12.2
- The Object Oriented Thought Process; Matt Weisfeld; capítulo 9
- UML Essential; Martin Fowler

UML

- *Unified Modeling Language* (<http://www.uml.org/>)
 - Linguagem visual para especificação, visualização, documentação e construção de sistemas de software
 - Padrão para modelagem orientada a objetos
 - Composta por vários diagramas
- Diagrama de Classes
 - Denota a estrutura estática do sistema
 - Representação das classes e seus relacionamentos

UML

- Representação de uma classe

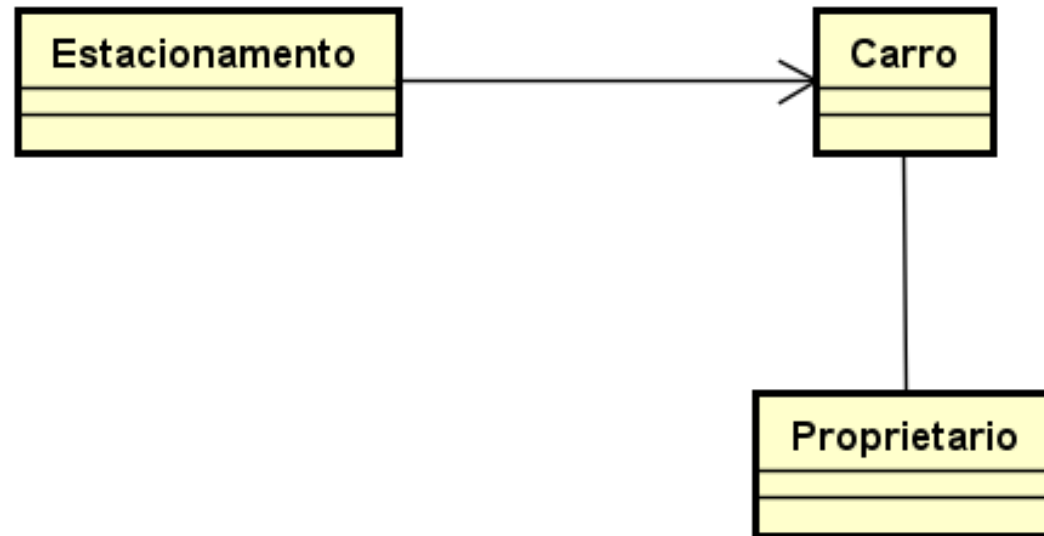


UML

- Em um programa orientado a objetos encontramos dois tipos de relacionamentos:
 - Relacionamentos entre classes:
 - Explicitam a forma pela qual os objetos são compostos, mas uma vez definidos não existe interação entre eles.
 - Relacionamentos entre objetos:
 - Representam interações entre diferentes objetos

UML

- Exemplo:



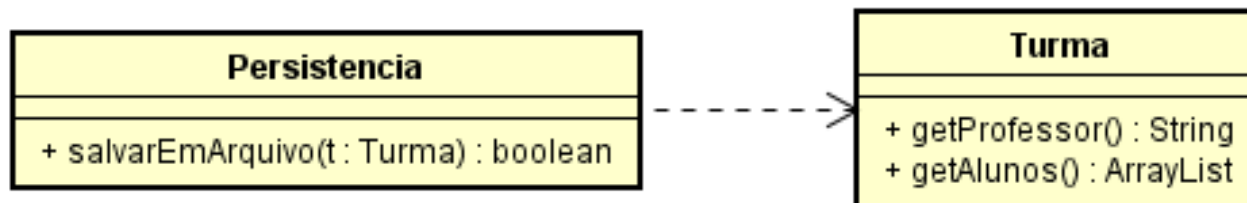
- O relacionamento entre *Estacionamento* e *Carro* ou entre *Carro* e *Proprietário* são relacionamentos entre objetos.
- Relacionamentos entre classes serão vistos na sequencia.

UML

- Relacionamentos básicos
 - **Dependência:**
 - Não trocam mensagens mas existe uma relação de dependência.
 - Usualmente implica na **implementação através de parâmetros ou criação de instâncias**

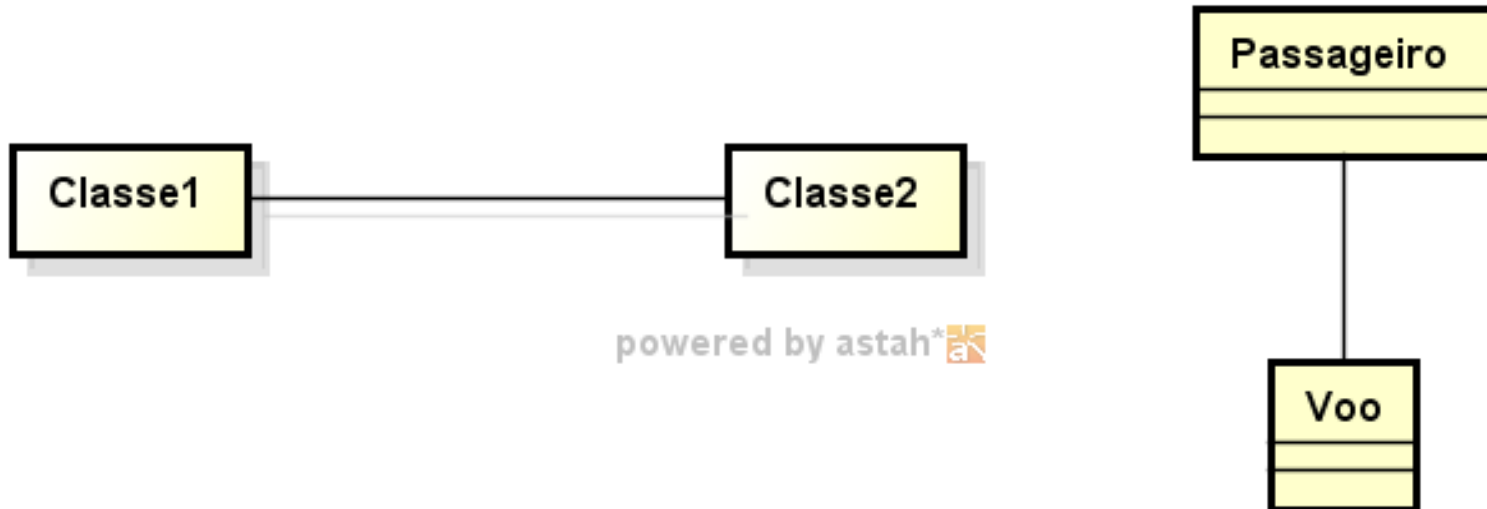


powered by astah*



UML

- Relacionamentos básicos
 - **Associação:**
 - Termo genérico para indicar que existe interação entre os objetos
 - Usualmente implica na **implementação via atributos**

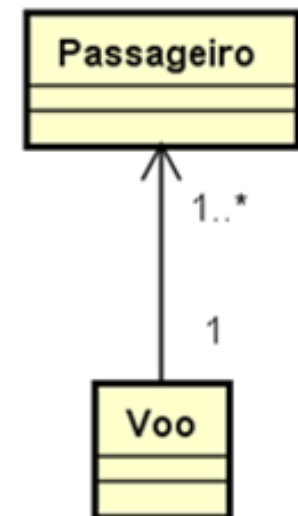


UML

- Relacionamentos básicos

- **Associação:**

- Expressa uma relação “container” – “contido”, indicando que é o “container” e que está “contido” (quem usa quem).
- Utiliza-se a seta de navegação direcional para indicar qual objeto possui referência(s) para outro(s) objeto(s).
- Pode conter um nome e especificação de multiplicidade (qualquer forma de associação pode expressar cardinalidade)

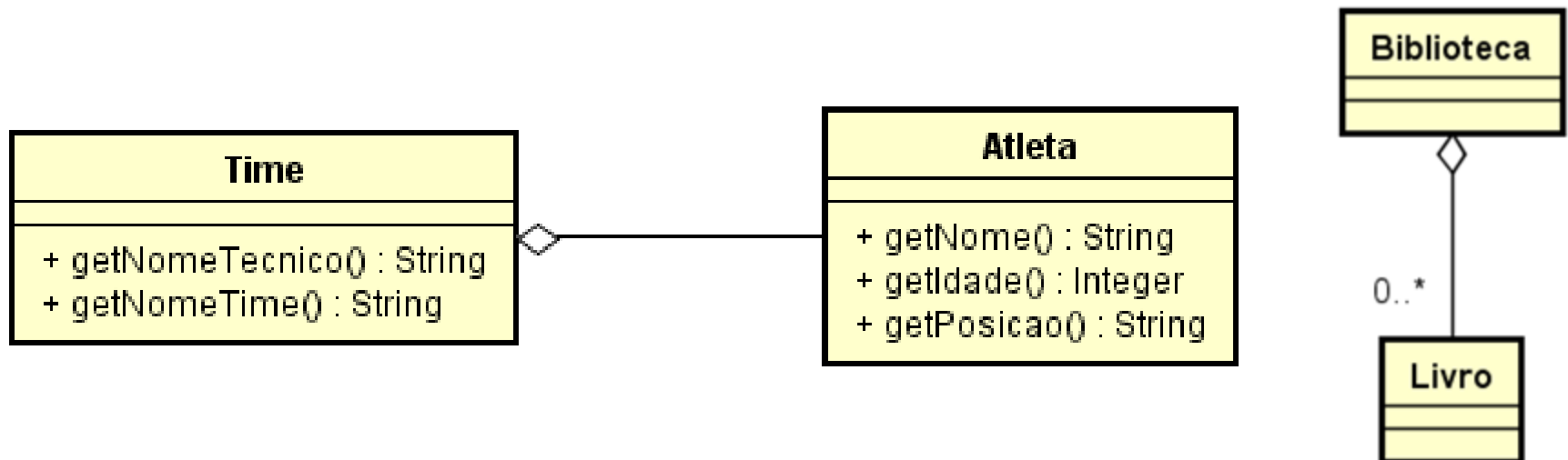


UML

- Relacionamentos básicos

- **Agregação:**

- Revela uma associação “todo-parte” onde as partes tem ciclo de vida independente do agregador
 - Objeto (o “todo”) contém referências para outros objetos (as “partes”)

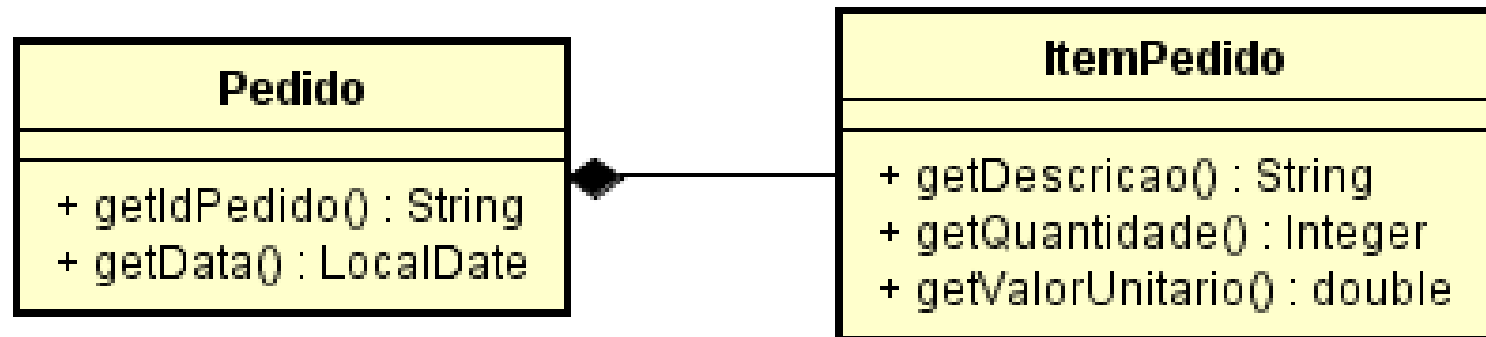


UML

- Relacionamentos básicos

- **Composição:**

- Agregação na qual o ciclo de vida das partes depende do ciclo de vida do agregador



Objetos


- Blocos de construção de um programa O.O
 - Um programa O.O. é basicamente uma coleção de objetos que se relacionam

- Compreendem:

Estado (definido por seus atributos)

+

Comportamento

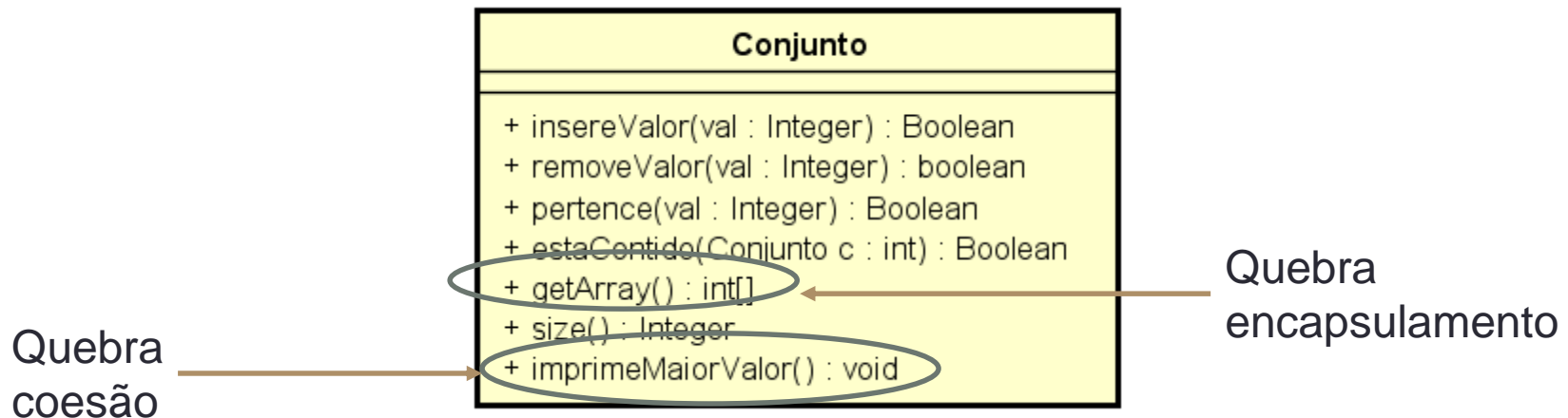
- Conjunto de métodos públicos ou
 - Operações que disponibiliza
- 
- “interface”

Características importantes

- Coesão
 - Objetos tem uma única responsabilidade claramente definida
 - Todos os métodos estão relacionados a essa única responsabilidade
 - As classes devem ser coesas:
 - Específicas para desempenhar um papel em um contexto
 - Se as responsabilidades não são relacionadas divide-se a mesma em novas classes.

Características importantes

- Encapsulamento
 - Garante que somente a interface é visível
 - Minimiza as interdependências entre módulos
 - Implementado através dos modificadores de acesso



Características importantes

- Acoplamento
 - Indica as interconexões (ou dependências) entre classes
 - Por exemplo:
 - Se uma classe A acessa atributos de outra classe B, há um acoplamento
 - Caso um atributo da classe B seja alterado, a classe A também terá que ser alterada

Características importantes

- Boas práticas na programação orientada a objetos para facilitar o desenvolvimento e manutenção de programas
 - Garantir o encapsulamento
 - Projetar classes com **baixo** acoplamento e **alta** coesão

