

Programação Orientada a Objetos

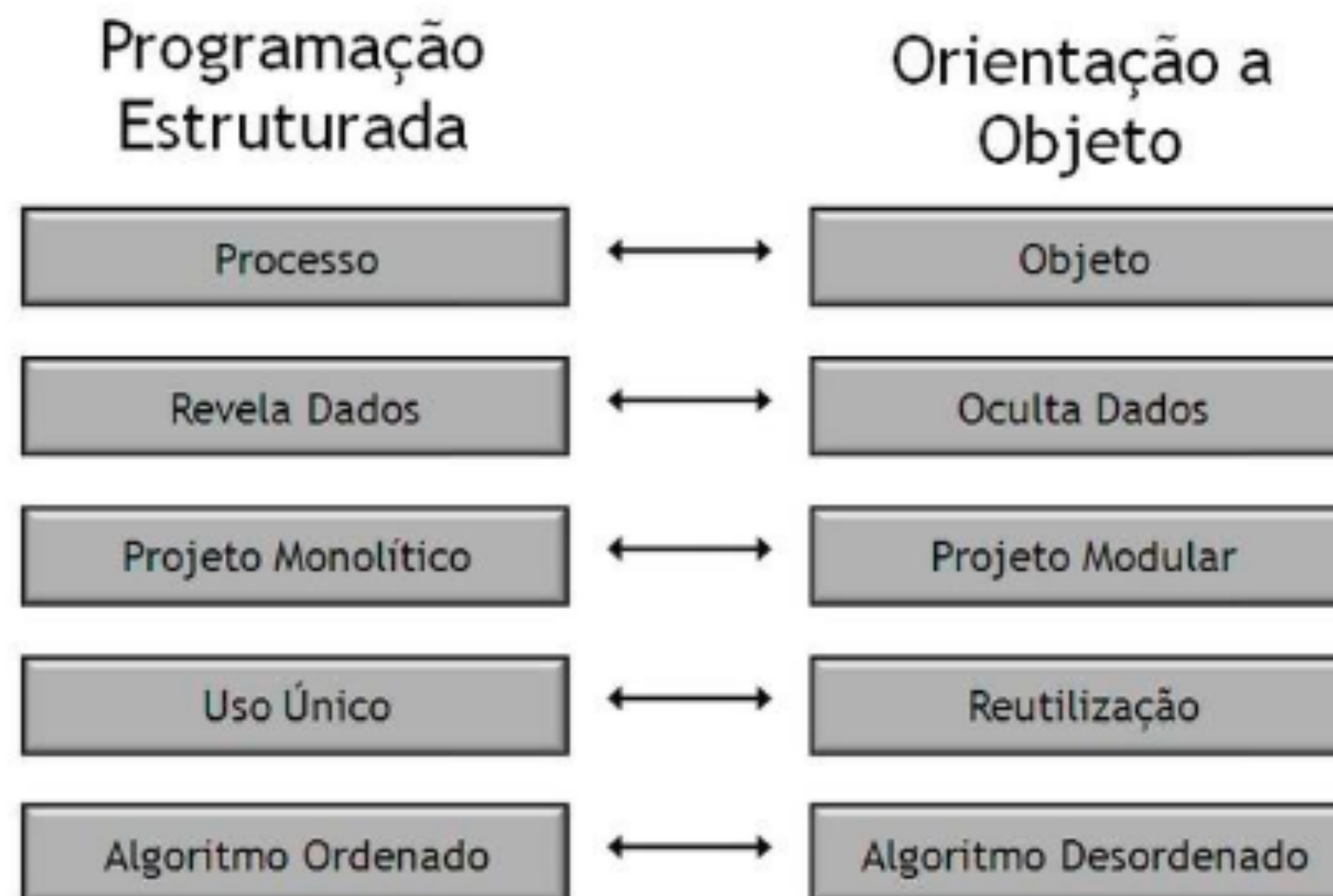
Prof. Dr. Wesin Ribeiro

Nesse capítulo

- Definição
- Classes e objetos
- Visibilidade
- Métodos especiais

Programação orientada a objetos é um paradigma para se criar **componentes reutilizáveis** de software por meio de **classes e objetos**. Geralmente, usamos classes e objetos para modelar **entidades** de um sistema especificando seus **atributos e comportamento**.

Programação estrutura x POO



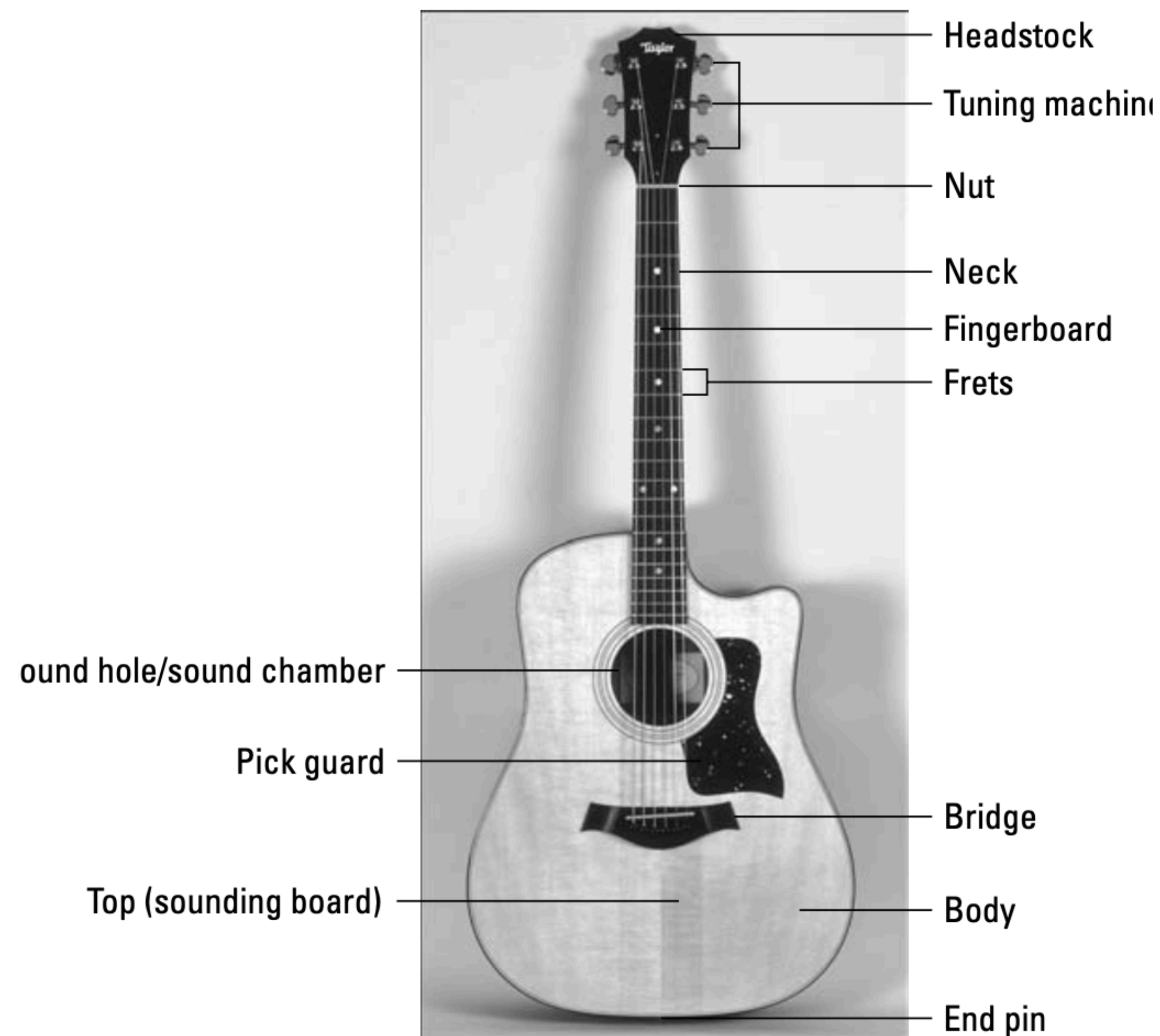
Um objeto é definido por meio de

- **Estado** - atributos e seus respectivos valores
- **Comportamento** - como o objeto reage a mudança de estados e relacionamento com os demais objetos do sistema
- **Identidade** - é a propriedade pela qual ele se distingue dos demais

Classe é uma estrutura que **abstrai um conjunto de objetos com características similares.**

O objeto violão

- Um violão pode ter vários formatos
- As cordas podem ser de nylon ou de aço
- Um violão possui vários componentes
- Cada componente é criado a partir de um molde



Fonte: Violão para Leigos, Phillips and Chappel

O violão jamais irá tocar sozinho. Então, o violonista deve usá-lo para fazer a música. O violonista pode dedilhar ou usar palheta para tocar seu violão.

Nesse contexto

- O molde do violão é a **classe**.
- O objeto violão em si é a **instanciação** de uma classe, ou seja, é um **objeto**.
- Assim como um violão possui um número de série, um objeto deve possuir um identificador.
- O modo de tocar o violão é o **método**
- As especificações do violão (marca, formato, tipo de corda, etc.) são os **atributos**.
- Os atributos e métodos são chamados de membros da classe.
- O “objeto” violonista se comunica com o objeto violão acessando seus membros, sem precisar conhecer os detalhes da fabricação. Isso se chama **encapsulamento**.

Herança

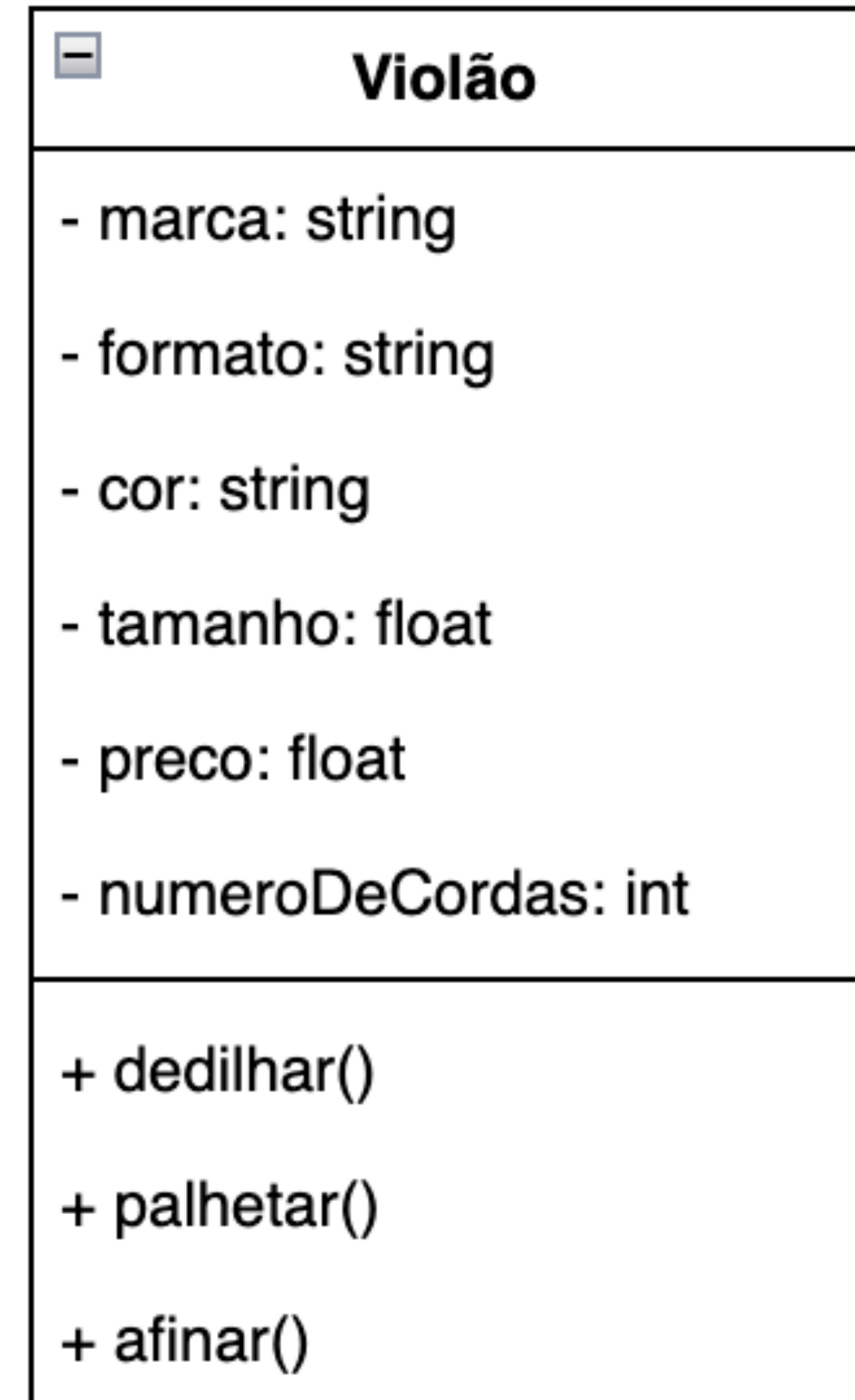
Quando uma classe é derivada de outra classe

- A classe mãe é chamada de **superclasse**
- A classe filha é chamada de **subclasse**
- Interfaces são coleções de métodos relacionados.
- Interfaces **não** dizem como implementar uma classe, apenas **o que** deve ser implementado.



Como representar?

Através do diagrama de classes da UML (unified modeling language)




Cada classe criada se torna um **novo tipo de dado** que você pode usar para criar objetos.

Visibilidad

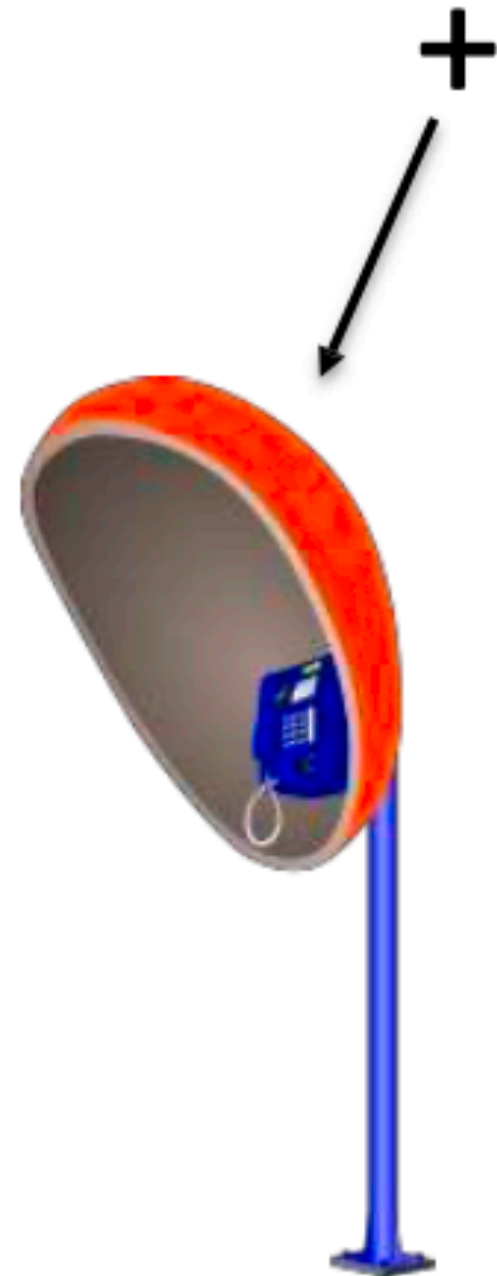
Modificadores de visibilidade dos membros de uma classe permite controlar seu nível de **encapsulamento.**

Tipos de modificadores

- + publico: visível para todas as classes
- - privado: visível somente para a classe atual
- # protegido: visível para classe atual e as respectivas sub-classes

 Violão
<div># marca: string</div> <div>- formato: string</div> <div>- cor: string</div> <div># tamanho: float</div> <div>- preco: float</div> <div>- numeroDeCordas: int</div>
<div>+ dedilhar()</div> <div>+ palhetar()</div> <div>+ afinar()</div>

Modificadores de visibilidade



Manter os atributos de uma classe com visibilidade privada e os métodos com visibilidade pública facilita a depuração de erros.

Métodos especiais

Tipos de métodos

- Construtores
- Destrutores
- Getters
- Setters

Métodos construtores

- Inicialização personalizada de membros da classe.
- Deve possuir o mesmo **nome** da classe.
- Não é recomendado chamar outros métodos dentro do construtor.
- O método construtor é chamado automaticamente na criação de um objeto.
- Se um método construtor for definido, não haverá um **construtor padrão**.
- Por padrão, construtores possuem visibilidade pública
- Não especificam **valor de retorno**

Violao
marca: string
- formato: string
- cor: string
tamanho: float
- preco: float
- numeroDeCordas: int
<<constructor>> Violao(marca: string)
+ dedilhar()
+ palhetar()
+ afinar()

A menos que a inicialização padrão de variáveis de instâncias seja aceitável, forneça um construtor personalizado para garantir a correta inicialização dos atributos quando um novo objeto for criado.

Métodos destrutores

- Chamados automaticamente quando o objeto está para ser destruído na memória
- Usado para liberar recursos alocados pelos construtores
- Acionar outros mecanismos necessários durante a fase de destruição
- Só pode haver um destrutor por classe
- Não recebe parâmetros.

Métodos getters

Métodos acessor

- Usado para verificar o valor de atributos privados ou protegidos
- Deve definir um valor de retorno apropriado, de acordo com o atributo.
- Controla como os dados são apresentados
- Não recebe parâmetros
- São usados por **clientes** da classe

Método setters

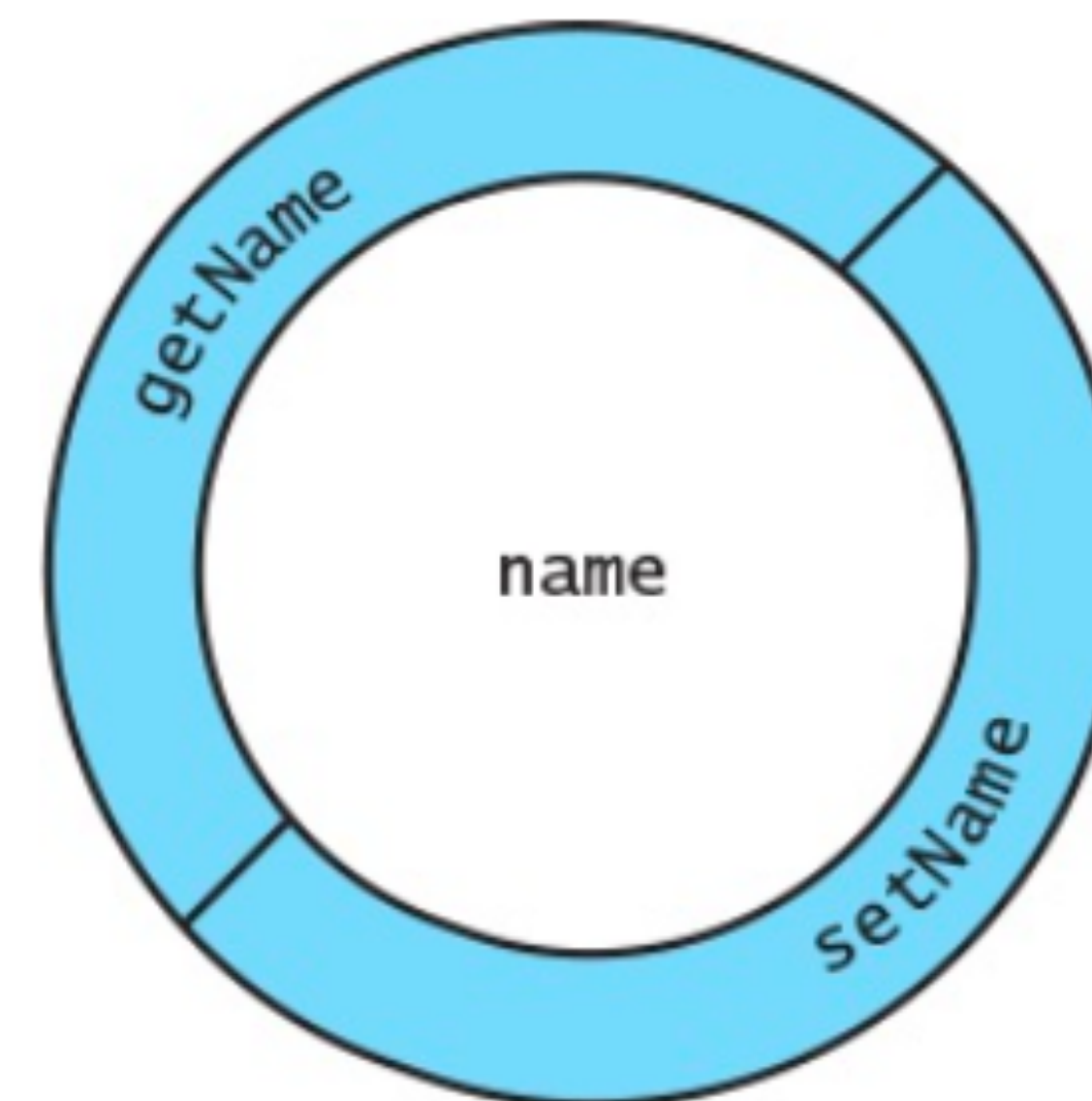
Método modificadores

- Usado para modificar algum atributo privado de uma classe
- Se um atributo privado não possuir um método set, ele não deve ser modificado
- Não há valor de retorno
- Recebe um parâmetro com o tipo apropriado
- São usados por **clientes** da classe

O controle adequado do acesso aos dados privados de um objeto ajuda a reduzir erros, aumentando a robustez, segurança e usabilidade de um programa.

Visão conceitual

Objeto com seus atributos privados encapsulados e protegidos por métodos públicos.



Fonte: C++ How to program, Deitel

Revisão

- Programação orientada a objetos é um paradigma de programação que modela as entidades do mundo real em classes e objetos.
- Os objetos possuem estado, comportamento e entidade
- As classes são moldes para objetos
- A visibilidade dos membros de classes podem ser pública, privada ou protegida
- Os métodos construtores são usados para inicialização personalizada dos atributos da classe
- Os métodos modificadores e assessores controlam e validam o acesso aos dados privados de uma classe.