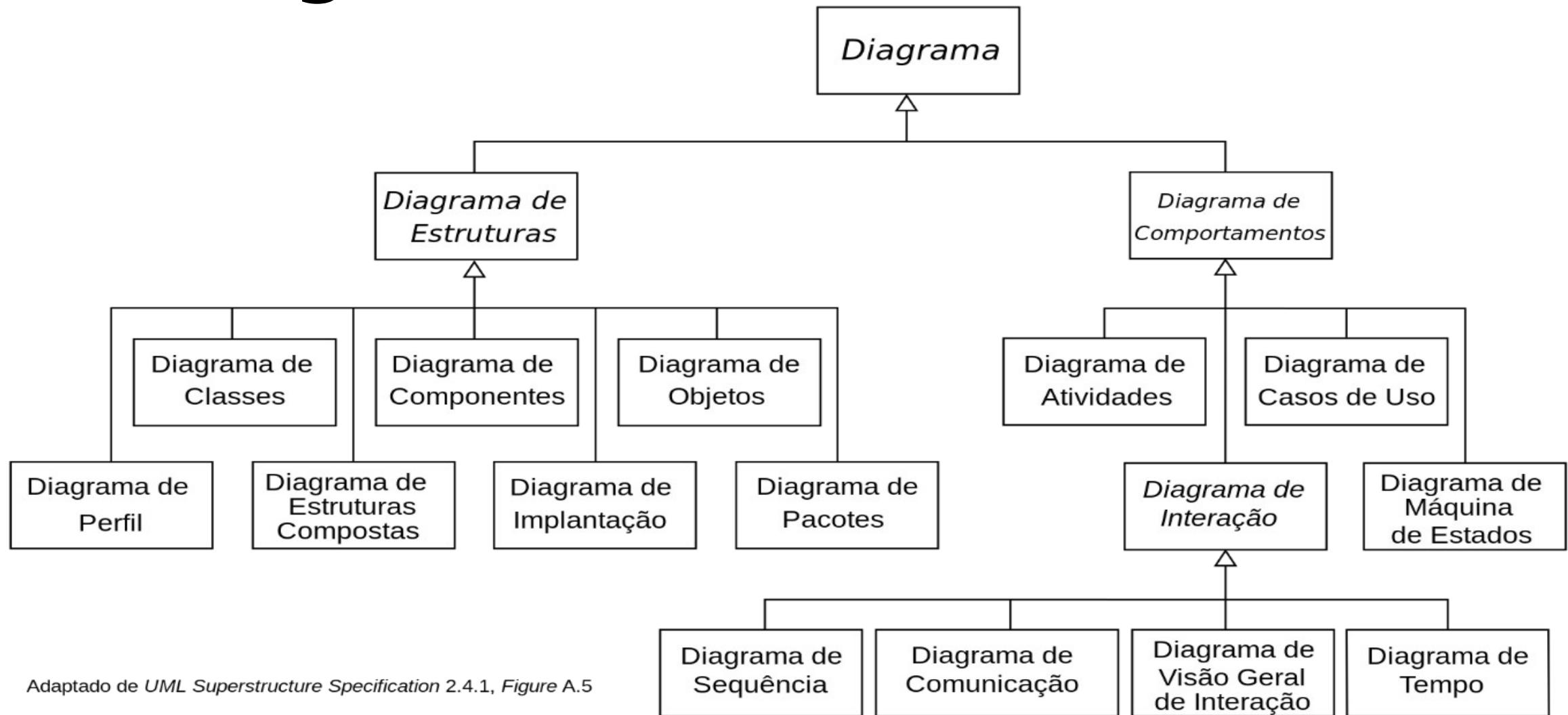


Engenharia de Requisitos de Software

3º período

Professora: Michelle Hanne Soares de Andrade
mhsandrade@sga.pucminas.br
michellehanne.andrade@gmail.com

Modelagem de Sistemas em UML

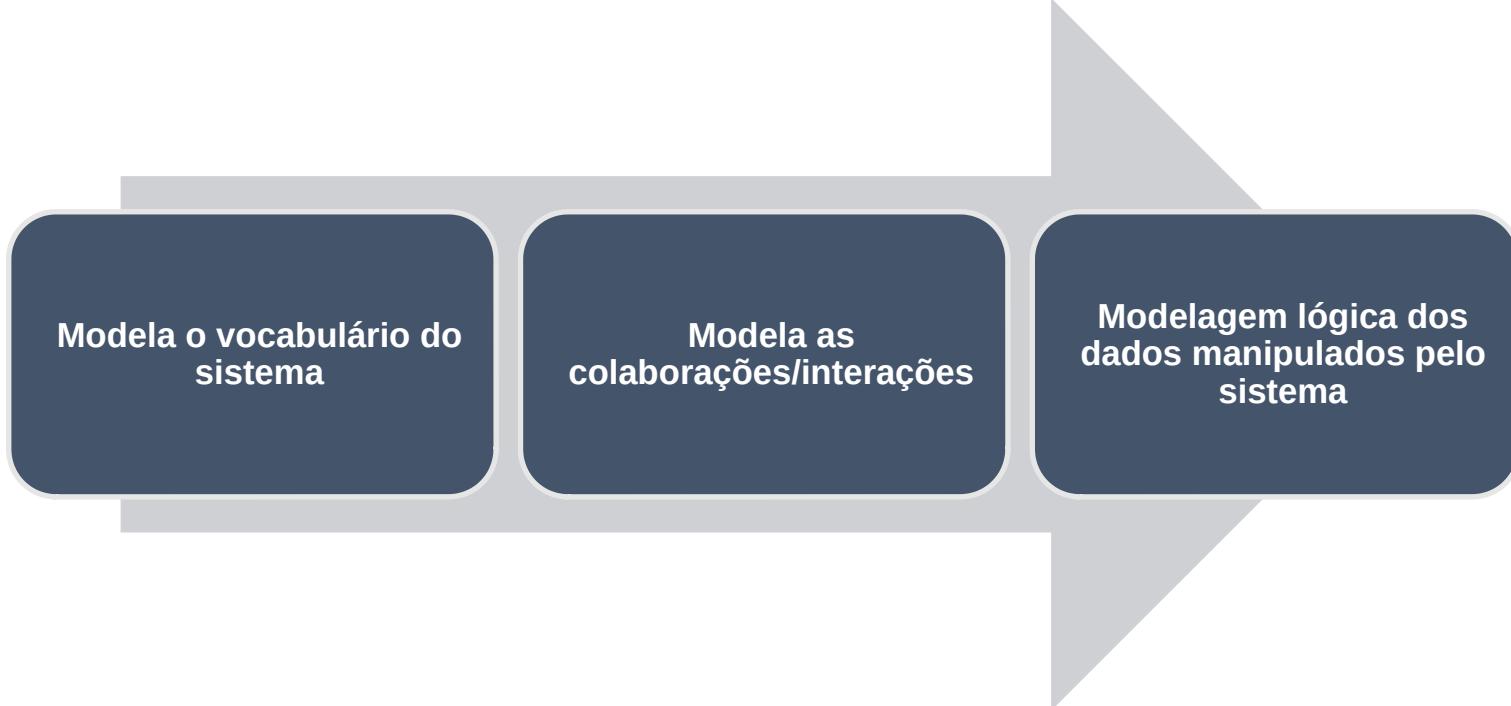


Adaptado de *UML Superstructure Specification 2.4.1, Figure A.5*

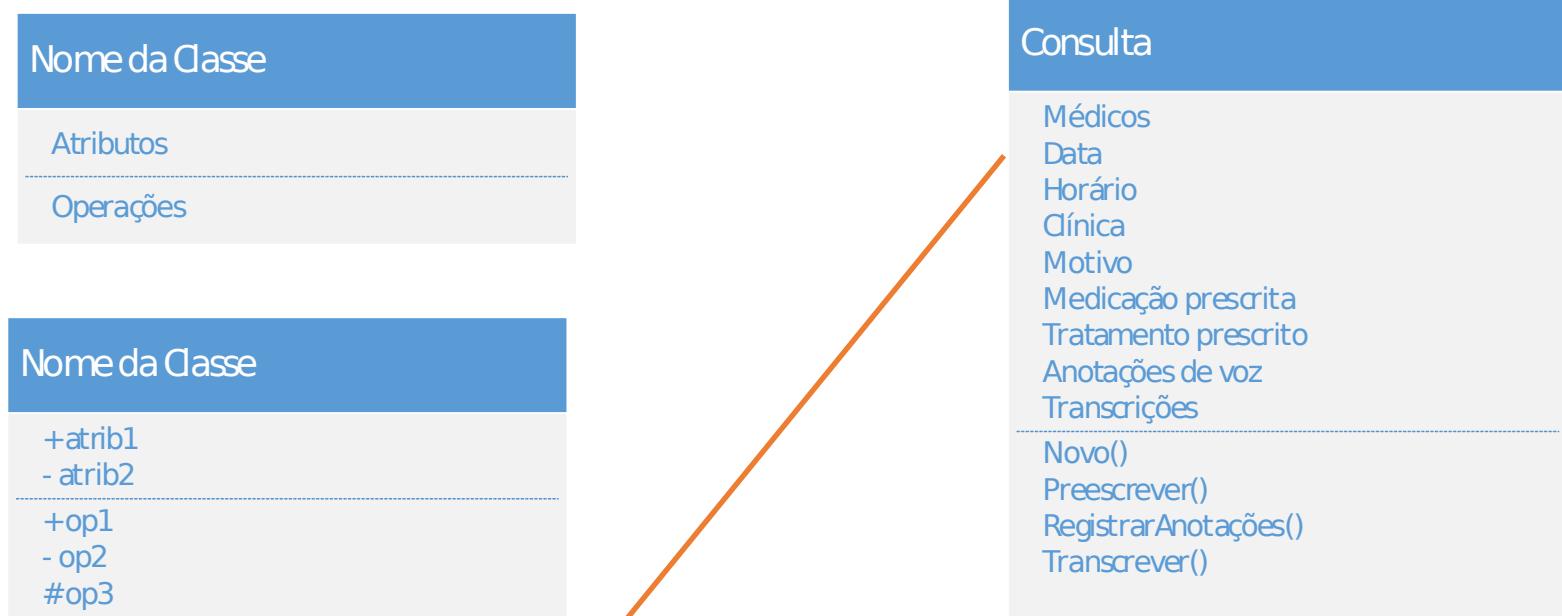
Diagrama de Classe

- Mostra a **estrutura do sistema**, subsistema ou componente projetado como **classes e interfaces** relacionadas, com seus **recursos, restrições e relacionamentos** - associações, generalizações, dependências, etc. (Sommerville, 2011).

Diagrama de Classe



Notação da Classe



Fonte: Sommerville (2011),
p. 91.

Classe é uma abstração de um conjunto de objetos com características similares.

Notação da Classe

Nome da Classe

+atrib1

- atrib2

+op1

- op2

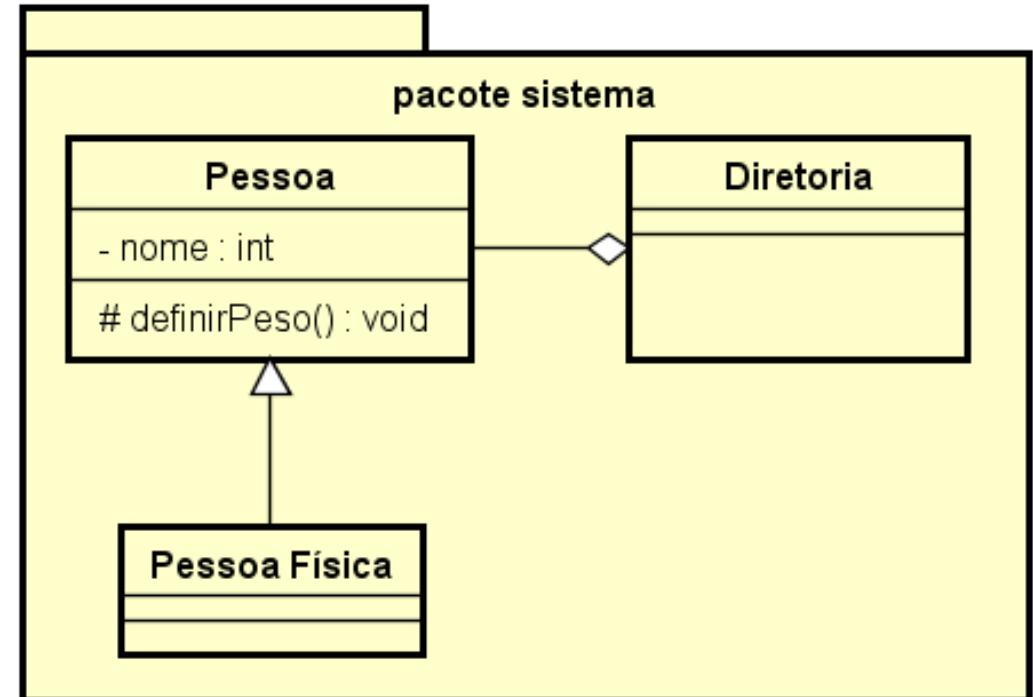
#op3

Indica o nível de acessibilidade de um atributo ou método:
+ Público
- Privado
Protegido

Notação da Classe

- (**private**): Atributos e métodos declarados como *private* são acessíveis somente pela classe que os declara. Métodos e atributos com o modificador *private* **não são herdados**.

(**Protected**): Atributos e métodos declarados como *protected* são acessíveis pela classe que os declara, suas subclasses em outros pacotes e outras classes dentro do mesmo pacote.



powered by Astah

Neste exemplo o método `definirPeso()` é visível na subclasse “Pessoa Física” e na classe “Diretoria” que está no mesmo pacote.

Notação da Classe

+ (Public): Atributos, métodos e classes declarados como public são acessíveis por qualquer classe do Java. Todos os métodos e atributos declarados como public são herdados pelas subclasses. Métodos e atributos declarados como public devem se manter public em todas as subclasses.

~ (Default): Modificador de acesso padrão, usado quando nenhum for definido. Neste caso os atributos, métodos e classes são visíveis por todas as classes dentro do mesmo pacote.

Atributo

- Permite a identificação de cada objeto de uma classe.
- Os valores dos atributos podem variar de instância para instância.
- Atributos devem conter o tipo de dados a ser armazenado: Byte, boolean, int, double, char, String, etc

Método

- São apenas declarados, não define a implementação.
- Outros diagramas permitem modelar o comportamento interno dos métodos: Diagrama de Sequência e Diagrama de Atividades.

Método

- Uma classe é representada por um retângulo com três divisões:
 - Nome da Classe
 - Atributos da Classe
 - Métodos da Classe

Pessoa
nome
email
enviarMensagem()

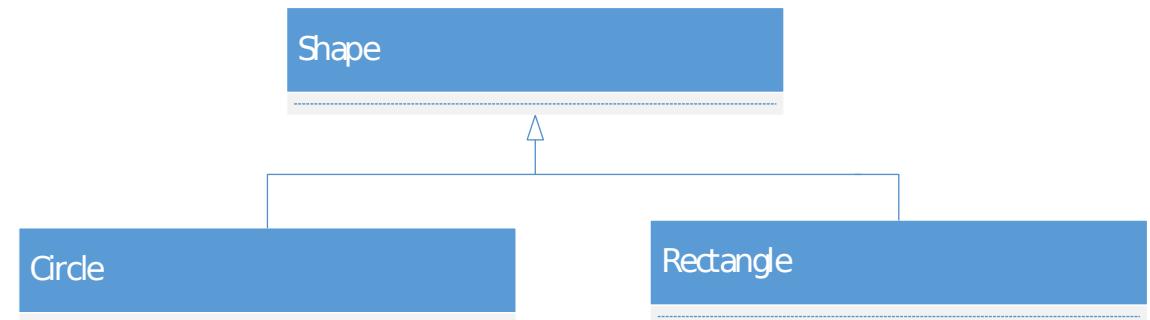
Relacionamentos

São conexões entre classes:

- 1. Dependência** - uma classe usa a outra.



- 2. Generalização** - geral (superclasse) e uma coisa mais específica (subclasse)



Relacionamentos

3. Associação - classes ou objetos estão interconectados.

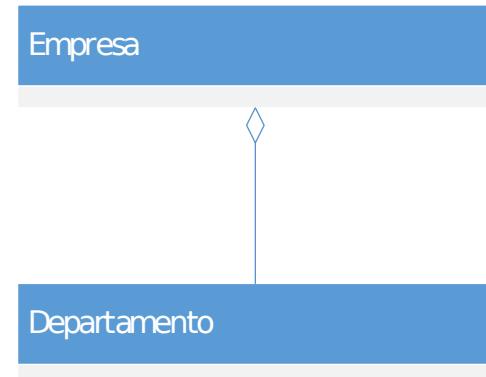


Ornamentos para Associações

- Multiplicidade

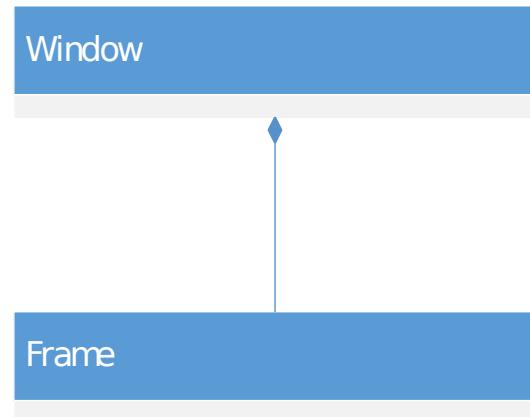


- Agregação: É uma relação do tipo “todo/parte” ou “possui um”



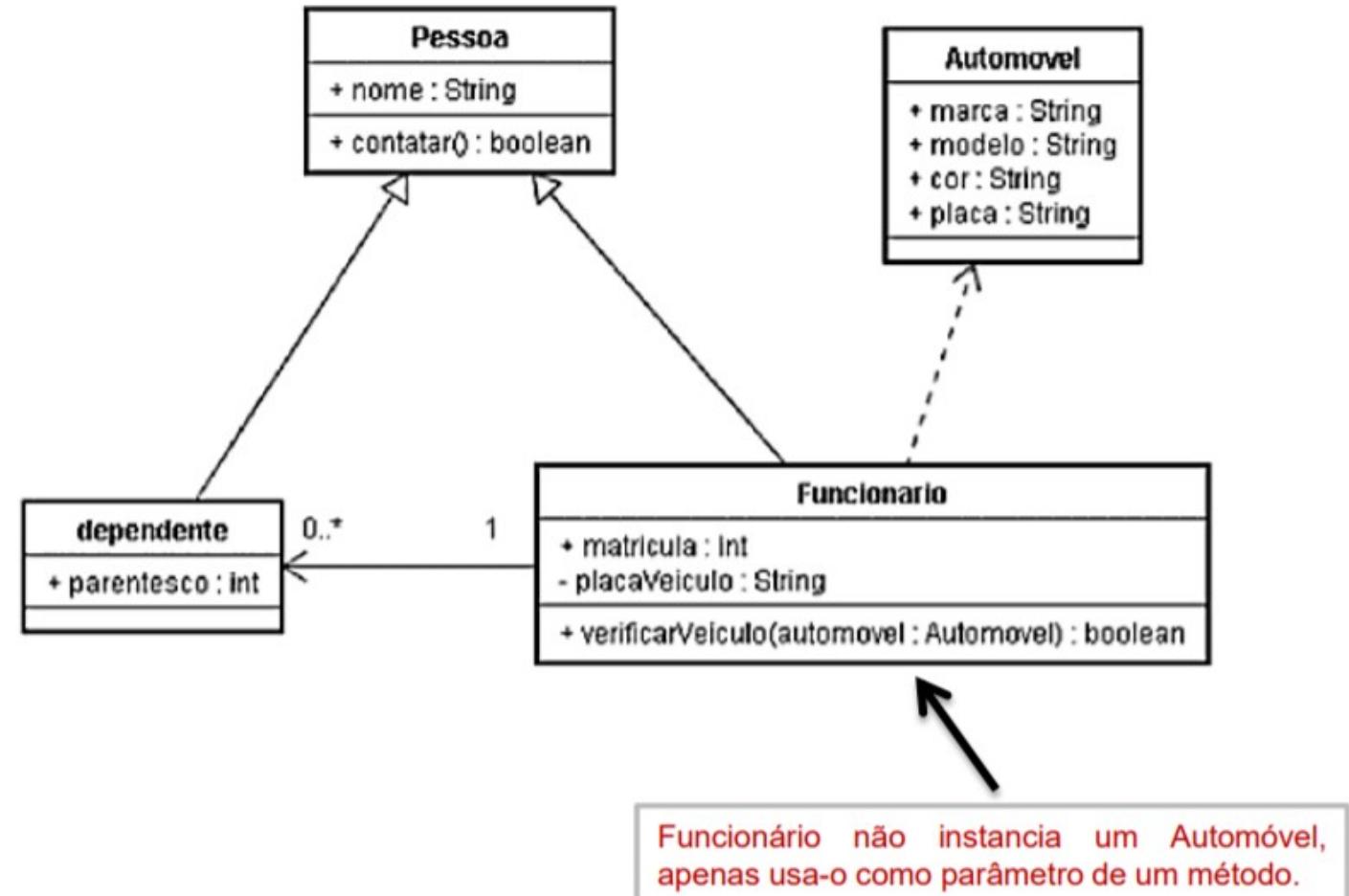
Ornamentos para Associações

- **Composição:** Um tipo de agregação na qual as partes são inseparáveis do todo.

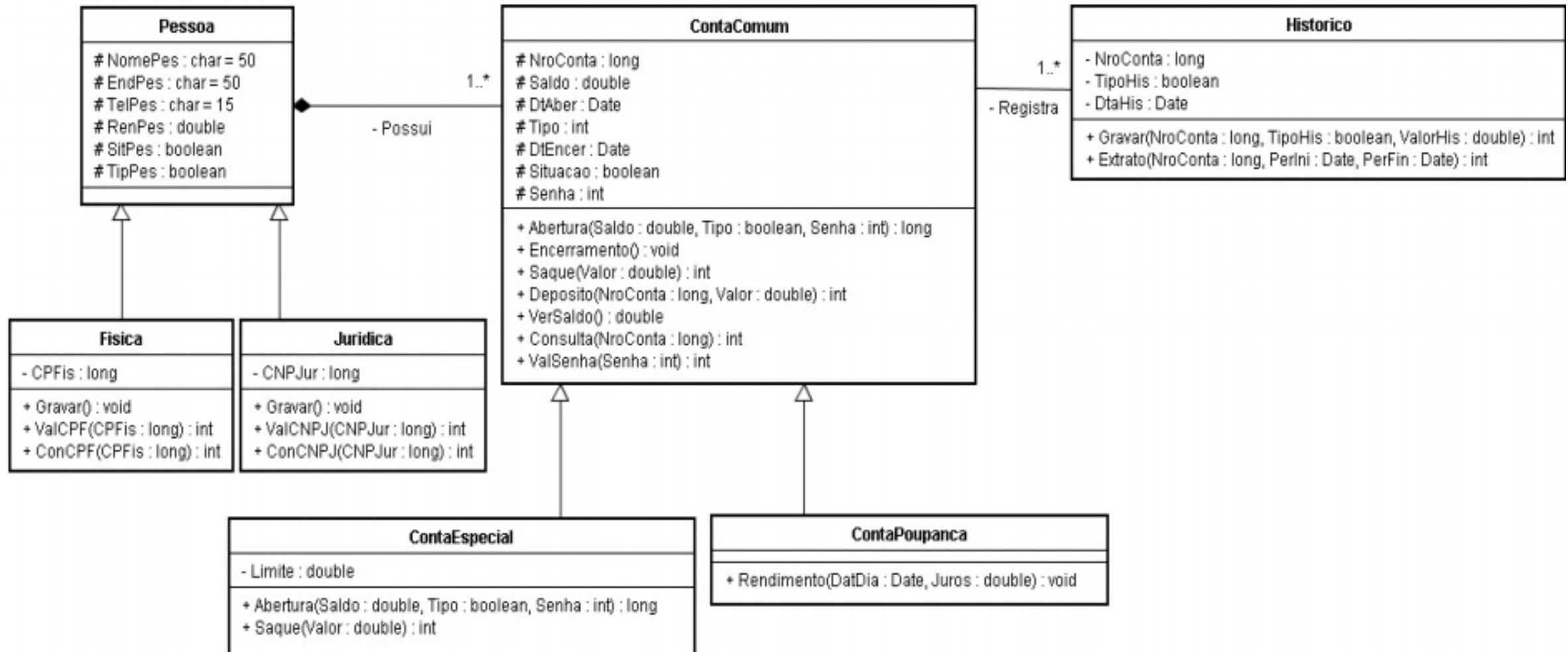


Dependência

- Uma dependência difere de uma associação porque a conexão entre as classes é temporária.



Exemplo de Diagrama de Classe



Exemplo de Diagrama de Classe

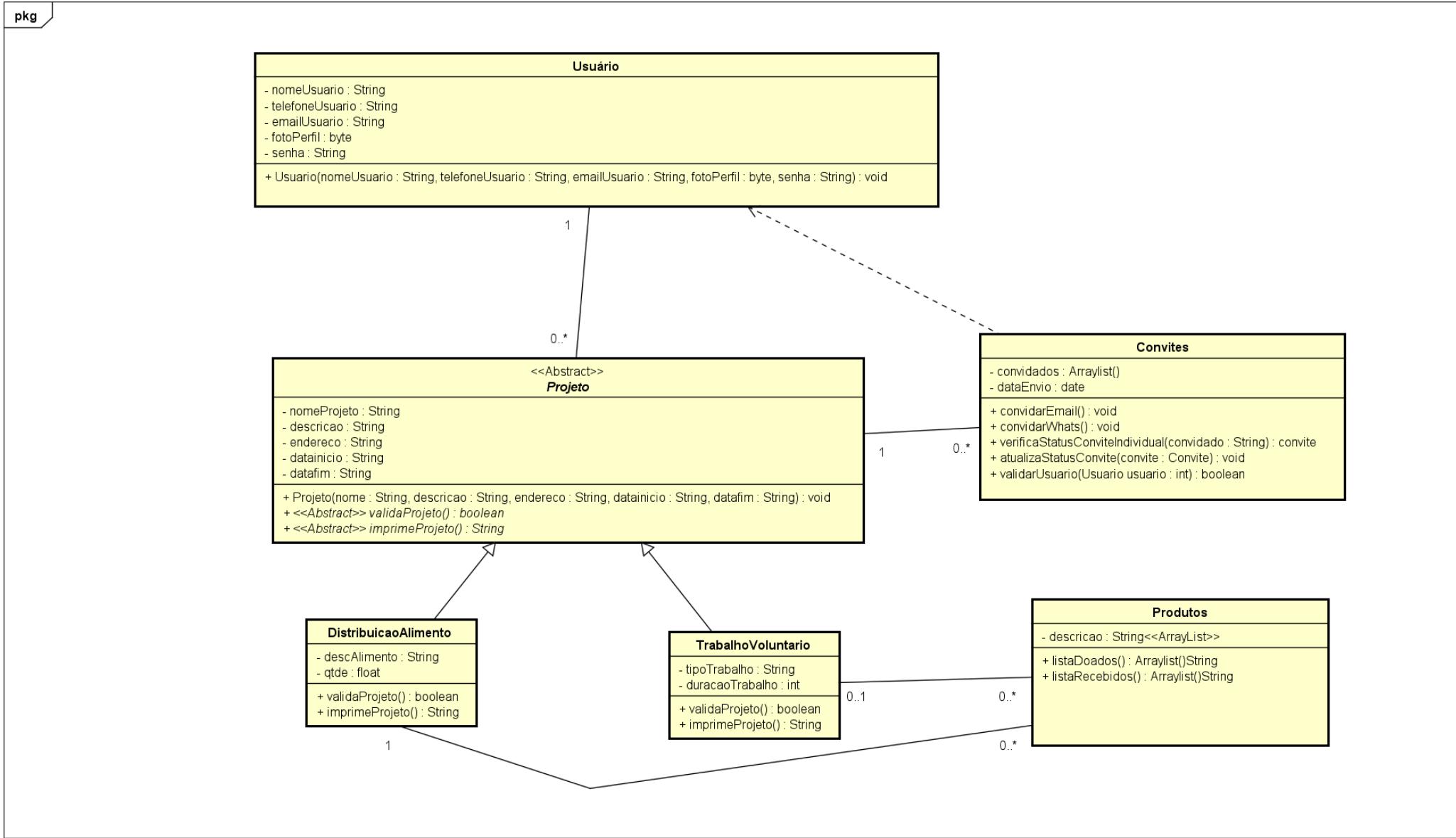
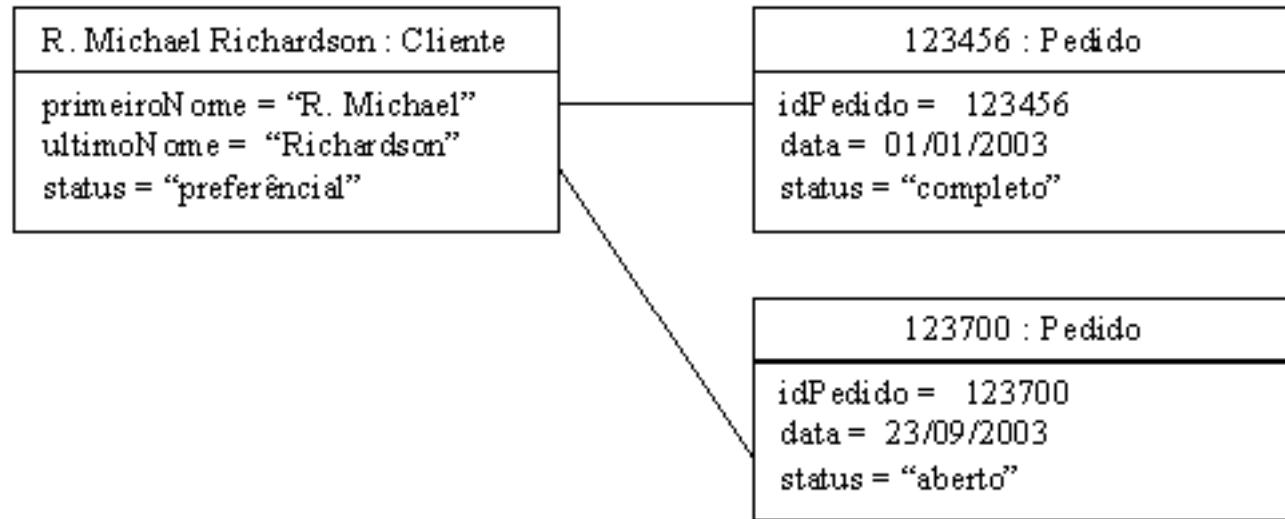


Diagrama de Objeto

Os diagramas de objetos fornecem uma captura instantânea das instâncias em um sistema e os relacionamentos entre as instâncias.

Os diagramas de objetos usam notação semelhante à usada nos diagramas de classe. No entanto, enquanto os diagramas de classe mostram os classificadores reais e seus relacionamentos em um sistema, os diagramas de objetos mostram instâncias específicas desses classificadores e os links entre essas instâncias em um determinado momento.

Diagrama de Objeto



O objeto R. Michael Richardson da classe Cliente está associado a ambos os objetos 123456 e 123700 da classe Pedido.

Usa-se o diagrama de objetos para modelar a visão estática de um sistema.

Exemplo de Diagrama de Objeto

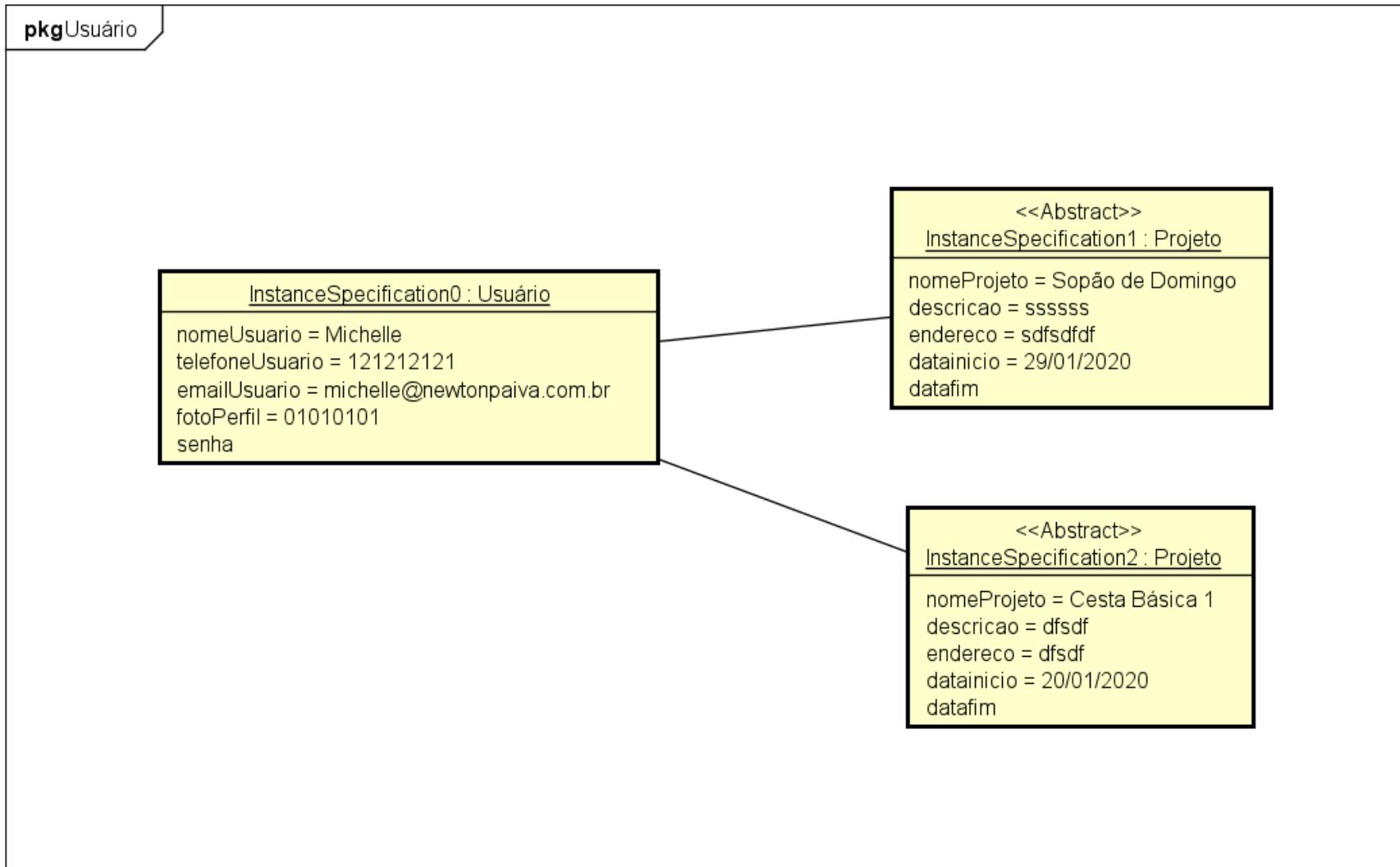


Diagrama de Pacotes

Um diagrama de pacotes são frequentemente utilizados para representar **subsistemas, módulos ou parte da arquitetura de um sistema**. Seu elemento principal é o pacote.

Um pacote pode representar muitas coisas: uma camada, um módulo, um subsistema, um espaço de nomes, etc.

Um pacote UML pode agrupar qualquer coisa: classes, outros pacotes, casos de uso, etc.

O elemento pacote pode ser utilizado em diversos outros diagramas como o digrama de classes ou o diagrama de casos de uso.

Diagrama de Pacotes

As classes representam a forma básica de estruturação de um sistema orientado a objetos. Um pacote é uma construção de agrupamento que permite agrupar seus elementos em unidades de nível mais alto. Seu uso mais comum é o agrupamento de classes, mas lembre-se de que você também pode usar pacotes para todos os outros elementos da UML.

Diagrama de Pacotes

- Cada pacote representa um espaço de nomes, o que significa que toda classe deve ter um nome exclusivo dentro do pacote a que pertence.
- A UML permite que as classes de um pacote sejam públicas ou privadas. Uma classe pública faz parte da interface do pacote e pode ser usada por classes de outros pacotes; uma classe privada fica oculta.

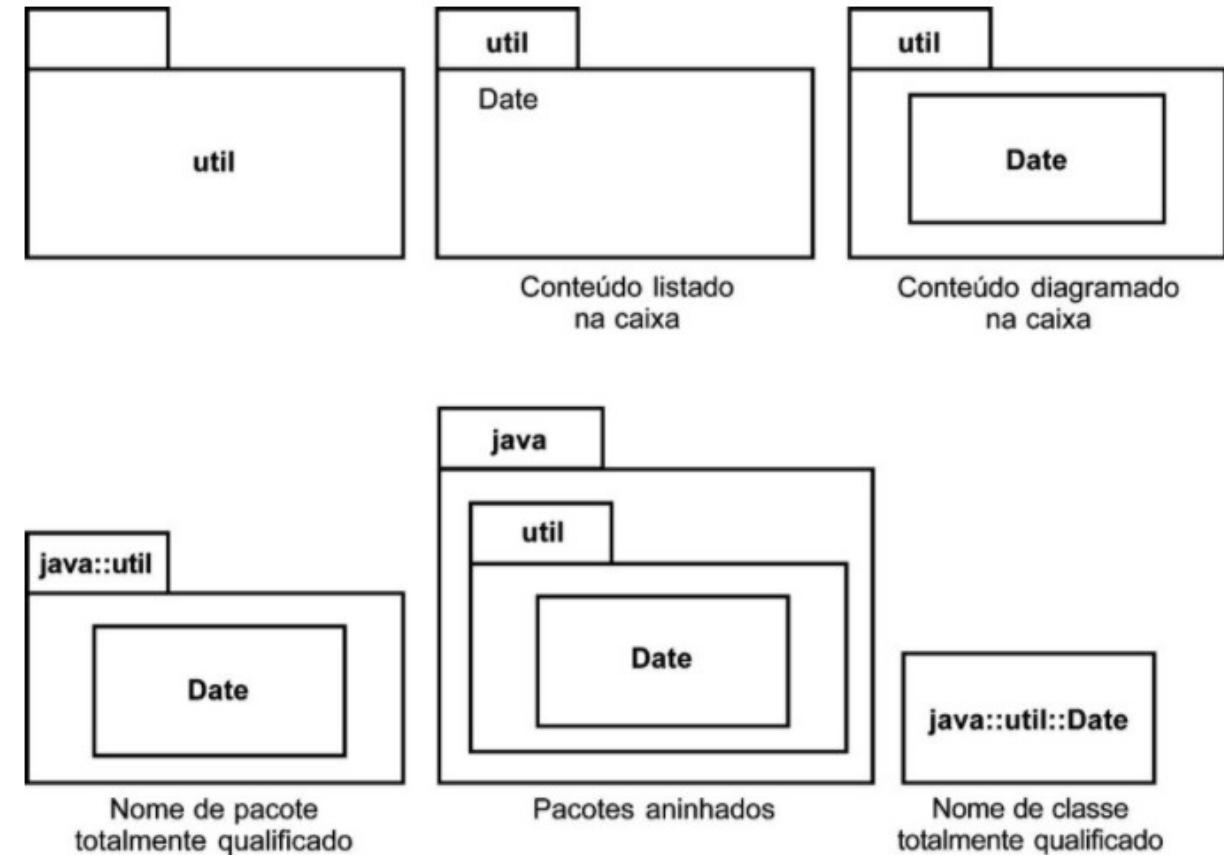


FIGURA 7.1 Maneiras de mostrar pacotes em diagramas.

Diagrama de Pacotes

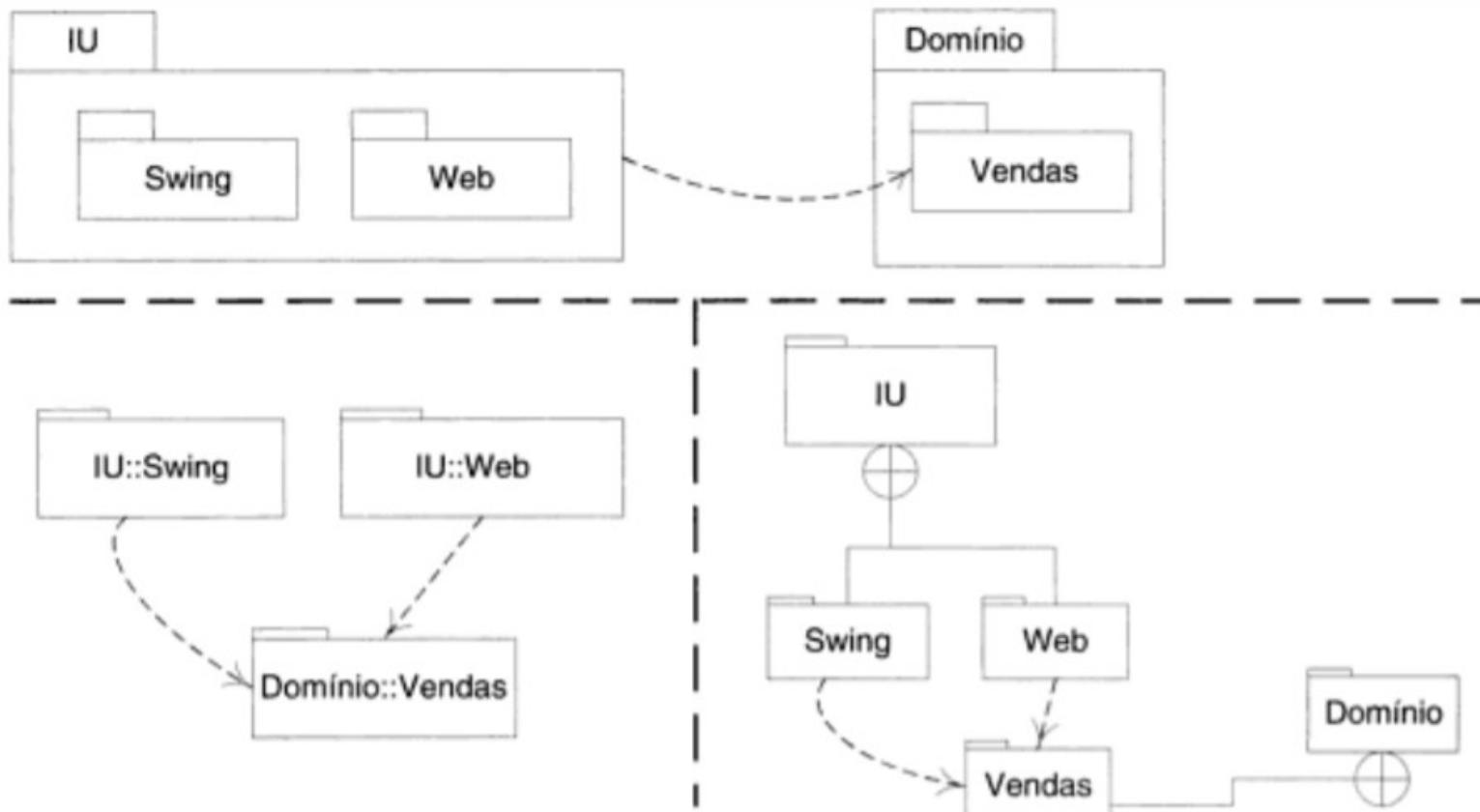


Figura 13.3 Abordagens alternativas UML para mostrar aninhamento de pacotes usando pacotes embutidos, nomes UML plenamente qualificados e o símbolo círculo-cruz.

Diagrama de Pacotes - Dependências

Acesso: indica que um pacote requer assistência das funções de outro pacote.

Exemplo:



Diagrama de Pacotes - Dependências

Importação: indica que a funcionalidade foi importada de um pacote para outro.
Exemplo:



Diagrama de Pacotes - Exemplos

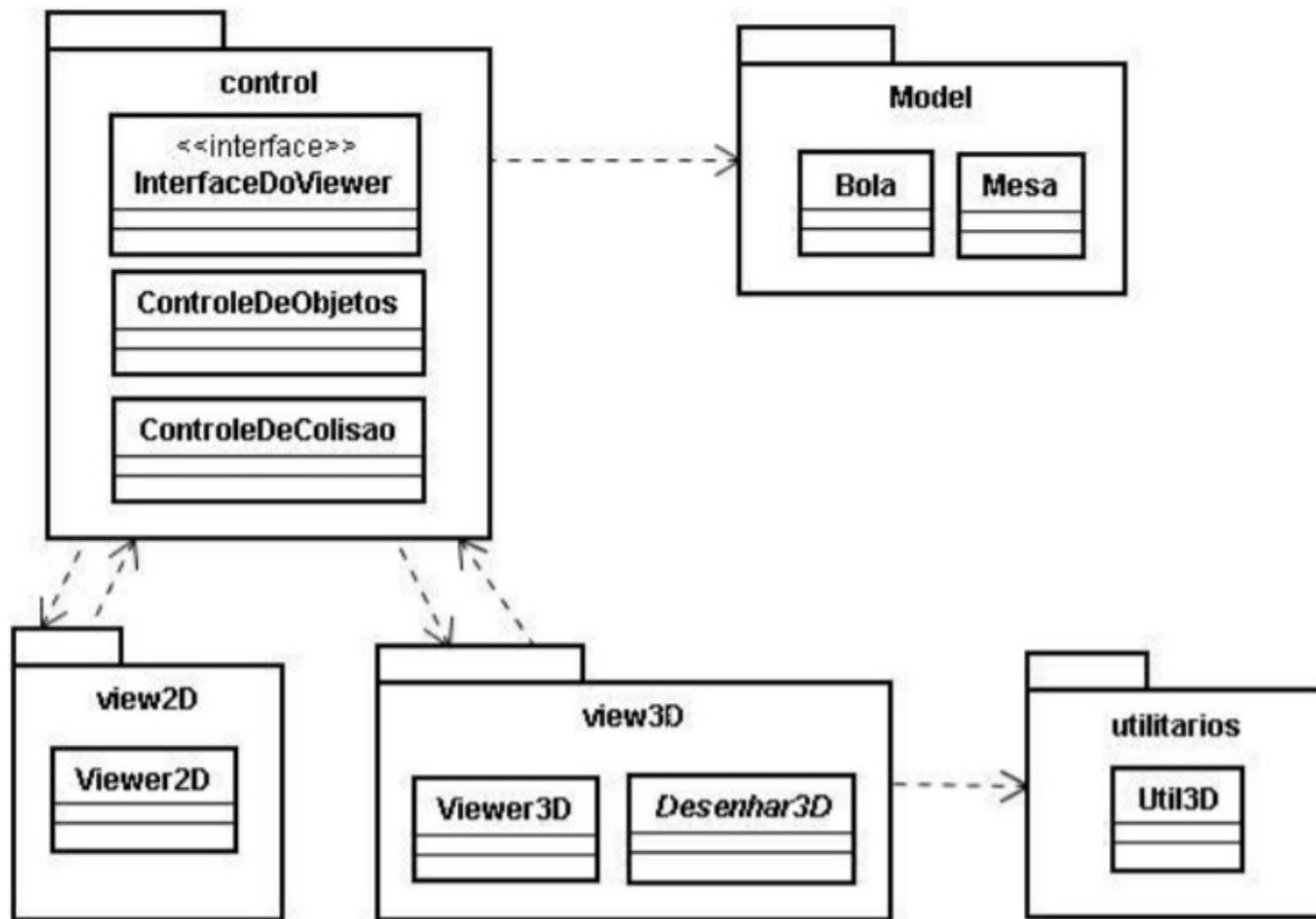


Diagrama de Pacotes - Exemplos

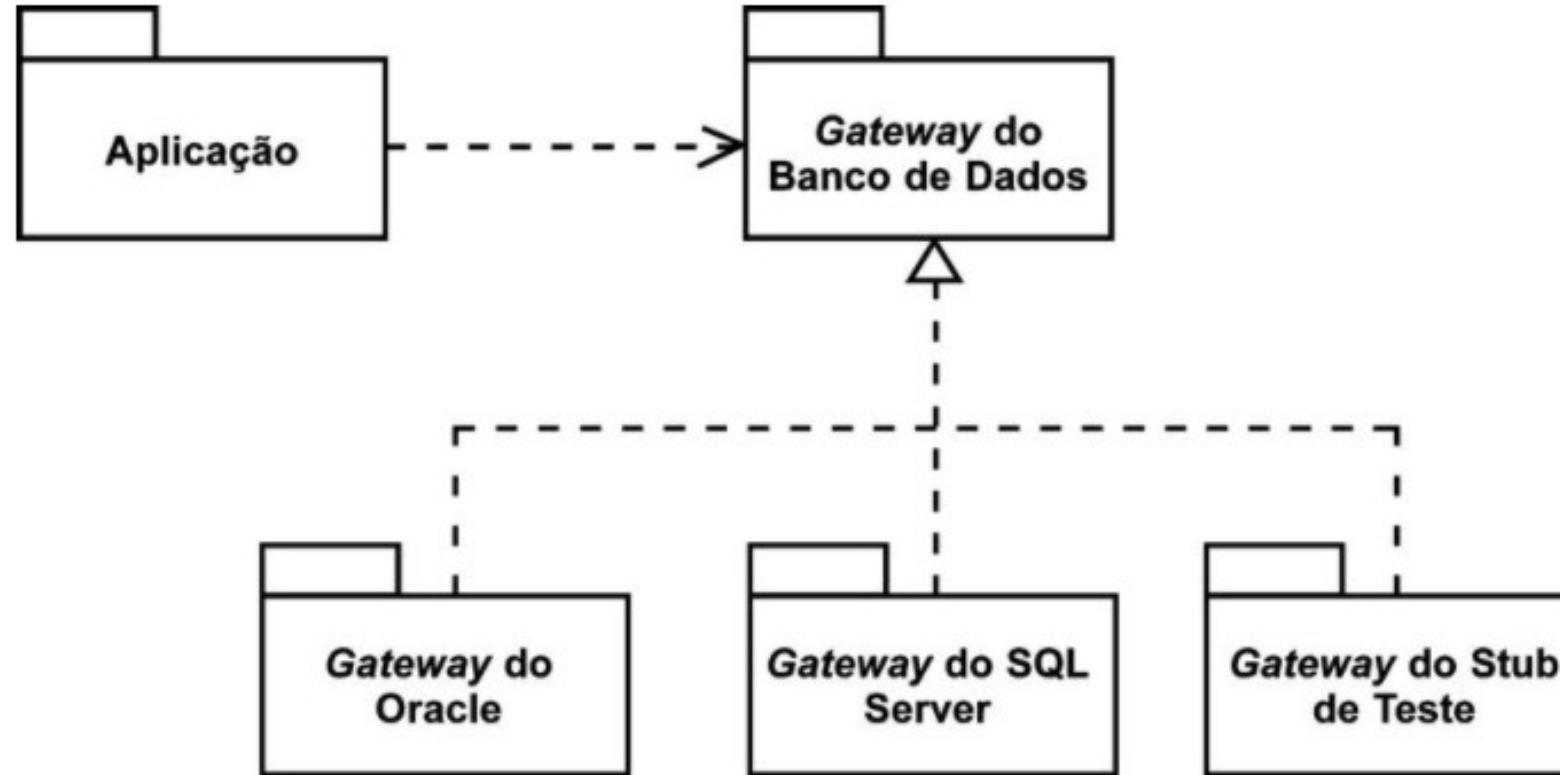


FIGURA 7.4 Um pacote implementado por outros pacotes

Diagrama de Pacotes - Exemplos

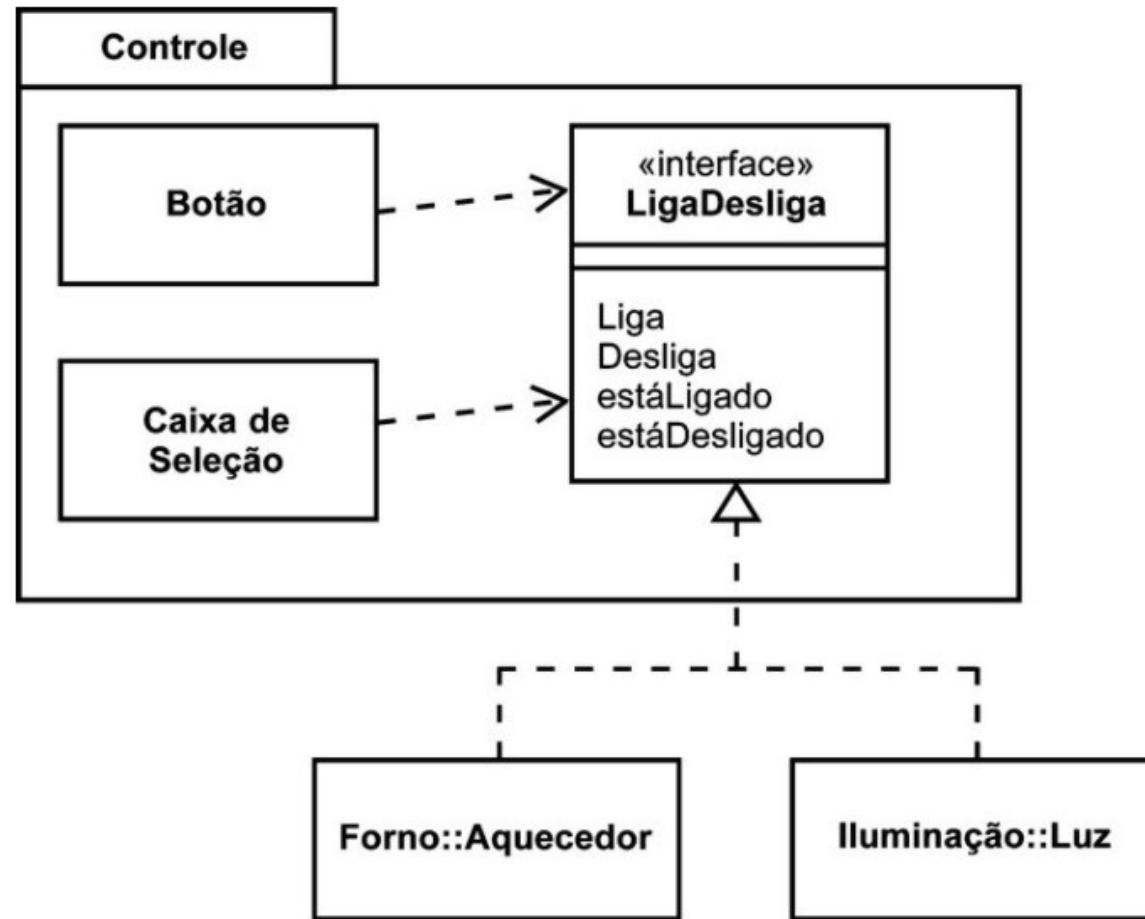


FIGURA 7.5 Definindo uma interface requerida em um pacote de cliente

Referências

LARMAN, Craig. **Utilizando UML e padrões**: uma introdução à análise e ao projeto orientados a objeto e ao desenvolvimento iterativo. Porto Alegre: Bookman, 2007, 3^a ed.

MARTIN, F. **UML Essencial**. Porto Alegre: 2011. 9788560031382.

PRESSMAN, Roger S. MAXIM, Bruce R. **Engenharia de Software - Uma Abordagem Profissional**. 8.ed. Porto Alegre: Amgh Editora, 2016. 968p. ISBN 9788580555332.

SOMMERVILLE, Ian. **Engenharia de software**. 8.ed. São Paulo: A. Wesley publishing company, 2010.