Engenharia de Software

Tópicos da aula

Gestão de Projetos Tecnológicos

Tópicos a serem discutidos:

- Conceitos básicos de orientação a objetos;
- Diagrama de classes;

Introdução à Orientação a Objetos

A orientação a objetos (OO) é um paradigma de programação baseado no conceito de "objetos", que são instâncias de classes.

A OO visa modelar o mundo real através da criação de objetos que interagem entre si, encapsulando dados e comportamentos específicos.

Princípios da Orientação a Objetos

Princípios fundamentais da orientação a objetos:

- Encapsulamento: Oculta os detalhes internos de um objeto e expõe apenas o necessário.
- Herança: Permite que uma classe (subclasse) herde atributos e métodos de outra classe (superclasse).
- Polimorfismo: Capacidade de um método ter diferentes implementações em diferentes classes.
- Abstração: Representação simplificada de conceitos complexos, focando nos aspectos essenciais.

Classes e Objetos

Uma classe define a estrutura e o comportamento de um objeto. Um objeto é uma instância de uma classe. As classes servem como um molde para criar objetos, especificando seus atributos (dados) e métodos (funções).

OBJETO

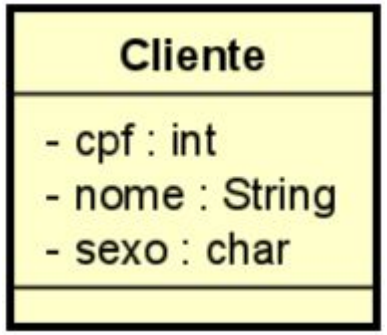
Segundo Melo (2004), um objeto é qualquer coisa, em forma concreta ou abstrata, que exista física ou apenas conceitualmente no mundo real. São exemplos de objetos: cliente, professor, carteira, caneta, carro, disciplina, curso, caixa de diálogo.

CLASSES

Uma classe é uma abstração de um conjunto de objetos que possuem os mesmos tipos de características e comportamentos, sendo representada por um retângulo que pode ter até três divisões. A primeira divisão armazena o nome da classe; a segunda, os atributos (características), enquanto a terceira carrega os métodos.

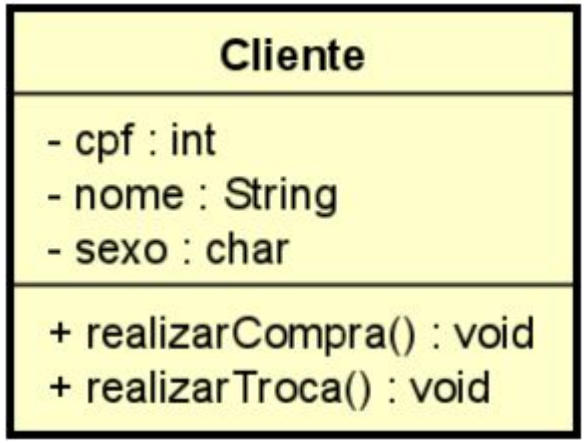
ATRIBUTOS

Os atributos representam as características que costumam variar de objeto para objeto, como o nome de um objeto da classe cliente ou a cor em um objeto da classe carro, por exemplo. Tais aspectos são os responsáveis por diferenciar um objeto de outro da mesma.



MÉTODOS

Métodos são processos ou serviços realizados por uma classe e disponibilizados para uso de outras classes em um sistema. Além disso, devem ficar armazenados na terceira divisão da classe. Os métodos são as atividades que uma instância de uma classe pode executar (GUEDES, 2007). Um método pode receber, ou não, parâmetros.



VISIBILIDADE (Encapsulamento)

De acordo com Guedes (2007), a visibilidade é um símbolo que antecede um atributo ou método e é utilizada para indicar o seu nível de acessibilidade.

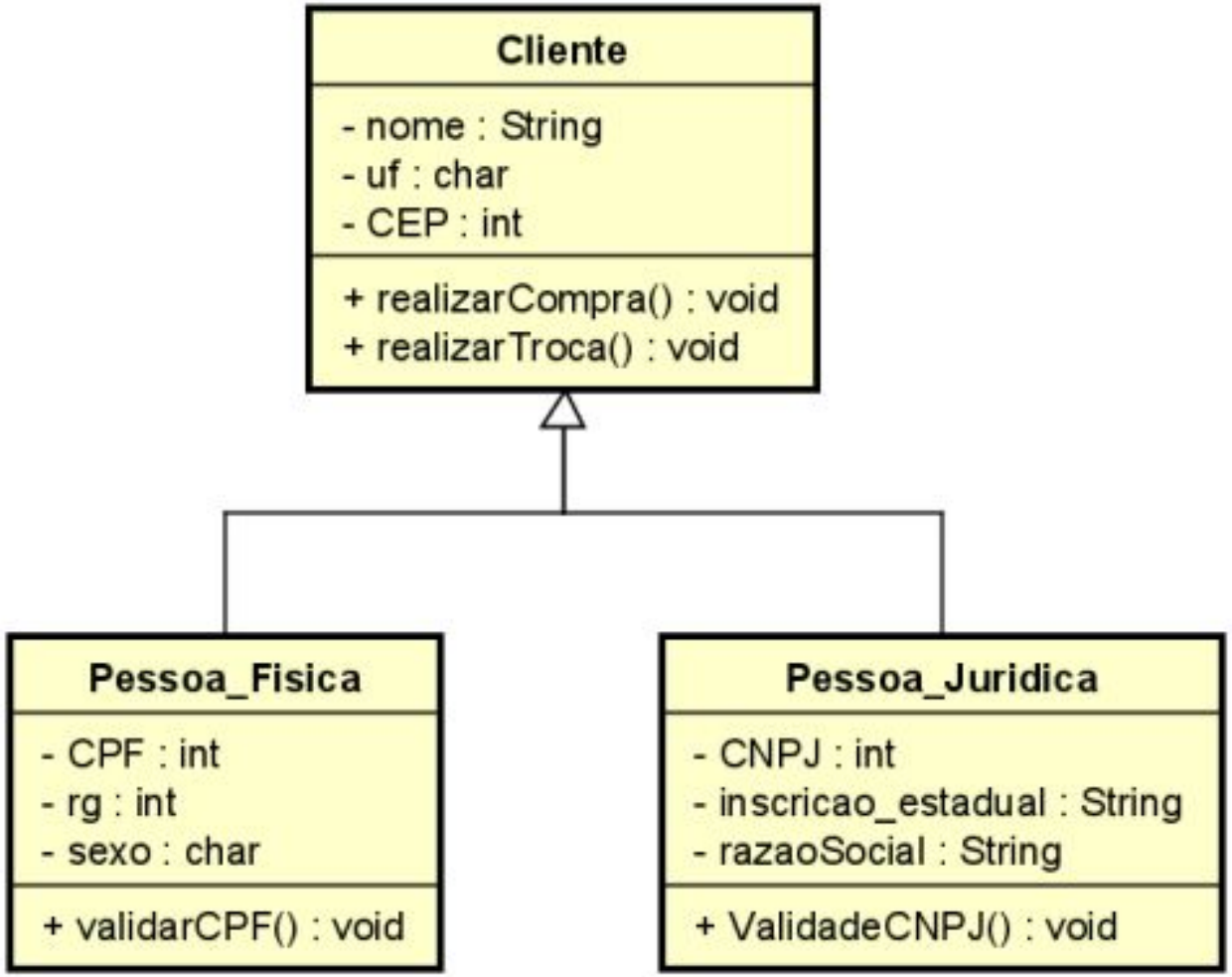
VISIBILIDADE

- O símbolo de mais (+) indica visibilidade pública, ou seja, significa que o atributo ou método pode ser utilizado por objetos de qualquer classe.
- O símbolo de sustenido (#) indica que a visibilidade é protegida, ou seja, determina que apenas objetos da classe possuidora do atributo ou método ou de suas subclasses podem acessá-lo.
- O símbolo de menos (-) indica que a visibilidade é privada, ou seja, somente os objetos da classe possuidora do atributo ou método poderão utilizá-lo.

HERANÇA (GENERALIZAÇÃO/ESPECIALIZAÇÃO)

A herança permite que as classes de um sistema compartilhem atributos e métodos, otimizando, assim, o tempo de desenvolvimento, com a diminuição de linhas de código, facilitando futuras manutenções (GUEDES, 2007).

HERANÇA (GENERALIZAÇÃO/ESPECIALIZAÇÃO)



POLIMORFISMO

É o princípio em que classes derivadas (as subclasses) e uma mesma superclasse podem chamar métodos que têm o mesmo nome (ou a mesma assinatura), mas que possuem comportamentos diferentes em cada subclasse, produzindo resultados diferentes.

POLIMORFISMO

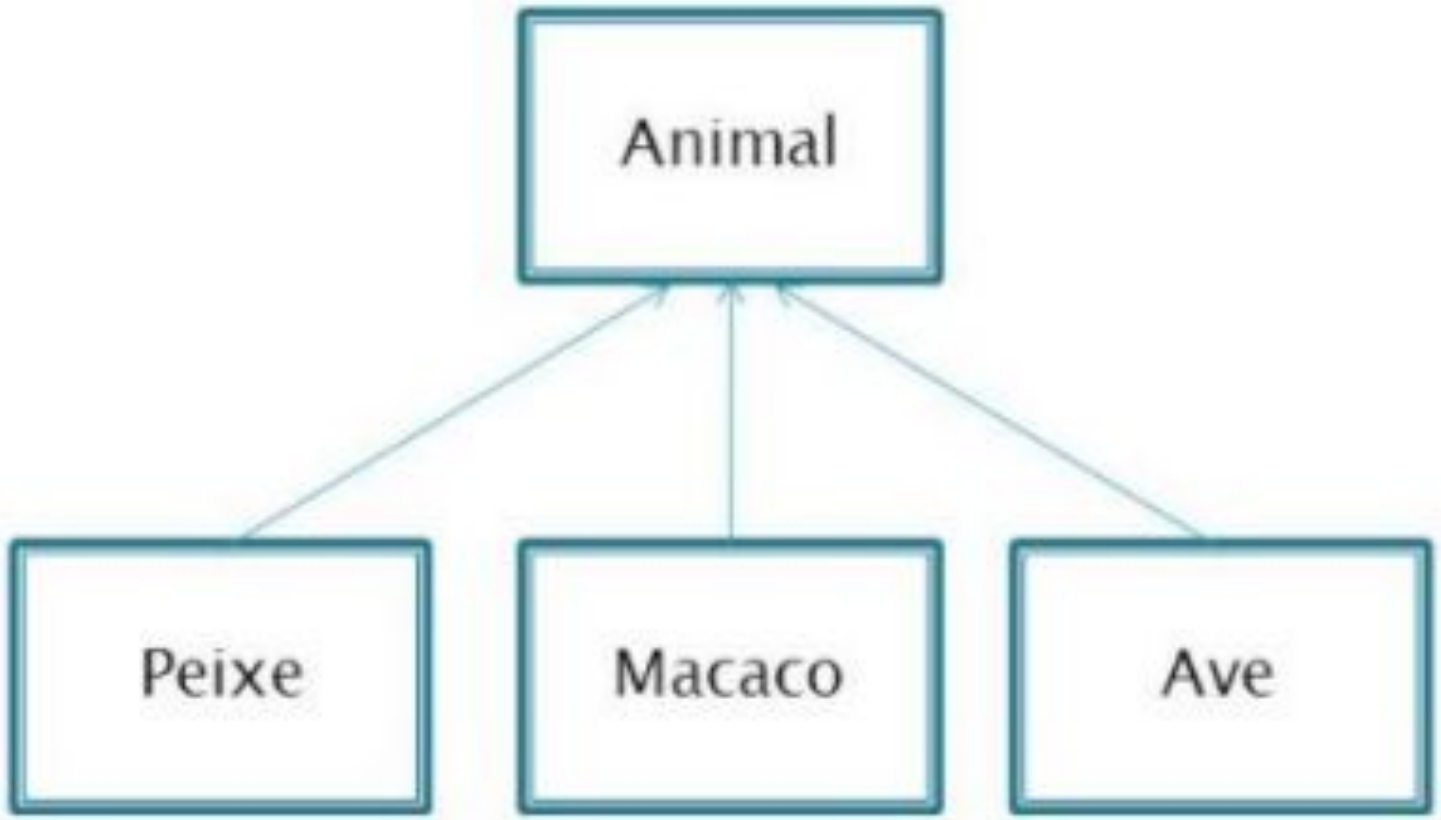


DIAGRAMA DE CLASSES

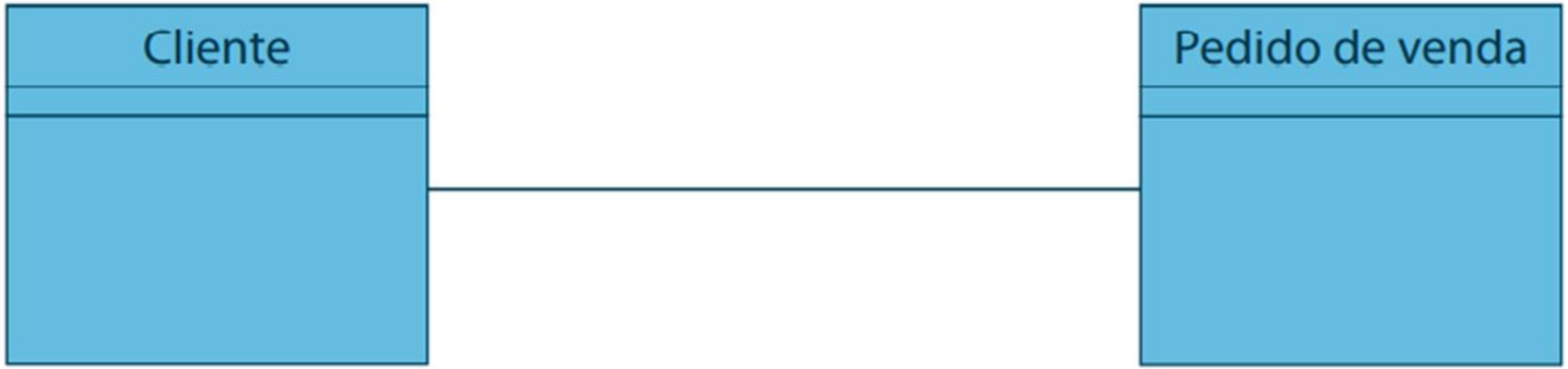
O diagrama de classes tem, como objetivo, permitir a visualização das classes utilizadas pelo sistema e como elas se relacionam, apresentando uma visão estática de como estão organizadas, preocupando-se apenas em definir sua estrutura lógica (GUEDES, 2007).

RELACIONAMENTOS

No diagrama de classes, temos alguns tipos de relacionamentos, como o de associação (que pode ser unária ou binária), generalização ou de agregação, por exemplo.

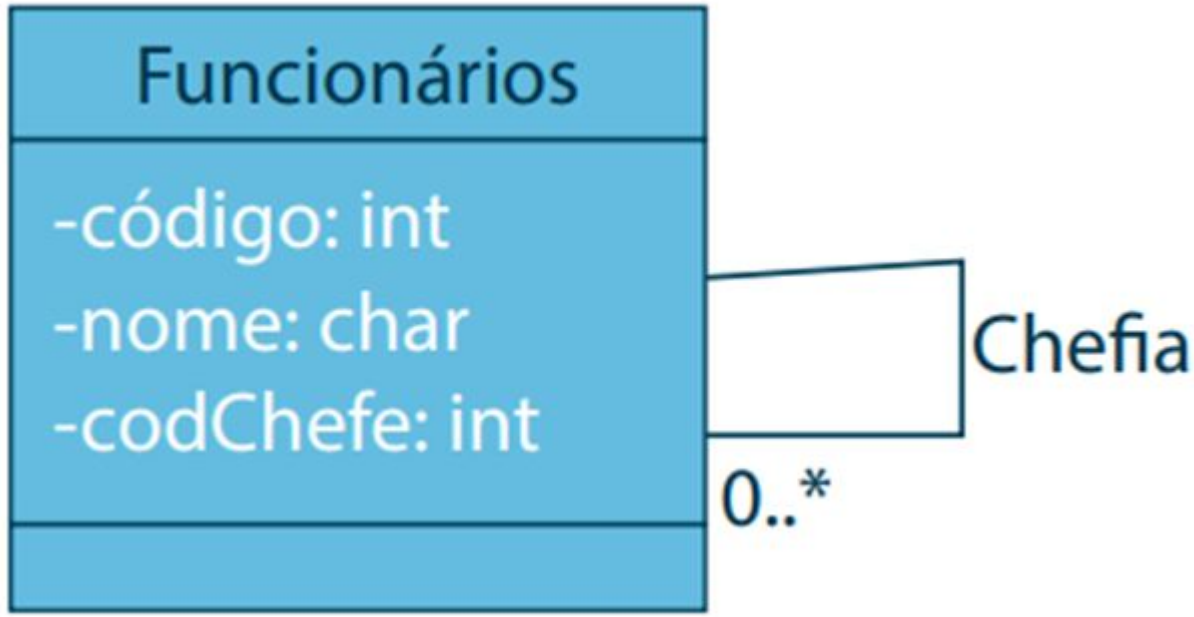
ASSOCIAÇÃO

É um relacionamento que conecta duas ou mais classes, demonstrando a colaboração entre as instâncias de classe.



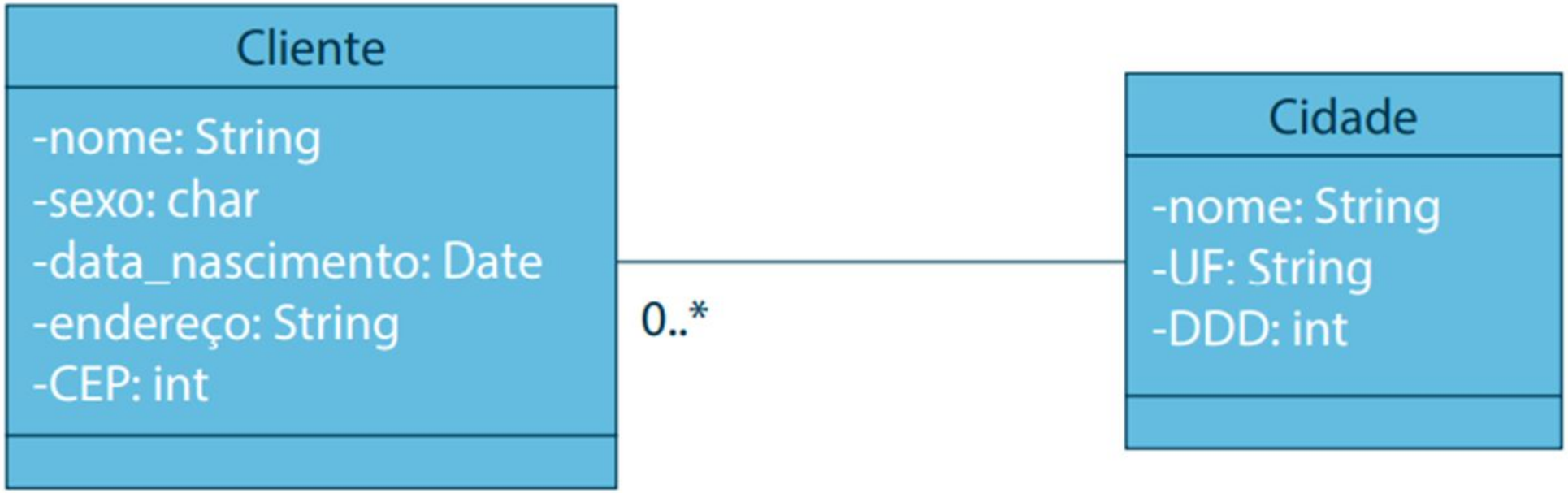
ASSOCIAÇÃO UNÁRIA OU REFLEXIVA

A associação unária ocorre quando existe um relacionamento de uma instância de uma classe com instâncias da mesma classe.



ASSOCIAÇÃO BINÁRIA

A associação binária conecta duas classes por meio de uma reta que liga uma classe a outra.



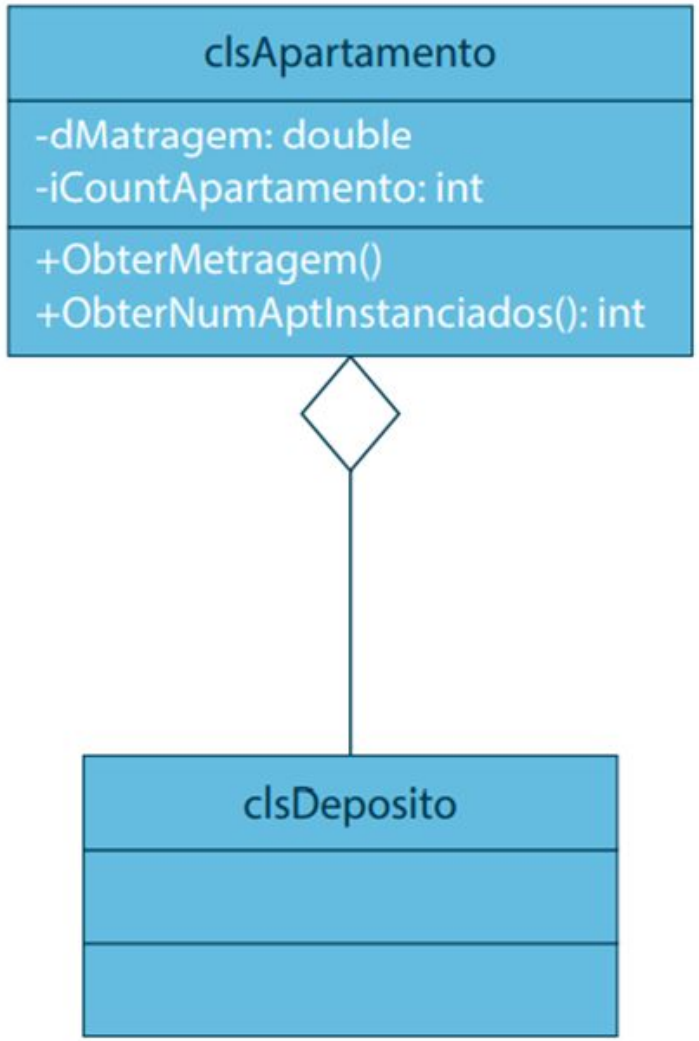
MULTIPLICIDADE

MULTIPLICIDADE	SIGNIFICADO
0..1	No mínimo zero (nenhum) e no máximo um. Indica que os objetos das classes associadas não precisam obrigatória-mente estar relacionadas, mas se houver relacionamento indica que apenas uma instância da classe se relaciona com as instâncias da outra classe.
1..1	Um e somente um. Indica que apenas um objeto da classe se relaciona com os objetos da outra classe.
0..*	No mínimo nenhum e no máximo muitos. Indica que pode ou não haver instância da classe participando do relacionamento.
*	Muitos. Indica que muitos objetos da classe estão envolvi-dos no relacionamento
1..*	No mínimo 1 e no máximo muitos. Indica que há pelo menos um objeto envolvido no relacionamento, podendo haver muitos objetos envolvidos.
2..5	No mínimo 2 e no máximo 5. Indica que existe pelo menos 2 instâncias envolvidas no relacionamento e que pode ser 3, 4 ou 5 as instâncias envolvidas, mas não mais do que isso.

AGREGAÇÃO

Uma agregação pode ocorrer somente entre duas classes. De acordo com as regras da UML, esse tipo de relacionamento ou associação é identificável a partir da notação de uma linha sólida entre duas classes: a classe denominada todo-agregado recebe um diamante vazio, enquanto a outra ponta da linha é ligada à classe denominada parte-constituinte. Ambas as classes podem “viver” de forma independente, ou seja, não existe uma ligação forte entre as duas.

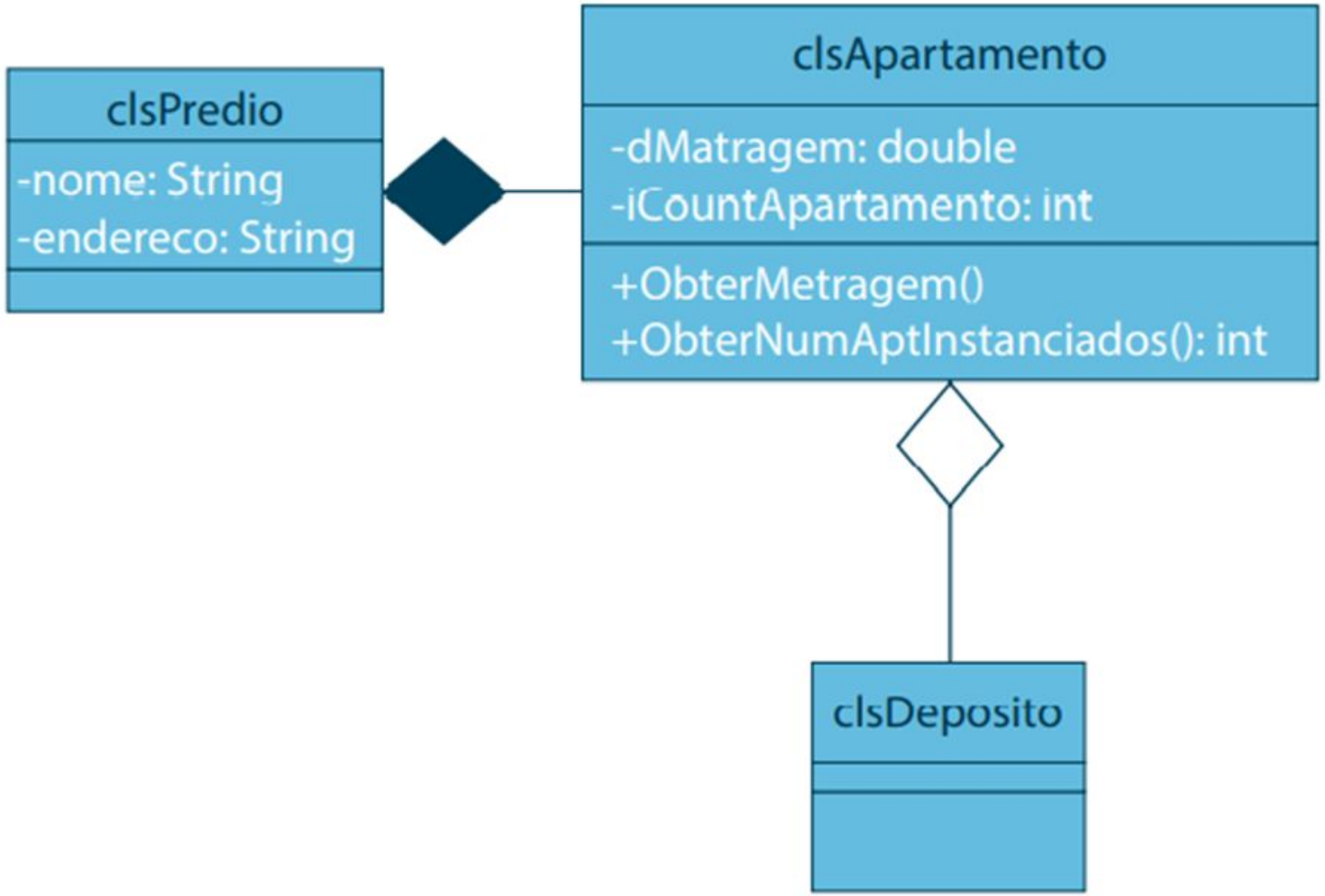
AGREGAÇÃO


clsApartamento
-dMatragem: double
-iCountApartamento: int
+ObterMetragem()
+ObterNumAptInstanciados(): int
clsDeposito

COMPOSIÇÃO

Uma composição ocorre quando temos uma situação semelhante à da agregação entre duas classes, mas os objetos da classe parte não podem viver quando o todo é destruído. Esse tipo de relacionamento é identificável pela notação de uma linha sólida entre duas classes: a classe denominada todo-composta, a qual recebe um diamante preenchido, enquanto a outra ponta da linha é ligada à classe denominada parte-componente

AGREGAÇÃO


clsPredio
-nome: String
-endereco: String
clsApartamento
-dMatragem: double
-iCountApartamento: int
+ObterMetragem()
+ObterNumAptInstanciados(): int
clsDeposito

CLASSE ASSOCIATIVA

Quando um relacionamento possui multiplicidade, ou seja, muitos (*) em suas duas extremidades, é necessário criar uma classe associativa para guardar os objetos envolvidos nessa associação.

- Uma classe associativa é representada por uma reta tracejada que parte do meio da associação e atinge uma classe.

CLASSE ASSOCIATIVA