IDENTIFICAR CLASSES E SEUS RELACIONAMENTOS EM UM DOMÍNIO DE APLICAÇÃO COM USO DE DIAGRAMA DE CLASSES

CONCEITO E ELEMENTOS DE UMA CLASSE

A abordagem de desenvolvimento de sistemas orientada a objetos assume que um sistema é uma coleção de entidades – os objetos – que se comunicam e realizam ações. A interação entre objetos é a forma de execução de uma funcionalidade do software.

Essa abordagem se baseia nos seguintes princípios listados por Bezerra (2015):

Qualquer coisa é um objeto.

Objetos realizam tarefas por meio da requisição de serviços a outros objetos.

Cada objeto pertence a uma classe, ou seja, classe é uma entidade que agrupa e abstrai objetos similares.

Uma classe funciona com repositório para comportamento associado aos objetos instanciados a partir dela.

Classes são organizadas em hierarquias.

Atenção
A UML é a linguagem padrão que permite a elaboração dos diversos modelos e das perspectivas de um sistema na abordagem da orientação a objetos. Classes (e seus objetos) são os componentes básicos dessa abordagem.
Objetos são entidades do mundo real que podemos observar. Por exemplo, um objeto pode ser um carro, uma fruta, uma empresa, um empregado da empresa, um concerto de música, um restaurante, um pedido de comida nesse restaurante, entre muitos outros.

Quando pensamos nesses objetos em termos de grupos de coisas, por exemplo, cada pessoa que assiste a aulas em uma turma é um objeto diferente, mas, se abstrairmos esses objetos, observando suas características comuns, criamos o conceito de Aluno, o qual é denominado, então, de Classe. Uma classe de objetos com caraterísticas comuns, tais como: estão matriculados em uma turma e assistem a aulas.

Uma classe pode ser definida como uma descrição de atributos (características) e serviços (ações) comuns a um grupo de objetos.

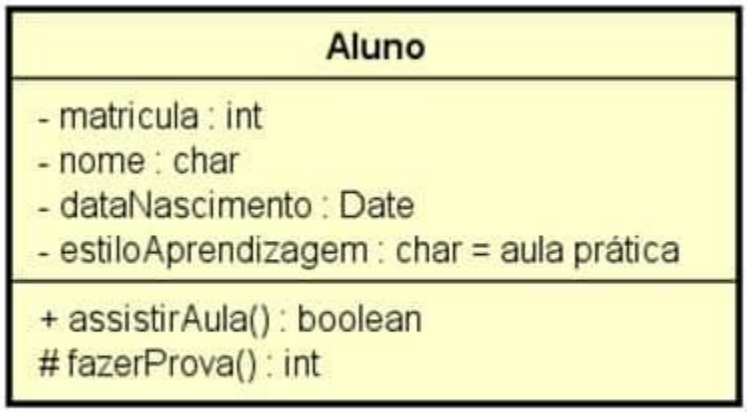
Bezerra (2015) ressalta que uma classe pode ser entendida como uma espécie de molde, a partir do qual objetos podem ser construídos. Portanto, é comum afirmarmos que um objeto é uma instância de uma classe. Cada objeto do mundo real é único, uma classe abstrai o tipo de objeto e permite que novos objetos sejam criados a partir desse modelo. Se uma nova pessoa se matricula na turma, ela é um novo objeto instanciado a partir da classe Aluno.

Quando estamos tratando da modelagem de um sistema, somente um subconjunto de características do grupo de objetos pode ser relevante. Portanto, uma classe representa uma abstração das características relevantes do mundo real para aquele conjunto de objetos.

Exemplo
Voltando aos nossos alunos, as características relevantes desses objetos, para um sistema acadêmico, poderiam ser número de matrícula, idade e estilo de aprendizagem; mas outras, tais como o esporte favorito ou número do calçado, talvez não façam parte do interesse desse sistema.
A estrutura de uma classe é composta de atributos e de operações.

Os atributos descrevem os dados que deverão ser armazenados pelos objetos dessa classe (suas características). Cada atributo de uma classe é de um tipo de dados e possui um conjunto de valores que esse atributo pode assumir, chamado de domínio. Veja um exemplo na figura 6.

As operações apresentam as ações que os objetos de uma classe podem realizar. Como são ações, costumam ser nomeadas com uso de um verbo e um complemento, e terminam com um par de parênteses (funções em linguagens de programação). Ao contrário dos atributos, que para cada objeto têm o seu próprio valor, os objetos de mesma classe compartilham as suas operações. Em outras palavras, todos os objetos de uma classe realizam as mesmas ações. Veja o exemplo de uma operação da classe Aluno na figura 6.



Na figura 6, observamos que a representação de uma classe é feita por meio de um retângulo com três compartimentos: nome da classe, lista de atributos e lista de operações. Você também pode perceber a sintaxe de especificação dos atributos e operações.

Um atributo é especificado no seguinte formato:

[visibilidade] nome: tipo = valor_inicial

Visibilidade de um atributo diz respeito ao seu nível de acesso, ou seja, permite definir quais atributos de um objeto são acessíveis por outros objetos. Essa propriedade serve para implementar o encapsulamento da estrutura interna da classe. A visibilidade de um atributo pode ser:

pública

(+)

protegida

(#)

privativa

(-)

de pacote

(~)

Somente as propriedades que são realmente necessárias ao exterior da classe devem ser definidas com visibilidade pública. Todo o resto é escondido dentro da classe por meio das visibilidades protegida (visível para subclasses da classe em que o atributo foi definido) e privativa (invisível externamente à classe em que o atributo está definido). Com visibilidade de pacote, o atributo é visível a qualquer classe que pertença ao mesmo pacote no qual está definida a classe. Como veremos no módulo seguinte, um pacote é um agrupamento de diagramas UML.

O elemento nome, única parte obrigatória na sintaxe do atributo, corresponde ao nome do atributo. O elemento tipo, que especifica o tipo do atributo, é dependente da linguagem de programação na qual a classe será implementada. Por exemplo: int, float, boolean. No exemplo da figura 6, o atributo matrícula é do tipo int.

É possível declarar o valor inicial que o atributo assume quando um objeto daquela classe for instanciado. Desse modo, sempre que um objeto de uma classe é instanciado, o valor inicial declarado é automaticamente definido para o atributo. Em nosso exemplo, quando um novo aluno é inserido no sistema, assumimos que seu estiloAprendizagem é aula prática.

Uma operação é especificada no seguinte formato:

visibilidade nome(parâmetros): tipo-retorno {propriedades}

O elemento nome corresponde ao nome dado à operação, em que normalmente usamos verbos que denotem a ação realizada por essa operação (no exemplo, assistirAula). Uma operação é ativada (executada) quando um objeto envia uma mensagem para outro, sendo essa mensagem o nome da operação.

Do mesmo modo que os atributos, a visibilidade pode ser pública (+), privada (-), protegida (#) ou de pacote (~).

Os parâmetros de uma operação são informações que ela recebe do objeto que envia a mensagem para requisitar a execução da operação. Uma operação pode ter zero ou mais parâmetros, que devem ser separados por vírgulas. Parâmetros têm seu tipo definido: tipo retorno representa o tipo de dados do valor retornado por uma operação.

VISÃO GERAL DO DIAGRAMA DE CLASSES

O Diagrama de Classes da UML é o modelo em que representamos as classes de um sistema, incluindo os detalhes sobre seus atributos e suas operações, e como essas classes se relacionam.

Portanto, a elaboração do Diagrama de Classes é fundamental em todo o processo de desenvolvimento que segue a abordagem da orientação a objetos.

Os diversos componentes de um Diagrama de Classes especificam as classes que serão realmente implementadas (ou seja, codificadas em uma linguagem de programação), sendo a base para a criação dos principais objetos e as interações entre eles.

Atenção

Góes (2014) explica que o Diagrama de Classes é o modelo mais importante da UML, pois permite a representação dos elementos da base de dados do sistema. Portanto, facilita a comunicação entre a equipe de desenvolvimento e com os usuários finais.

O Diagrama de Classes também explicita as demandas de dados e informações dos atores, que são declaradas nos casos de uso. Diagramas de classes são usados para expressar graficamente os conceitos a serem manipulados por um sistema, bem como as ações esperadas, e para fornecer uma descrição independente de implementação dos tipos utilizados em um sistema, ou seja, os modelos de dados.

Comentário

Costumamos dizer que esse diagrama destaca um aspecto estrutural estático do sistema, pois representa a estrutura interna do sistema, mas não como os objetos do sistema interagem no decorrer do tempo. As operações mostram quais são as possíveis interações entre os objetos, mas não explicitam em que momento serão invocadas.

O Diagrama de Classes é utilizado na construção do modelo de classes desde o nível de análise até o nível de especificação. O diagrama evolui à medida que o processo de desenvolvimento do sistema avança.

O que queremos dizer com “evolui”?

Resposta

Significa que vamos acrescentando detalhes e refinando o diagrama, tanto em termos de atributos e operações quanto na forma como associamos as classes.

É comum chamarmos esse diagrama que evolui sucessivamente de modelo de classes de análise, modelo de classes de projeto e modelo de classes de implementação, fazendo referência às etapas do processo de desenvolvimento.

Clique nas barras para ver as informações.

MODELO DE CLASSES DE ANÁLISE

É composto por elementos identificados na etapa de análise; é basicamente formado por classes relacionadas a conceitos do domínio da aplicação. Seu objetivo é descrever o problema representado pelo sistema a ser desenvolvido. É um modelo conceitual; com foco em o que o sistema deve tratar, não considera características técnicas da solução a ser utilizada. Em termos práticos, nesse modelo, incluímos as classes, a lista de atributos e a lista básica de operações, sem os detalhes de tipos de dados, ou parâmetros das funções, por exemplo. O modelo de casos de uso e o modelo de classes de análise são os dois principais modelos criados na fase de análise.

MODELO DE CLASSES DE PROJETO

É construído na atividade de projeto do desenvolvimento e considera alguns detalhes da solução de software a ser utilizada, mas ainda em um nível alto de abstração. Nessa etapa do processo de desenvolvimento, é comum percebermos a necessidade de criar novas classes, uma vez que o foco agora está em como o sistema deve funcionar. Ou seja, como o sistema vai entregar suas funcionalidades. Os detalhes incluídos são relacionados, por exemplo, com tipos e domínio dos atributos; parâmetros, entradas e saídas das operações etc. Novas classes criadas podem ser de interfaces do sistema, bancos de dados, entre outras.

MODELO DE CLASSES DE IMPLEMENTAÇÃO

É a codificação das classes definidas nas fases de análise e projeto em alguma linguagem de programação, normalmente uma linguagem orientada a objetos (por exemplo, C++, Java, C# etc.). O modelo de implementação é construído na atividade de codificação de um processo de desenvolvimento

O exemplo da figura 7, extraído da obra de Bezerra (2015), ilustra a representação de uma classe em diferentes níveis de abstração. Observe que, em cada iteração do processo de desenvolvimento, são acrescentados mais detalhes sobre os atributos e operações.

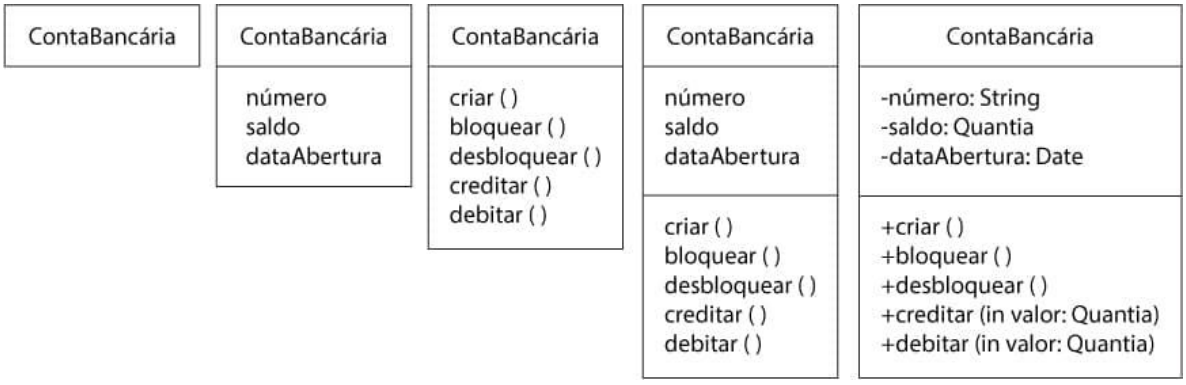


Figura 7 – Exemplo de Classe ContaBancária em diferentes níveis de abstração.

Na especificação mais detalhada da classe ContaBancária do exemplo da figura 7, você pode observar a presença dos símbolos – e + antes dos nomes dos atributos e das operações, que representam sua visibilidade. Nessa mesma representação, temos as operações com seus parâmetros de entrada especificados (a Quantia em creditar e debitar).

RELACIONAMENTOS NO DIAGRAMA DE CLASSES

Você já viu que especificar uma classe diz respeito a definir seus atributos e operações. Mas classes não existem isoladas, elas só fazem sentido quando se relacionam a outras, pois os objetos originados a partir delas trocam informações uns com os outros; e nisso reside a implementação de um sistema orientado a objetos.

Dizemos que os objetos colaboram uns com os outros. O desenho do Diagrama de Classes mostra como os objetos instanciados a partir dessas classes podem se relacionar. Existem diferentes tipos de relacionamentos possíveis, bem como diversos detalhes que podem ser acrescentados no diagrama para melhorar sua semântica. Trataremos deles a seguir.

A forma de relacionamento mais comum é chamada de associação e é representada no Diagrama de Classes por uma linha (normalmente um segmento de reta) ligando as classes às quais pertencem os objetos relacionados.

A figura 8 ilustra associações entre classes:

- Cliente faz Pedido;
- Professor ministra Disciplina.

Mas qual é o significado dessas associações? Quando ligamos as classes Cliente e Pedido significa que, durante a execução do sistema, haverá a possibilidade de troca de mensagens entre objetos dessas classes.

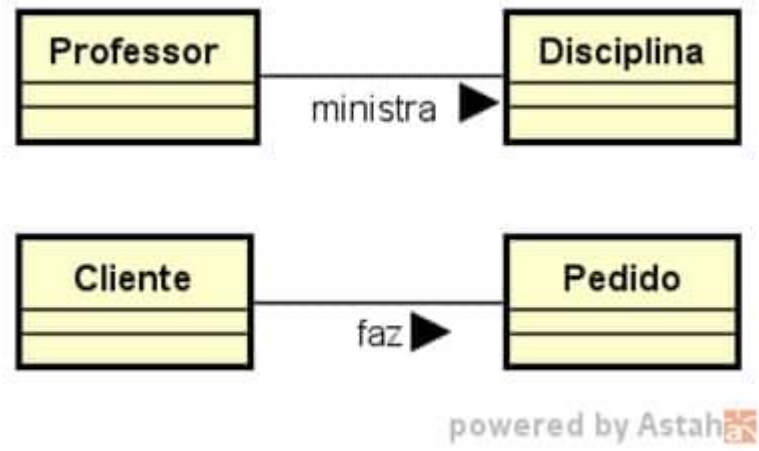


Figura 8 – Associações simples entre classes, elaborado com ferramenta ASTAH (2021)

As associações possuem diversas características que podem ser representadas, por escolha do analista, para prover um melhor entendimento do modelo: nome, multiplicidades, tipo de participação, direção de leitura e papéis.

Cliques nas barras para ver as características das associações.

NOME

Conforme a figura 8 mostra, os nomes das associações (ministra, faz) descrevem a relação que é estabelecida entre os objetos das classes associadas. Normalmente, são usados verbos ou expressões verbais para nomear associações.

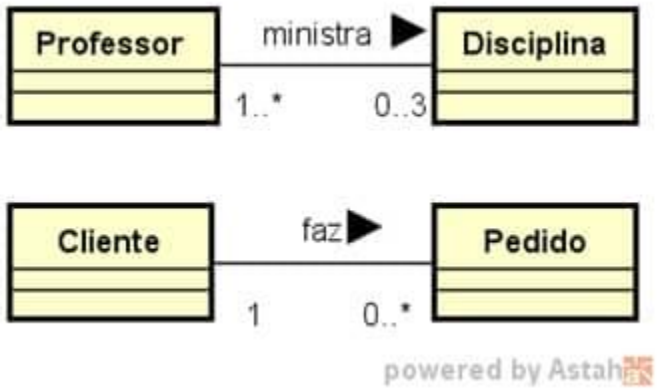
MULTIPLICIDADES

As associações permitem representar a informação dos limites inferior e superior da quantidade de objetos aos quais outro objeto pode estar associado: são as chamadas multiplicidades. Cada associação em um Diagrama de Classes possui duas multiplicidades, uma em cada extremo da linha que a representa. Os símbolos que representam uma multiplicidade são apresentados na tabela 1.

Apenas 1	1..1 (ou 1)
Zero ou Muitos	0..* (ou *)
Um ou Muitos	1..*
Zero ou Um	0..1
Intervalo Específico	li..ls

Tabela 1 – Multiplicidades em um Diagrama de Classes. Tabela: Flavia Maria Santoro

Na figura 9, mostramos o exemplo com as multiplicidades correspondentes. Esses novos diagramas expressam mais do que o anterior. Eles informam que um professor pode ministrar nenhuma (zero) e no máximo três disciplinas (0..3). Uma disciplina precisa ser ministrada por pelo menos um professor, mas pode ser ministrada por muitos (1..*). Já um cliente pode fazer nenhum (zero) pedido, sem um limite máximo (0..*), e um pedido precisa ser feito por um cliente (1).



TIPO DE PARTICIPAÇÃO

Indica a necessidade ou não da existência dessa associação entre objetos. A participação pode ser **obrigatória** ou **opcional**. Se o valor mínimo da multiplicidade de uma associação é igual a 1 (um), significa que a participação é obrigatória. Caso contrário, a participação é opcional. Por exemplo, a participação de Cliente e Professor nos exemplos da figura 9 é obrigatória, e a de Disciplina e Pedido é opcional. Isso significa que pode existir professores e clientes sem necessariamente estarem ministrando disciplinas ou fazendo pedidos.

DIREÇÃO DA LEITURA

A direção de leitura indica como a associação deve ser lida. Essa direção é representada por um pequeno triângulo posicionado próximo a um dos lados do nome da associação (como nos exemplos da figura 9).

PAPÉIS

Quando um objeto participa de uma associação, ele tem um papel específico nela. Uma característica complementar à utilização de nomes e de direções de leitura é a indicação de papéis (roles) para cada uma das classes participantes em uma associação.

CLASSES ASSOCIATIVAS

São classes que estão ligadas a associações, em vez de estarem ligadas a outras classes. São também chamadas de classes de associação. Esse tipo de classe normalmente aparece quando duas ou mais classes estão associadas e seja necessário manter informações sobre a associação que existe entre elas.

É muito comum a necessidade de classes associativas em relacionamentos muitos para muitos, ou seja 0..* 0..*. Uma classe associativa é representada pela mesma notação utilizada para uma classe comum. A diferença é que aquela é ligada por uma linha tracejada a uma associação.

A figura 10 mostra um exemplo de classe associativa. Um professor ministra uma disciplina em determinada turma (que possui número, ano e semestre específicos).

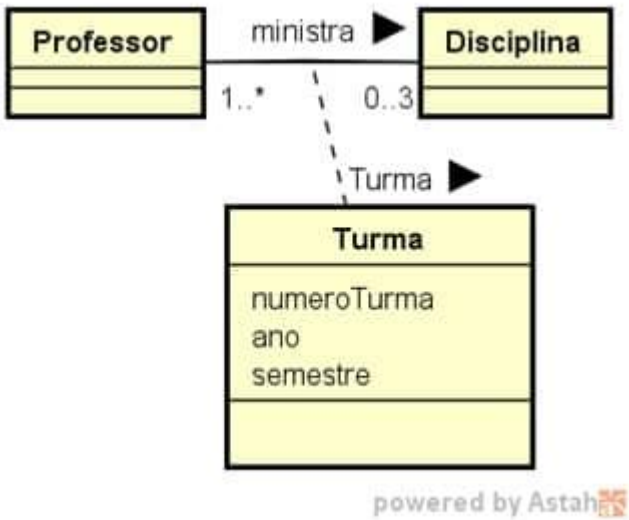


Figura 10 – Exemplo de classe associativa, elaborado com ferramenta ASTAH (2021)

Autoassociação

Tem como objetivo relacionar objetos da mesma classe, mas cada objeto tem um papel distinto nessa associação. Por exemplo, na figura 11, objetos da classe Empregado estão associados entre si, um empregado como supervisor, que supervisiona outros empregados.

A figura 11 ilustra, também, o uso de papéis (roles) para distinguir o lado supervisor (multiplicidade 1) do lado supervisionado (lado muitos) da autoassociação.

Figura 11 – Exemplo de autoassociação, elaborado com ferramenta ASTAH (2021)

Todo-parte

Existe uma categoria especial de relacionamento que possui uma semântica (significado) particular: todo-parte.

Uma relação todo-parte entre dois objetos indica que um dos objetos está contido (é uma parte do) no outro, ou seja, o objeto do lado todo contém o do lado parte.
A UML define dois tipos de relacionamentos todo-parte: a agregação e a composição.

Atenção
A **agregação** e a **composição** são casos especiais da associação, logo, todas as características válidas para uma associação valem para agregações e composições.
Agregações/composições são assimétricas no sentido que, se um objeto A é parte de um objeto B, o objeto B não pode ser parte do objeto A.
Agregações e composições propagam comportamento, pois um comportamento que se aplica a um todo, automaticamente se aplica às suas partes. Nas agregações/composições, as partes são normalmente criadas e destruídas pelo todo.

Uma agregação é representada graficamente como uma linha que conecta as classes relacionadas, com um diamante (losango) branco perto da classe que representa o todo.
Uma composição é representada por meio de um diamante negro.

A figura 12 apresenta um exemplo de relacionamento de composição. Um pedido é composto de vários itens de pedidos.

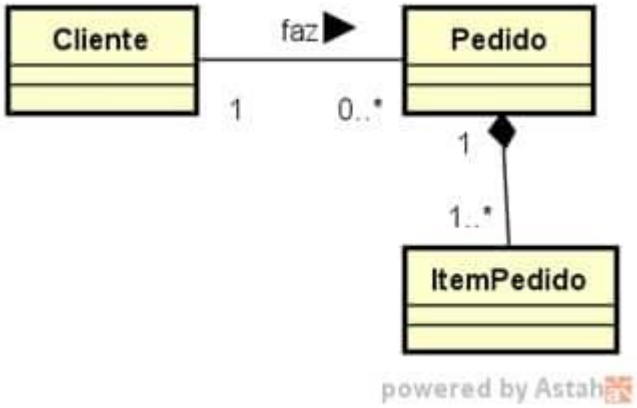


Figura 12 – Exemplo de relacionamento de composição, elaborado com ferramenta ASTAH (2021)

A diferença entre agregação e composição é sutil, pois não é muito bem definida na UML. Ambas são relacionamentos todo-parte, mas pode-se dizer que, na agregação, a parte pode existir independentemente do todo; na composição, a existência da parte depende da existência do todo.

No exemplo da figura 12, não faria sentido a existência de ItemPedido se não existisse Pedido.

Exemplo
Um exemplo de agregação seria a associação entre Equipe e Jogador, em que este último é parte da Equipe; entretanto, um Jogador existe independentemente da existência da Equipe. Isso significa que a destruição de uma Equipe (todo) não implicaria na destruição do Jogador (parte), como ocorre na composição.

HERANÇA

Outro relacionamento com semântica específica é o de herança, ou relacionamento de generalização/especialização. Esse tipo de relacionamento, diferente dos que você estudou, significa um relacionamento de generalidade ou especialidade entre as classes envolvidas. Por exemplo, o conceito animais é mais genérico que o conceito mamífero, que, por sua vez, é mais genérico que o conceito macaco.

Nesse tipo de relacionamento, as classes mais específicas herdam características das classes mais genéricas.

Usamos o termo subclasse para denotar a classe que herda as propriedades de outra classe mediante uma generalização. A classe que possui propriedades herdadas por outras classes é chamada de superclasse. No Diagrama de Classes, a herança é representada por uma flecha partindo da subclasse em direção à superclasse. Um relacionamento de herança representa um conjunto de objetos que compartilham um conjunto comum de propriedades (atributos, operações, associações), características inerentes à superclasse, além de possuírem características próprias.

A figura 13 mostra um exemplo do relacionamento de herança.

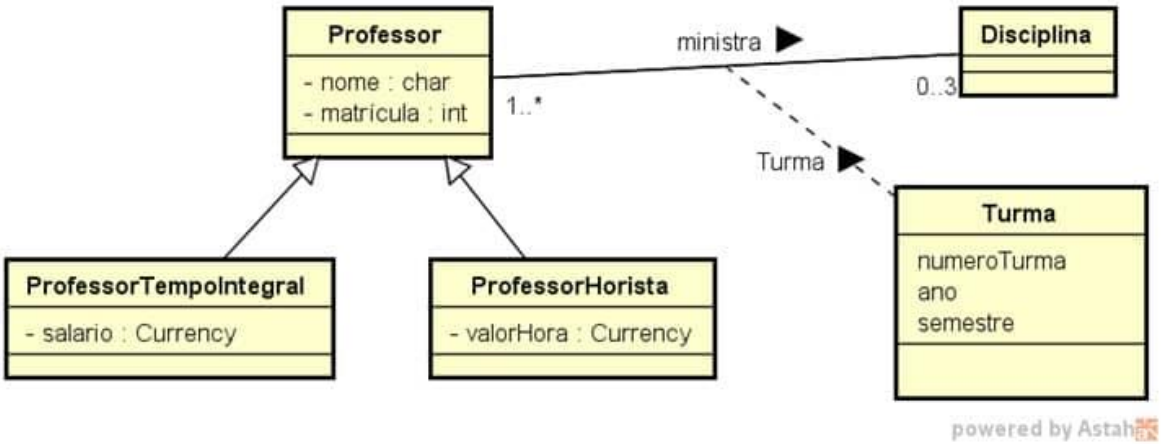


Figura 13 – Exemplo de relacionamento de herança, elaborado com ferramenta ASTAH (2021)

Nesse exemplo, um professor pode ser do tipo Tempo Integral ou Horista. Ambos possuem nome e matrícula, porém, o Professor Tempo Integral possui salário, e o Professor Horista possui valor de hora paga. Note que, além dos atributos da classe genérica (no exemplo, Professor), as classes especializadas herdam também as operações (não especificadas no exemplo), assim como os relacionamentos (no exemplo, a associação ministra) da classe genérica.