FACULDADE DE COMPUTAÇÃO E INFORMÁTICA
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

Projeto de Software – Aula 6 – 2º SEMESTRE/2018

Análise, Projeto e Desenvolvimento de Sistemas de Software – Qualificador, Herança, Generalização e Especialização

---

Objetivos:

- Compreender conceitos avançados de objetos, classes, ligações e associações através do uso de qualificador.
- Aprender novos tipos de associação: generalização/especialização para Diagramas de Classe

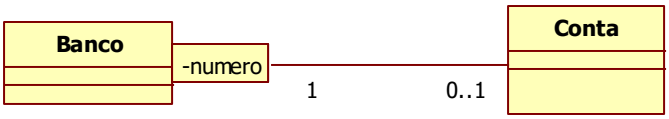
---

Qualificador- Associações Qualificadas

Observe este exemplo:

Neste exemplo um banco atende várias contas, mas uma conta pertence a único banco. Certo?
Entretanto, no contexto de um banco, o número da conta especifica uma conta exclusiva, ou seja, Banco e conta são classes o número da conta é Qualificador.
Desta forma, um qualificador seleciona entre os objetos de destino, reduzindo a multiplicidade efetiva de “muitos” para “um”.
As Associações qualificadas são usadas com associações de um para vários (1..*) ou vários para vários (*). O “qualificador” (identificador da associação qualificada) especifica como um determinado objeto no final da associação “n” é identificado, e pode ser visto como um meio de identifica de modo único todos os objetos na associação.

Definição: Associação qualificada é uma associação em que um atributo denominado qualificador distingue os objetos para uma extremidade de associações “muitos.
Observe novamente o exemplo:



Pode-se notar que este diagrama acrescenta informações: A combinação de um banco e um número da conta gera no máximo uma conta, ou seja, o diagrama com uso de qualificação destaca o significado do número da conta na travessia do modelo: você primeiro encontra o banco e depois especifica o número da conta para encontrá-la.

A classe origem (Banco) mais o qualificador (numero) geram a classe destino (Conta).

→ Banco + Número geram Conta

Importante: O qualificador é desenhado como uma pequena caixa no final da associação junto à classe de onde a navegação deve ser feita.

Herança

Uma hierarquia é a classificação ou ordenação de abstrações.
Existem duas importantes hierarquias em sistemas complexos, que são:
- sua estrutura de classes → hierarquia "é uma"
- sua estrutura de objetos → hierarquia "parte de"

Observe a figura a seguir.

Extraído do livro do Booch (2007).

Herança é a mais importante hierarquia “é um”, e é considerado um elemento essencial em sistemas orientados a objetos segundo Booch (2007).
A Herança define um relacionamento entre classes na qual uma classe compartilha estrutura e/ou comportamento definido em uma ou mais classes.
Portanto, herança representa uma hierarquia de abstrações, em que uma subclasse herda de uma ou mais superclasses, ou seja, a subclasse aumenta ou redefine a estrutura e comportamento existentes de suas superclasses.

Semanticamente, herança denota um relacionamento "é um". Por exemplo:

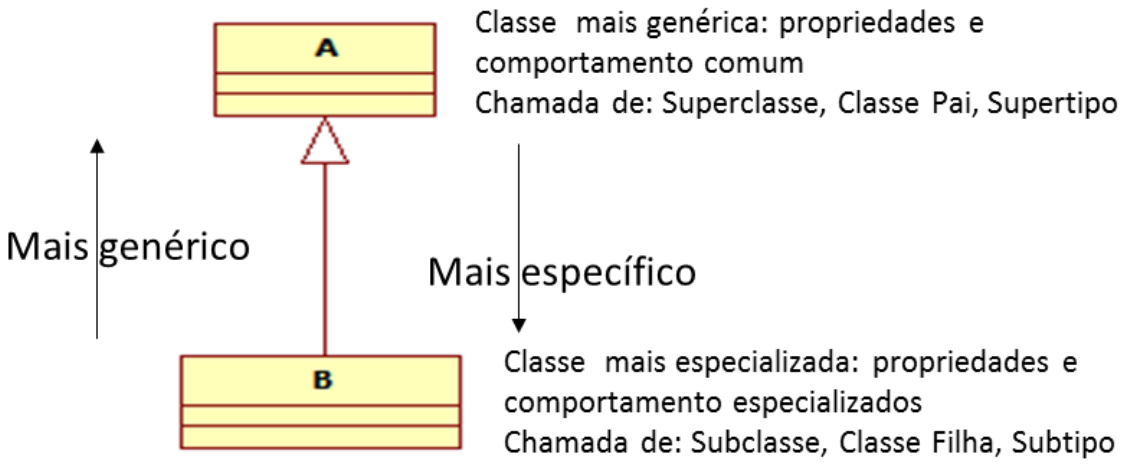
- um Homem “é uma” espécie de mamífero;
- um cachorro “é uma” espécie de mamífero;
- Um gato "é uma" espécie de mamífero;
- uma bicicleta “é um” meio de locomoção
- um carro “é um” meio de locomoção.

Pode-se perceber que a Herança implica uma hierarquia de **generalização / especialização**, em que uma subclasse especializada a estrutura mais geral ou o comportamento de suas superclasses.

Generalização é o relacionamento entre uma classe (**a superclasse**) e uma ou mais variações da classe (**as subclasses)**.

A generalização e a especialização são dois pontos de vista do mesmo relacionamento, ou seja, dado duas classes **A** e **B**, se **A** é uma generalização de **B**, então **B** é uma especialização de **A**.

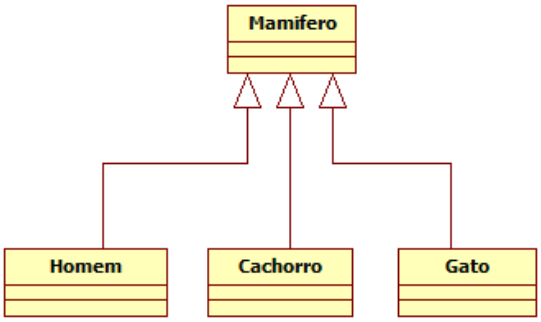
Como isso, é representado em UML?



Segundo Bezerra (2007), o termo subclasse é utilizado para denotar que a classe herda as propriedades de outra classe (superclasse) através de uma generalização, ou ainda, que a classe que possui as propriedades herdadas por outras é a superclasse. Também podem ser usados os seguintes nomes:

- Supertipo (superclasse) e subtipo (subclasse);
- Classe base (superclasse) e classe herdeira (subclasse);
- subclasse é uma especialização da sua superclasse, e que a uma superclasse é uma generalização de suas subclasses;
- o termo ancestral e descendente também são usados para fazer referência ao uso do relacionamento de generalização em vários níveis.

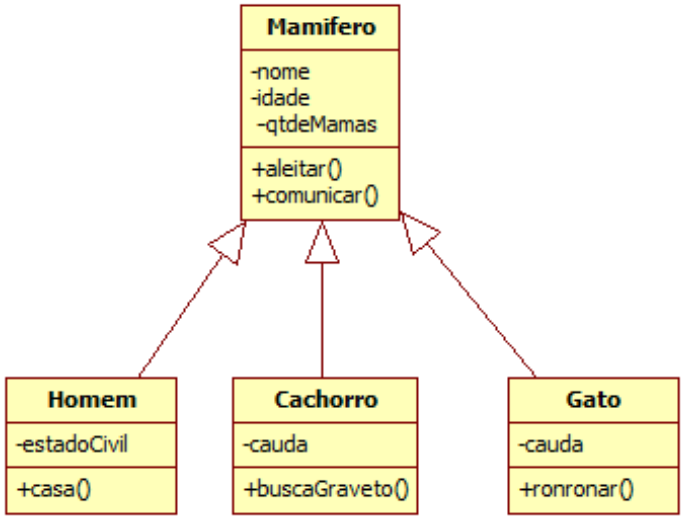
Assim, no exemplo dos mamíferos temos:



Semanticamente, a generalização/especialização denota um relacionamento "é um" relacionamento. Por exemplo, um **Homem** "é uma" espécie de **Mamífero**, um **Cachorro** "é uma" espécie de **Mamífero**, um **Gato** "é um" tipo de **Mamífero**.

A hierarquia de generalização / especialização, em que uma subclasse é a estrutura ou comportamento mais especializado de suas superclasses. A herança é o compartilhamento de atributos e operações (recursos) entre classes com base em um relacionamento hierárquico.

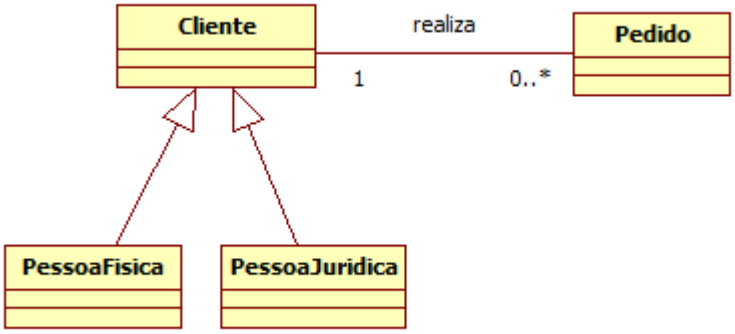
A superclasse mantém atributos, operações e associações comuns; as subclasses acrescentam atributos, operações e associações específicos. Assim, pode-se afirmar que a subclasse herda as características de sua superclasse. Veja o exemplo dos animais mamíferos:



Assim, para entender semântica de um relacionamento de herança, é preciso lembrar que uma classe representa um conjunto de objetos que compartilham um conjunto comum de propriedades (atributos, operações e associações). Entretanto, alguns objetos embora bastante semelhantes a outros podem possuir um conjunto de propriedades que outros não possuem.

Como por exemplo, todos os Mamíferos compartilham os atributos nome, idade e quantidade de mamas, entretanto, apenas alguns certos tipos de mamíferos que possuem estado civil e podem casar como os Homens. Já os mamíferos do tipo Gato e Cachorro possuem cauda, apenas os cachorros sabem buscar gravetos, e também apenas os gatos sabem ronronar.

Os cachorros pertencem à classe Mamífero, mas os cachorros pro sim só formam a classe Cachorro, portanto, os cachorros possuem todas as características inerentes a Mamífero, além de possuir as suas próprias características, desta forma, a classe cachorros herda as propriedades da classe Mamífero.
Outro aspecto importante a ser destacado é que, não somente atributos e operações são herdados pelas subclasses, mas também as associações que estão definidas na superclasse.
Veja o exemplo a seguir:



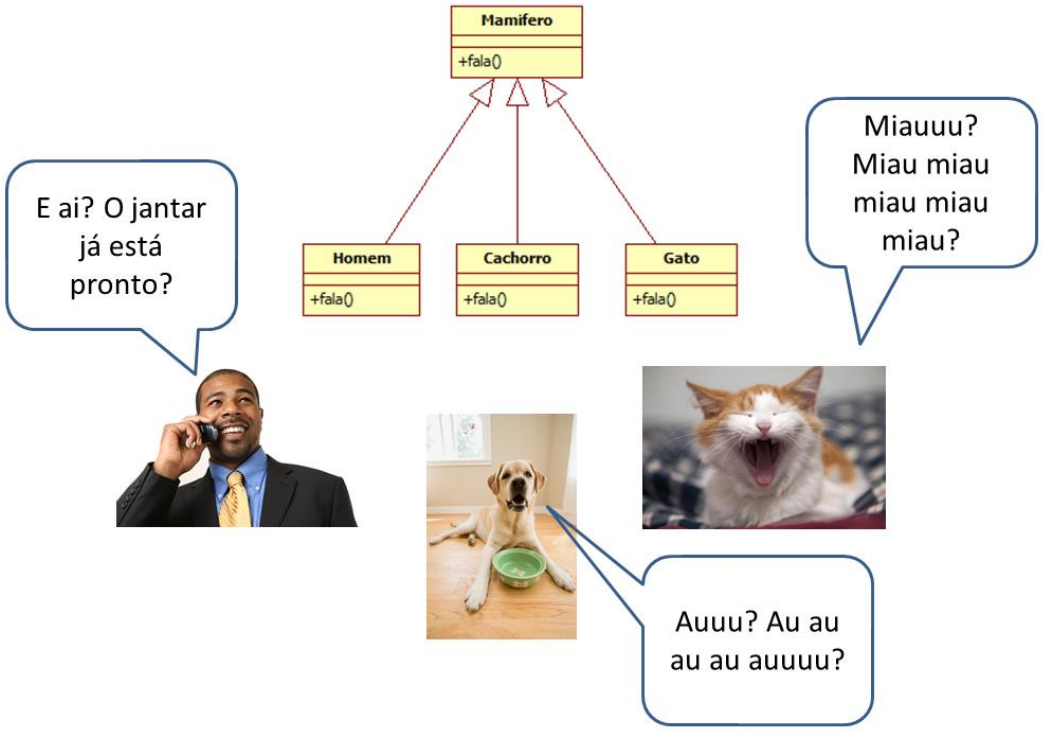
Neste exemplo existe uma associação entre Cliente e Pedido, não há necessidade de criar uma associação Pedido e PessoaFisica ou entre Pedido e PessoaJuridica, pois as subclasses herdam de a associação de sua superclasse.

Entretanto, se existe alguma associação que não é comum a todas as subclasses, mas, sim a alguma subclasse particular, então essa associação deve ser representada em separado, envolvendo somente as subclasses em questão.

Sobrescrevendo Características

- ➔ Uma subclasse pode sobrescrever uma característica da subclasse definindo uma característica com mesmo nome, ou seja, a característica que sobrescrever (a característica da subclasse) refina e substitui a característica sobrescrita (a característica da superclasse).
- ➔ Motivos para você querer sobrescrever uma característica:
    1. Para especificar o comportamento que depende da subclasse
    2. Para refinar a especificação de uma característica
    3. Para melhorar o desempenho

Veja este exemplo:

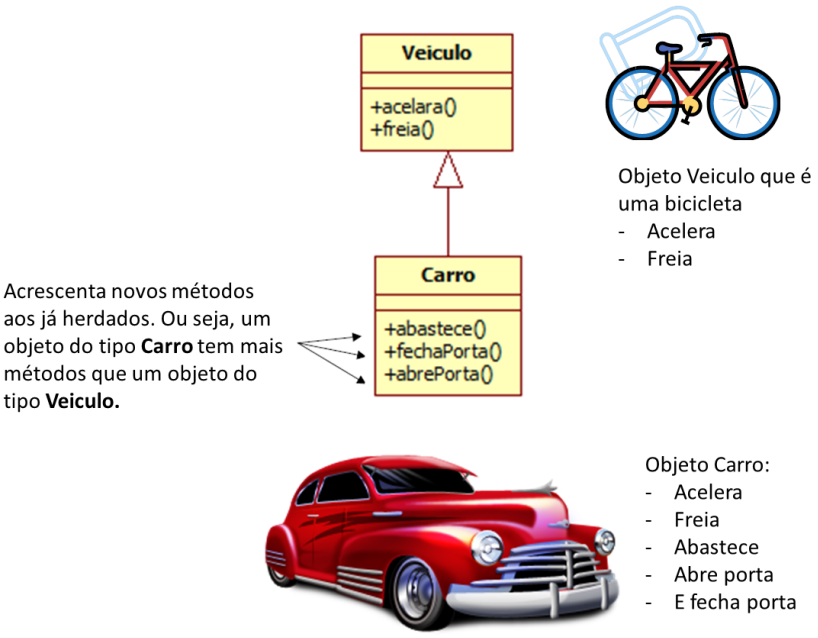


Uma subclasse herda todas as propriedades de sua superclasse que tenham **visibilidade pública (+)** ou **protegida (#).**

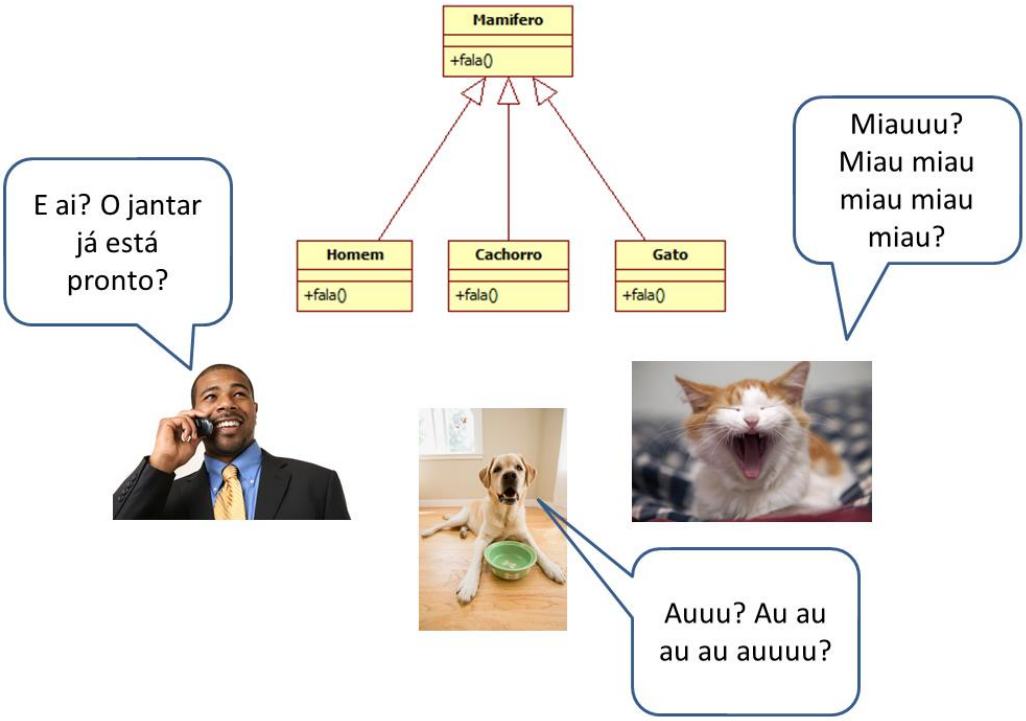
Pode-se observar que, uma subclasse pode receber uma mensagem para que uma operação (pública ou protegida) definida na superclasse seja executada.

A subclasse pode estender e sobrescrever comportamento da superclasse.

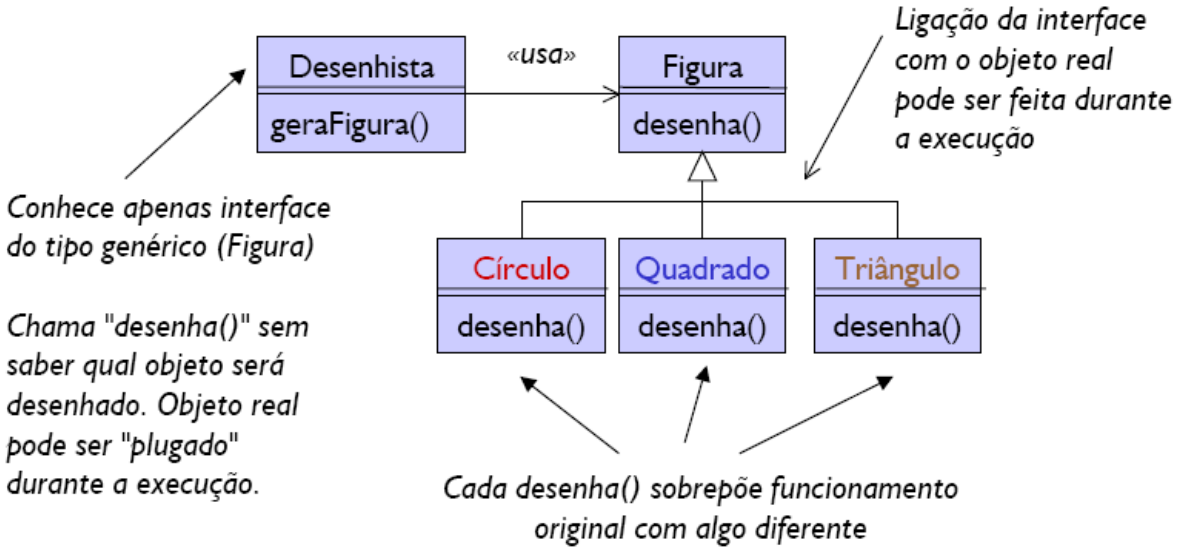
Vejam este exemplo:



Ou seja, o comportamento do de alguma operação da classe herdada é diferente para a algumas subclasses. Neste caso, a subclasse tem que redefinir o comportamento da operação da superclasse. Desta forma, o comportamento de alguma operação herdada tenha que ser diferente, como mostra o exemplo da classe de mamíferos apresentado na aula anterior, como relação a forma como homem, o cachorro e gato implementam o método **fala()**.
Vocês lembram este exemplo:



Ou seja, **operações polimórficas** são operações de mesma assinatura definidas em diversos níveis de uma hierarquia de herança e que possuem métodos diferentes, sendo que, estas operações devem ter a assinatura repedida na (s) subclasse(s) para enfatizar que elas são redefinidas, desta forma, somente as assinaturas são herdadas, mas não a implementação.
Polimorfismo é o princípio da orientação a objetos que em que duas ou mais classes derivadas de uma mesma superclasse podem invocar métodos que têm a mesma assinatura, mas comportamentos distintos, especializados para cada classe derivada, usando para tanto uma referência a um objeto do tipo da superclasse.



Importante:

A decisão sobre qual o método que deve Ser selecionado, de acordo com o tipo da classe derivada, é tomada em tempo de execução, através do mecanismo de ligação tardia.

Ligação Tardia: Para a utilização de polimorfismo, a linguagem de programação orientada a objetos deve suportar o conceito de **ligação tardia** *(late binding)*, em que a definição do método que será efetivamente invocado só ocorre durante a execução do programa. Também é conhecido pelos termos *dynamic binding* ou *run-time binding.*

Ligação prematura: Quando o método a ser invocado é definido durante a compilação do programa, o mecanismo de **ligação prematura** *(early binding).*

Outro conceito importante:

- ➔ Sobrescrita dos métodos não é mesma coisa de sobrecarga (***overloading***) de métodos, e assim, não deve ser confundido com o mecanismo de sobrecarga de métodos.
- ➔ Sobrecarga de métodos: Métodos podem compartilhar o mesmo nome, mas possuir diferentes parâmetros, que variam em número ou tipo.

Um método aplicado a um objeto é selecionado para execução através da sua assinatura e da verificação a qual classe o objeto pertence.

Mecanismo de sobrecarga (**overloading**) - dois métodos de uma mesma classe podem ter o mesmo nome, desde que, suas listas de parâmetros sejam diferentes, constituindo assim uma assinatura diferente.

Essa situação não gera conflito, uma vez que, o compilador é capaz de detectar qual método deve ser escolhido a partir da análise dos tipos de argumentos do método.

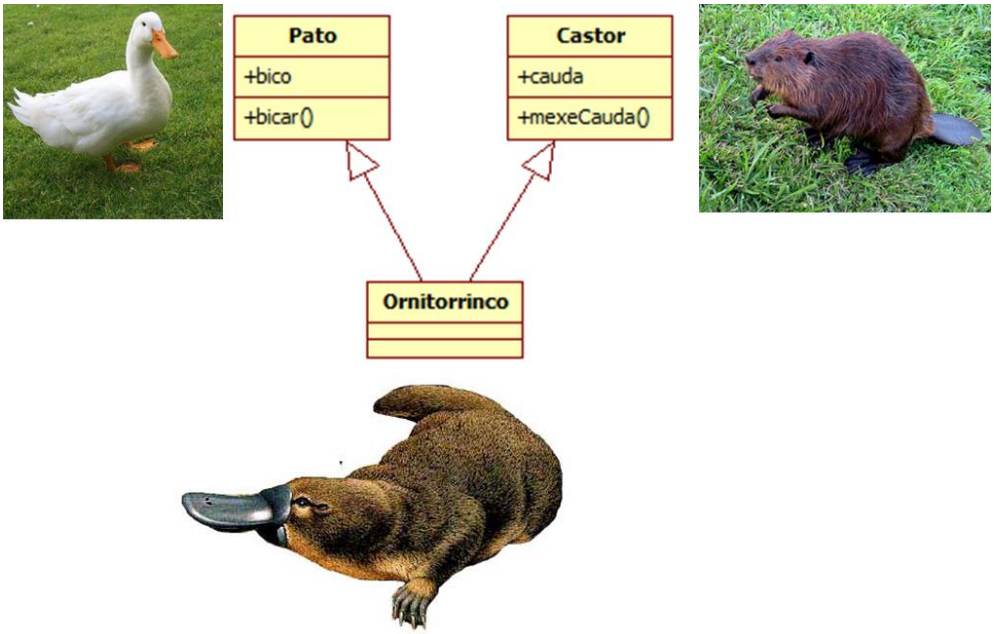
Pode ser citado como exemplo em Java - os métodos abs(), max() e min() da classe Math, que têm implementações alternativas para quatro tipos de argumentos distintos.

Herança Múltipla

Existem diversas categorizações que podem ser feitas a respeito do conceito de herança, uma delas é com relação a quantidade de superclasses que uma certa classe pode ter. Assim uma classe pode ter mais de uma superclasse, esse é o conceito de **herança múltipla**.

Desta forma, a herança múltipla permite que uma classe possua mais de uma superclasse e herde características de todos os seus ancestrais. O que possibilita a mesclagem de informações de duas ou mais origens.

Exemplo:



A herança múltipla é uma forma mais complicada de generalização do que a herança simples, que restringe a hierarquia de classes a uma árvore.

Vantagem da herança múltipla: maior capacidade de especificação de classes e a maior oportunidade de reutilização. A herança múltipla habilita a modelagem de objetos para mais próximo da maneira como se costuma pensar.

Desvantagem da Herança Múltipla: perda em simplicidade conceitual e de implementação.

Nem todas as linguagens de programação orientadas a objetos implementam a herança múltipla. Algumas linguagens que implementam herança múltipla, como por exemplo: C++ e Phyton. As linguagens Java, C# e Smalltalk não implementam herança múltipla.

Referências bibliográficas

Bezerra, E. (2017). Princípios de Análise e Projeto de Sistema com UML, 3a. Edição. Elsevier Brasil.

BLAHA, M., RUMBAUGH, J. Modelagem e projetos baseados em objetos com UML 2. Rio de Janeiro:Elsevier-Campus, 2006.

BOOCH, G. Object-oriented analysis and design with applications. 3ª.ed. Addison-Wesley, 2007.