Herança Múltipla

João Vítor Moreira Dos Santos Neris¹, Matheus Godoy Dos Reis²

¹Faculdade de Tecnologia – Faculdade de tecnologia de Itaquera (FATEC)  
Caixa Postal 08295 - 005 – São Paulo – SP – Brasil

[joao.neris@fatec.sp.gov.br](mailto:joao.neris@fatec.sp.gov.br), [matheus.reis19@fatec.sp.gob.br](mailto:matheus.reis19@fatec.sp.gob.br)

**Abstract.** This article talks about multiple inheritance, we introduce the reader by showing that inheritance trees, it is important that you know the concept so that we can approach the topic of multiple inheritance using this important resource. Having this basis, the reader will be able to more easily understand the concept of Multiple Inheritance, which is the real objective of this article. Below we will address and analyze how it works in the Java programming language and discuss its advantages and disadvantages.

**Resumo.** Este artigo fala sobre herança múltipla, introduzimos o leitor mostrando que árvores de herança, é importante que conheça o conceito para abordamos o tema de herança múltipla usando este importante recurso. Possuindo esta base, o leitor poderá entender mais facilmente o conceito de Herança Múltipla que é o real objetivo deste artigo. A seguir abordaremos e analisaremos seu funcionamento na linguagem de programação Java e discutiremos as suas vantagens e desvantagens.

1. **Introdução**

## Árvore de Herança

A árvore de Herança é uma representação visual para a hierarquia de classe, onde os relacionamentos são feitos usando herança, todas os relacionamentos de herança podem ser representados usando árvores de herança, ela lembra muito um diagrama UML, porém acaba sendo mais específica tratando apenas sobre a hierarquia entre as classes, enquanto o UML acaba sendo mais abrangente. Como neste artigo nós tratarem apenas de herança múltipla, a árvore de herança acaba sendo mais do que suficiente para exemplificar situações de diagrama de classe. Logo, usaremos este recurso neste artigo para fazer a representação e explicação de classe se relacionando com outras através de Herança.

## Conceitos de árvore de Herança

## Classe pai

Também conhecida como superclasse, é uma classe que fica imediatamente acima de outra classe, ela vai ter seus comportamentos e atributos herdados por outras classes que estão abaixo dela.

## Classe filha

Também conhecida como subclasse, é uma classe que fica imediatamente abaixo de outra classe, ela vai herdar todos os comportamentos e atributos da classe ou classes que está ou estão acima dela.

## Ancestral

A classe ancestral é uma classe que está acima de outra classe na hierarquia, porém não é a classe pai. A classe ancestral também tem seus comportamentos e características herdados pelas classes que estão abaixo dela.

## Descendente

A classe descendente é uma classe que está abaixo de outra classe na hierarquia, porém não é a classe filha.

## Classe raiz

Classe raiz ou classe base é a primeira classe que aparece na hierarquia de uma árvore. Ela é a classe mais alta na hierarquia e todas as outras classes herdam comportamento e características de outras classes.

## Classe folha

A classe folha é uma classe que não possui outras subclasse associadas a ela.

## Generalização

A generalização é um processo de criação de uma classe que encapsula características e comportamentos que são comuns a duas ou mais classes. Esta criação gera uma classe mais abstrata em relação àquelas que vão herdar suas características e comportamentos.

## Especialização

A especificação é um processo em que é gerado uma classe mais específica a partir de outra classe mais abstrata, é possível gerar mais de uma classe específica a partir dessa classe abstrata. As classes mais específicas herdam características e comportamentos de suas classes abstrata.

Diagrama

Descrição gerada automaticamente

Figura 1. Representação da árvore de herança e suas respectivas relações.

Nesta imagem é possível perceber a relação de hierarquia entre as classes, cada retângulo nesta imagem representa uma classe, a classe A é a primeira classe de nossa hierarquia, ela é a classe mais abstrata de todas, é conhecida como classe raiz, abaixo dela teremos suas subclasses, a classe AA é uma subclasse quando comparada com a classe A, porém ela pode ser considerada uma superclasse quando com comparada com a classe AAA, as classes AAA, ABAA, ABAB, ABB, ACA podem ser consideradas como classes folhas, pois abaixo delas não temos nenhuma outra classe associada, quanto mais descemos em nesta hierarquia, mais específicas vão ser as classes, processo conhecido como especialização e quanto mais subimos em nesta hierarquia, mais genérica a classe vai ser, processo conhecido como generalização.

1. **Herança**

A Herança é um dos conceitos fundamentais da programação orientada a objeto que permite basear uma nova classe na definição de uma outra classe existente, ou seja, uma classe pode herdar características e comportamento de uma outra classe. Essa relação permite a reutilização de código de uma outra classe, ao mesmo tempo, adiciona ou modifica um comportamento em específico.

## Subclasse

A subclasse, também conhecida como classe derivada, é uma classe que herda métodos e atributos de uma outra classe.

## Superclasse

A superclasse, também conhecida como classe base ou classe pai, é uma classe onde outras classes podem herdar métodos e atributos.

1. **Tipos de herança**

## Herança simples

Herança simples consiste em uma superclasse tendo seus métodos e atributos herdados por uma ou mais subclasses. Por exemplo, no contexto de um sistema para uma escola, a classe Pessoa (Superclasse) pode ter seus atributos (características) e métodos (comportamentos) herdados por outras subclasses, como a classe Aluno, Professor e Funcionário.

Diagrama

Descrição gerada automaticamente

Figura 2. Uma árvore de herança representando a herança simples.

## Herança múltipla

Herança múltipla é quando uma subclasse herda atributos e métodos de mais de uma superclasse. Por exemplo, a classe Animal pode ter o método comer e a classe mamífero pode ter o método amamentar, logo eu poderia fazer uma classe Humano que herdaria o método amamentar da classe mamífero e o método comer da classe animal, tendo assim os dois métodos.

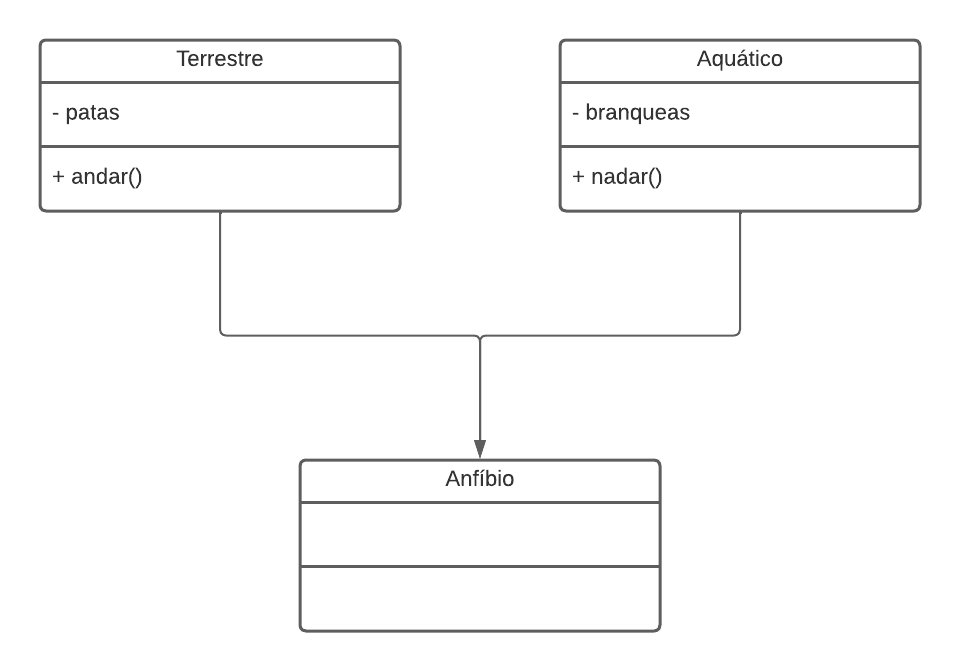


Figura 3. Uma árvore de herança representando a herança simples.

1. **Vantagens da Herança Múltipla**

A herança múltipla apresenta a diversas vantagens, e uma delas está na reutilização de código de múltiplas fontes, é possível criar uma nova classe a partir de outras herdando seus comportamentos e características, por exemplo, a classe Pessoa possui métodos e atributos que podem ser usados em outras classes como a classe Aluno, Professor, Diretor, entre diversas outras, ao invés de implementar todos os métodos e atributos em cada classe, nós utilizamos uma única classe para conter todos esses comportamentos e características para compartilhar com diversas outras classes.

1. **Desvantagens da Herança Múltipla**

Existe alguns fatores que pode implicar em uma linguagem de programação não permitir a herança múltipla de forma direta.

Um deles é que a herança múltipla por herdar métodos e atributos de diversas classes pode trazer certa ambiguidade, principalmente quando métodos e atributos de diferentes classes possuem o mesmo nome.

Outro problema que isso pode gerar é um conhecido como diamante, seu nome é devido ao seu formato adquirido na árvore de herança que lembra um diamante, esse problema ocorre quando uma classe herda comportamentos e atributos de duas classes que herdam de outra classe em comum, por exemplo, eu tenho a classe que B e a classe que C, ambas herdam características e comportamento da classe A, quanto eu introduzo uma classe D herdando características e comportamento da classe B e classe C, ou sejam, em Herança múltipla, eu acabo criando uma problema de ambiguidade, pois ambas, classe C e D, tem um conjunto de métodos e atributos iguais que foram herdados de uma mesma classe, no momento em que eu vou fazer o uso desses métodos.

Diagrama

Descrição gerada automaticamente

Figura 4. Árvore herança representando a estrutura do problema diamante.

A herança múltipla pode trazer complexidade nos códigos, muitas linguagens priorizam o princípio de simplicidade e legibilidade do código, trazer o conceito de herança múltipla pode adicionar complexidade desnecessária a certos programas, em que fazer a manutenções pode se tornar algo mais desafiador.

1. **Herança múltipla em Java**

No Java uma classe pode herdar métodos e atributos apenas de uma classe, em teoria, não é permitido fazer herança múltipla em Java. O que acontece é que pelo fato de a herança múltipla apresentar alguns problemas na implementação como ambiguidade, complexidade, entre outras coisas, Java acabou não permitindo acesso direto à Herança múltipla, é possível usar os conceitos de Herança múltipla em Java, porém utilizando interfaces.

Java possui um recurso para simular a herança múltipla, trazendo o conceito de interface. Dessa forma, uma classe pode implementar várias interfaces podendo assim herdar diversos comportamentos (métodos). Neste sentido, é possível entender que, em Java, não é permito herança múltipla em classes, porém é permitido herança múltipla em interfaces.

## Interface em Java

Uma interface é um contrato que a classe deve seguir, nesta interface, veremos a presença de métodos abstratos e atributos finais constantes. Esta interface pode ser implementada por uma classe, após isso, a classe deverá implementar todos os métodos abstratos que estão na interface.

Tabela 1. Tabela de comparação entre a interface e a classe abstrata

|  |  |
| --- | --- |
| **Interface** | **Classe** **Abstrata** |
| A herança múltipla é permitida: uma classe pode estender várias interfaces. | A herança múltipla não é permitida: Uma classe pode estender somente uma classe. |
| Na interface os métodos, por padrão, vêm definidos como público e abstrato, mas se necessário pode se utilizar outros modificadores. | Na classe abstrata os métodos por padrão vêm definidos como público, porém não como abstrato, mas ainda assim, é possível usar modificadores para realizar alterações. |
| Uma classe utiliza a palavra implements para implementar uma classe. | Uma classe utiliza a palavra extends para estender uma classe. |
| Interfaces não tem métodos construtores, já que todos seus atributos são constantes. | A classe abstrata pode ter métodos construtores. |
| Todos os métodos, por padrão, precisam ser sobrescritos. | Todos os métodos que utilizarem o modificador abstract precisam ser sobrescritos. |
| Todos as variáveis que forem declaradas em uma interface são constantes, ou seja, vem com o modificador final. | As variáveis podem ser declaradas como constantes, mas isto é opcional. |
| Interfaces, por padrão, não tem implementações | Classes abstratas podem ou não ter implementações e também podem ter implementação parciais. |

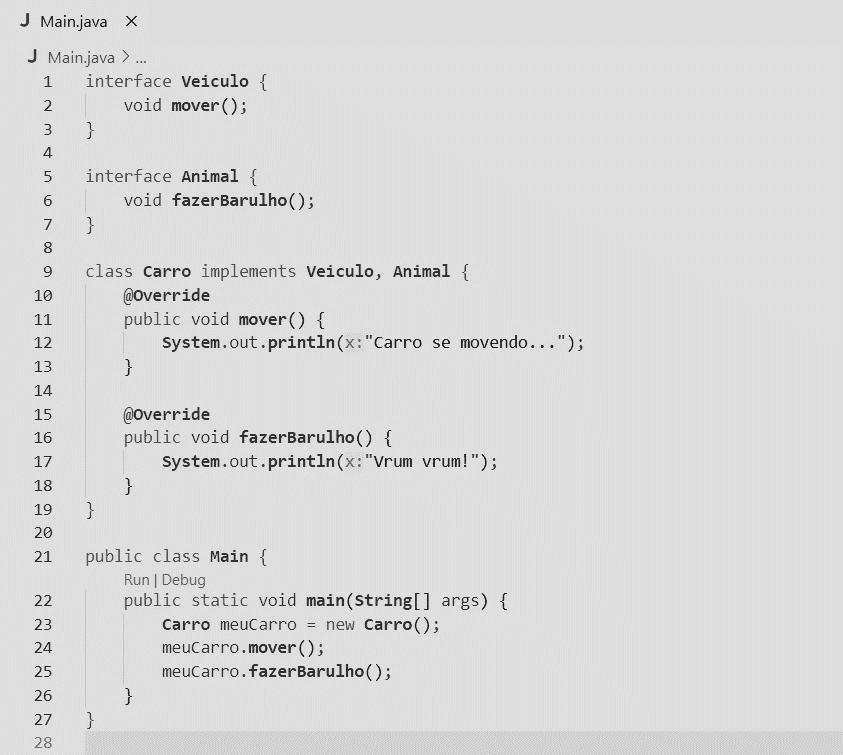


Figura 5. Código exemplificando Herança Múltipla em Java.

Neste código temos um exemplo de herança múltipla em Java, nele criamos duas interfaces, uma chamada Veículo com a assinatura de um método mover e na outra chamada Animal com a assinatura de um método chamado mover, amboas os métodos são abstratos e precisam ser implementados na classe que implementar estas interfaces. Logo abaixo, temos a classe Carro que implementa a interface Veículo e a interface Animal, na classe os métodos foram implementados de acordo com as necessidades da classe carro. Perceba que a classe Carro pode abstrair o método fazerBarulho da classe Animal e implementá-lo de forma que se adeque em sua classe. Por fim, temos na classe Main a classe carro sendo instanciada e criando o objeto meuCarro, nestes objetos nós temos acesso aos métodos mover e fazerBarulho que foram implementados na classe carro.

**Referências**

<https://materialpublic.imd.ufrn.br/curso/disciplina/4/8/7/4#:~:text=Descendente%3A%20dada%20uma%20classe%2C%20toda,a%20classe%20topo%20da%20hierarquia>

<https://www.cin.ufpe.br/~gta/rup-vc/core.base_rup/guidances/guidelines/generalization_94751549.html>

[https://web.tecgraf.puc-rio.br/~marcio/cursos/oo/hermul.htm#:~:text=A%20vantagem%20da%20heran%C3%A7a%20m%C3%BAltipla,simplicidade%20conceitual%20e%20de%20implementa%C3%A7%C3%A3o](https://web.tecgraf.puc-rio.br/~marcio/cursos/oo/hermul.htm).

<https://www.inf.pucrs.br/~manssour/LinguagemC++/HerancaMultipla.pdf>

<https://www.dca.fee.unicamp.br/cursos/POOCPP/node55.html>

<https://edisciplinas.usp.br/pluginfile.php/4671040/mod_resource/content/3/heranca.pdf>

http://www.decom.ufop.br/guillermo/BCC221/slides/Heranca\_Multiple\_\_\_POO.pdf