

PARADIGMAS DE LINGUAGEM DE PROGRAMAÇÃO

PROFESSOR: DANIEL ABELLA

ALUNA: FABRÍCIA MARTINS SILVA

SLIDE 1 - ATIVIDADE 1

**PROBLEMA DO DIAMANTE**

**Herança**

Basicamente o uso de herança faz com que uma classe herde atributos e métodos de outra classe existente.

Em algumas linguagens também há a possibilidade de se herdar de mais de uma classe, a chamada herança múltipla. C++ e Python são uma das linguagens que suportam esse recurso, mas Java está entre as linguagens que não suporta esse recurso.

**O Problema**

Se Java se espelha em C++, então porque ele não suporta herança múltipla?

O uso de herança múltipla pode gerar uma grande dor de cabeça para os desenvolvedores.

Imagine uma classe Carro que implementa um método chamado obterNumeroDeRodas que retorna o valor 4. Outra classe chamada Moto que também implementa um método chamada obterNumeroDeRodas com a mesma assinatura, mas que retorna o valor 2. Agora você cria outra classe chamada Quadriciclo que faz uso da herança múltipla, herdando de Carro e Moto. Então quando o método obterNumeroDeRodas for chamado, qual método retornará? O da classe Carro ou o da classe Moto?

Essa situação gera uma confusão e faz com que o resultado seja imprevisível. Esse problema é chamado "Diamond problem" (Problema Diamante).

**A Solução**

Uma forma de contornar esse problema seria o uso de Interface. Nesse caso, não haveria implementação dos métodos, mas sim um contrato que toda classe que implementar as interfaces deverão assinar.

Por exemplo, Carro e Moto poderiam ser uma interface e a classe Quadriciclo que as implementaria, teria que implementar o seu próprio método obterNumeroDeRodas.

interface ICarro {

int obterNumeroDeRodas();

}

interface IMoto {

int obterNumeroDeRodas();

}

class Quadriciclo extends ICarr, IMoto {

public int obterNumeroDeRodas() {

// sua implementação

}

}

Com essa solução não há reuso de código. Para resolver esse impasse o Java 8 disponibiliza o chamado método default, que possibilita a implementação dos métodos em interfaces.

Então como é possível implementar mais de uma interface, a classe Quadriciclo pode implementar as duas interfaces Carro e Moto, e como as duas interfaces possuem um método com a mesma assinatura, o compilador não aceita, forçando a classe Quadriciclo implementar seu próprio método obterNumeroDeRodas, mas como as interfaces já tem suas implementações, a classe Quadriciclo pode escolher uma delas quando for sobreescrever o método.

public interface ICarro {

default public int obterNumeroDeRodas(){

return 4;

}

}

public interface IMoto {

default public int obterNumeroDeRodas(){

return 2;

}

}

public class Quadriciclo implements ICarro, IMoto {

@Override

public int obterNumeroDeRodas(){

return ICarro.super.obterNumeroDeRodas();

}

}

**Referências**

<http://www.brunonardini.com.br/desenv-back-end/metodo-padrao-e-heranca-multipla-no-java-8>

<http://pt.stackoverflow.com/questions/42970/por-que-todo-mundo-odeia-heran%C3%A7a-m%C3%BAltipla-em-c-e-qual-sua-diferen%C3%A7a-para-mixin>

<http://www.devmedia.com.br/heranca-conceitos-e-o-problema-diamante-parte-i/2697>

<http://blog.victormonte.com/heranca-multipla-e-o-problema-do-diamante/>