Resumo: Problema do diamante

Aluno: Danilo de Queiroga

A linguagem de programação Java possui diversas funcionalidades por ser uma linguagem orientada a objetos. Dessa forma, ela dispõe de um recurso de herança, que permite a criação de novas classes a partir de outras previamente criadas. Essas novas classes são chamadas de subclasses, ou classes derivadas; e as classes já existentes, que deram origem às subclasses, são chamadas de superclasses, ou classes base. Deste modo é possível criar uma hierarquia dessas classes, tornando, assim, classes mais amplas e classes mais específicas. Uma subclasse herda métodos e atributos de sua superclasse; apesar disso, pode escrevê-los novamente para uma forma mais específica de representar o comportamento do método herdado.

Entretanto, a linguagem Java não permite heranças múltiplas. A herança múltipla torna-se um problema quando um filho tem dois pais, e esses dois pais têm dois métodos não relacionados, mas com o mesmo nome, nesse caso temos o problema do diamante.

Considerando uma herança em que uma classe A tenha 2 filhos, B e C. E uma classe D seja filho ao mesmo tempo de B e C. A classe D enxerga ambiguidade em todos os métodos vindos de A, já que ela implementa o A que vem de B e o A que vem de C. Essa funcionalidade pode ser encontrada em outra linguagem de programação, mas como existem mais exceções do que regras pros casos onde ela pode ser útil, e dezenas de situações onde ela pode ser confusa, os projetistas do Java resolveram remove-la.

Porém, Java implementa o conceito de interfaces. Interface é uma classe que não tem implementação ou atributos. Ou seja, ela representa um contrato, uma coisa que a classe precisa fazer. A vantagem é que os compiladores podem ter certeza de que com interfaces, nunca há ambiguidade, já que a interface por si só não tem comportamento. Por isso uma classe pode implementar diversas interfaces.