Resumo – Herança Aberta e Projeto de Herança

Projeto de Herança

Josh Bloch é quem melhor resume projeto de herança: “desenvolva e documente muito bem as heranças ou proiba-as”. Devemos tomar cuidado pra ver quais métodos podem ser herdados e verificar se não estamos fazendo muita sobrescrita.
A herança é algo muito íntimo, muito entranhado no código, então deve ser muito bem desenhado, projetado, planejado, desenvolvido, para que não fiquem pontos abertos no código. Muitos desenvolvedores, principalmente aqueles mais desinibidos (ou desavisados), acham esse argumento não muito convincente, mas em algum momento veem que é algo a se pensar.
As heranças são muito observadas em bibliotecas e conjuntos de bibliotecas, então podemos dizer que elas existem para simplificar as tecnologias para os usuários, por meio de herança ou de instância. “APIs são escritas por experts para não experts” é uma citação uilizada pelo autor do texto.

Então como exemplo dado por Martin Fowler, o XOM é uma API open source para processamento de XML com Java. Ela deve permitir que possamos sobrecrever as classes das bibliotecas caso queiramos e mesmo assim a API me trará os mesmos resultados. Ou seja, não importa o código que eu inserir ali, desde que esteja de acordo com as normas da API e das bibliotecas do XOM, o XML será produzido (por exemplo).

É sobre isso que o Martin Fowler diz ao dizer que o projeto de herança é importante no desenvolvimento. Não queremos, como usuários, perder muito tempo com coisas específicas de uma linguagem. Queremos utilizar diretamente as ferramentas que elas nos proporcionam de acordo com nossas necessidades.
Apesar disso, diz que sua mente ainda insiste em preferir herança aberta. Uma alternativa para apaziguar a discussão seria o modo como você sai de seu padrão para sobrescreer algo que não foi antes pensado(desenvolvido). Se você mantém a classe concreta, então o compilador só permite herança normal, mas se você utilizar o mecanismo abstrato, o compilador permitirá que você faça mais alterações do que se já fosse um bloco selado, concreto.
Martin Fowler diz, então, que há ainda muito o que pensar sobre interface, tanto como por instancia como por herança. Precisamos de mais ideias e estudos a respeito de providers e consumers, utilizados em pacotes de serviços, para que possamos fazer um melhor uso das interfaces.

A Herança aberta é o oposto do projeto de Herança (e é por isso que há tanto embate entre elas). Apesar de serem opostas, a comunidade não costuma discordar quanto aos perigos da herança. Por ser uma relação muito íntima, é dificultosa para os desavisados.

Um trecho que nos ajuda a entender melhor a diferença entre herança aberta e fechada é que o projeto de herança, a herança fechada exige muito conhecimento por parte do desenvolvedor. A herança aberta permite que o usuário da classe possa consertá-las ou até atribuir a ela novas funcionalidades, aproveitando-se do trabalho de outros desenvolvedores.