Princípio do aberto/fechado

Origem: Wikipédia, a enciclopédia livre.

Na programação orientada a objeto, o **princípio do aberto/fechado** estabelece que "*entidades de software (classes, módulos, funções, etc.) devem ser abertas para extensão, mas fechadas para modificação*";[1] isto é, a entidade pode permitir que o seu comportamento seja estendido sem modificar seu código-fonte.

O nome do *princípio aberto/fechado* tem sido usado de duas maneiras. Ambas as maneiras usam generalizações (por exemplo, herança , ou delegação de funções) para resolver o aparente dilema, mas os objetivos, as técnicas e os resultados são diferentes.

Índice

Princípio aberto/fechado de Meyer

Bertrand Meyer geralmente é creditado por ter originado a expressão do princípio *aberto/fechado*,[2] que apareceu em sua obra Software Orientação a objetos de Construção, de 1988.

> - Um módulo será dito para ser aberto se ele ainda está disponível para extensão. Por exemplo, deve ser possível adicionar campos para as estruturas de dados que ele contém, ou novos elementos para o conjunto de funções que executa.
> - Um módulo será dito ser fechado se [ele] está disponível para uso por outros módulos. Isso pressupõe que o módulo tenha sido bem-definido, estável descrição (o interface no sentido de ocultar informações).

No momento que Meyer estava escrevendo aquela obra, adicionar campos ou funções em uma biblioteca, inevitavelmente, geraria alterações necessárias para todos os programas que dependiam daquela biblioteca. A solução proposta por Meyer para este dilema dependia da noção de herança presente na orientação a objeto (especificamente a herança de implementação):

> Uma classe é fechado, uma vez que podem ser compilados, armazenados em uma biblioteca, baselined, e utilizados pelo cliente classes. Mas também é aberto, uma vez que qualquer nova classe pode usá-lo como pai, a adição de novas funcionalidades. Quando uma classe descendente é definida, não há necessidade de alterar o original ou perturbar seus clientes.

Princípio de aberto/fechado polimórfico

Durante a década de 1990, o princípio de aberto/fechado tornou-se redefinido popularmente para se referir ao uso de interfaces abstratas, onde as implementações podem ser alterados e várias implementações poderiam ser criadas e polimorficamente substituídas por outras.

Em contraste com o uso de Meyer, esta definição defende a herança de classes base abstratas. Especificações de Interface podem ser reutilizadas através de herança, não sendo necessário uma implementação. A interface existente é fechada para modificações e novas implementações devem, no mínimo, implementar essa interface.

Robert C. Martin's no artigo, de 1996, "Open-Closed Princípio"[3] foi um dos seminais escritos para tomar essa atitude. Em 2001 Craig Larman relacionou o princípio aberto/fechado com o padrão, de Alistair Cockburn, chamado Variações Protegidas, e com a discussão de David Parnas *sobre esconder informações*.[4]

Veja também

- SOLID – o "O" em "SOLID" está para o aberto/fechado princípio

Referências

1. Meyer, Bertrand. *Object-Oriented Software Construction*. [S.l.: s.n.] ISBN 0-13-629049-3
2. Robert C. Martin "The Open-Closed Principle", C++ Report, January 1996, pp. 1 (http://docs.google.com/a/cleancoder.com/viewer?a=v&pid=explorer&chrome=true&srcid=0BwhCYaYDn8EgN2M5MTkwM2EtNWFkZC00ZTI3LWFjZTUtNTFhZGZiYmUzODc1&hl=en) Arquivado em (https://web.archive.org/web/20060822033314/http://www.objectmentor.com/resources/articles/ocp.pdf) agosto 22, 2006[Erro data trocada], no Wayback Machine.
3. Robert C. Martin "The Open-Closed Principle", C++ Report, January 1996 (http://docs.google.com/a/cleancoder.com/viewer?a=v&pid=explorer&chrome=true&srcid=0BwhCYaYDn8EgN2M5MTkwM2EtNWFkZC00ZTI3LWFjZTUtNTFhZGZiYmUzODc1&hl=en) Arquivado em (https://web.archive.org/web/20060822033314/http://www.objectmentor.com/resources/articles/ocp.pdf) agosto 22, 2006[Erro data trocada], no Wayback Machine.
4. Craig Larman, "Protegidos da Variação: A Importância de Ser Fechado", IEEE Software de Maio/junho de 2001, pp. 89-91 *[1] (http://codecourse.sourceforge.net/materials/The-Importance-of-Being-Closed.pdf)*

Links externos

- Os Princípios de INUNDAÇÃO (http://butunclebob.com/ArticleS.UncleBob.PrinciplesOfOod)
- Aberto/Fechado Princípio: Preocupações sobre a Mudança no Design de Software (http://blog.symprise.net/2009/06/23/open-closed-principle-concerns-about-change-in-software-design/)
- Aberto Fechado Princípio (https://8thlight.com/blog/uncle-bob/2014/05/12/TheOpenClosedPrinciple.html)
- O Aberto-Fechado Princípio-e o Que se Esconde por Trás (https://medium.com/@wrong.about/the-open-closed-principle-c3dc45419784)
