Open–closed principle

In object-oriented programming, the **open/closed principle** states "*software entities (classes, modules, functions, etc.) should be open for extension, but closed for modification*";[1] that is, such an entity can allow its behaviour to be extended without modifying its source code.

The name *open/closed principle* has been used in two ways. Both ways use generalizations (for instance, inheritance or delegate functions) to resolve the apparent dilemma, but the goals, techniques, and results are different.

Open-closed principle is one of the five SOLID principles of object-oriented design.

Contents

Meyer's open/closed principle

Bertrand Meyer is generally credited for having originated the term *open/closed principle*,[2] which appeared in his 1988 book *Object Oriented Software Construction*.

> - A module will be said to be open if it is still available for extension. For example, it should be possible to add fields to the data structures it contains, or new elements to the set of functions it performs.
> - A module will be said to be closed if [it] is available for use by other modules. This assumes that the module has been given a well-defined, stable description (the interface in the sense of information hiding).[3]

At the time Meyer was writing, adding fields or functions to a library inevitably required changes to any programs depending on that library. Meyer's proposed solution to this dilemma relied on the notion of object-oriented inheritance (specifically implementation inheritance):

> A class is closed, since it may be compiled, stored in a library, baselined, and used by client classes. But it is also open, since any new class may use it as parent, adding new features. When a descendant class is defined, there is no need to change the original or to disturb its clients.[4]

Polymorphic open/closed principle

During the 1990s, the open/closed principle became popularly redefined to refer to the use of abstracted interfaces, where the implementations can be changed and multiple implementations could be created and polymorphically substituted for each other.

In contrast to Meyer's usage, this definition advocates inheritance from abstract base classes. Interface specifications can be reused through inheritance but implementation need not be. The existing interface is closed to modifications and new implementations must, at a minimum, implement that interface.

Robert C. Martin's 1996 article "The Open-Closed Principle"[5] was one of the seminal writings to take this approach. In 2001 Craig Larman related the open/closed principle to the pattern by Alistair Cockburn called *Protected Variations*, and to the David Parnas discussion of *information hiding*.[6]

See also

- SOLID – the "O" in "SOLID" stands for the open/closed principle

References

1. Meyer, Bertrand (1988). *Object-Oriented Software Construction*. Prentice Hall. 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) Archived (https://web.archive.org/web/20060822033314/http://www.objectmentor.com/resources/articles/ocp.pdf) August 22, 2006, at the Wayback Machine
3. Meyer, Bertrand (1988). *Object-oriented software construction*. New York: Prentice Hall. p. 23. ISBN 0136290493.
4. Meyer, Bertrand (1988). *Object-oriented software construction*. New York: Prentice Hall. p. 229. ISBN 0136290493.
5. 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) Archived (https://web.archive.org/web/20060822033314/http://www.objectmentor.com/resources/articles/ocp.pdf) August 22, 2006, at the Wayback Machine
6. Craig Larman, "Protected Variation: The Importance of Being Closed", IEEE Software May/June 2001, pp. 89-91 *[1] (http://codecourse.sourceforge.net/materials/The-Importance-of-Being-Closed.pdf)*

External links

- The Principles of OOD (http://butunclebob.com/ArticleS.UncleBob.PrinciplesOfOod)
- The Open/Closed Principle: Concerns about Change in Software Design (http://blog.symprise.net/2009/06/23/open-closed-principle-concerns-about-change-in-software-design/)
- The Open Closed Principle (https://8thlight.com/blog/uncle-bob/2014/05/12/TheOpenClosedPrinciple.html)
- The Open-Closed Principle -- and What Hides Behind It (https://medium.com/@wrong.about/the-open-closed-principle-c3dc45419784)
