The Open-Closed Principle

Software entities (classes, modules, functions, etc.) should be open for extension, but closed for modification.

1. They are "Open For Extension". This means that the behavior of the module can be extended. That we can make the module behave in new and different ways as the requirements of the application change, or to meet the needs of new applications.

2. They are "Closed for Modification". The source code of such a module is inviolate. No one is allowed to make source code changes to it.

If we wish for a `Client` object to use a different server object, then the `Client` class must be changed to name the new server class.

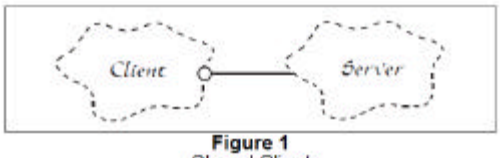


Figure 1
Closed Client

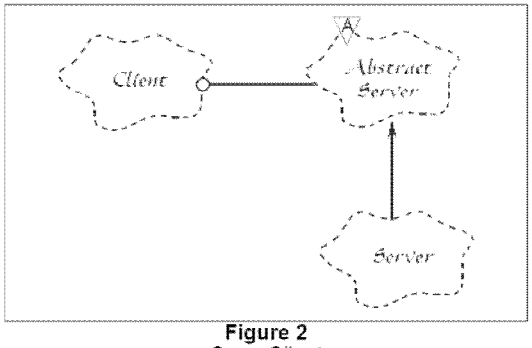


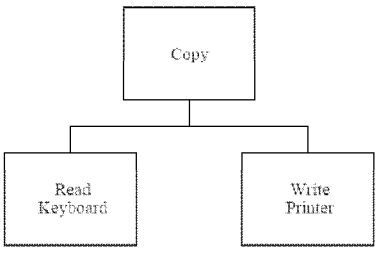
Figure 2
Open Client

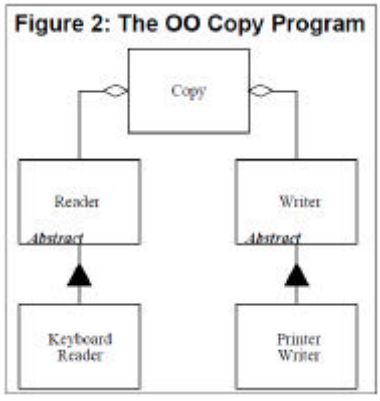
Member Variables Private.
No Global Variables.

The Dependency Inversion Principle

A. High level modules should not depend upon low-level modules. Both should depend upon abstractions.

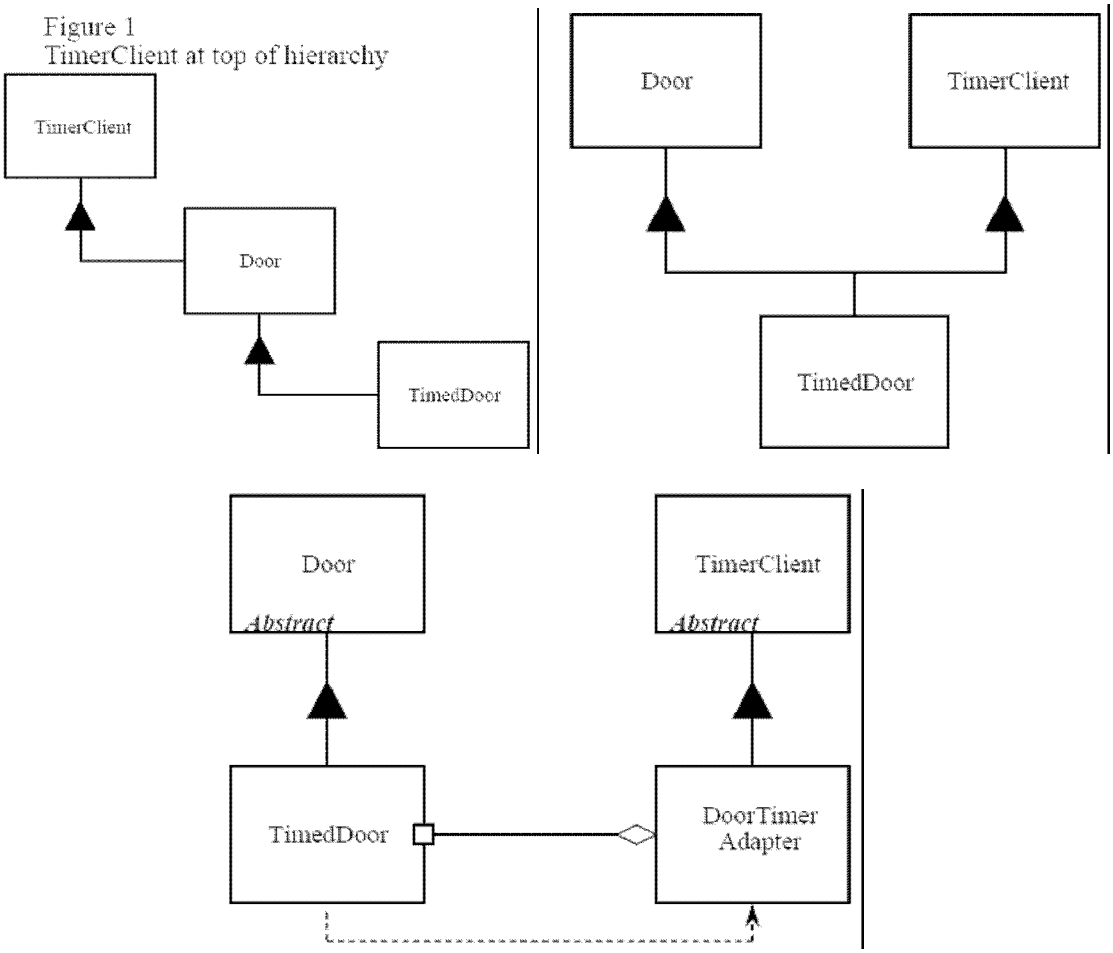
B. Abstractions should not depend upon details. Details should depend upon abstractions.





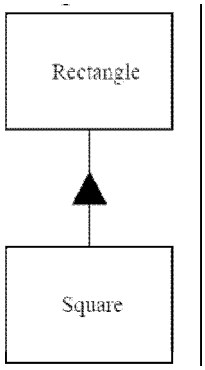
The Interface Segregation Principle

Clients should not be forced to depend upon interfaces that they do not use.



Even if the change to Timer were to be made, none of the users of Door would be affected. Moreover, TimedDoor does not have to have the exact same interface as TimerClient.

Liskov Substitution Principle



If S is a subtype of T, then objects of type T in a program may be replaced with objects of type S without altering any of the desirable properties of that program.

LSP is a particular definition of a subtyping relation, called (strong) behavioral subtyping.
The dimensions of a Square cannot (or rather should not) be modified independently.

> More information on Wikipedia.