Object-Oriented Design Principles Cheat Sheet

by scottashipp via cheatography.com/18826/cs/1958/

Basic Principles

Encapsulate what varies.

Code to an interface rather than to an implementation.

Each class in your application should have only one reason to change.

Classes are about behavior and functionality.

Basic OO Terms

Abstraction	The process of separating ideas from specific instances of those ideas at work.
Polymorphism	The provision of a single interface to entities of different types. Subtyping.
Inheritance	When an object or class is based on another object or class, using the same implementation; it is a mechanism for code reuse. The relationships of objects or classes through inheritance give rise to a hierarchy.
Encapsulation	Enclosing objects in a common interface in a way that makes them interchangeable, and guards their states from invalid changes

Favor the following over inheritance

Delegation	When you hand over the responsibility for a particular task to another class or method.
Composition	Use behavior from a family of other classes, and change that behavior at runtime.
Aggregation	When one class is used as part of another class, but still exists outside of that other class.

Don't Repeat Yourself (DRY)

Avoid duplicate code by abstracting out things that are common and placing those things in a single location.
DRY is about having each piece of information and behavior in your system in a single, sensible place.

Single Responsibility Principle

Every object in your system should have a single responsibility, and all the object's services should be focused on carrying out that single responsibility.

Open-Closed Principle

Classes should be open for extension, and closed for modification.

Liskov Substitution Principle (LSP)

Subtypes must be substitutable for their base types.

Interface Segregation Principle (ISP)

Clients should not be forced to depend upon interfaces that they don't use.

Dependency Inversion Principle (DIP)

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.

References

Wood, David, Bert Bates, Kathy Sierra, Brett D. McLaughlin, Gary Pollice, and David West. Head First Object-Oriented Analysis and Design: A Brain Friendly Guide to Object Oriented Programming. Cambridge: O'Reilly Media, Incorporated, 2006. Print.
"Abstraction (computer Science)."- Wikipedia. Wikimedia Foundation, 18 Apr. 2014. Web. 25 Apr. 2014. <http://en.wikipedia.org/wiki/Abstraction_(computer_science)>
"Polymorphism (computer Science)."- Wikipedia. Wikimedia Foundation, 25 Apr. 2014. Web. 25 Apr. 2014. <http://en.wikipedia.org/wiki/Polymorphism_(computer_science)>
"Inheritance (object-oriented Programming)." Wikipedia. Wikimedia Foundation, 21 Apr. 2014. Web. 25 Apr. 2014. <http://en.wikipedia.org/wiki/Inheritance_(object-oriented_programming)>.
"Encapsulation (object-oriented Programming)." Wikipedia. Wikimedia Foundation, 21 Apr. 2014. Web. 25 Apr. 2014. <http://en.wikipedia.org/wiki/Encapsulation_(object-oriented_programming)>