# An Introduction to SOLID programming principles.

SOLID is an acronym, first articulated by Bob Martin, used to describe a style of programming that aims at producing software that can be easily read, maintained, and expanded upon.

It's main principles are these:

S - single responsibility principle

O - open-closed principle

L - Liskov substitution principle

I - interface segregation principle

D - dependency inversion principle

In this talk, I provide an introduction to these principles, along with foundational computer science concepts like modularity, cohesion, and loose coupling from which they were developed. I begin by explaining the problem that SOLID programming aims to solve, viz. that without careful design, the velocity at which software can be added to will slow, making the addition of new features more and more difficult as software grows. I then detail each of the principles, providing practical examples throughout, and end with a demo showing how some of these principles might be applied in our code.

Links:

Why focus on software quality: [Code quality: a concern for businesses, bottom lines, and empathetic programmers - Stack Overflow Blog](https://stackoverflow.blog/2021/10/18/code-quality-a-concern-for-businesses-bottom-lines-and-empathetic-programmers/)

Bob Martin’s *Clean Code:* [Clean Code: A Handbook of Agile Software Cr... - Kindle (amazon.com)](https://read.amazon.com/kp/embed?asin=B001GSTOAM&tag=bing08-20&linkCode=kpp&reshareId=EVFMFNENM1ETW7VENFVE&reshareChannel=system)

[Summary of 'Clean code' by Robert C. Martin (github.com)](https://gist.github.com/wojteklu/73c6914cc446146b8b533c0988cf8d29)

D. L. Parnas – “On the Criteria To Be Used in Decomposing Systems into Modules” - [criteria\_for\_modularization.pdf (tue.nl)](https://www.win.tue.nl/~wstomv/edu/2ip30/references/criteria_for_modularization.pdf)

Modular design and microservice development examples: [High Scalability –](http://highscalability.com/blog/category/example)

Hexagonal Architecture: [Migrating Monolithic Apps to Multi-Platform Product Lines with .NET 5 (codemag.com)](https://codemag.com/Article/2103081/Migrating-Monolithic-Apps-to-Multi-Platform-Product-Lines-with-.NET-5)

Yegor Bugayenko’s Blog on object-oriented programming: [Seven Virtues of a Good Object (yegor256.com)](https://www.yegor256.com/2014/11/20/seven-virtues-of-good-object.html)

[Type Theory (Stanford Encyclopedia of Philosophy)](https://plato.stanford.edu/entries/type-theory/)

Some sample projects:

Employee App Refactor - [JacobArchambault/EmployeeAppRefactor: A git tutorial showing how to change an application printing out employee info to implement best practices like strong encapsulation, composition over inheritance, immutability, and dependency injection. (github.com)](https://github.com/JacobArchambault/EmployeeAppRefactor)

Folder Search (Decorator patterns, Interface Segregation): [JacobArchambault/FolderSearch: A file I/O application recursively copying files to an output directly based on a user prompt (github.com)](https://github.com/JacobArchambault/FolderSearch)

Bowling: [JacobArchambault/Bowling (github.com)](https://github.com/JacobArchambault/Bowling)