

The SOLID Principles



Mel Grubb

Developer

@melgrubb | www.melgrubb.com

The SOLID Principles

S

Single Responsibility Principle

O

Open-Closed Principle

L

Liskov Substitution Principle

I

Interface Segregation Principle

D

Dependency Inversion Principle





For More Information

SOLID Principles for C# Developers

Steve Smith



Overview

Introduction to the SOLID principles

Examine influence on design decisions

Compare that influence with pillars



| Single Responsibility Principle

Single Responsibility Principle

“A class should have one, and only one, reason to change.”

or

“There should never be more than one reason for a class to change.”

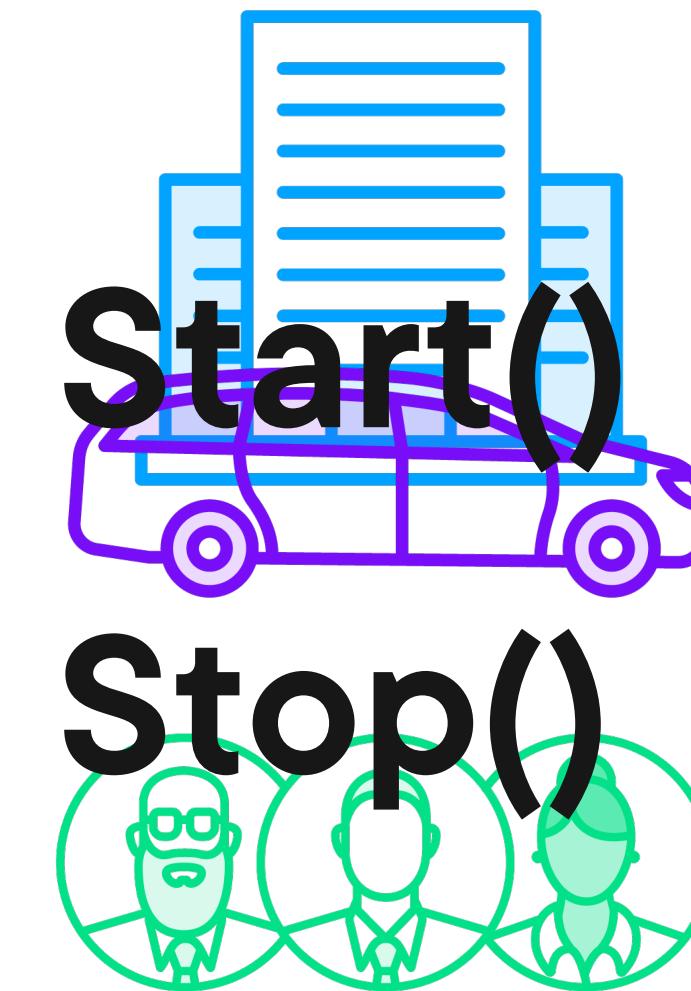
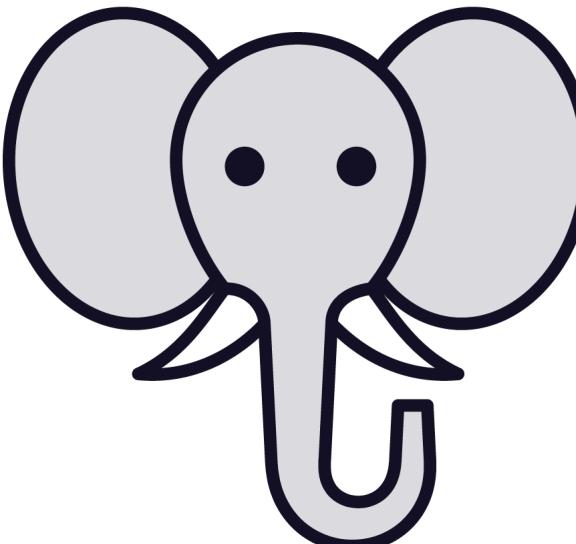
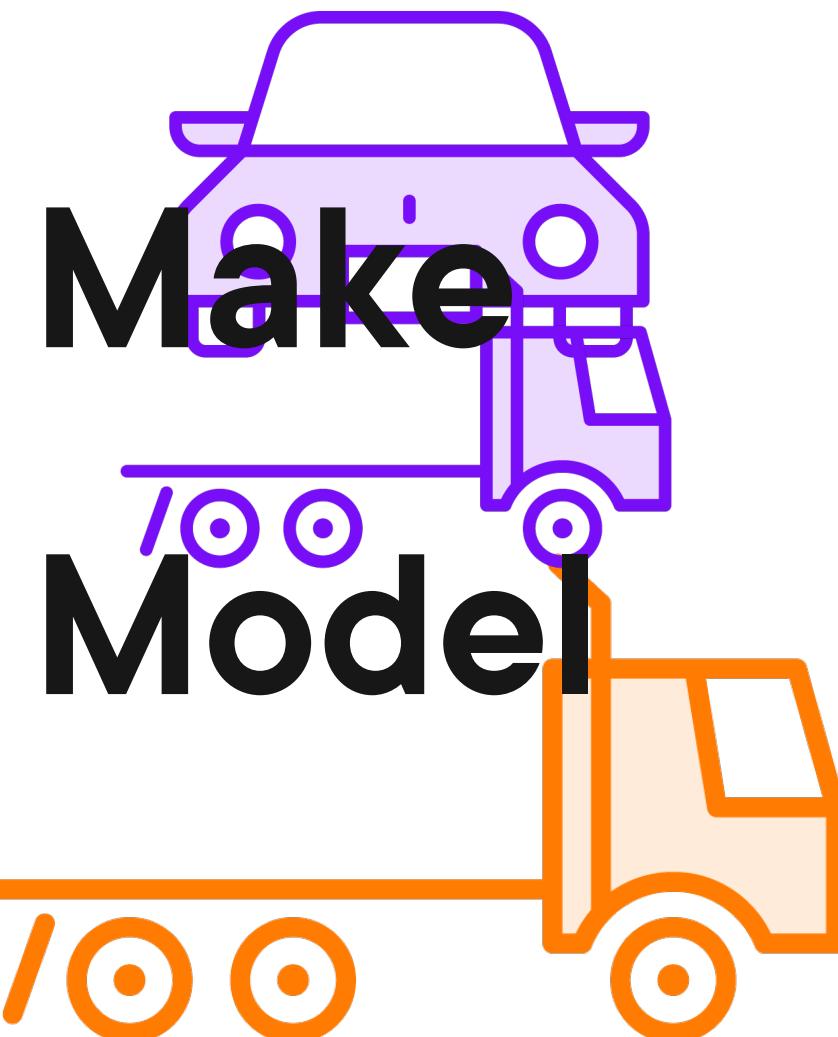


Single Responsibility Principle

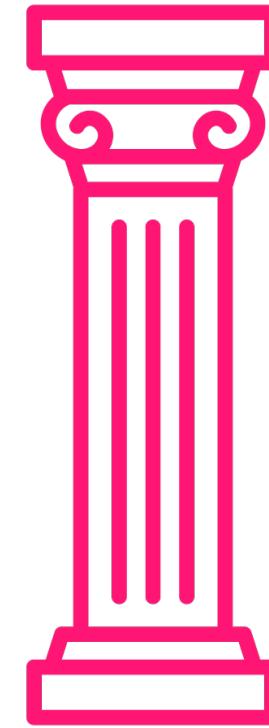
“A module should have a clearly-defined job”



Scopes of Responsibility



The Single Responsibility Principle



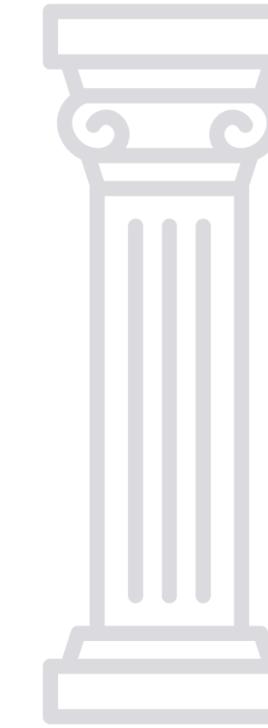
Abstraction



Encapsulation



Inheritance



Polymorphism





Design Tips

- Identify “responsibilities” and “jobs”
- Watch out for “reasons to change”
- Favor multiple small classes over fewer, more complex ones



| Open/Closed Principle



Open/Closed Principle

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



Open/Closed Principle

“You should be able to extend a class’s behavior without modifying it.”

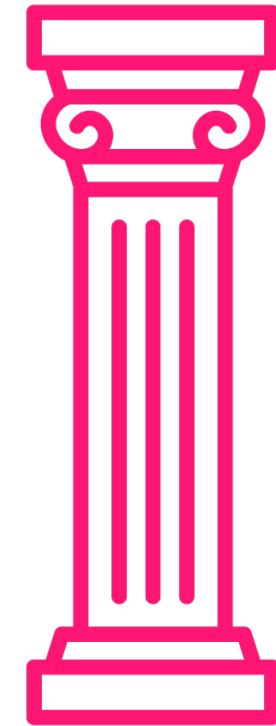


Code Changes

Class	Change Type	Start Line	End Line
com.example.model	Addition	100	200
com.example.model	Deletion	150	180
com.example.model	Modification	120	140
com.example.model	Modification	130	150
com.example.model	Modification	140	160
com.example.model	Modification	150	170
com.example.model	Modification	160	180
com.example.model	Modification	170	190
com.example.model	Modification	180	200
com.example.model	Modification	190	210
com.example.model	Modification	200	220
com.example.model	Modification	210	230
com.example.model	Modification	220	240
com.example.model	Modification	230	250
com.example.model	Modification	240	260
com.example.model	Modification	250	270
com.example.model	Modification	260	280
com.example.model	Modification	270	290
com.example.model	Modification	280	300
com.example.model	Modification	290	310
com.example.model	Modification	300	320
com.example.model	Modification	310	330
com.example.model	Modification	320	340
com.example.model	Modification	330	350
com.example.model	Modification	340	360
com.example.model	Modification	350	370
com.example.model	Modification	360	380
com.example.model	Modification	370	390
com.example.model	Modification	380	400
com.example.model	Modification	390	410
com.example.model	Modification	400	420
com.example.model	Modification	410	430
com.example.model	Modification	420	440
com.example.model	Modification	430	450
com.example.model	Modification	440	460
com.example.model	Modification	450	470
com.example.model	Modification	460	480
com.example.model	Modification	470	490
com.example.model	Modification	480	500
com.example.model	Modification	490	510
com.example.model	Modification	500	520
com.example.model	Modification	510	530
com.example.model	Modification	520	540
com.example.model	Modification	530	550
com.example.model	Modification	540	560
com.example.model	Modification	550	570
com.example.model	Modification	560	580
com.example.model	Modification	570	590
com.example.model	Modification	580	600
com.example.model	Modification	590	610
com.example.model	Modification	600	620
com.example.model	Modification	610	630
com.example.model	Modification	620	640
com.example.model	Modification	630	650
com.example.model	Modification	640	660
com.example.model	Modification	650	670
com.example.model	Modification	660	680
com.example.model	Modification	670	690
com.example.model	Modification	680	700
com.example.model	Modification	690	710
com.example.model	Modification	700	720
com.example.model	Modification	710	730
com.example.model	Modification	720	740
com.example.model	Modification	730	750
com.example.model	Modification	740	760
com.example.model	Modification	750	770
com.example.model	Modification	760	780
com.example.model	Modification	770	790
com.example.model	Modification	780	800
com.example.model	Modification	790	810
com.example.model	Modification	800	820
com.example.model	Modification	810	830
com.example.model	Modification	820	840
com.example.model	Modification	830	850
com.example.model	Modification	840	860
com.example.model	Modification	850	870
com.example.model	Modification	860	880
com.example.model	Modification	870	890
com.example.model	Modification	880	900
com.example.model	Modification	890	910
com.example.model	Modification	900	920
com.example.model	Modification	910	930
com.example.model	Modification	920	940
com.example.model	Modification	930	950
com.example.model	Modification	940	960
com.example.model	Modification	950	970
com.example.model	Modification	960	980
com.example.model	Modification	970	990
com.example.model	Modification	980	1000



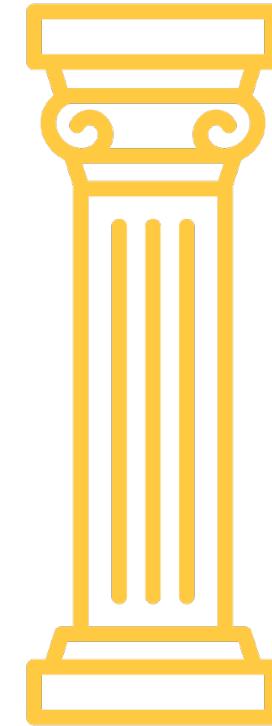
The Open/Closed Principle



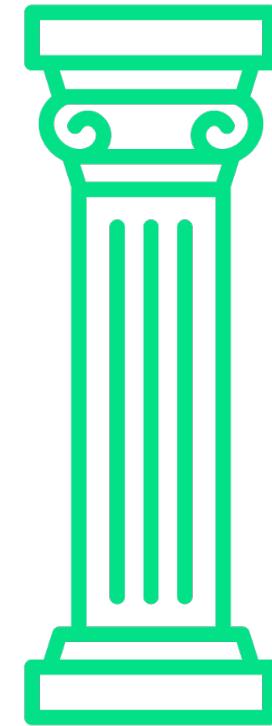
Abstraction



Encapsulation



Inheritance



Polymorphism





Design Tips

- Separate “bedrock” code into base classes
- Implement specializations on top of that bedrock
- Watch for classes with multiple “modes”
- Break these into smaller classes



Liskov Substitution Principle



Liskov Substitution Principle

“Let $\varphi(x)$ be a property provable about objects x of type T . Then $\varphi(y)$ should also be true for objects y of type S where S is a subtype of T .”

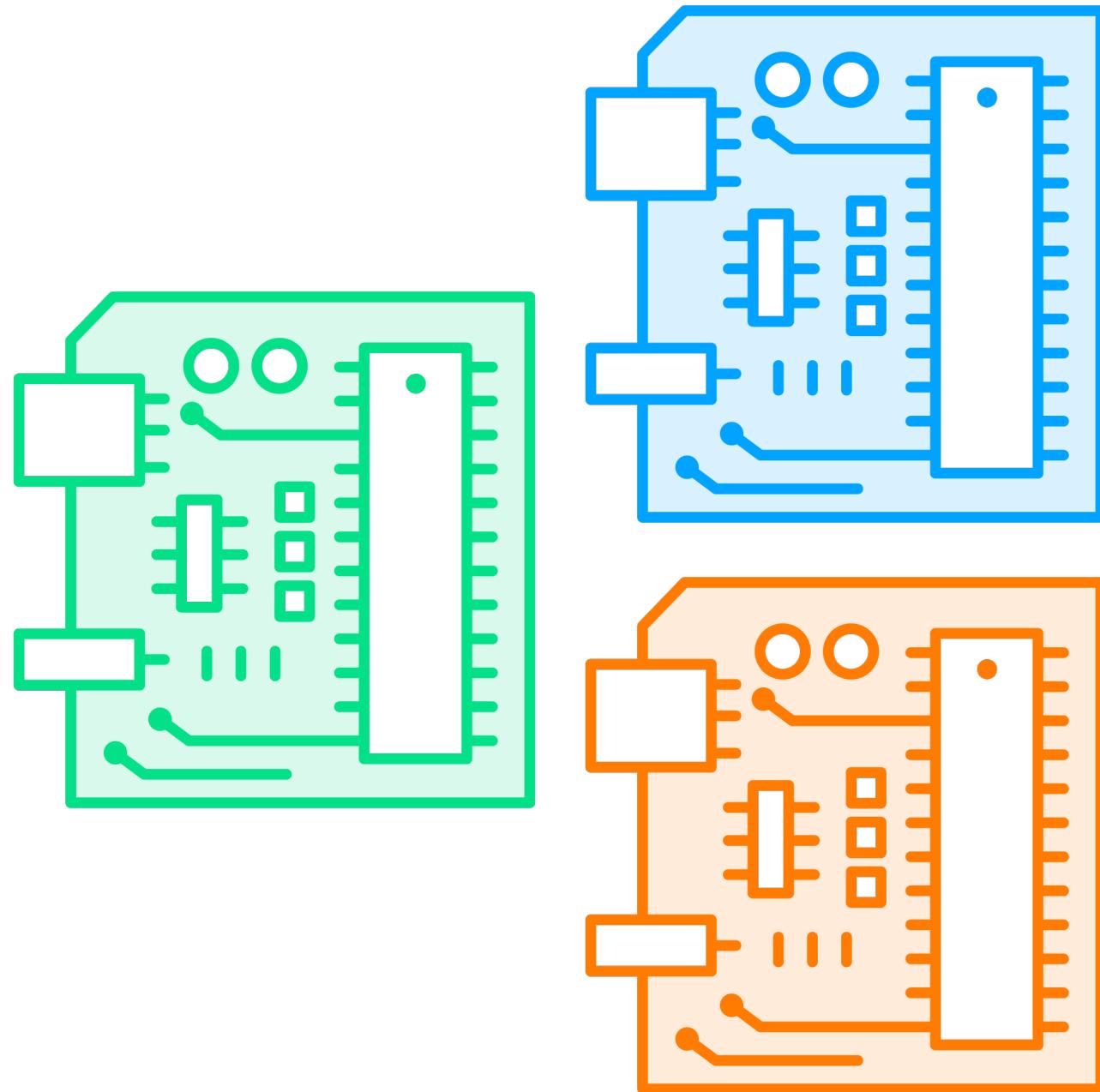
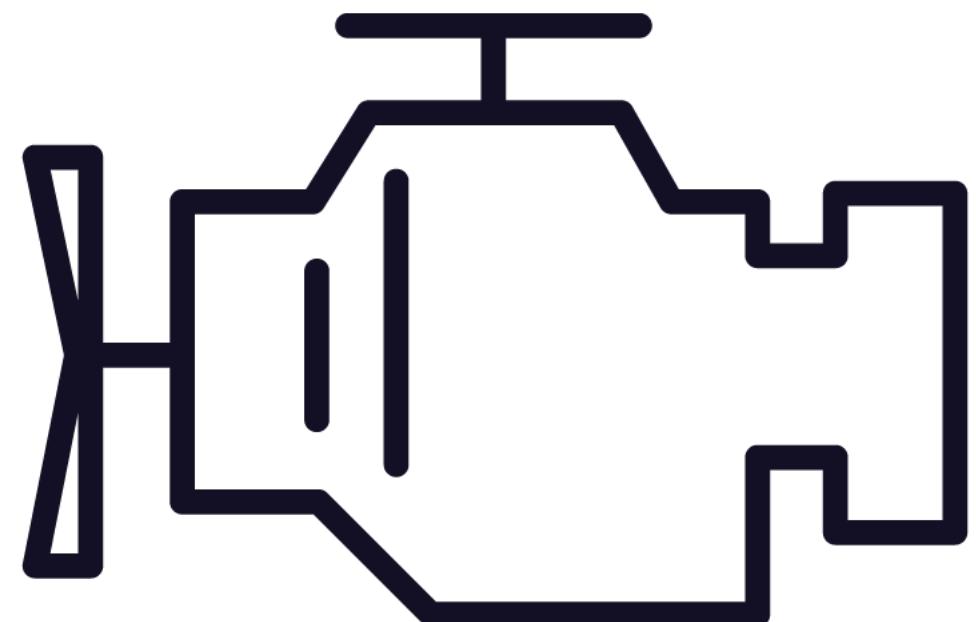


Liskov Substitution Principle

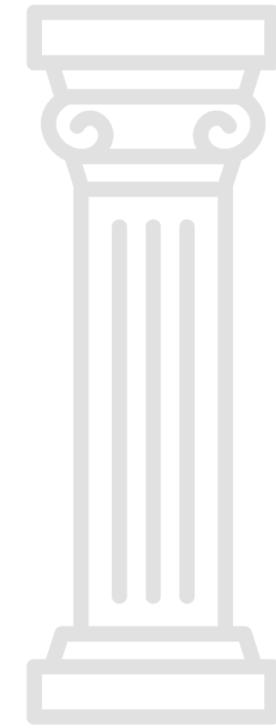
“Derived types must be substitutable for their base types.”



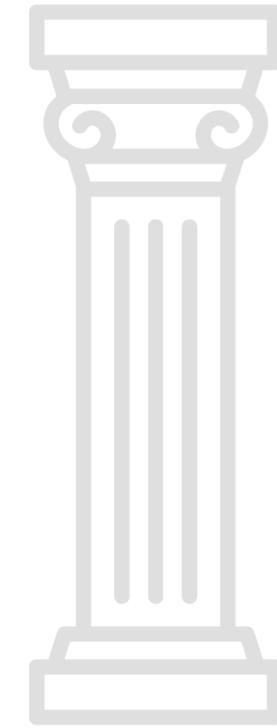
Liskov Substitution Principle



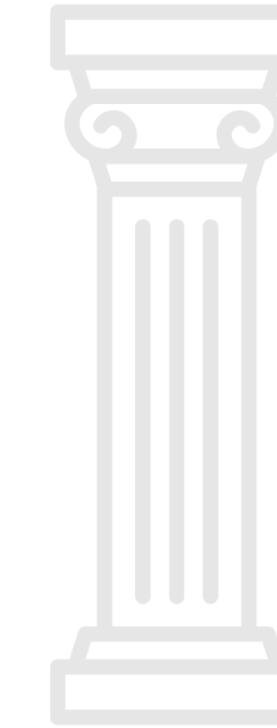
The Liskov Substitution Principle



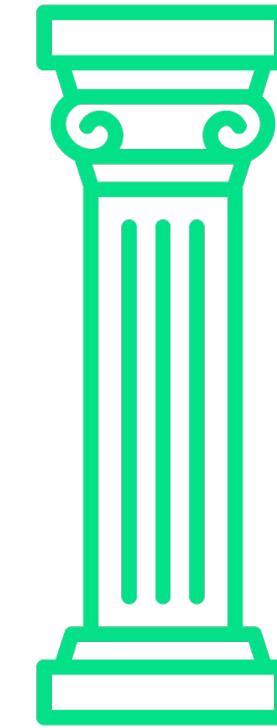
Abstraction



Encapsulation



Inheritance



Polymorphism





Design Tips

- **Test subtypes individually**
- **Test that they each perform all tasks**
- **Watch out for NotImplementedExceptions**



Interface Segregation Principle



Interface Segregation Principle

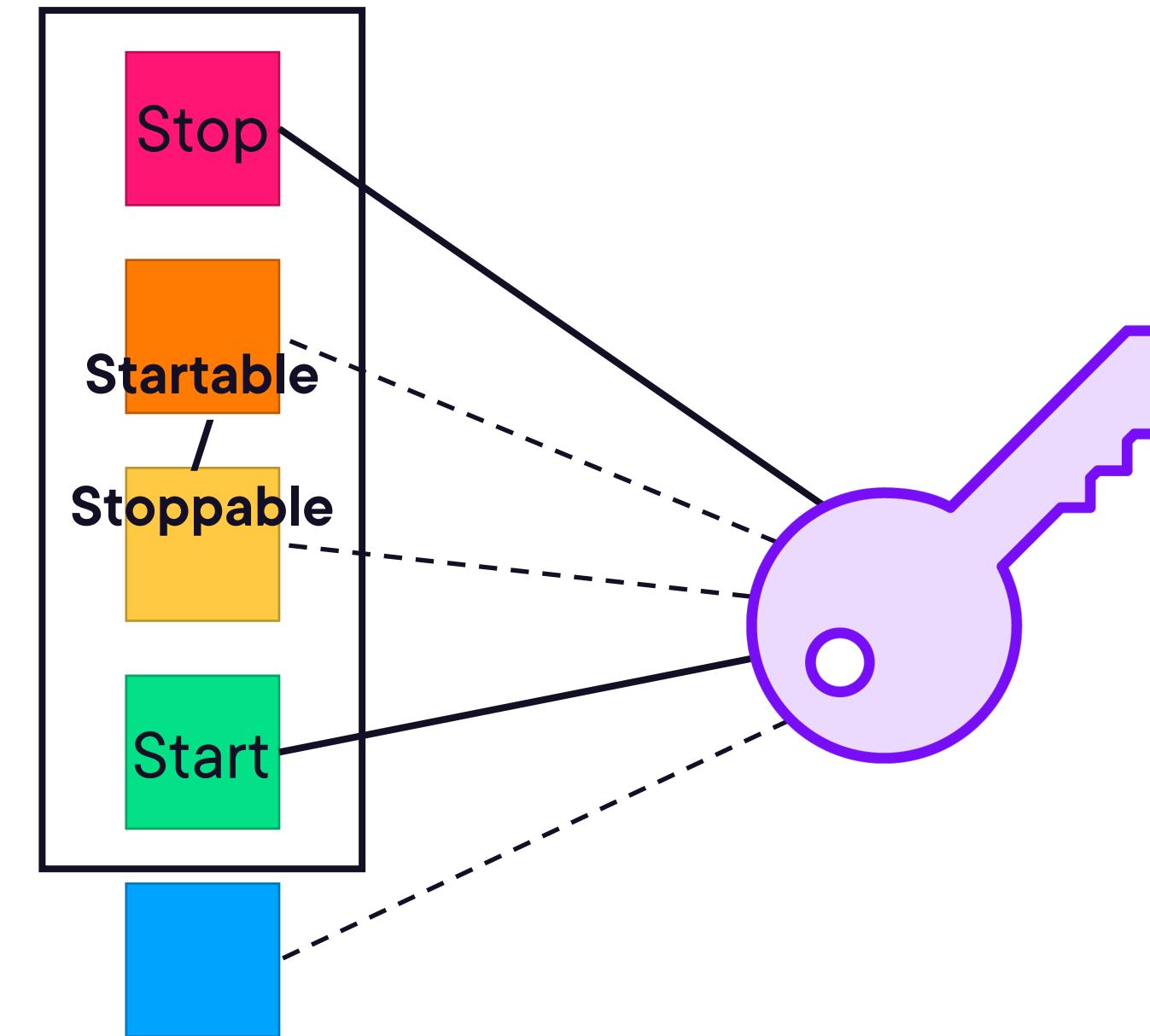
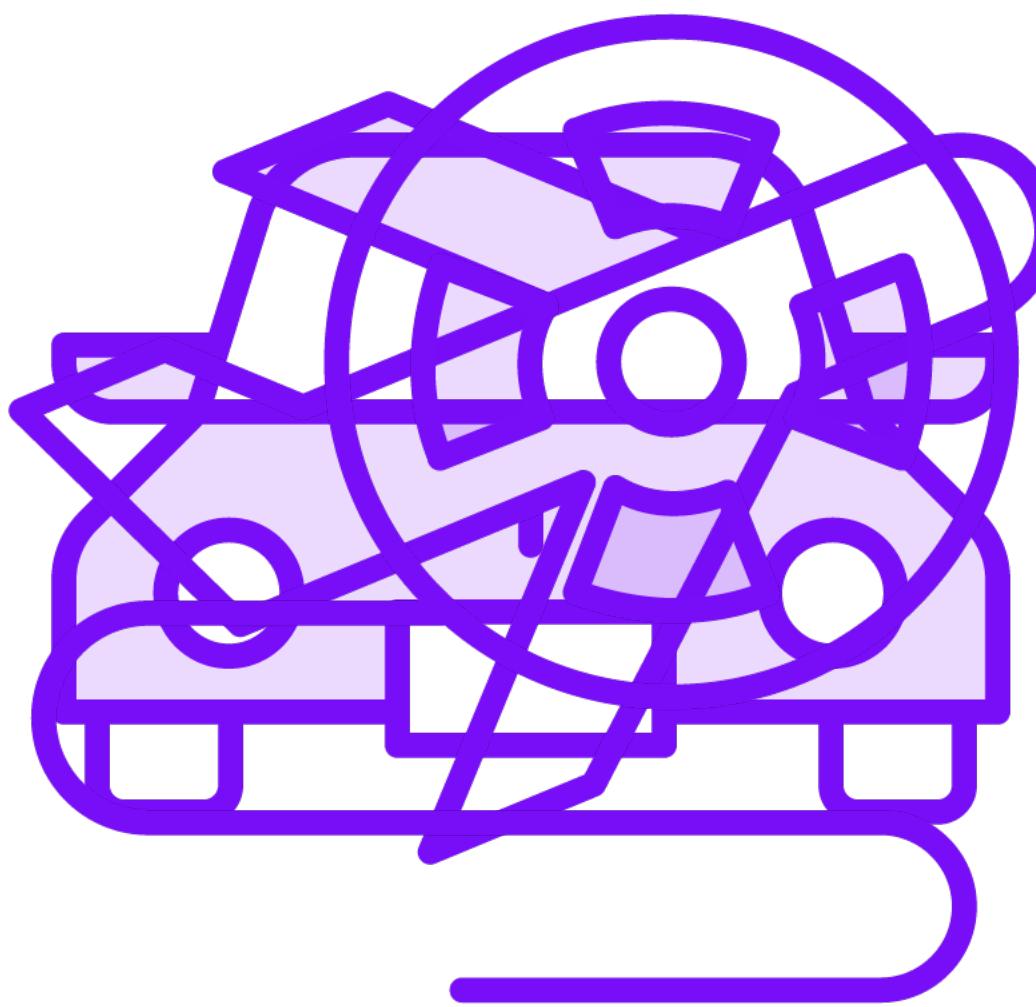
“Clients should not be forced to depend upon interfaces that they do not use.”

or

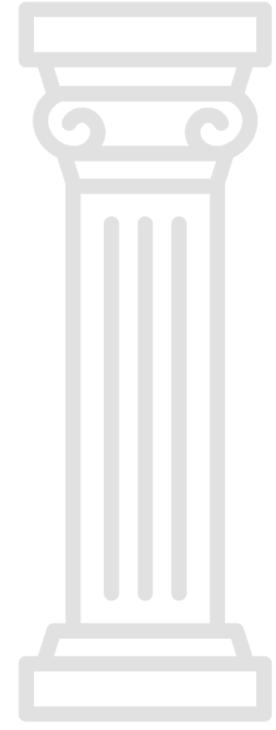
“Make fine-grained interfaces that are client-specific.”



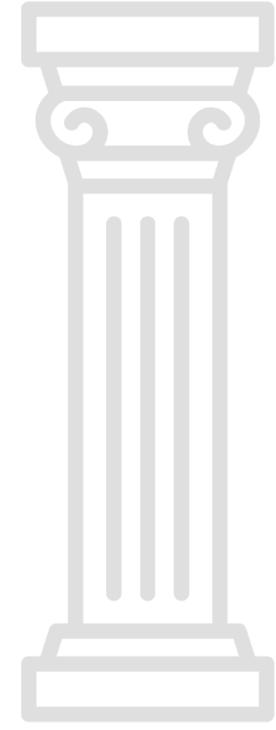
Interface Segregation Principle



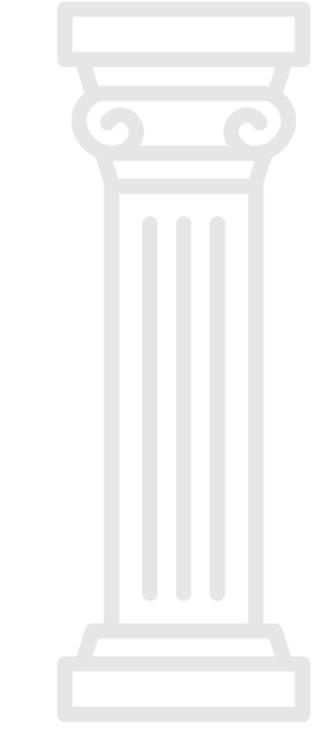
The Interface Segregation Principle



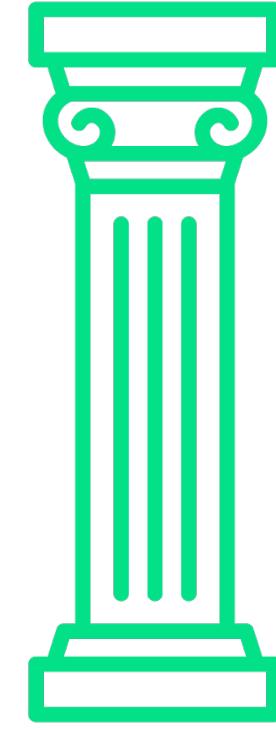
Abstraction



Encapsulation



Inheritance



Polymorphism





Design Tips

- Create interfaces for a class's abilities
- Group related functionality together
- Keep interfaces cohesive



Dependency Inversion Principle



Dependency Inversion Principle

“High-level modules should not depend on low-level modules.

Both should depend on abstractions.”

or

“Depend on abstractions, not concretions.”



Dependency Inversion Principle

!= Dependency Injection

!= Inversion of Control Container

Dependency Injection and IoC containers are just implementation details



Dependency Inversion Principle

Don't depend on specific classes
Don't instantiate your own dependencies
Prefer dependencies as interfaces
...provided from an outside source





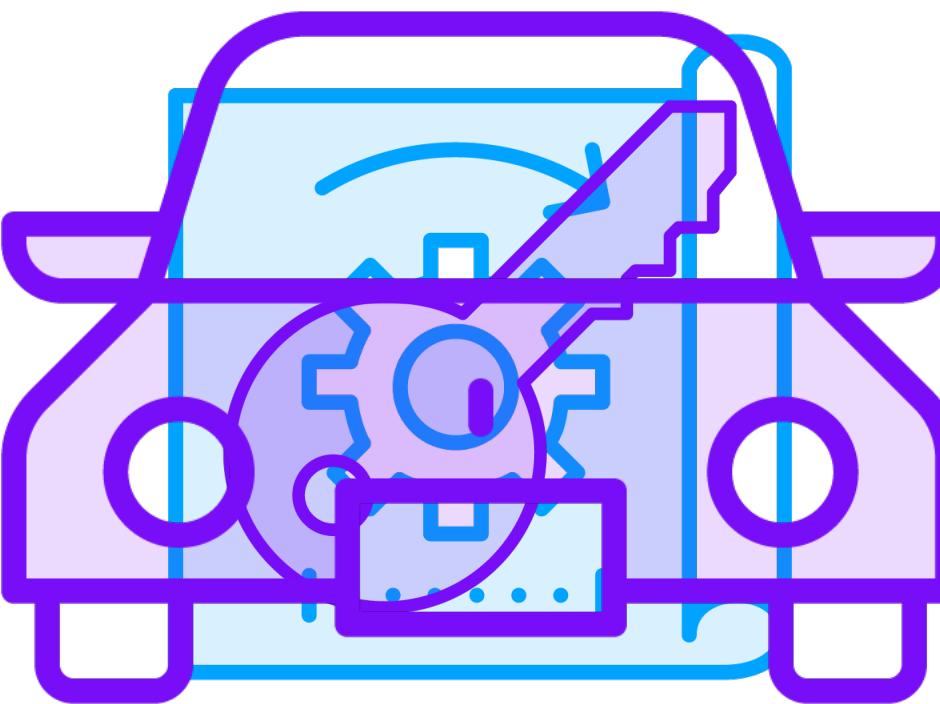
For more information

C# Dependency Injection

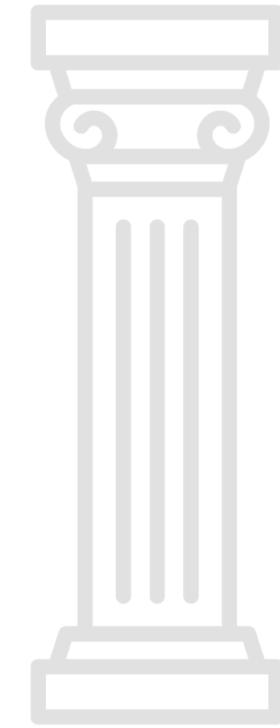
Henry Been



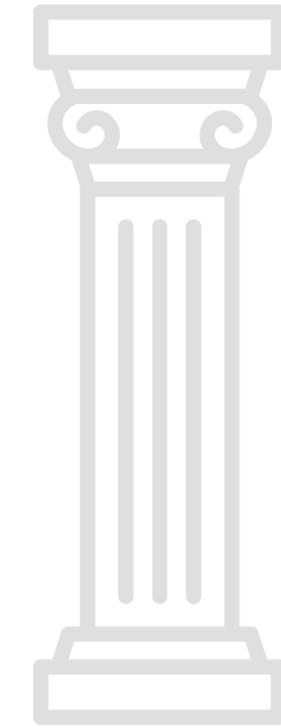
IoC Containers



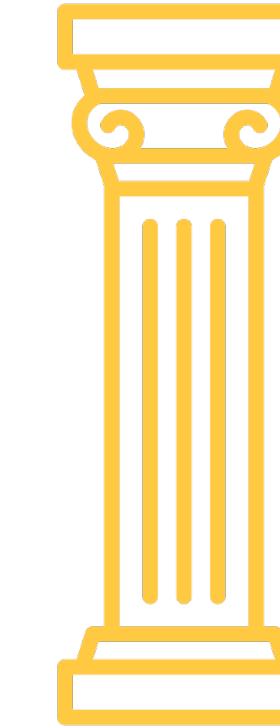
The Dependency Inversion Principle



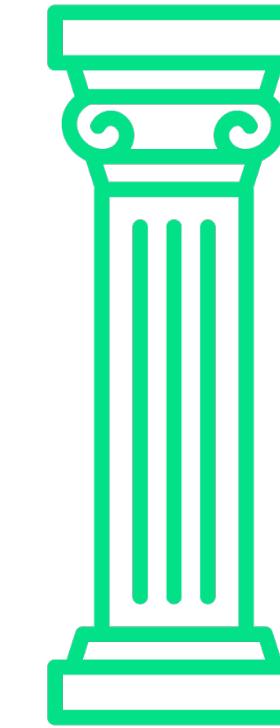
Abstraction



Encapsulation



Inheritance



Polymorphism





Design Tips

- Don't instantiate your own dependencies
- Require them as constructor parameters...
- ...or inject them as properties
- Consider using an IoC container





Summary

The principles justify the pillars

They span multiple pillars

They can overlap each other

Single Responsibility Principle

Open/Closed Principle

Liskov Substitution Principle

Interface Segregation Principle

Dependency Inversion Principle



Introduction to Design Patterns

