Solid

Motivation:

Maintainability

Testability

Flexibility and Extensibility (add new feature easily)

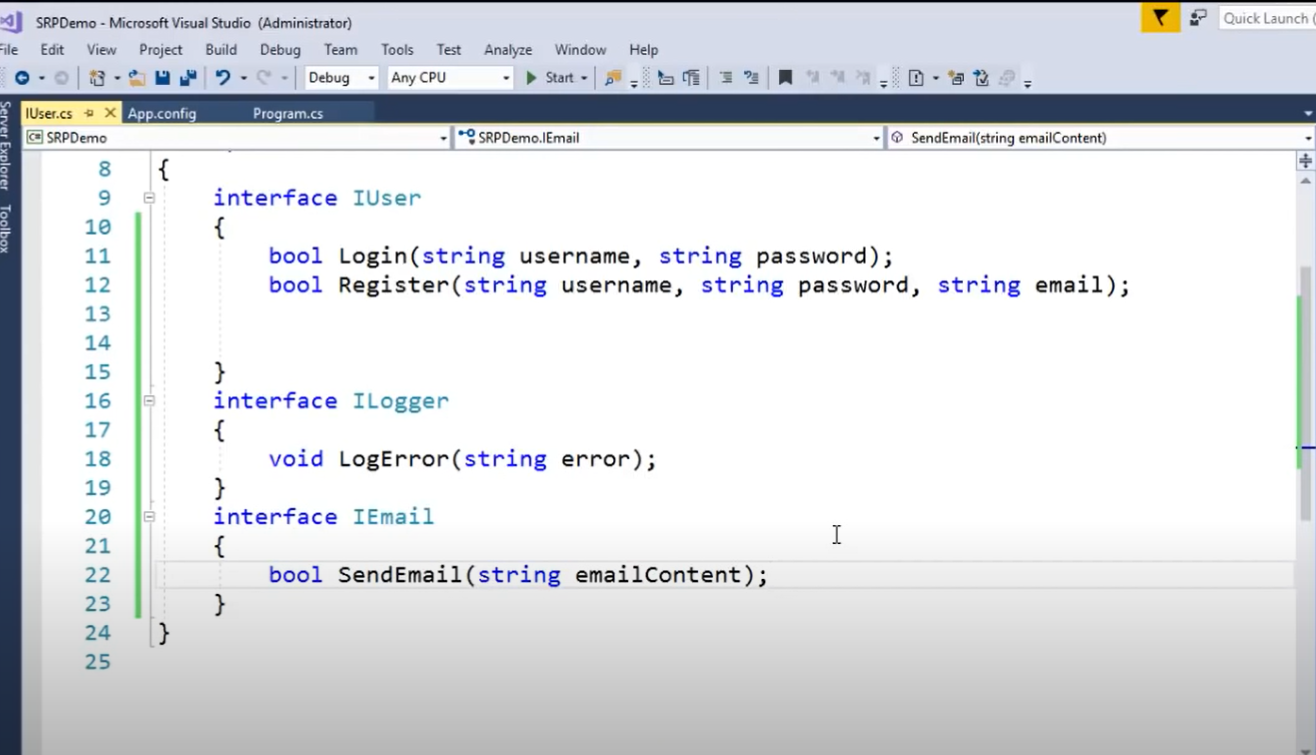
Parallel Development

Loose Coupling

S-Single Responsibility Principle

A class should have responsibility over a single part of the functionality

Responsibility should be entirely encapsulated by class

Example

Three interface instead of one interface

O-Open closed principle

Software should be open for extension but closed for modification

Any new classes should be implemented by adding new classes, attributes and methods instead of changing the current ones

Example

Create abstract class Employee

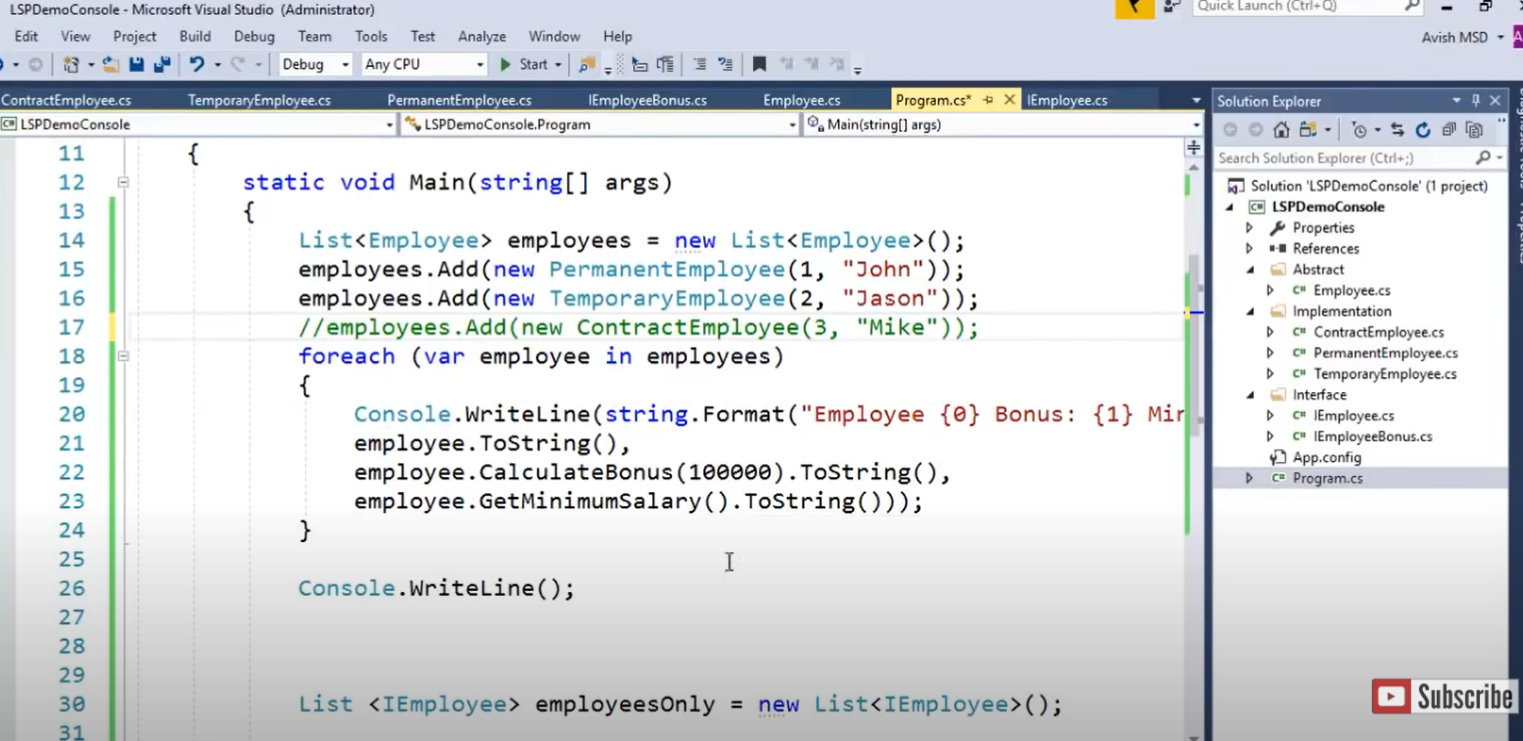
And extend by permanent, temporary and many other in the future

Liskov Substitution principle

Objects in a program should be replaceable with instances of their subtypes

Extension of Open close principle

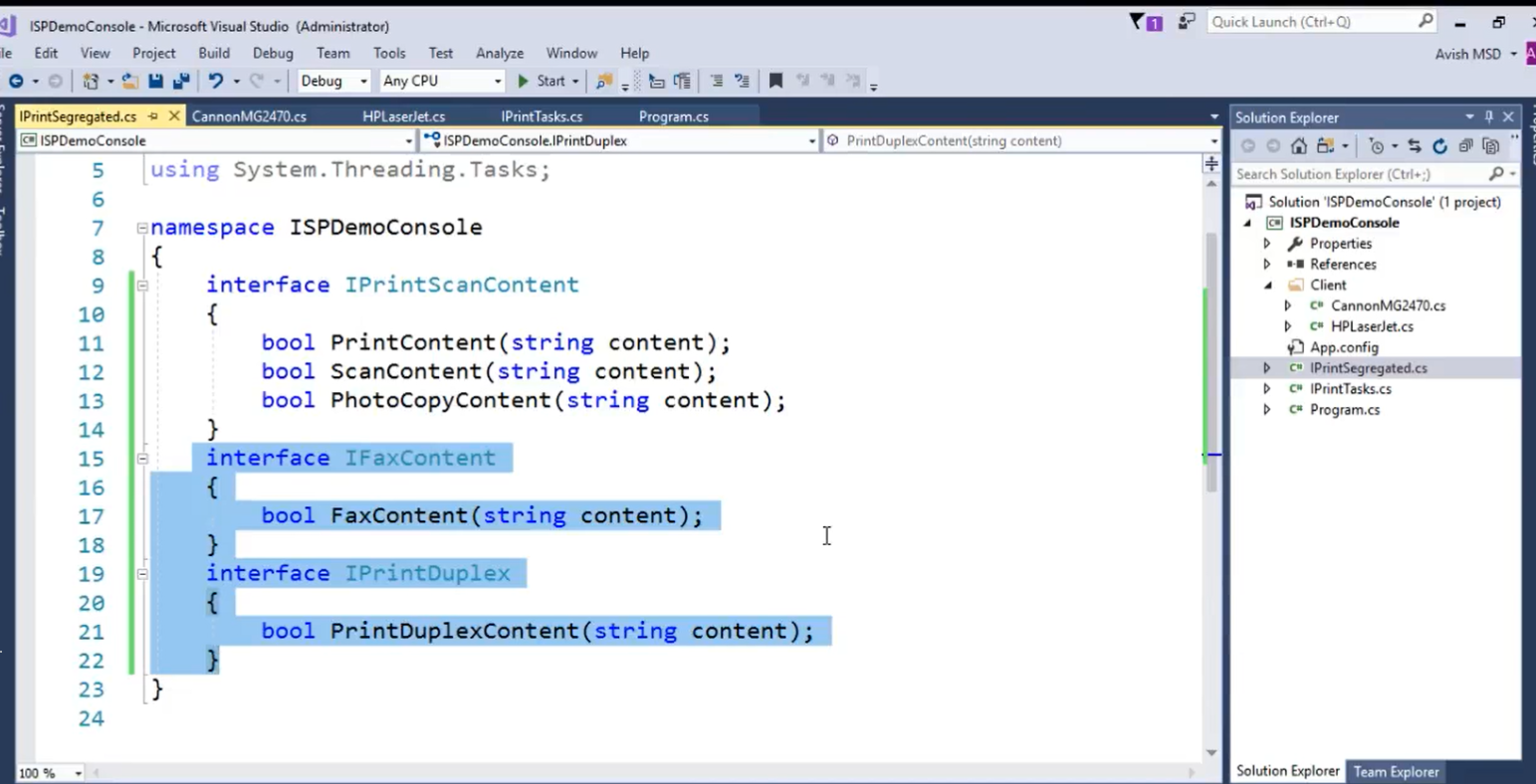
Example

Employee John = ne w PermanentEmployee()

Interface segregation principle

Many specific interfaces are better than one general-purpose interface

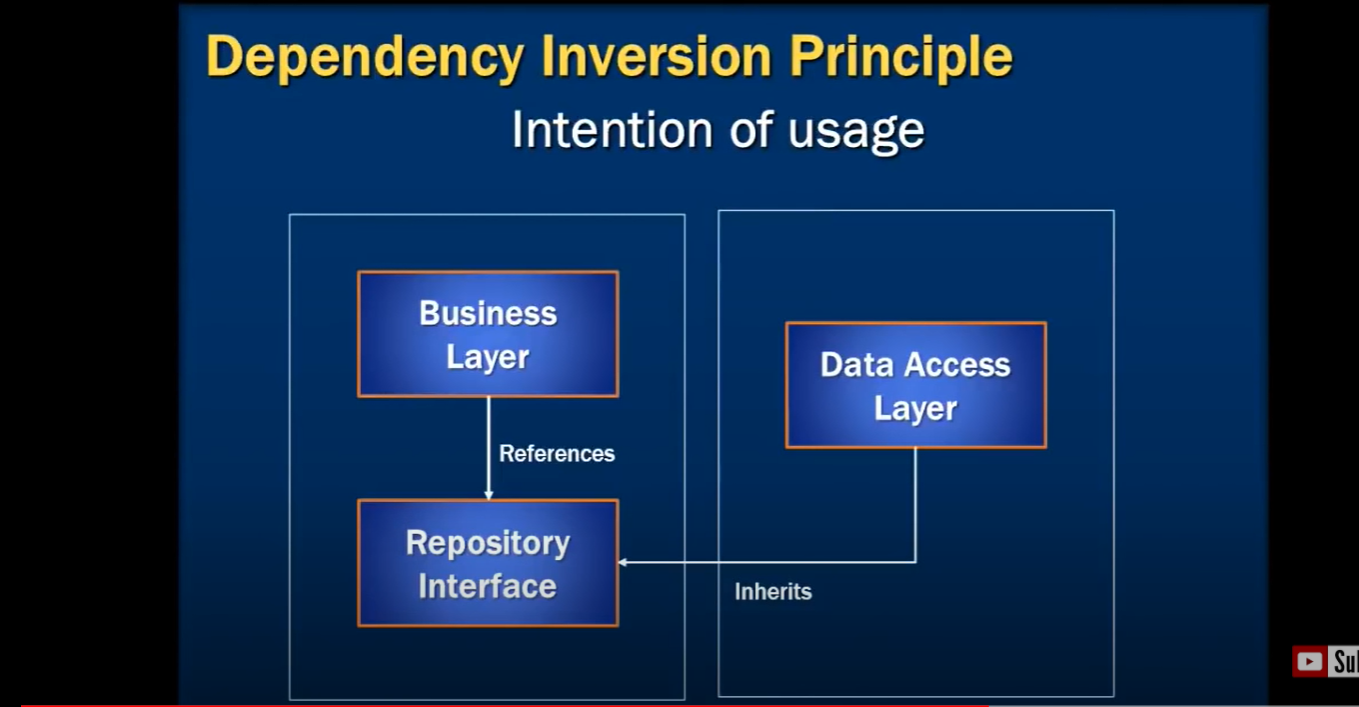
We should not force client to install the interface that they don’t use

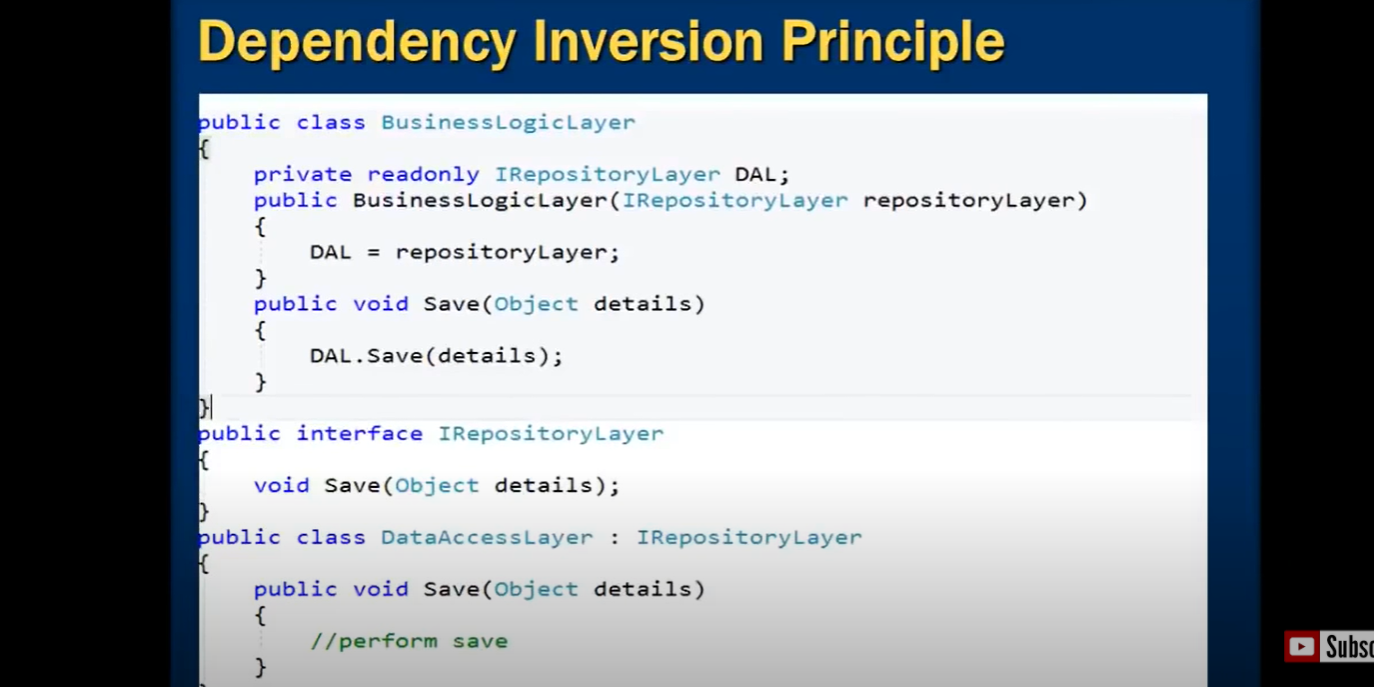
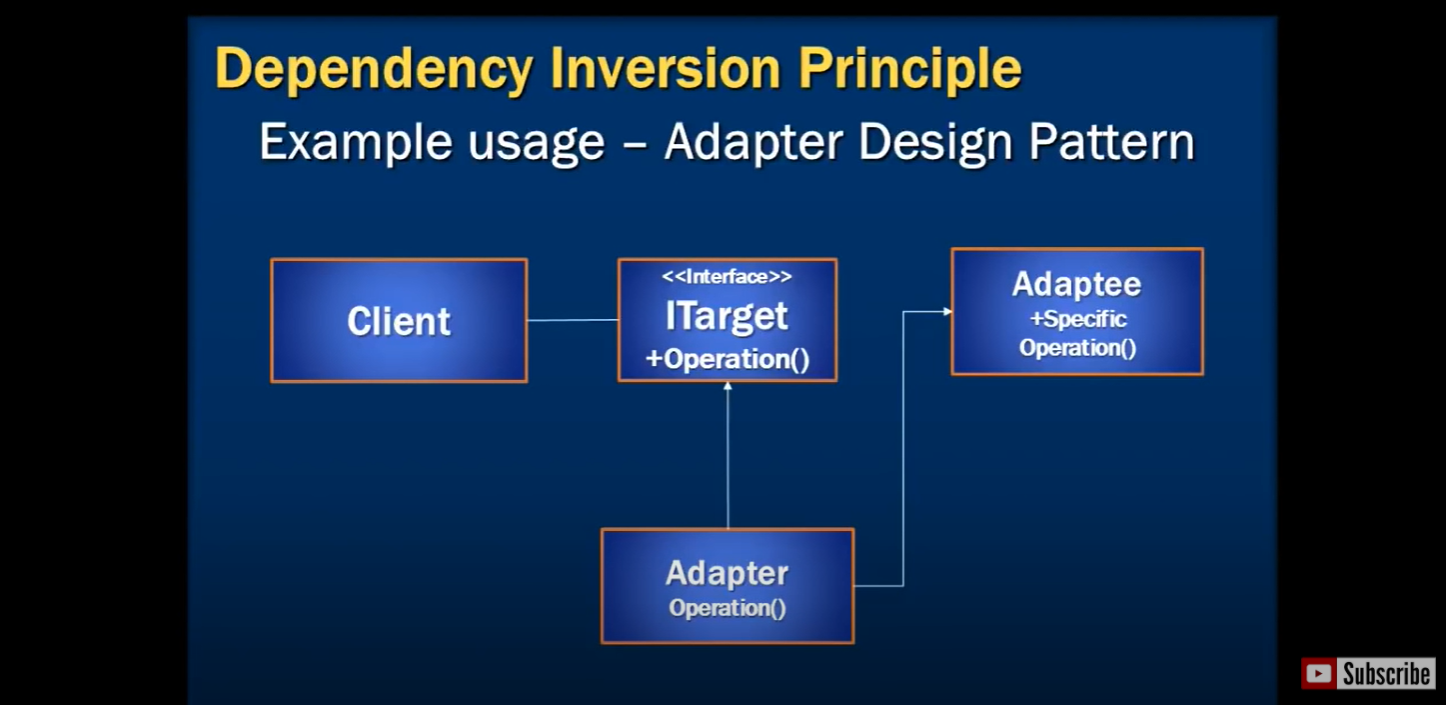
Example

Dependency Inversion Principle

Abstraction should not depend on details

High level modules should not depend on low level modules. Both should depend on abstractions

Example



Why SOILD

Avoid tight coupling (Tight coupling is when a group of classes are highly dependent on one another)