DESIGN PRINCIPLES

Clean Code:
Principles, Patterns, and Practices for Better Code

KEEP IT SIMPLE, STUPID (KISS)

“KEEP IT SIMPLE STUPID”

by Kelly Johnson

http://blogs.smarter.com/blogs/Lego%20Brick.jpg

© Theo Jungeblut 2014

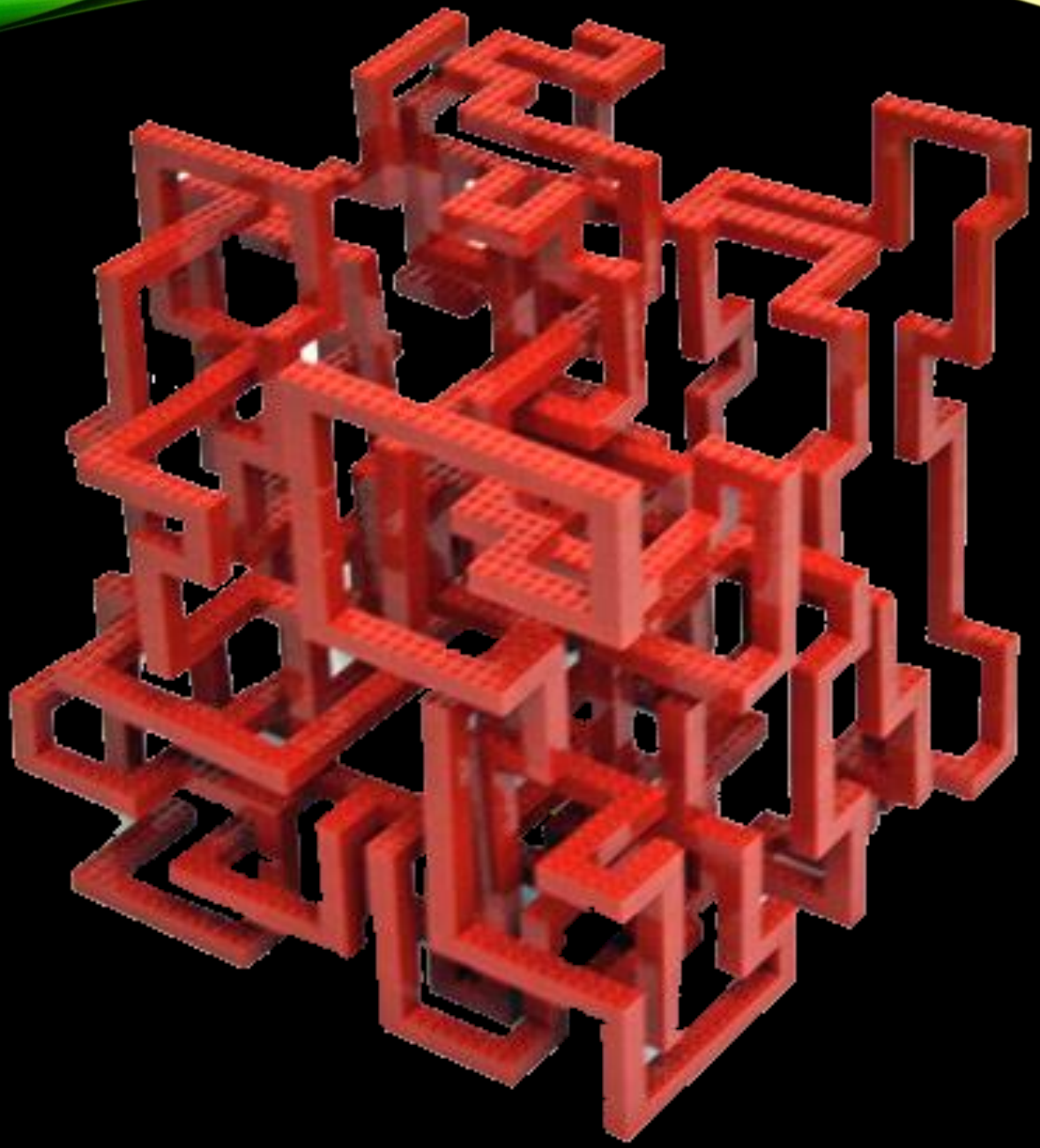
THE POWER OF SIMPLICITY

Graphic by Nathan Sawaya
courtesy of brickartist.com

Graphic by Nathan Sawaya courtesy of brickartist.com

http://www.geekalerts.com/lego-iphone/

© Theo Jungeblut 2014



© Theo Jungeblut 2014

Graphic by Nathan Sawaya courtesy of brickartist.com

DON’T REPEAT YOURSELF (DRY)

DON'T REPEAT YOURSELF (DRY)

by Andy Hunt and Dave Thomas in their book "The Pragmatic Programmer"

```
// Code Copy and Paste Method
public Class Person
 {
    public string FirstName { get;  set;}
    public string LastName { get; set;}

    public Person(Person person)
    {
        this.FirstName =
string.IsNullOrEmpty(person.FirstName)
                    ? string.Empty : (string)
person.FirstName.Clone();

        this.LastName =
string.IsNullOrEmpty(person.LastName)
                    ? string.Empty : (string)
person.LastName.Clone();
    }

    public object Clone()
    {
        return new Person(this);
    }
}
```

```
// DRY Method
public Class Person
 {
    public string FirstName { get;  set;}
    public string LastName { get; set;}

    public Person(Person person)
    {
        this.FirstName = person.FirstName.CloneSecured();
        this.LastName = person.LastName.CloneSecured();
    }

    public object Clone()
    {
        return new Person(this);
    }
}
```

```
public static class StringExtension
 {
    public static string CloneSecured(this string original)
    {
      return string.IsNullOrEmpty(original) ? string.Empty : (string)original.Clone();
    }
 }
```

© Theo Jungeblut 2014

SEPARATION OF CONCERNS (SOC)

SINGLE RESPONSIBILITY PRINCIPLE (SRP)

THE PRODUCT

http://www.technicopedia.com/8865.html

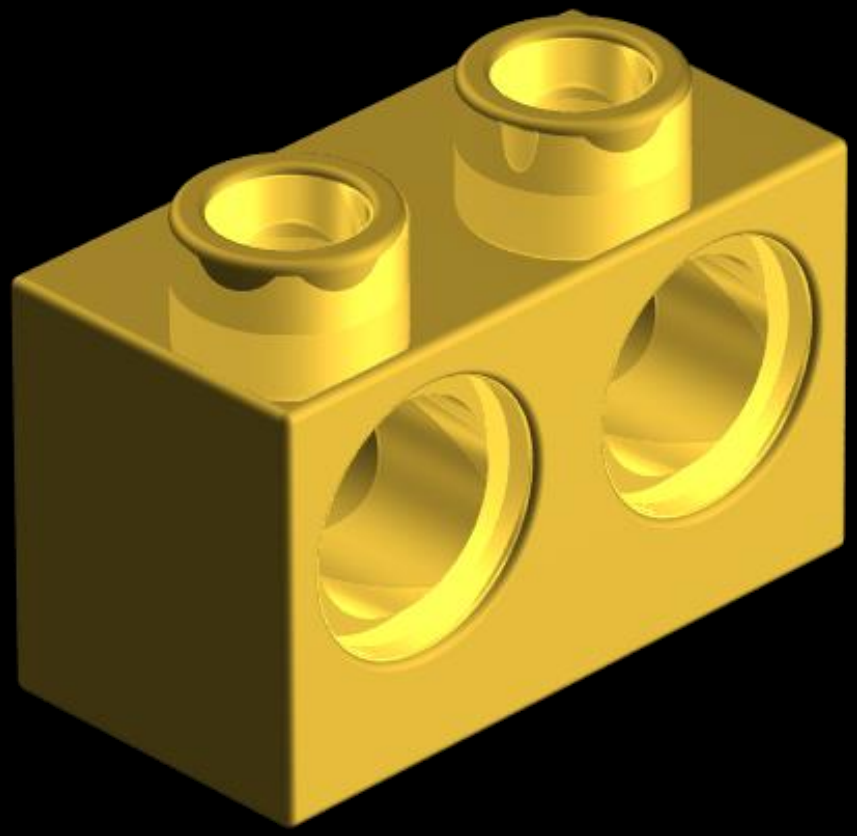
© Theo Jungeblut 2014

COMPONENT / SERVICE

http://www.technicopedia.com/8865.html

© Theo Jungeblut 2014

CLASS, STRUCT, ENUM ETC.



http://technicbricks.blogspot.com/2009/06/tbs-techpoll-12-results-2009-1st.html

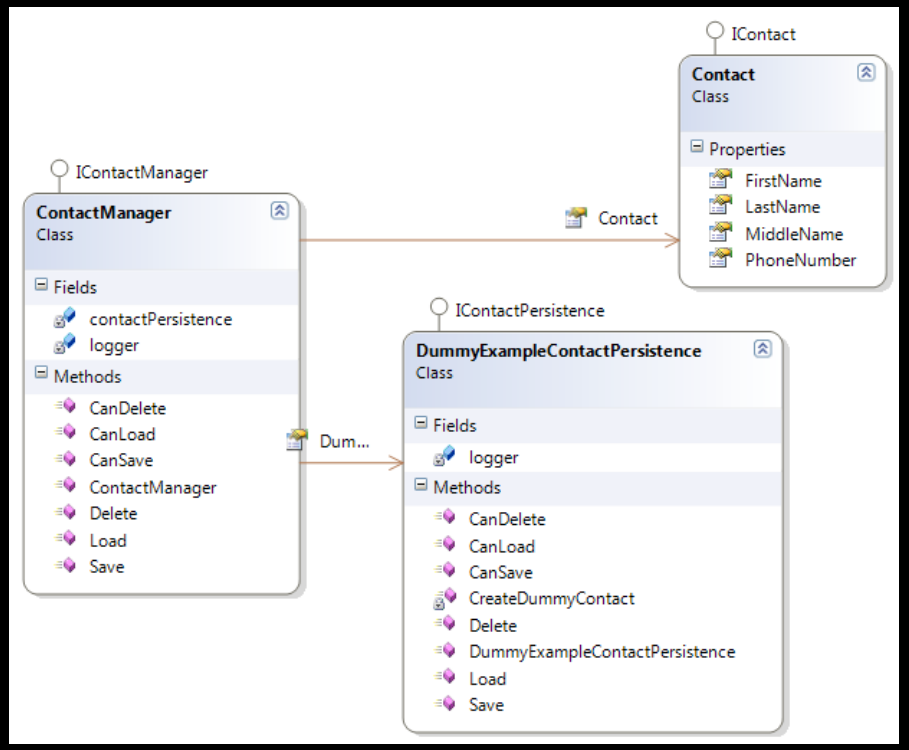
© Theo Jungeblut 2014

SEPARATION OF CONCERNS (SOC)

probably by Edsger W. Dijkstra in 1974

- "In computer science, separation of concerns (SoC) is the process of separating a computer program into distinct features that overlap in functionality as little as possible.

- A concern is any piece of interest or focus in a program. Typically, concerns are synonymous with features or behaviors. "



http://en.wikipedia.org/wiki/Separation_of_Concerns

© Theo Jungeblut 2014

SINGLE RESPONSIBILITY PRINCIPLE (SRP)

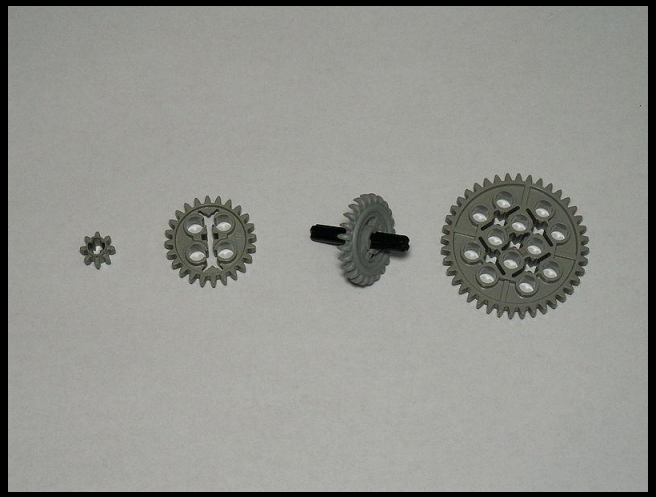
by Robert C Martin

"Every object should have a single responsibility, and that responsibility should be entirely encapsulated by the class."

http://en.wikipedia.org/wiki/Single_responsibility_principle

```
public class Logger : ILogger
{
    public Logger(ILogRepository  repository)
    {}

    public void Log(string message)
    {}
}
```



http://www.ericalbrecht.com

© Theo Jungeblut 2014

INFORMATION HIDING PRINCIPLE (IHP)

INFORMATION HIDING PRINCIPLE (IHP)

by David Parnas (1972)

"... information hiding is the principle of segregation of the design decisions on a computer program that are most likely to change, .."

http://en.wikipedia.org/wiki/Information_hiding

© Theo Jungeblut 2014

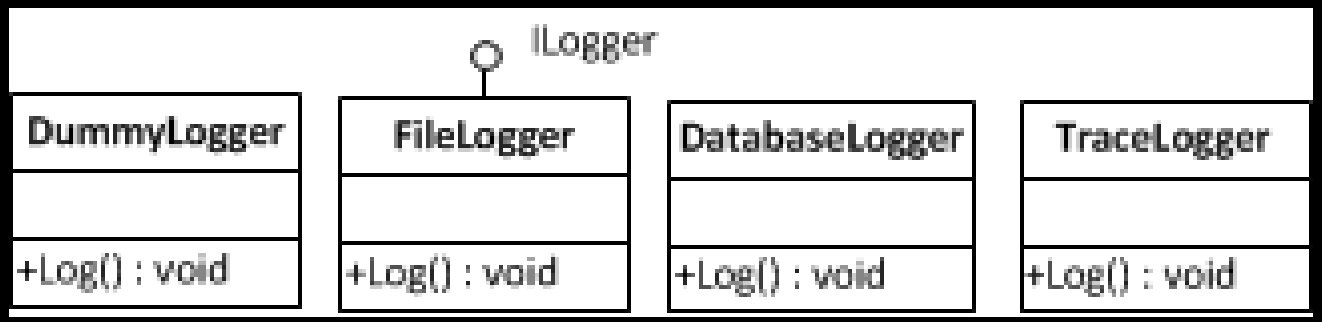
INTERFACES / CONTRACTS

- Decouple Usage and Implementation through introduction of a contract
  - Allows to replace implementation without changing the consumer

```
public interface ILogger
{
  void Log(string message);
}
```

```
public class Logger : ILogger
{
    public Logger(ILogRepository repository)
    {}

    public void Log(string message)
    {}
}
```



© Theo Jungeblut 2014

OPEN CLOSED PRINCIPLE (OCP)

OPEN/CLOSED PRINCIPLE (OCP)

by Bertrand Meyer (1988)

An implementation is open for extension but closed for modification

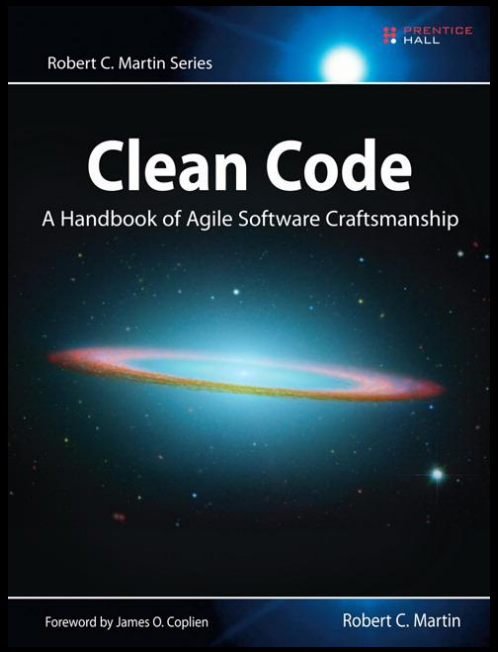
http://en.wikipedia.org/wiki/Open/closed_principle

© Theo Jungeblut 2014

SUMMARY DESIGN PRINCIPLES

Maintainability is achieved through:

- Simplification and Specialization *(KISS, SoC, SRP, OCP,)*
- Decoupling *(LSP, DIP, IHP, Contracts, LoD, CoP, IoC or SOA)*
- Avoiding Code Bloat (DRY, YAGNI)
- Quality through Testability (all of them!)



© Theo Jungeblut 2014

REFERENCES…

http://clean-code-developer.com
http://blogs.telerik.com/justteam/posts/13-05-16/clean-code-developer-school
http://michael.hoennig.de/2009/08/08/clean-code-developer-ccd/
http://butunclebob.com/ArticleS.UncleBob.PrinciplesOfOod
http://www.manning.com/seemann/
http://en.wikipedia.org/wiki/Keep_it_simple_stupid
http://picocontainer.org/patterns.html
http://en.wikipedia.org/wiki/Separation_of_concerns
http://en.wikipedia.org/wiki/Single_responsibility_principle
http://en.wikipedia.org/wiki/Information_hiding
http://en.wikipedia.org/wiki/Liskov_substitution_principle
http://en.wikipedia.org/wiki/Dependency_inversion_principle
http://stackoverflow.com/questions/6766056/dip-vs-di-vs-ioc
http://en.wikipedia.org/wiki/Open/closed_principle
http://en.wikipedia.org/wiki/Law_Of_Demeter
http://en.wikipedia.org/wiki/Don't_repeat_yourself
http://en.wikipedia.org/wiki/You_ain't_gonna_need_it
http://en.wikipedia.org/wiki/Component-oriented_programming
http://en.wikipedia.org/wiki/Service-oriented_architecture
http://www.martinfowler.com/articles/injection.html
http://www.codeproject.com/KB/aspnet/IOCDI.aspx
http://msdn.microsoft.com/en-us/magazine/cc163739.aspx
http://msdn.microsoft.com/en-us/library/ff650320.aspx
http://msdn.microsoft.com/en-us/library/aa973811.aspx
http://msdn.microsoft.com/en-us/library/ff647976.aspx
http://msdn.microsoft.com/en-us/library/cc707845.aspx
http://www.idesign.net/idesign/DesktopDefault.aspx?tabindex=5&tabid=11

DESIGN PRINCIPLES

Next Up: Dependency Injection