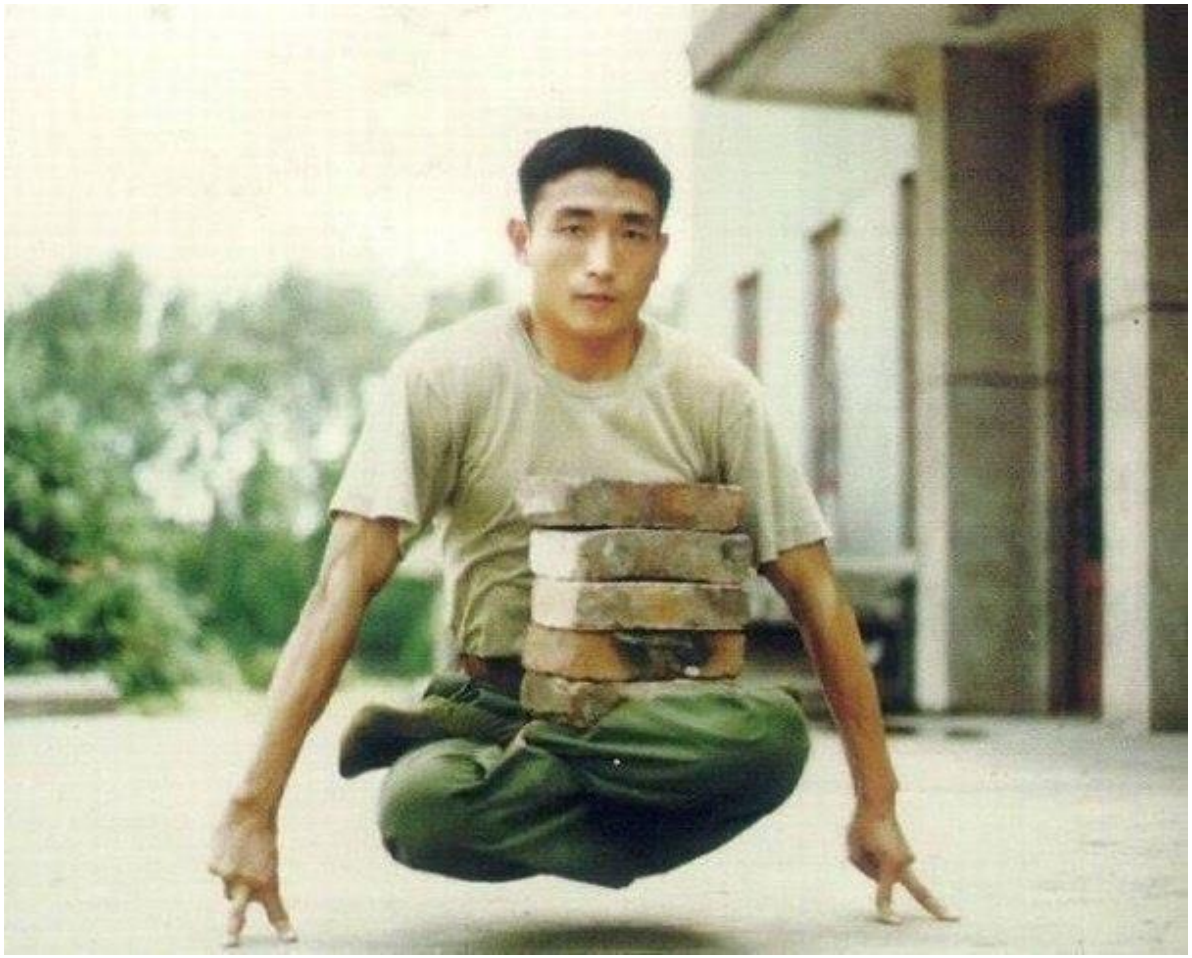
Good coding practice in real life

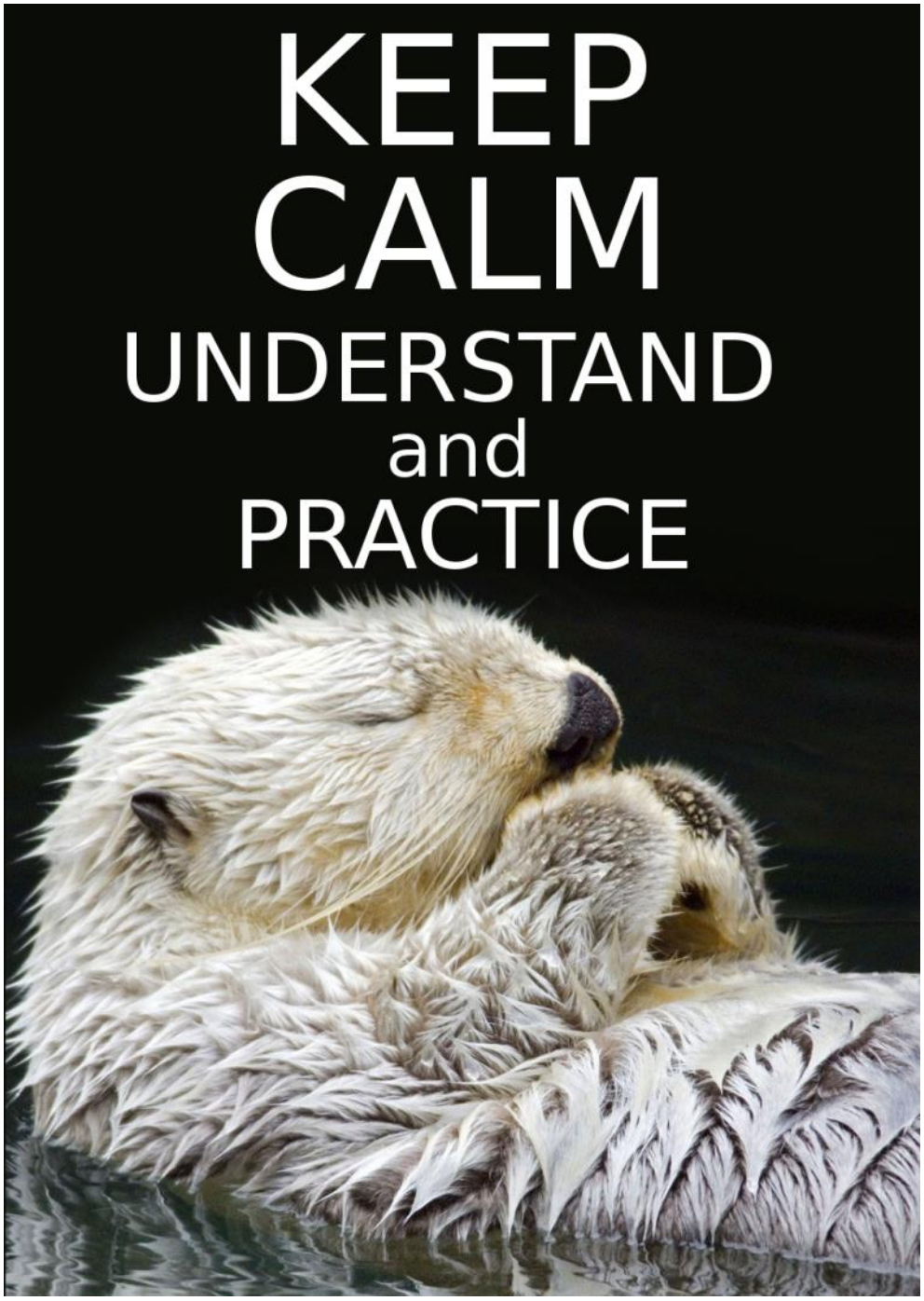
(with a focus on SRP)

Vladimir Alekseichenko

Good practices make life easier

Good practices are not easy




KEEP
CALM
UNDERSTAND
and
PRACTICE

Agenda

- Curly's Law: Do One Thing
- Related concepts
- The single responsibility principle
- What is a responsibility?
- The five examples

Curly's Law: **do one thing**

http://www.youtube.com/watch?v=2KAF5AASdYo

Curly's Law: **do one thing**

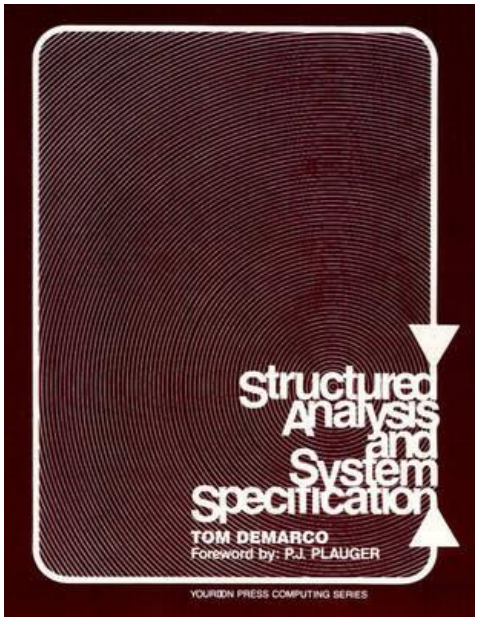
Jack Palance played grizzled cowboy **Curly Washburn** in the 1991 comedy **City Slickers**.

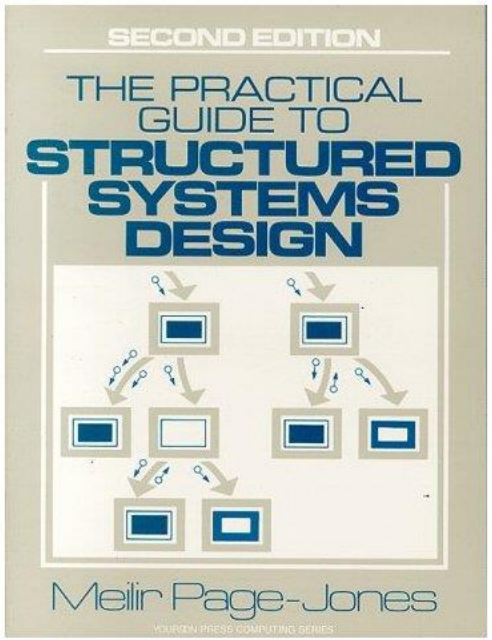
- *Curly*: Do you know what the secret of life is?
- *Curly*: This. [holds up one finger]
- *Mitch*: Your finger?
- *Curly*: One thing. Just one thing. You stick to that and the rest don't mean shit.
- *Mitch*: But what is the „one thing”?
- *Curly*: [smiles] That's what you have to find out.

http://www.youtube.com/watch?v=2KAF5AASdYo

Cohesion

This principle was described in the work of **Tom DeMarco** and **Meilir Page-Jones**. They called it *cohesion*.





http://www.objectmentor.com/resources/articles/srp.pdf

Low cohesion

http://persbaglio.ilcannocchiale.it/?TAG=disordine

High cohesion

http://www.3deko.info/inter/ofor/415-uyutnyj-balkon-mesto-otdyxa-chast-ii.html

Separation of concerns

Create distance between dissimilar concepts in your code.

This allows you to change one without affecting the other.

The single responsibility principle

Robert Cecil Martin aka „Uncle Bob”

http://www.butunclebob.com/CleanCoder.UncleBob

Single responsibility principle

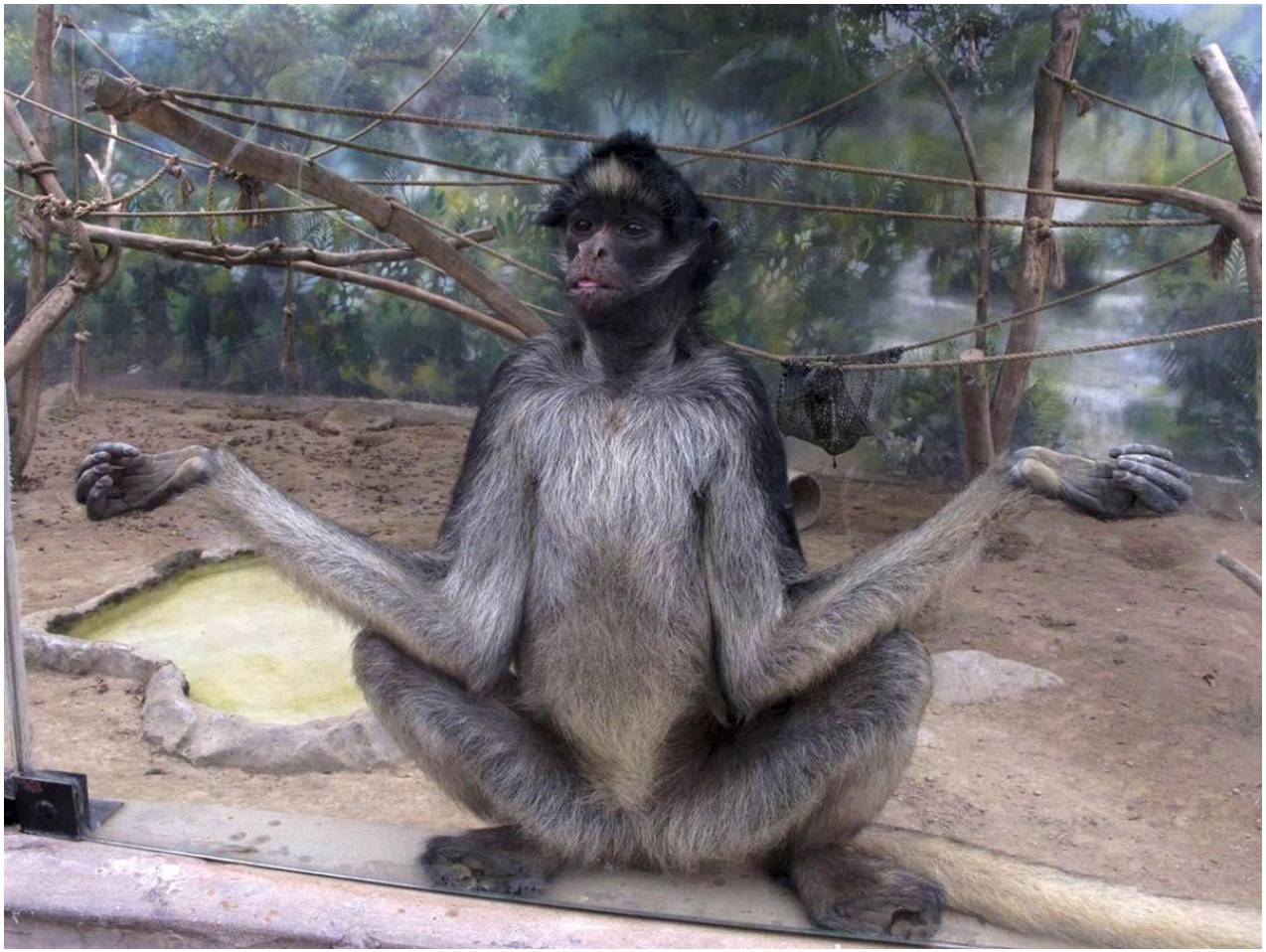
The term was introduced by **Robert C. Martin** in an article by the same name as part of his *Principles of Object Oriented Design.*

Single responsibility principle

The class should have **one**, and only **one**, **reason** to change.

http://butunclebob.com/ArticleS.UncleBob.PrinciplesOfOod

What is a responsibility?



What is a responsibility?

In the context of the SRP we define a

responsibility to be

a **reason** for **change**.

This is sometimes **hard** to see.

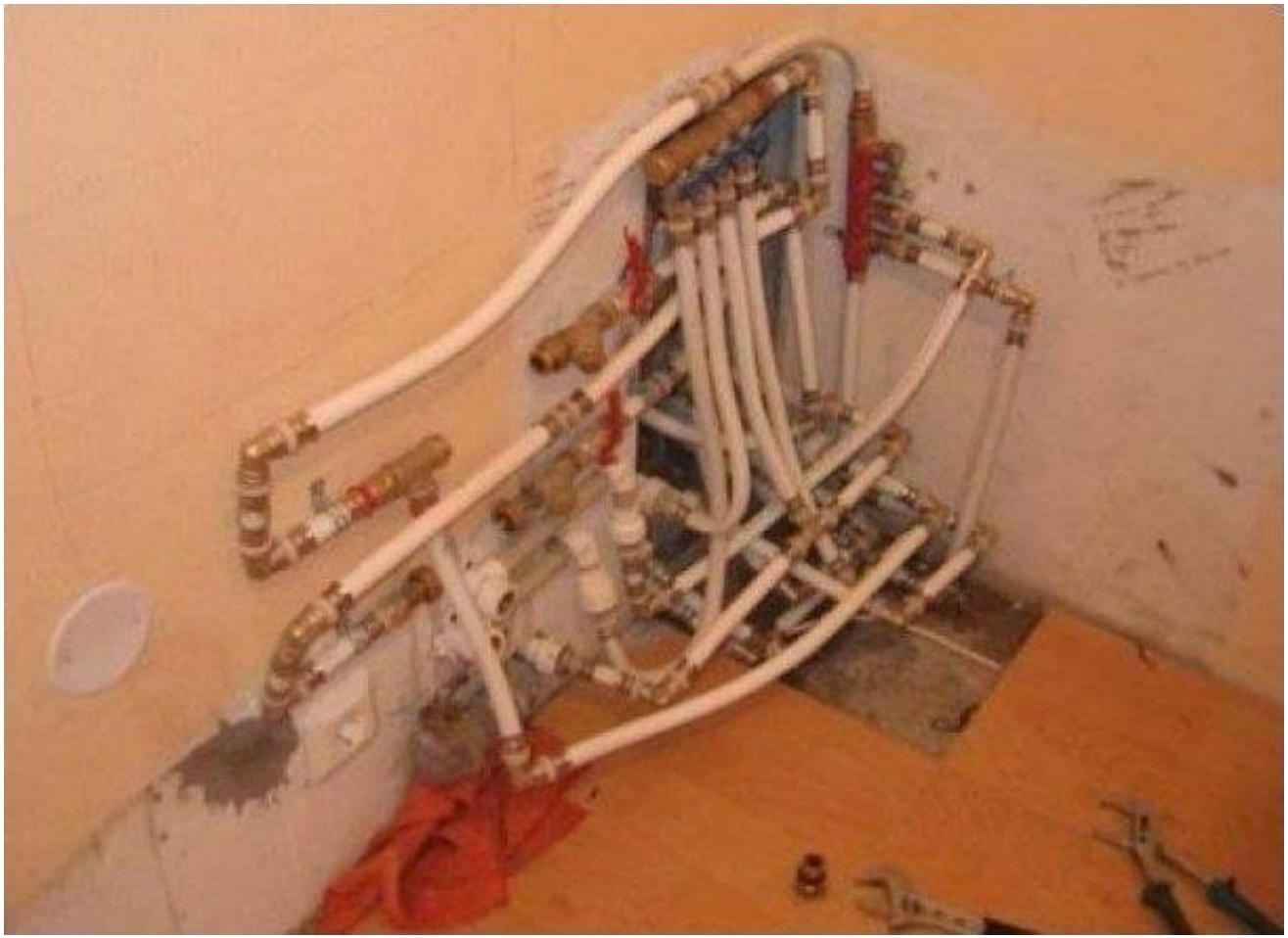
http://www.objectmentor.com/resources/articles/srp.pdf

…only **one reason** to change.

It's a dog or a plane?

…only **one reason** to change.

Is this plumbing nightmare?



Important concept

SRP is **one** of the more important concept in OO design.

It’s also one of the simpler concepts to understand and adhere to.

Clean Code: A Handbook of Agile Software Craftsmanship by *Robert C. Martin*

Why?

Single responsibility principle

The SRP is one of the **simplest** of the principle, and one of the **hardest** to get right.

http://www.objectmentor.com/resources/articles/srp.pdf

Single responsibility principle

The reason it is important to keep a class focused on a **single** concern is that it makes the class more **robust**.

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

It isn’t difficult, we can handle it

The first example

A modem

```
public interface IModem
{
    void Dial(string pno);
    void Hangup();
    void Send(char c);
    char Recv();
}
```

http://www.objectmentor.com/resources/articles/srp.pdf

How many responsibilities do you see?

http://siriusbuzz.com/what-happens-with-that-sirius-lifetime-subscription.php/big-question-mark

There are two responsibilities

- connection management (*Dial, Hangup*),
- data communication (Send, Recv).

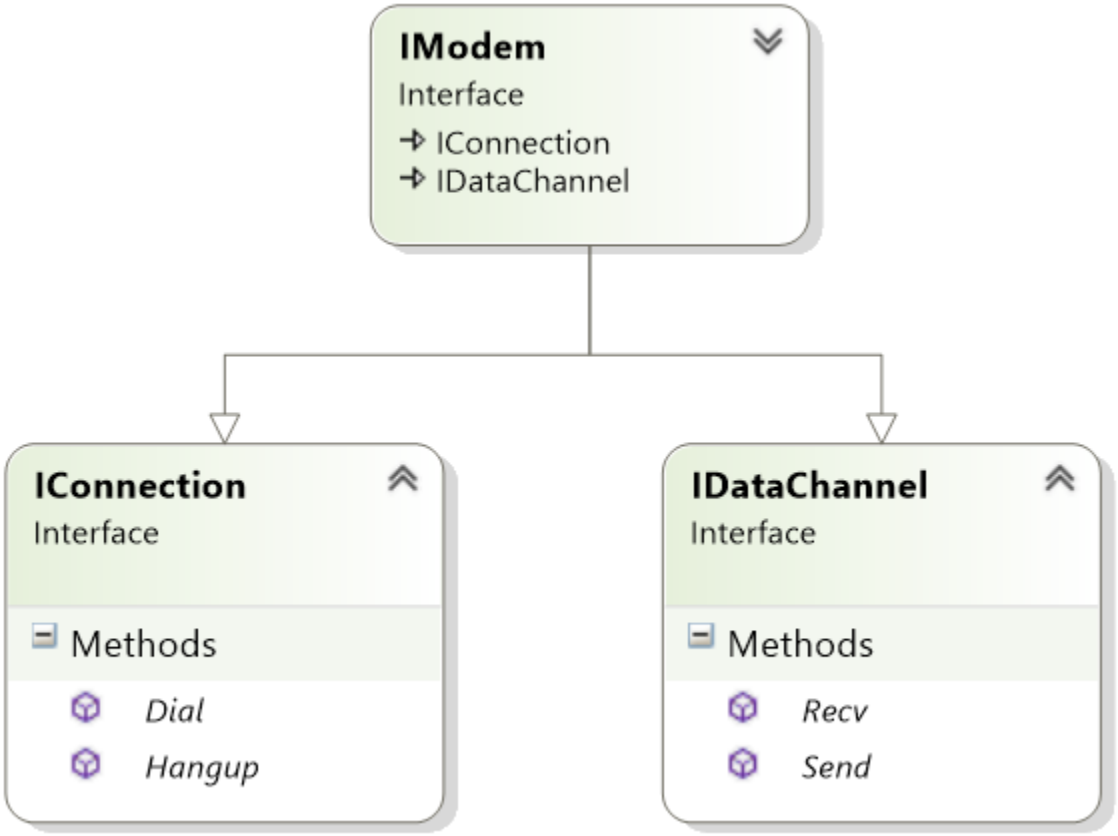
```
public interface IModem
{
    void Dial(string pno);
1   void Hangup();
    void Send(char c);
2   char Recv();
}
```

http://www.objectmentor.com/resources/articles/srp.pdf

How to do well?

http://siriusbuzz.com/what-happens-with-that-sirius-lifetime-subscription.php/big-question-mark

The solution



http://www.objectmentor.com/resources/articles/srp.pdf

The solution

```
public interface IModem : IConnection, IDataChannel
{
}

public interface IConnection
{
    void Dial(string pno);
    void Hangup();
}

public interface IDataChannel
{
    void Send(char c);
    char Recv();
}
```

http://www.objectmentor.com/resources/articles/srp.pdf

The second example

A radio

```
public interface IRadio
{
    void ChangeStation();
    void VolumeDown();
    void VolumeUp();
}
```

http://www.remondo.net/solid-principles-csharp-single-responsibility

How many responsibilities do you see?

http://siriusbuzz.com/what-happens-with-that-sirius-lifetime-subscription.php/big-question-mark

There are two responsibilities

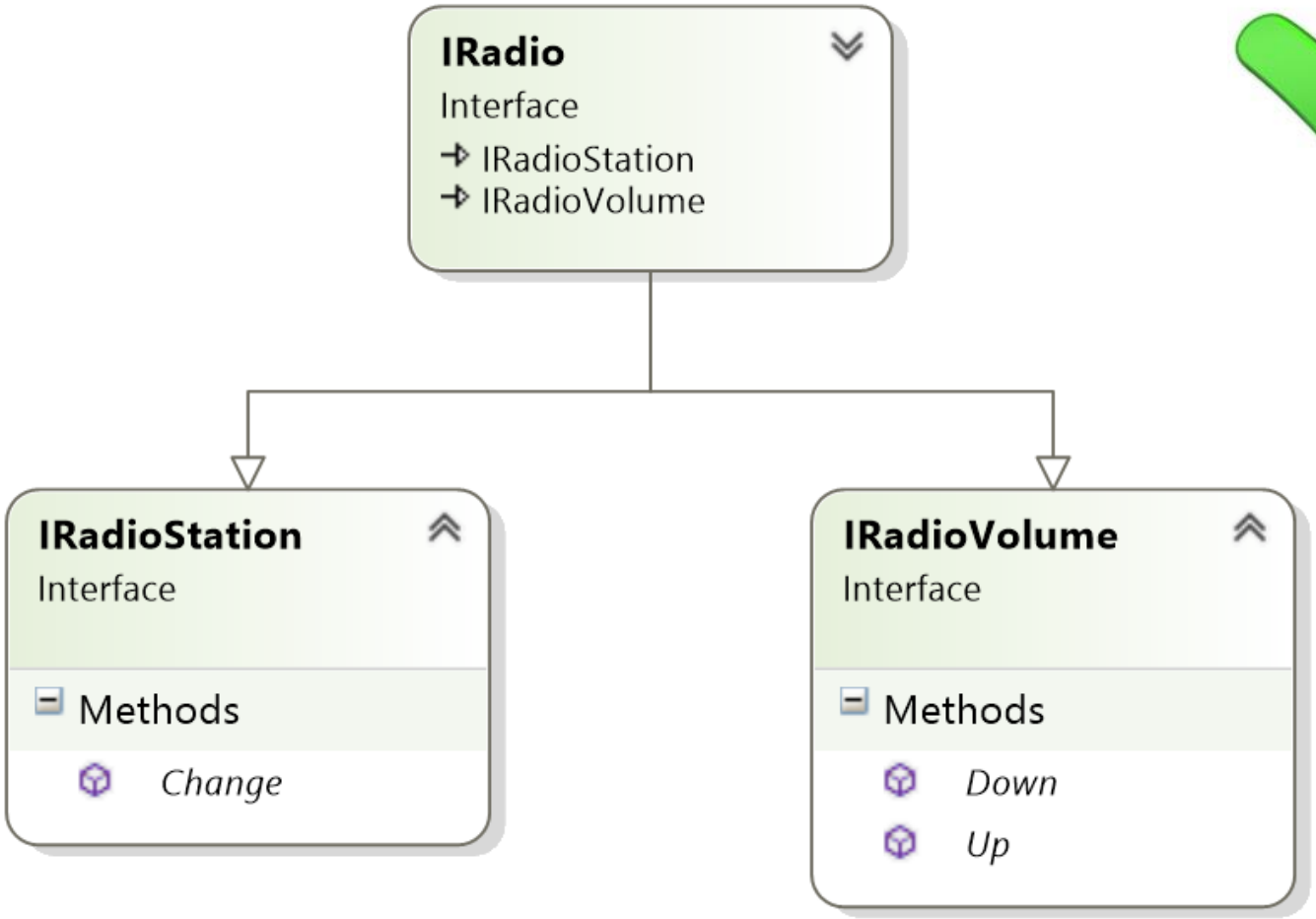
```
public interface IRadio
{
1   void ChangeStation();
    void VolumeDown();
2   void VolumeUp();
}
```

http://www.remondo.net/solid-principles-csharp-single-responsibility

How to do well?

http://siriusbuzz.com/what-happens-with-that-sirius-lifetime-subscription.php/big-question-mark

The solution



The solution

```
public interface IRadio
    : IRadioStation, IRadioVolume
{
}

public interface IRadioStation
{
    void Change();
}

public interface IRadioVolume
{
    void Down();
    void Up();
}
```

The third example

An user

```
public class User
{
    public string Name { get; set; }

    public bool IsValid()
    {
        return false == string.IsNullOrEmpty(Name);
    }
}
```

How many responsibilities do you see?

http://siriusbuzz.com/what-happens-with-that-sirius-lifetime-subscription.php/big-question-mark

There are two responsibilities

```
public class User
{
    public string Name { get; set; }

    public bool IsValid()
    {
        return false == string.IsNullOrEmpty(Name);
    }
}
```

1

2

How to do well?

http://siriusbuzz.com/what-happens-with-that-sirius-lifetime-subscription.php/big-question-mark

The solution

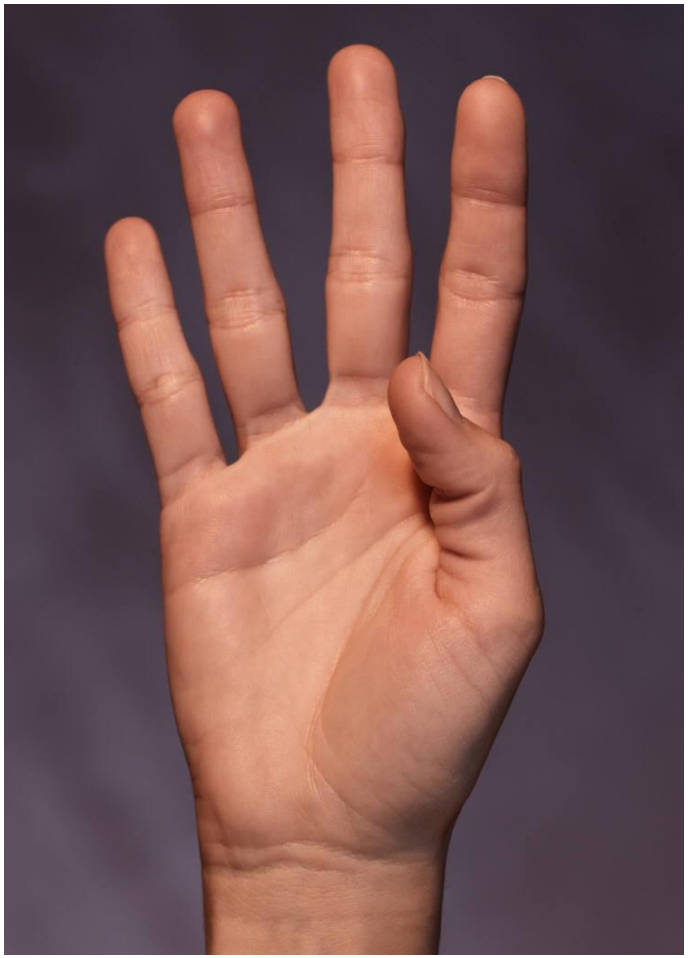
```
public interface IUserValidator
{
    bool IsValid(string name);
}

public class UserNameValidator : IUserValidator
{
    public bool IsValid(string name)
    {
        return false == String.IsNullOrEmpty(name);
    }
}

public class User
{
    private readonly IUserValidator _validator;

    public User(IUserValidator validator)
    {
        _validator = validator;
    }

    public string Name { get; set; }


    public bool IsValid()
    {
        return _validator.IsValid(Name);
    }
}
```

The solution

```
public class LengthStringValidator : IUserValidator
{
   private readonly int? _min;
   private readonly int? _max;

   public LengthStringValidator(int? min = null, int? max = null)
   {
      _min = min;
      _max = max;
   }

   public bool IsValid(string name)
   {
      if (String.IsNullOrEmpty(name))
         return false;

      if (null != _min && _min > name.Length)
         return false;

      if (null != _max && _max < name.Length)
         return false;

      return true;
   }
}
```

```
var user = new User(new LengthStringValidator(1, 10));
user.IsValid();
```

The fourth example



A hasher

```
public class Hasher
{
    public byte[] Compute(string path)
    {
        using (var fileStream = File.OpenRead(path))
        {
            var algorithm = HashAlgorithm.Create(HashNames.MD5.ToString());
            Debug.Assert(algorithm != null, "algorithm != null");
            return algorithm.ComputeHash(fileStream);
        }
    }
}
```

How many responsibilities do you see?

http://siriusbuzz.com/what-happens-with-that-sirius-lifetime-subscription.php/big-question-mark

There are three responsibilities

```
public class Hasher
{
    public byte[] Compute(string path)
    {
        using (var fileStream = File.OpenRead(path))
        {
            var algorithm = HashAlgorithm.Create(HashNames.MD5.ToString());
            Debug.Assert(algorithm != null, "algorithm != null");
            return algorithm.ComputeHash(fileStream);
        }
    }
}
```

1

2

3

The dangerous way

http://www.break.com/pictures/dangerous-way-to-sleep-1519154

The dangerous way

```
public class Hasher
{
    public byte[] Compute(string path)
    {
        using (var fileStream = File.OpenRead(path))
        {
            var algorithm = HashAlgorithm.Create(HashNames.MD5.ToString());
            Debug.Assert(algorithm != null, "algorithm != null");
            return algorithm.ComputeHash(fileStream);
        }
    }
}
```

The dangerous way!

How to do well?

http://siriusbuzz.com/what-happens-with-that-sirius-lifetime-subscription.php/big-question-mark

The solution

```
public class Hasher
{
    private readonly HashAlgorithm _algorithm;

    public Hasher(HashNames hashName = HashNames.MD5)
    {
        _algorithm = HashAlgorithm.Create(hashName.ToString());
    }

    public byte[] Compute(FileStream fileStream)
    {
        return _algorithm.ComputeHash(fileStream);
    }
}
```

What has changed?

```
public class Hasher
{
    private readonly HashAlgorithm _algorithm;

    public Hasher(HashAlgorithm algorithm)
    {
        _algorithm = algorithm;
    }

    public byte[] Compute(FileStream fileStream)
    {
        return _algorithm.ComputeHash(fileStream);
    }
}
```

Dependency injection

The safe way

The only one responsible

In summary

- Violation of SRP causes spurious transitive dependencies between modules that are hard to anticipate, in other words *fragility.*
- If your classes do not match the SR-principle, you have a bad smell, or a reason for refactoring (Fowler).
- SRP avoids responsibility coupling.

Resources

Books and papers

- **The Single Responsibility Principle** by *Robert C. Martin.*
- **Clean Code: A Handbook of Agile Software Craftsmanship** by *Robert C. Martin.*
- **Head First Software Development** by *Dan Pilone, Russ Miles*.
- **Head First Object-Oriented Analysis and Design** by *Brett D. McLaughlin, Gary Pollice, David West.*
- **Head First Design Patterns** *by Eric Freeman, Elisabeth Freeman, Kathy Sierra, Bert Bates*