Liskov Substitution Principle

Steve Smith
FORCE MULTIPLIER FOR DEV TEAMS
@ardalis | ardalis.com | weeklydevtips.com

SOLID Principles

SRP

Single Responsibility Principle

OCP

Open / Closed Principle

LSP

Liskov Substitution Principle

Interface Segregation Principle

ISP

Dependency Inversion Principle

DIP

Liskov Substitution Principle

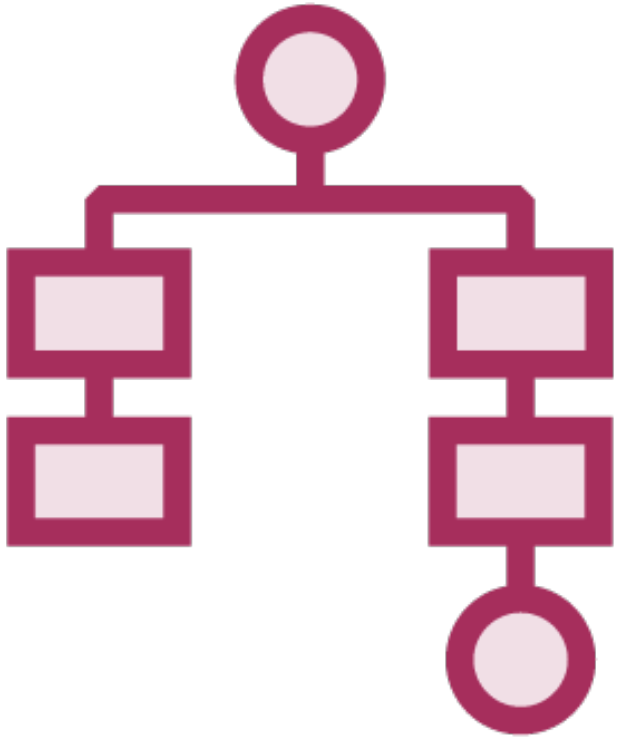
Let Φ(x) be a property provable about objects x of type T. Then Φ(y) should be true for objects y of type S where S is a subtype of T.

Liskov Substitution Principle

Subtypes must be substitutable for their base types.

Barbara Liskov introduced the principle in a conference keynote in 1987.

Basic Object-Oriented Design



Something IS-A something else

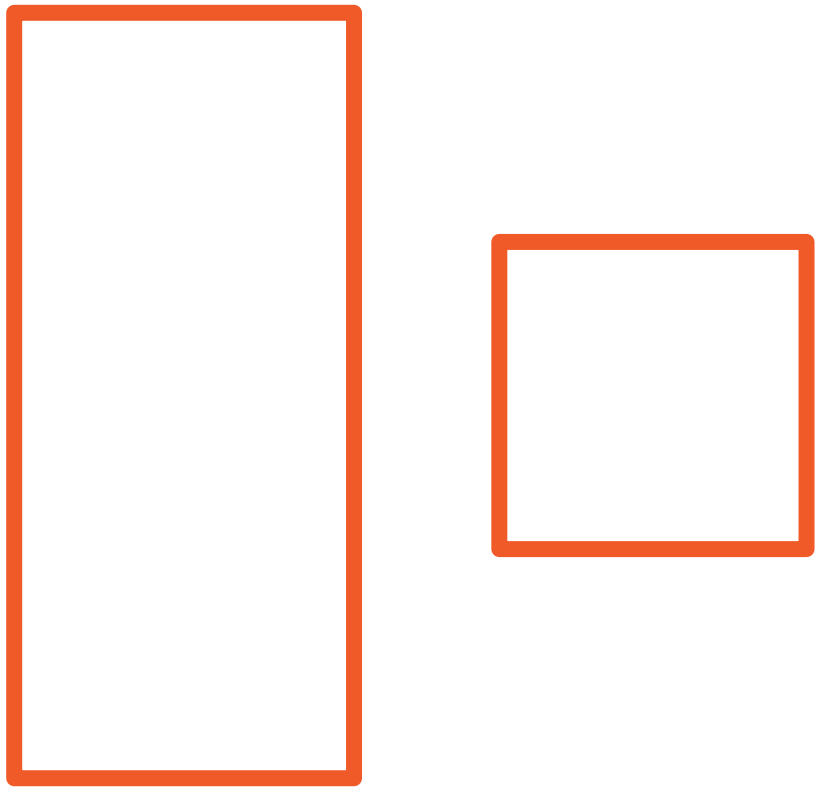
- An eagle IS-A bird

Something HAS-A property

- An address HAS-A city

LSP states that the IS-A relationship is insufficient and should be replaced with IS-SUBSTITUTABLE-FOR.

Classic Rectangle-square Problem



A rectangle has four sides and four right angles

A square has four equal sides and four right angles

Per geometry, a square is a rectangle

Rectangle

```
public class Rectangle
{
    public virtual int Height { get; set; }
    public virtual int Width { get; set; }
}
```

Area Calculation Utility

```
public class AreaCalculator
{
    public static int CalculateArea(Rectangle r)
    {
        return r.Height * r.Width;
    }
}
```

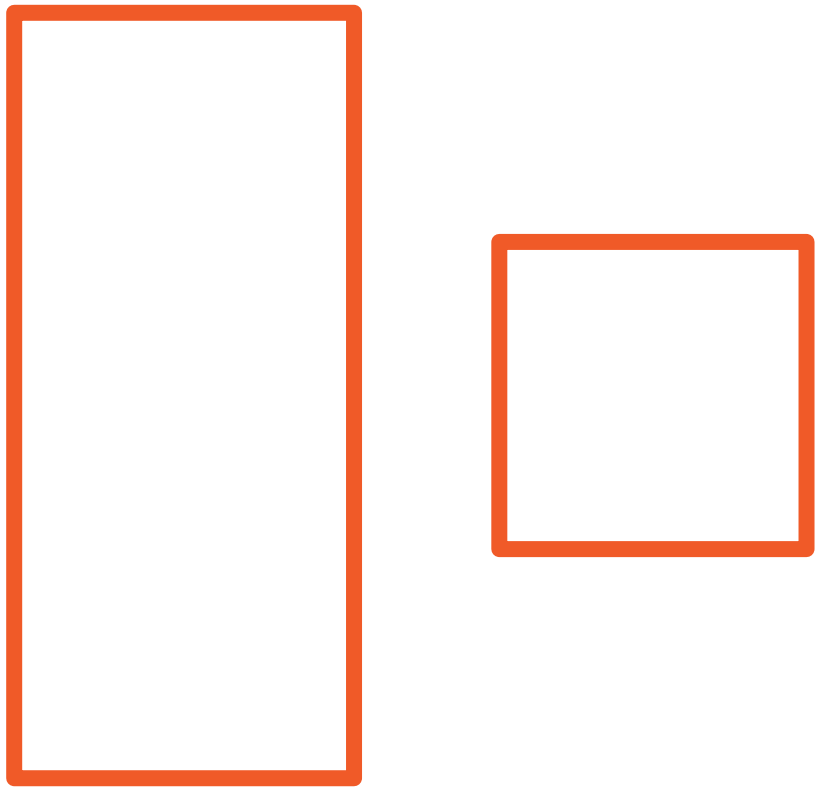
Square (a Subtype of Rectangle)

```
public class Square : Rectangle
{
    private int _height;
    public int Height
    {

        get { return _height; }
        set
        {
            _width = value;
            _height = value;
        }
    }
    // Width implemented similarly
}
```

The Problem

```
Rectangle myRect = new Square();
myRect.Width = 4;
myRect.Height = 5;

Assert.Equal(20, AreaCalculator.CalculateArea(myRect));

// Actual Result: 25
```

What Happened?



Square has an invariant

- Its sides must be equal

Rectangle has an invariant

- Its sides are independent

This design breaks rectangle's invariant and thus violates LSP

One Solution

```
public class Rectangle
{
    public int Height { get; set; }
    public int Width { get; set; }

    public bool IsSquare => Height == Width;
}
```

Another Solution

```
public class Rectangle
{
    public int Height { get; set; }
    public int Width { get; set; }
}

public class Square
{
    public int Side { get; set; }
}
```

Detecting LSP Violations in Your Code

Type checking with `is` or `as` in polymorphic code

Null checks

`NotImplementedException`

Type Checking

```
foreach(var employee in employees)
{
    if(employee is Manager)
    {
        Helpers.PrintManager(employee as Manager);
        break;
    }
    Helpers.PrintEmployee(employee);
}
```

Type Checking (Corrected)

```
foreach(var employee in employees)
{
    employee.Print();
}

// OR

foreach(var employee in employees)
{
    Helpers.PrintEmployee(employee);
}
```

Null Checking

```
foreach(var employee in employees)
{
    if(employee == null)
    {
        Console.WriteLine("Employee not found.");
        break;
    }
    Helpers.PrintEmployee(employee);
}
```

Type Checking

```
foreach(var employee in employees)
{
    if(employee is Manager)
    {
        Helpers.PrintManager(employee as Manager);
        break;
    }
    Helpers.PrintEmployee(employee);
}
```

Learn More

Nulls Break Polymorphism

- ardalis.com/nulls-break-polymorphism

Pluralsight courses

- “Design Patterns Library”

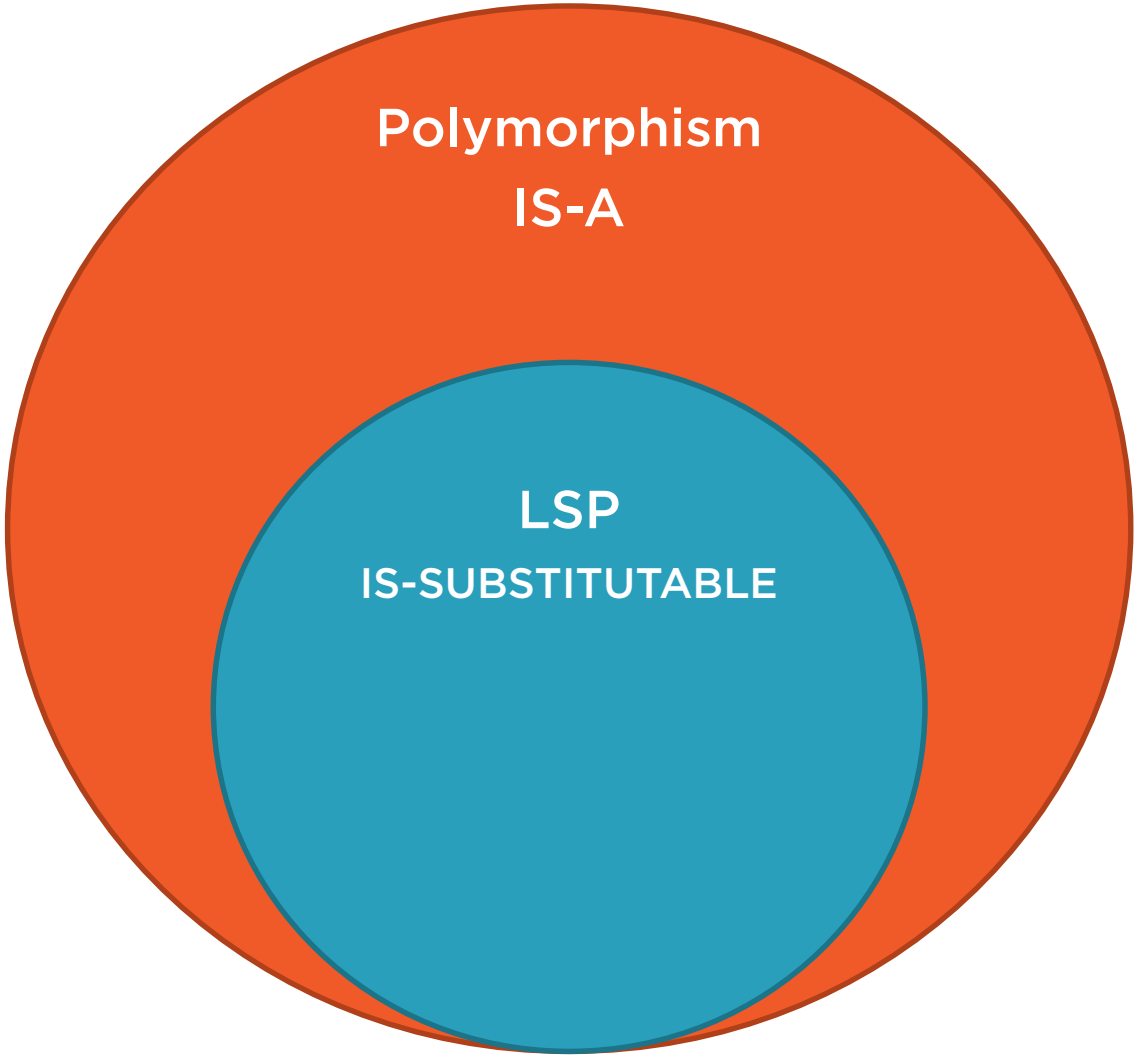
Not Implemented Exceptions

```
public interface INotificationService
{
    void SendText(string SmsNumber, string message);

    void SendEmail(string to, string from,
                   string subject, string body);
}
```

Not Implemented Exceptions

```
public class SmtpNotificationService : INotificationService
{
    public void SendEmail(string to, string from,
                          string subject, string body)
    {
        // actually send email here
    }

    public void SendText(string SmsNumber, string message)
    {
        throw new NotImplementedException();
    }
}
```

LSP Is a Subset of Polymorphism



Fixing LSP Violations

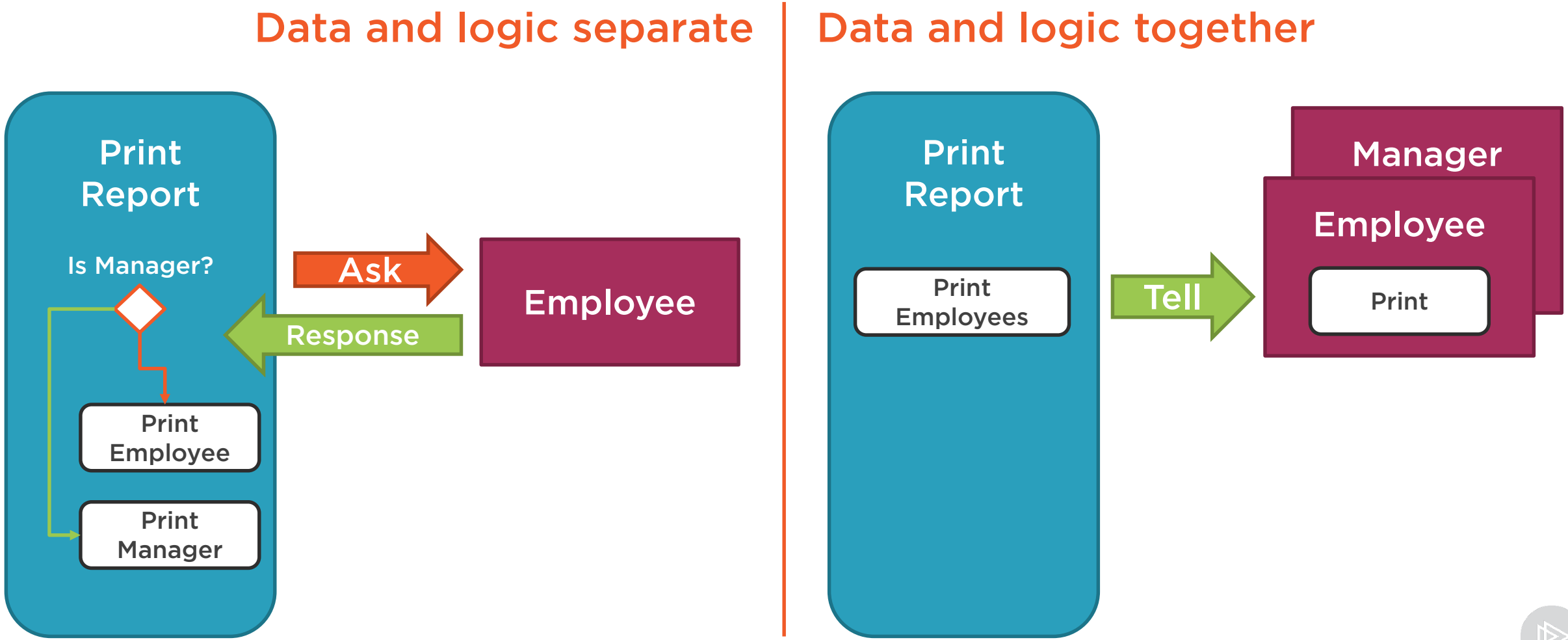
Follow the “Tell, Don’t Ask” principle

Minimize null checks with

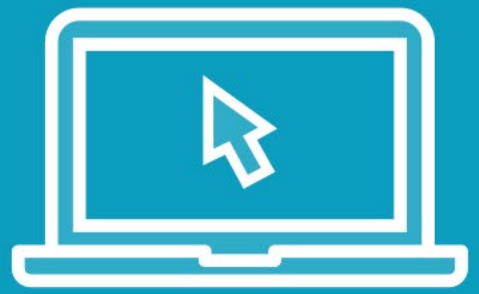
- C# features
- Guard clauses
- Null Object design pattern

Follow ISP and be sure to fully implement interfaces

Tell, Don't Ask



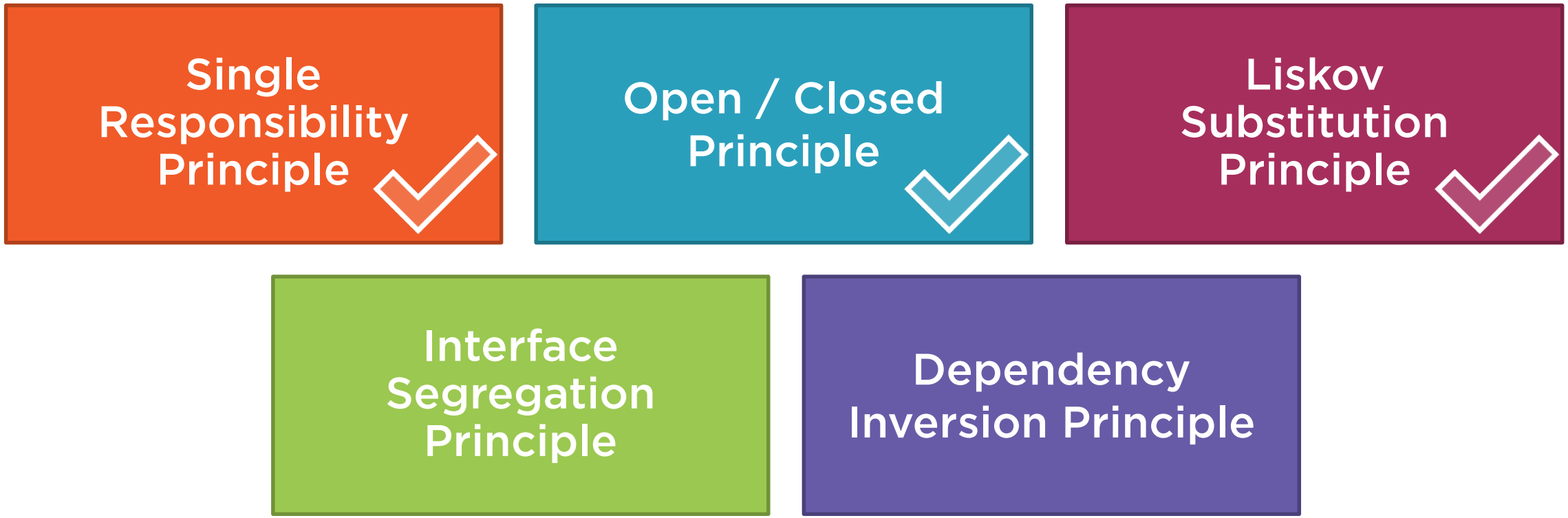
Demo



Applying LSP to ArdalisRating

Available at
https://github.com/ardalis/solidsample

SOLID Principles



Key Takeaways

Subtypes must be substitutable for their base types

Ensure base type invariants are enforced

Look for

- Type checking
- Null checking
- `NotImplementedException`