C# Interfaces

A Practical Guide to Interfaces

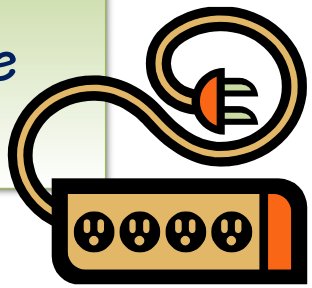
Jeremy Clark
www.jeremybytes.com
jeremy@jeremybytes.com

pluralsight
hardcore developer training

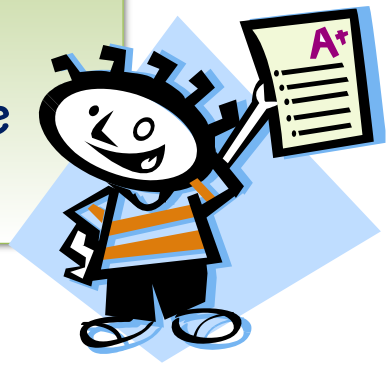
Why Interfaces?

Maintainable

Extensible



Easily Testable



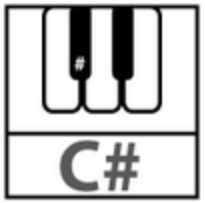
Interfaces help us get there

Goals

- **Learn the "Why"**
  - Maintainability
  - Extensibility
- **Implement Interfaces**
  - .NET Framework Interfaces
  - Custom Interfaces
- **Create Interfaces**
  - Add Abstraction
- **Peek at Advanced Topics**
  - Mocking
  - Unit Testing
  - Dependency Injection

Pre-requisites

- **Basic understanding of C#**
  - Classes
  - Inheritance
  - Properties
  - Methods



C# Fundamentals - Part 1

This course is designed to give you everything you need to become a productive C# developer on the .NET platform

Interfaces, Abstract Classes, and Concrete Classes

The “What”

Jeremy Clark
www.jeremybytes.com
jeremy@jeremybytes.com

pluralsight
hardcore developer training

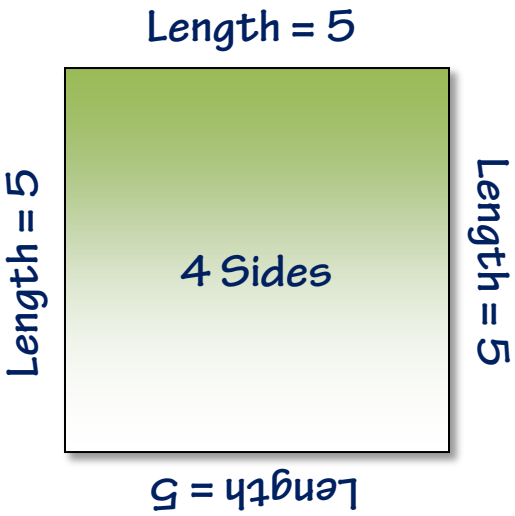
What are Interfaces?

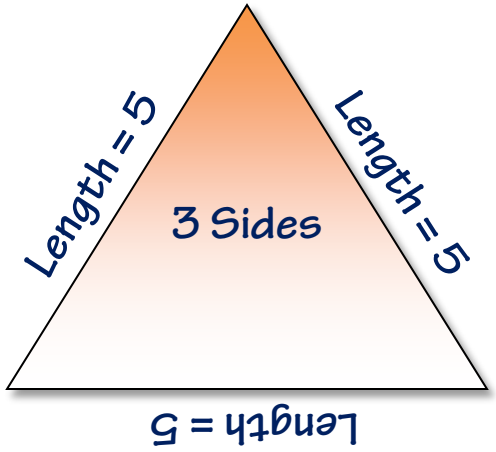
Interfaces describe a group of related functions that can belong to any class or struct.

Public set of members:

- Properties
- Methods
- Events
- Indexers

Scenario: Regular Polygons





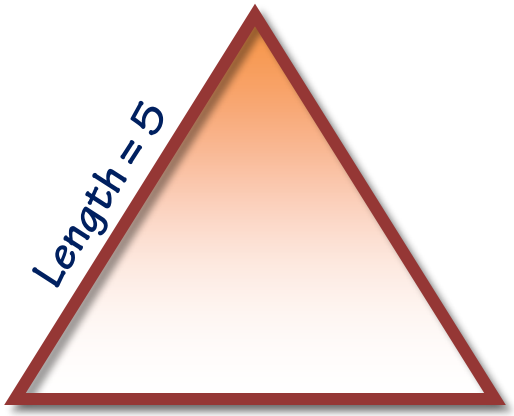
- **3 or more Sides**
- **Each Side has the same Length**

Perimeter

Length = 5

Perimeter =
Number of Sides x Side Length

Perimeter = 4 x 5
Perimeter = 20



Perimeter =
Number of Sides x Side Length

Perimeter = 3 x 5
Perimeter = 15

Area

Length = 5

Area =
Side Length x Side Length

Area = 5 x 5
Area = 25

Area =
Side Length x Side Length x Sqrt(3) / 4

Area = 5 x 5 x Sqrt(3) / 4
Area = 10.8 (approximately)

Concrete Class, Abstract Class, or Interface?

- **Concrete Class**  No Compile-time checking
- **Abstract Class**  Compile-time checking
- **Interface**  Compile-time checking

Comparison: Implementation Code

- **Abstract Classes may contain implementation**

```
public abstract class AbstractRegularPolygon
{
    public double GetPerimeter()
    {
        return NumberOfSides * SideLength;
    }
}
```

- **Interfaces may not contain implementation (declarations only)**

Comparison: Inheritance

- **Inherit from a single Abstract Class (Single Inheritance)**
- **Implement any number of Interfaces**

```
public class List<T> : IList<T>,
    ICollection<T>,IList, ICollection,
    IReadOnlyList<T>,IReadOnlyCollection<T>,
    IEnumerable<T>,IEnumerable
```

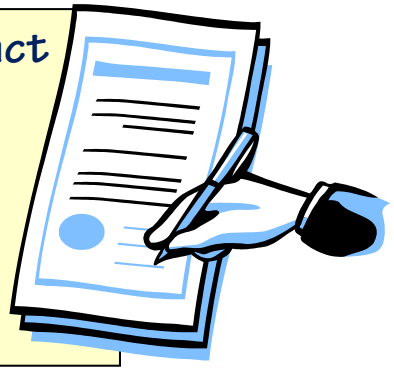
Comparison: Access Modifiers

- **Abstract Class Members can have access modifiers**

```
public abstract class AbstractRegularPolygon
{
    public int NumberOfSides { get; set; }
    public int SideLength { get; set; }
    public double GetPerimeter()...
    public abstract double GetArea();
}
```

- **Interface Members are automatically public**

```
public interface IRegularPolygon
{
    int NumberOfSides { get; set; }
    int SideLength { get; set; }
    double GetPerimeter();
    double GetArea();
}
```



Comparison: Valid Members

Abstract Classes	Interfaces
Fields	Properties
Properties	Methods
Constructors	Events
Destructors	Indexers
Methods	
Events	
Indexers	

Comparison Summary

Abstract Classes

- May contain implementation code
- A class may inherit from a single base class
- Members have access modifiers
- May contain fields, properties, constructors, destructors, methods, events and indexers

Interfaces

- May not contain implementation code
- A class may implement any number of interfaces
- Members are automatically public
- May only contain properties, methods, events, and indexers

Summary

- **The "What" of Interfaces**

Public set of members:

- Properties
- Methods
- Events
- Indexers

- **Compiler-enforced implementation**
- **Comparison between Abstract Classes and Interfaces**

- **Next up: The "Why" of Interfaces**