# C# Interfaces

A PRACTICAL GUIDE TO INTERFACES

**Jeremy Clark**
AUTHOR TITLE

@authortwitter www.jeremybytes.com

## Why Interfaces?

Maintainable

Extensible

Easily testable

## Goals

**Learn the ‘Why”**

- Maintainability
- Extensibility

**Implement Interfaces**

- .NET Framework Interfaces
- Custom Interfaces

## Goals

**Create Interfaces**

- Add Abstraction

**Peek at Advanced Topics**

- Mocking
- Unit Testing
- Dependency Injection

## Pre-requisites

**Basic Understanding of C#**

- Classes
- Inheritance
- Properties
- Methods

## Interfaces, Abstract Classes, and Concrete Classes

## What are Interfaces?

## Interface

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

Microsoft

## What are Interfaces?

Contract

**Public set of members**

- Properties
- Methods
- Events
- Indexers

## Regular Polygons

3 or more sides

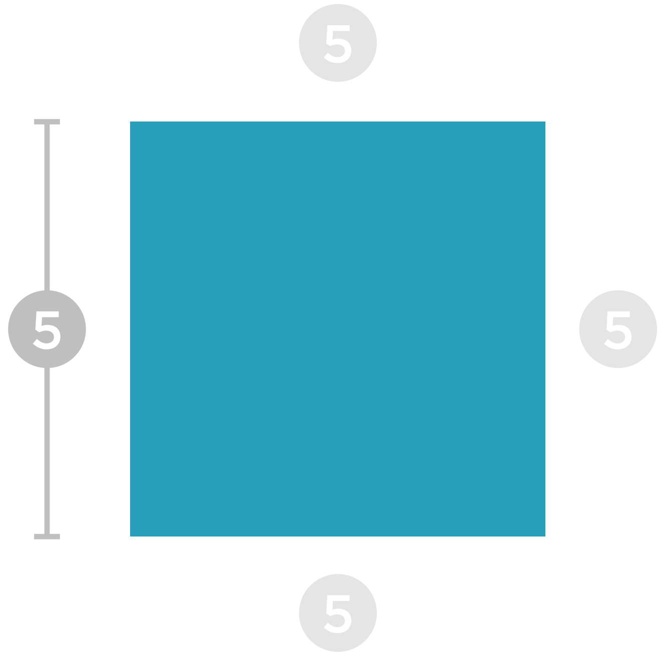
Each side has the same length

## Scenario: Regular Polygons

3 or more sides

Each side has the same length

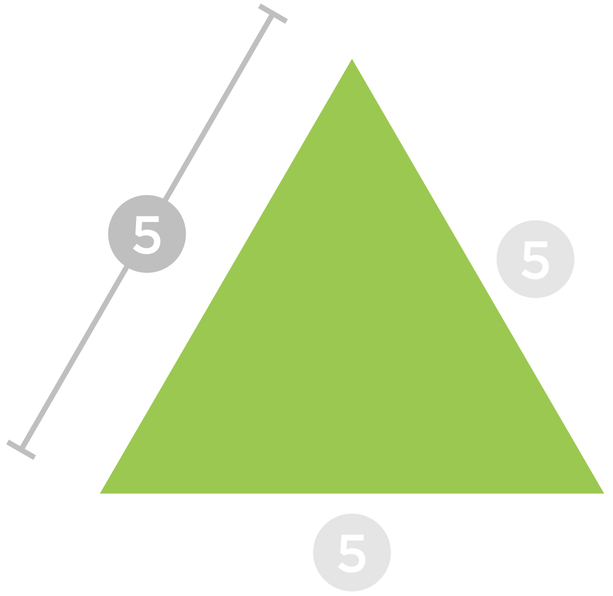
# Scenario: Regular Polygons



**Square**

4 sides

Each side has same length

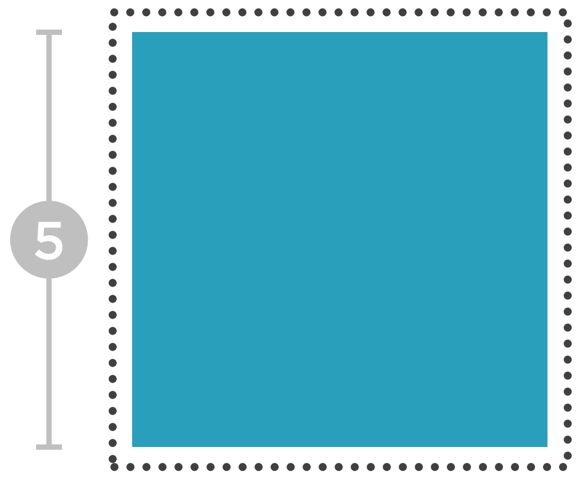


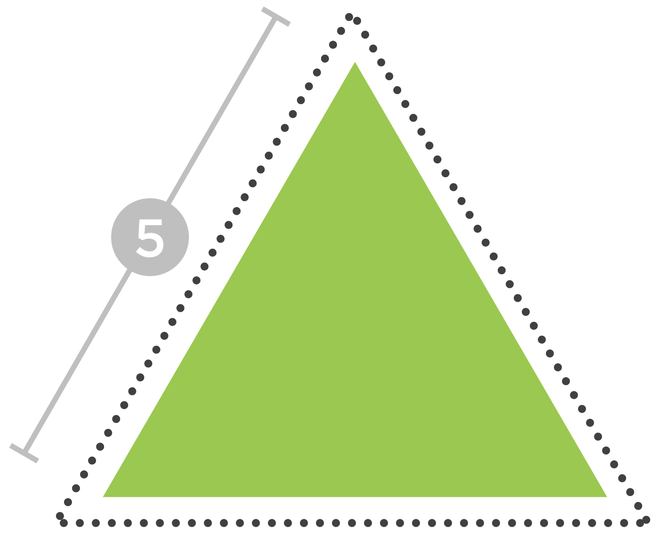
**Equilateral Triangle**

3 sides

Each side has same length

# Perimeter





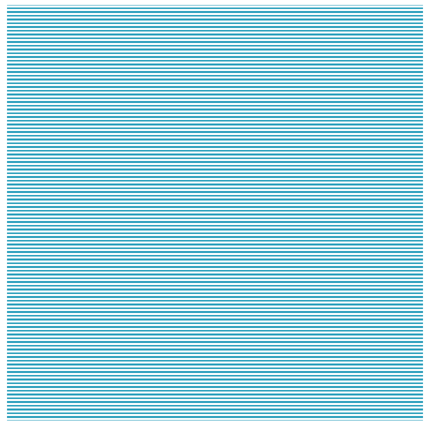
**Perimeter =**
**Number of Sides x Side Length**

**Perimeter = 4 x 5**
**Perimeter = 20**

**Perimeter =**
**Number of Sides x Side Length**

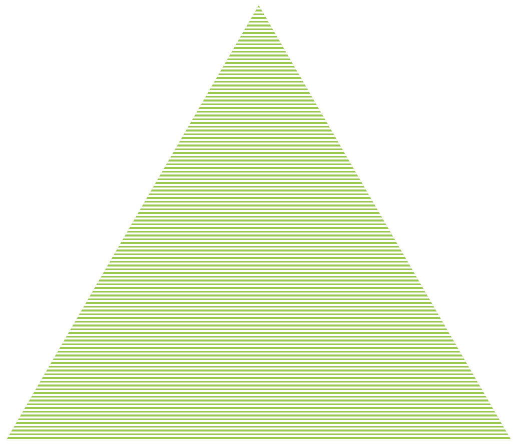
**Perimeter = 3 x 5**
**Perimeter = 15**

## Area



Area =

Side Length x Side Length



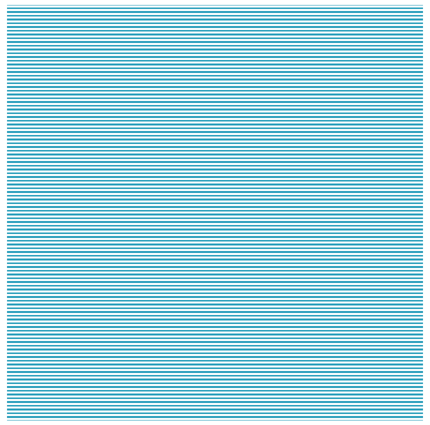
Area =

Side Length x Side Length

x Square Root of 3

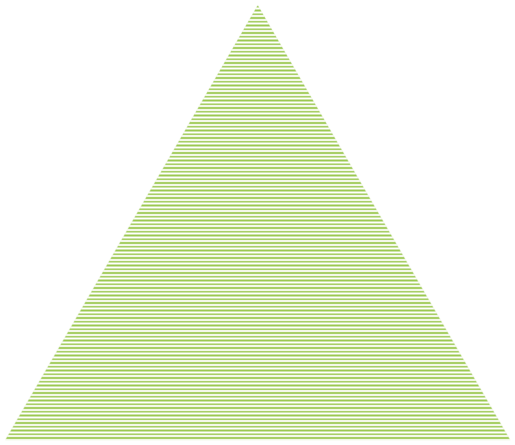
Divided by 4

## Area



Area = 5 x 5

Area = 25



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

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

}
```

## Comparison: Implementation Code

**Abstract Classes may contain implementation**

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

```
Public class List<T> : Ilist<T>
        ICollection<T>, Ilist, Icollection,
        IReadOnlyList<T>, IReadOnlyCollection<T>
        IEnumerable<T>, IEnumerable
```

## Comparison: Inheritance

**Inherit from a single Abstract Class (Single Inheritance)**

**Implement any number of Interfaces**

```
public abstract class AbstractRegularPolygon
{
    public int NumberOfSide { get; set; }
    public int SideLength { get; set; }
    public double GetPErimeter()…
    public abstract double GetArea();
}
```

## Comparison: Access Modifiers

**Abstract Classes Members can have access modifiers**

```
public interface IRegularPolygon
{
    int NumberOfSide { get; set; }
    int SideLength { get; set; }
    double GetPErimeter();
    double GetArea();
}
```

## Comparison: Access Modifiers

**Interface Members are automatically public**

Contract

# Comparison: Valid Members

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

# Comparison Summary

| Abstract Classes | Interfaces |
|---|---|
| May contain implementation code | May not contain implementation code |
| A class may inherit from a single base class | A class may implement any number of interfaces |
| Members have access modifiers | Members are automatically public |
| May contain fields, properties, constructors, destructors, methods, events and indexers | May only contain properties, methods, events, and indexers |

## Comparison Summary

| | Abstract Classes | Interfaces | |
|---|---|---|---|
| 👍 | May contain implementation code | May not contain implementation code | 👎 |
| 👎 | A class may inherit from a single base class | A class may implement any number of interfaces | 👍 |
| | Members have access modifiers | Members are automatically public | |
| | May contain fields, properties, constructors, destructors, methods, events and indexers | 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**

UP NEXT:

## The "Why" of Interfaces