C# Interfaces

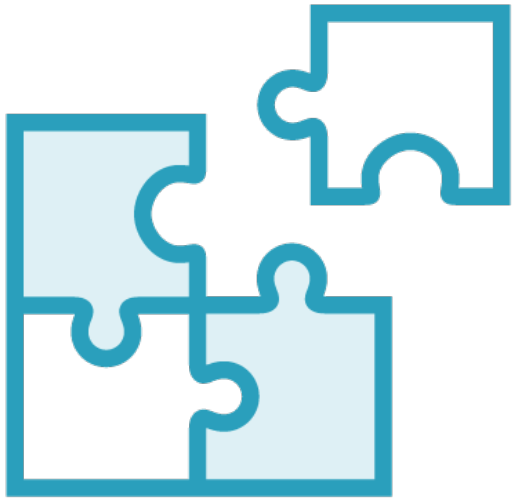
Introducing Interfaces

Jeremy Clark
Developer Betterer

@jeremybytes www.jeremybytes.com

Why Interfaces?

Maintain



Extend

Test

Trying to Learn Interfaces

Interfaces

IFoo and Bar class

NO!

Much reading

Conversations

This is awesome!

Our Roadmap

What?

Definition and technical bits

Why?

Maintain, extend, test

How?

Create and implement

Where?
Practical bits, dependency injection, and unit testing

Prerequisites

- **Classes**
- **Inheritance**
- **Properties**
- **Methods**

Development Environment

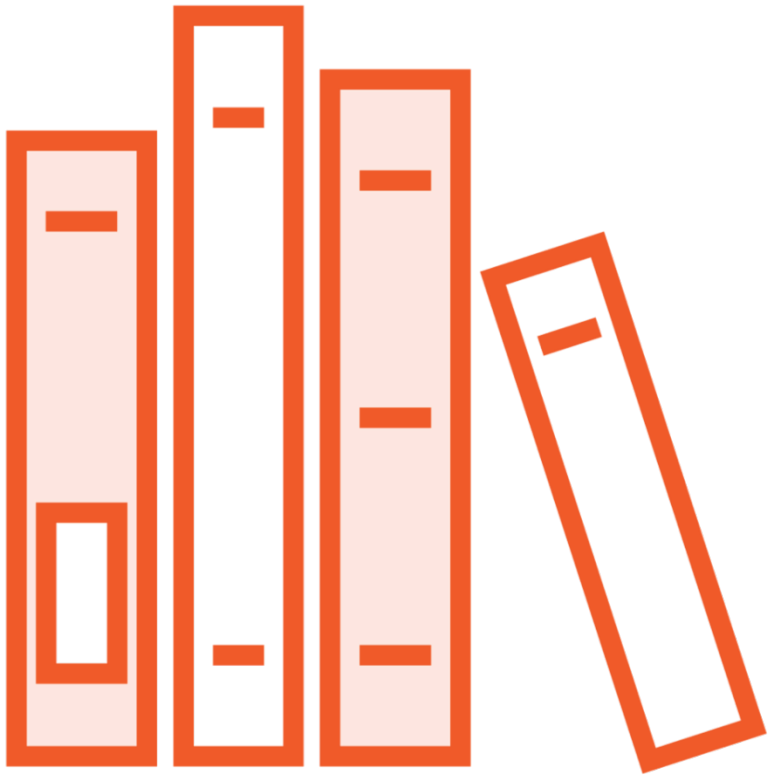
Visual Studio 2019 (Community edition)

- **ASP.NET and web development**

.NET / C#

- **.NET 5.0**
- **C# 9**

Note: Visual Studio Code will also work for running the samples



Additional Resources

https://bit.ly/3tYeAee

https://github.com/jeremybytes/csharp-interfaces-resources

What & Why

Definition

Differences

- Concrete classes
- Abstract classes
- Interfaces

Interfaces and flexible code

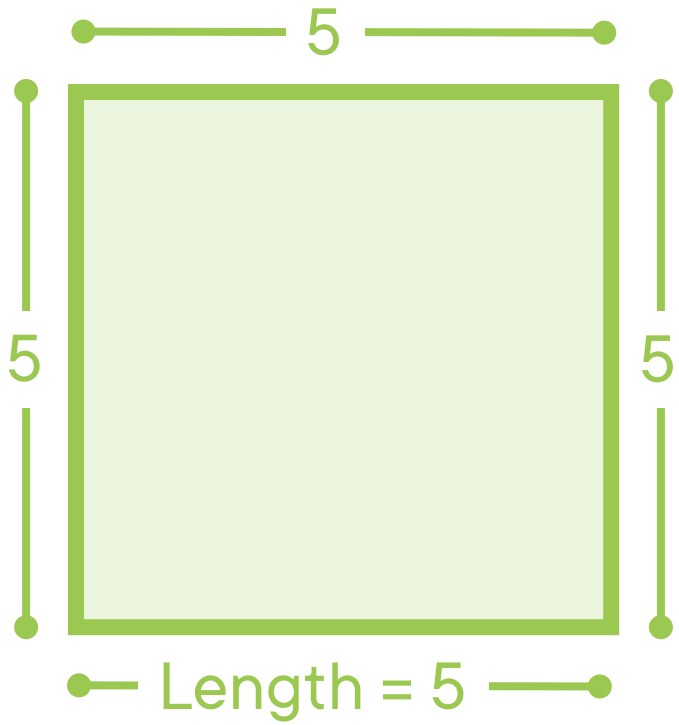
An interface contains definitions for a group of related functionalities that a non-abstract class or struct must implement.

Contract

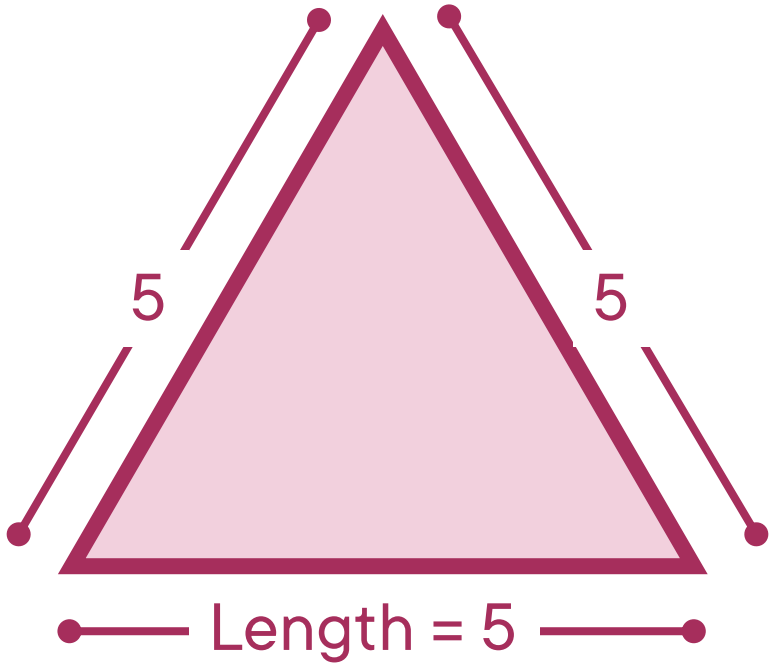
“I have these functions.”

Properties, methods, events, indexers

Regular Polygons

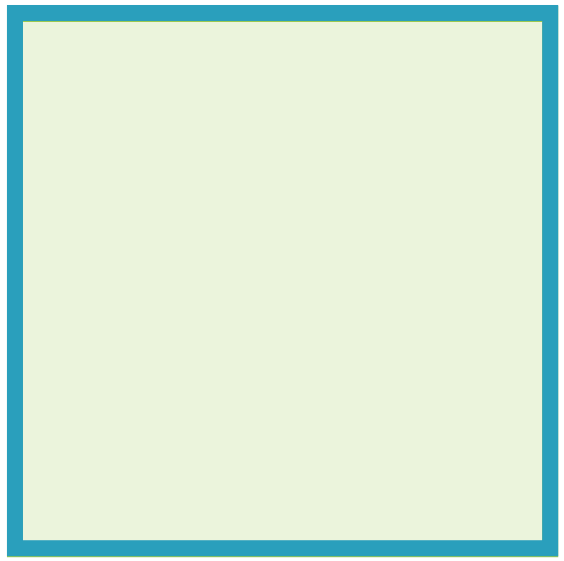


3 or more sides
Sides are same length



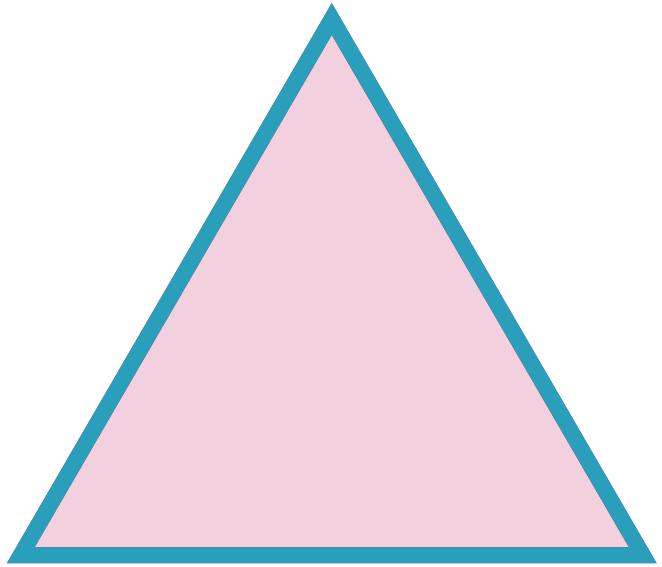
3 or more sides
Sides are same length

Perimeter



Length = 5

perimeter = sides x length

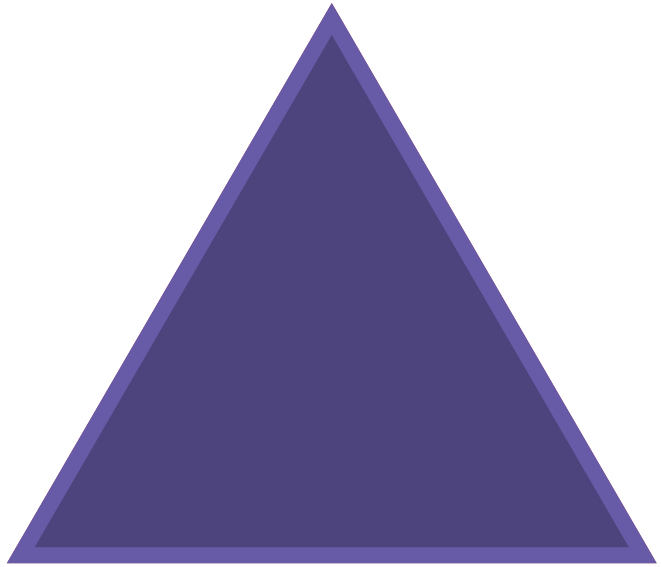


Length = 5

perimeter = sides x length

Area

Length = 5

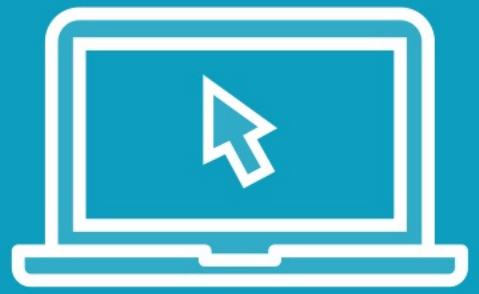


Length = 5

area = length x length

area = length x length x sqrt(3) / 4

Demo



Regular polygon as

- Concrete class
- Abstract class
- Interface

Interfaces and Flexible Code

Resilience in the face of change

Insulation from implementation details

Program to an abstraction rather than a concrete type

Program to an interface rather than a concrete class

Concrete Collection Types

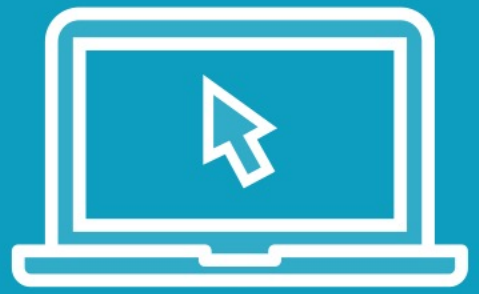
`List<T>`	◂ **Strongly-typed collection of objects**
`Array`	
`ArrayList`	
`SortedList<TKey, TValue>`	
`HashTable`	◂ **Unordered bag of objects**
`Queue / Queue<T>`	◂ **First in / first out collection**
`Stack / Stack<T>`	◂ **Last in / first out collection**
`Dictionary<TKey,TValue>`	
`ObservableCollection<T>`	

List<T> Interfaces

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

- **Allows iteration**
  Used with ‘foreach’

Demo



Code against a class and an interface

Change the type

See how the code responds

What & Why

Definition

Differences

- Concrete classes
- Abstract classes
- Interfaces

Interfaces and flexible code