Interfaces

Agenda

Interfaces

Interface implementation

Implement IComparable

Explicit implementation

The logger project

Interface in software development

Interfaces

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

.NET Documentation

Interfaces

Interface = contract

Interface declaration

- Interface : *“Here are the terms of the contract, you must give a concrete implementation of …”*

Interface implementation

- Type : *“Agreed ! I implement all your members”*

Interfaces

Has public members

All the types that implement an interface must implement all its members

A type can implement multiple interfaces

Interface declaration

```
public interface IMyInterface
{
    // members
}
```

Interface members declaration

```
public interface ILogger
{
    void Log(string message);
}
```

Interface members

Can contain static members

Can contain a default implementation for members

Demo

Create an ILogger interface

Demo

Create an ILogger interface

Implementations

Implementations

Is a type that implements the interface

Interface members must be implemented by concrete types that implement it

Implemented members must be public and non static

Implemented methods must have the same signature

Interface implementation

```
public class TxtLogger : ILogger
{
    public void Log(string message)
    {
        // Log to txt file
    }
}
```

Interface implementation

```
public class JsonLogger : ILogger
{
    public void Log(string message)
    {
        // Log to json file
    }
}
```

Demo

Implement the ILogger interface

Demo

Implement the IComparable interface

Interfaces in software development

Interfaces benefits

Promote polymorphism

Allow code reuse

Establish an abstraction upon concrete classes

Enable loose coupling

Allow dynamic loading

Dynamic loading

```csharp
ILogger logger;
if (condition)
{
    logger = new TxtLogger();
}
else
{
    logger = new JsonLogger();
}
logger.Log("I love C#!");
```

Interface usage

Is a pillar of good unit testing

Allows to write extensible

Largely used in software design (design patterns, clean coding, SOLID principles, dependency injection)

Abstract classes vs interfaces

Abstract classes	Interfaces
Multiple interface implementation	Single class inheritance
Access modifiers	Public members
Structures can’t inherit from abstract classes	Structures can implement an interface
Field member ok	No field member
Constructor ok	No constructor

⚠ Prefer small and consistent interfaces

Demo

Create a logger factory

Interfaces explicit implementation

Standard implementation

Interface members are accessible through the interface type and the concrete type

Can have an access modifier

Implemented interface

```
public interface ILogger
{
    void Log(string message);
}
```

Interface standard implementation

```
ILogger logger = new TxtLogger();
logger.Log("Hello!");
```

Interface standard implementation

```
var logger = new TxtLogger();
logger.Log("Hello!");
```

Explicit implementation

A class can implement an interface member explicitly

Accessible through the interface type

No access modifier

The two methods can be mixed

Solution for a class implementing two members of two interfaces with the same signature

Interface explicit implementation

```
public class TxtLogger : ILogger
{
    void ILogger.Log(string message)
    {
        // Log in txt file
    }
}
```

Interface explicit implementation

```
ILogger logger = new TxtLogger();
logger.Log();
```

Interface explicit implementation

```
TxtLogger logger = new();
logger.Log("Hello!");
```

Interface explicit implementation

```
var logger = new TxtLogger();
((ILogger)logger).Log("Hello!");
```

Demo

Implement interface members explicitly

Call explicitly implemented methods

Implement 2 interface members with the same name

Challenge

Add a layer of abstraction over concrete classes

Reduce code duplication

Work with abstractions in your program instead of concrete classes

2 loggers

`JsonLogger`

`TxtLogger`

Calling program

```
private static TextLogger textLogger = new TextLogger();
private static JsonLogger jsonLogger = new JsonLogger();
```

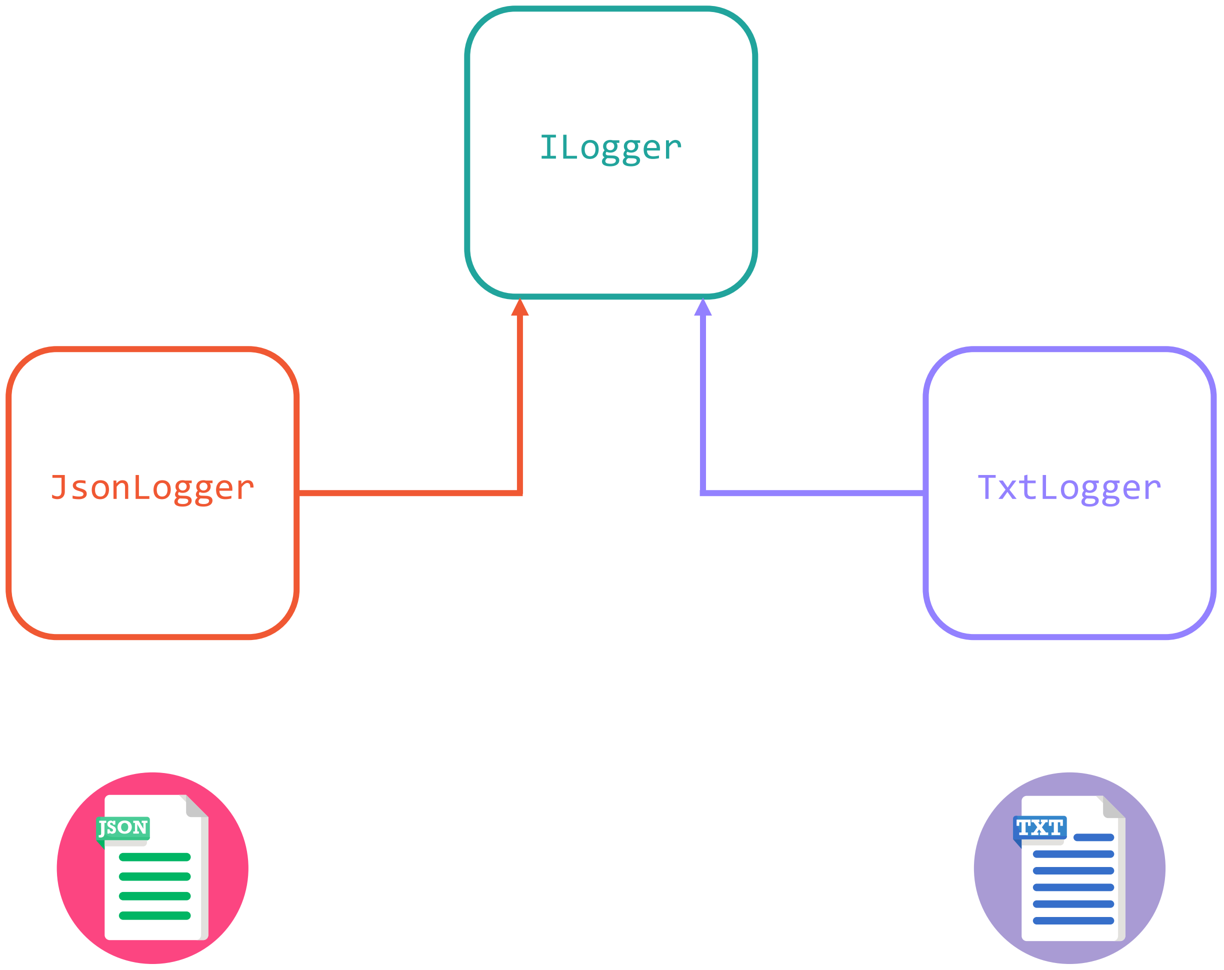
Requirements

Create an interface and/or abstract class

Your interface can have a default implementation for some methods

Work with the created abstractions in your calling program

2 loggers, 1 interface



Summary

An interface is an abstraction defines a contract between a type that will implement the members of the interface and the interface

A type can implement multiple interfaces

Its members are public

Its members can have a default implementation

Interface members can be implemented explicitly, the member can be called through the interface type

Interfaces (and abstract classes) are heavily used in software development and clean software design