**Interface**

An interface contains only the signatures of [methods](https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/classes-and-structs/methods), [properties](https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/classes-and-structs/properties), [events](https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/events/index) or [indexers](https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/indexers/index). A class or struct that implements the interface must implement the members of the interface that are specified in the interface definition. In the following example, class ImplementationClass must implement a method named SampleMethod that has no parameters and returns void.

## **Example**

interface ISampleInterface

{

void SampleMethod();

}

class ImplementationClass : ISampleInterface

{

// Explicit interface member implementation:

void ISampleInterface.SampleMethod()

{

// Method implementation.

}

static void Main()

{

// Declare an interface instance.

ISampleInterface obj = new ImplementationClass();

// Call the member.

obj.SampleMethod();

}

}

An interface is a contract between itself and any class that implements it.

The main benefit is about code readability, code maintainability and code "semantics".

* **Code readability:** An interface constitutes a declaration about intentions. It defines a capability of your class, what your class is capable of doing. If you implement ISortable you're clearly stating that your class can be sorted, same for IRenderable or IConvertible.
* **Code semantics:** By providing interfaces and implementing them you're actively separating concepts in a similar way HTML and CSS does. A class is a concrete implementation of an "object class" some way of representing the reality by modeling general properties of real life objects or concepts. An interface define a behavioral model, a definition of what an object can do. Separating those concepts keeps the semantics of your code more clear. That way some methods may need an instance of an animal class while other may accept whatever object you throw at them as long as it supports "walking".
* **Code maintainability:** Interfaces helps to reduce coupling and therefore allow you to easily interchange implementations for the same concept without the underlying code being affected. You can change the implementation of a IMessage easily by defining a new class that implements the interface. Compare that to sistematically replacing all references from CMessage to CMyNewMessageClass.

**Benefit of Interface:**

The main benefits of interfaces is mostly related to project design.

If you use an interface:

1. The consumer of the interface should implement that interface.
2. Designing bridge patters.
3. Creating a contract so that user must adhere the rules of the interface.
4. Can take only interface part (Object) from the main class.
5. Even class is private, can obtain the interface object from that
6. Multiple inheritance kind of style.
7. Need not be should implement, simple go for if implements that means if you want you can implement other wise can drop it..
8. Cleaner code.