**Interfaces in C# Explained – In-Depth guide on how to use interfaces**

[**https://www.youtube.com/watch?v=VT9ueWBqquU**](https://www.youtube.com/watch?v=VT9ueWBqquU)

A picture containing graphical user interface

Description automatically generated

This allows you to have several different classes that can be used to the same way by using a common interface and you don’t care what else that class is doing.

Well think as Interfaces as a contract. Where a class that implements an interface agrees to provide implementations for all objects defined by that interface.

This means an interface will contain the contract terms, methods and properties, but how we implement them is up to the class that implements the interface.

As long as these interface methods are implemented in our class the interface will remain happy.

Interfaces cannot contain any implementations. That’s important. Their names are generally prefixed with an “I”, to distinguish them from other C# objects.

We create interfaces using the interface keyword.

Interfaces are used between two similar and non-similar classes, which do not care about the type of class implementing the interface.

**Interfaces help with code readability:**

Interfaces constitutes a declaration about intentions. It defines the capability of your class. So what your class is capable of doing.

**Interfaces help with Code semantics:**

By providing interfaces and implementing them, you’re actively separating concepts. An interface defines a behavioral model. A definition of what an object can do. Separating those concepts keeps semantics of your code a lot clearer.

**Interfaces help with code maintainability:**

Interfaces reduces coupling and allow you to easily interchange implementations for the same concept, without the underlying code being affected.

**Interfaces help with Design Patterns:**

It is the bigger picture of using contracts, abstraction, and interfaces are pivotal for object-orientated programming. Human understanding and complex system architecture so it’s really basically understand it better as a human being.

**Interfaces help with Multi-Inheritance:**

Interfaces can be our gateway to use multiple inheritance in C#. So suppose you write a library and want it to be modifiable by users. You write an interface and it’s class implementation. Other developers who will use your library can still write their own implementation class which may use difference technologies or algorithms that achieve the same result. This is also why we meet so many interfaces in libraries we use but rarely feel the need to write our own interfaces because we don’t write libraries ourselves.

On a final note, the important thing is to understand how interfaces are implemented at this point. As we go through different technologies like wpf and asp.net will start to use interfaces more which will help you better understand them better in the future.

An interface contains methods, properties and object that do not contain any implementation.

When a class inherits the interface, the class defines the implementations for the methods within the interface.

A contract is basically formed between the interface and class and the class agrees to use all the objects, methods, and properties that are contained in the interface, but the class can determine their behavior and implementation.

<https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/interface>

https://docs.microsoft.com/en-us/dotnet/csharp/fundamentals/types/interfaces