**What is an Abstract Class?**

An abstract class is a special kind of class that cannot be instantiated. So the question is why we need a class that cannot be instantiated? An abstract class is only to be sub-classed (inherited from). In other words, it only allows other classes to inherit from it but cannot be instantiated. The advantage is that it enforces certain hierarchies for all the subclasses. In simple words, it is a kind of contract that forces all the subclasses to carry on the same hierarchies or standards.

**What is an Interface?**

An interface is not a class. It is an entity that is defined by the word Interface. An interface has no implementation; it only has the signature or in other words, **just the definition of the methods without the body**. As one of the similarities to Abstract class, it is a contract that is used to define hierarchies for all subclasses or it defines specific set of methods and their arguments. The main difference between them is that a class can implement more than one interface but can only inherit from one abstract class. Since C# doesn’t support multiple inheritance, interfaces are used to implement multiple inheritance.

**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 otherwise can drop it..
8. Cleaner code.
9. Implementation which changes depends on class can go ahead with interface.
10. If each class have separate implementation of a method better to go for interfaces. For example IEnumerable in collections.

According to C# Architect, in a simple word it's a contract. Consumer must adhere to it.

* **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.

Interfaces are very useful for:

* Dependency Injection (DI)
* Inversion of Control (IC)
* Test Isolation

An interface is a contract; it's a guarantee that the methods and properties specified will be available. It provides no implementation, which is what makes it different than a class which does provide implementation.

Interfaces are the highest level of abstraction, it provides no implementation details to the consumer.

**Purposes of Interfaces**  
-create loosely coupled software  
-support design by contract (an implementor must provide the entire interface)  
-allow for pluggable software  
-allow different objects to interact easily  
-hide implementation details of classes from each other  
-facilitate reuse of software