# Interface vs Abstract Class

## C# abstract class explained

An abstract class is a special type of class that cannot be instantiated. An abstract class is designed to be inherited by subclasses that either implement or override its methods. In other words, abstract classes are either partially implemented or not implemented at all. You can have functionality in your abstract class—the methods in an abstract class can be both abstract and concrete. An abstract class can have constructors—this is one major difference between an abstract class and an interface. You can take advantage of abstract classes to design components and specify some level of common functionality that must be implemented by derived classes.

## C# interface explained

An interface is basically a contract—it doesn’t have any implementation. An interface can contain only method declarations; it cannot contain method definitions. Nor can you have any member data in an interface. Whereas an abstract class may contain method definitions, fields, and constructors, an interface may only have declarations of events, methods, and properties. Methods declared in an interface must be implemented by the classes that implement the interface. Note that a class can implement more than one interface but extend only one class. The class that implements the interface should implement all its members. Like an abstract class, an interface cannot be instantiated.

| Abstract Class | Interface |
| --- | --- |
| It contains both declaration and definition part. | It contains only a declaration part. |
| Multiple inheritance is not achieved by abstract class. | Multiple inheritance is achieved by interface. |
| It contain constructor. | It does not contain constructor. |
| It can contain static members. | It does not contain static members. |
| It can contain different types of access modifiers like public, private, protected etc. | It only contains public access modifier because everything in the interface is public. |
| The performance of an abstract class is fast. | The performance of interface is slow because it requires time to search actual method in the corresponding class. |
| It is used to implement the core identity of class. | It is used to implement peripheral abilities of class. |
| A class can only use one abstract class. | A class can use multiple interface. |
| If many implementations are of the same kind and use common behavior, then it is superior to use abstract class. | If many implementations only share methods, then it is superior to use Interface. |
| Abstract class can contain methods, fields, constants, etc. | Interface can only contains methods, properties, indexers, events. |
| It can be fully, partially or not implemented. | It should be fully implemented. |