abstract (C# Reference)

The **abstract** modifier indicates that the thing being modified has a missing or incomplete implementation. The abstract modifier can be used with classes, methods, properties, indexers, and events. Use the **abstract** modifier in a class declaration to indicate that a class is intended only to be a base class of other classes. Members marked as abstract, or included in an abstract class, must be implemented by classes that derive from the abstract class.

Abstract classes have the following features:

- An abstract class cannot be instantiated.
- An abstract class may contain abstract methods and accessors.
- It is not possible to modify an abstract class with the sealed (C# Reference) modifier because the two modifers have opposite meanings. The **sealed** modifier prevents a class from being inherited and the **abstract** modifier requires a class to be inherited.
- A non-abstract class derived from an abstract class must include actual implementations of all inherited abstract methods and accessors

virtual (C# Reference)

The **virtual** keyword is used to modify a method, property, indexer, or event declaration and allow for it to be overridden in a derived class.

The implementation of a virtual member can be changed by an overriding member in a derived class.

override (C# Reference)

The **override** modifier is required to extend or modify the abstract or virtual implementation of an inherited method, property, indexer, or event.

An **override** method provides a new implementation of a member that is inherited from a base class. The method that is overridden by an **override** declaration is known as the overridden base method. The overridden base method must have the same signature as the **override** method.

You cannot override a non-virtual or static method. The overridden base method must be **virtual**, **abstract**, or **override**.
An **override** declaration cannot change the accessibility of the **virtual** method. Both the **override** method and the **virtual** method must have the same access level modifier.

sealed (C# Reference)

When applied to a class, the **sealed** modifier prevents other classes from inheriting from it.

It is an error to use the abstract modifier with a sealed class, because an abstract class must be inherited by a class that provides an implementation of the abstract methods or properties.