You use the const keyword to declare a constant field or a local constant. Constant fields and locals aren't variables and can't be modified. Constants can be numbers, Boolean values, strings, or a null reference

The **[readonly](https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/readonly)** keyword differs from the const keyword. A const field can only be initialized at the declaration of the field. A readonly field can be initialized either at the declaration or in a constructor. Therefore, readonly fields can have different values depending on the constructor used. Also, although a const field is a compile-time constant, the readonly field can be used for run-time constants, as in this line: public static readonly uint l1 = (uint)DateTime.Now.Ticks;

The yellow highlighted example is really good to demonstrate where can we use readonly fields.  
  
Also check in above the field is defined as static, so it’s a class level field and doesnot depend on instance of the class:  
for example:  
using System;

class MyClass

{

public static readonly uint l1 = (uint)DateTime.Now.Ticks;

}

class Program

{

static void Main()

{

// Accessing the static field 'l1' without creating an instance Console.WriteLine(MyClass.l1);

}

}  
  
  
Sealed:  
When applied to a class, the sealed modifier prevents other classes from inheriting from it. In the following example, class B inherits from class A, but no class can inherit from class B.

C#Copy

class A {}

sealed class B : A {}

You can also use the sealed modifier on a method or property that overrides a virtual method or property in a base class. This enables you to allow classes to derive from your class and prevent them from overriding specific virtual methods or properties.