Indexer:

C# indexers are usually known as smart arrays. A C# indexer is a class property that allows you to access a member variable of a class or struct using the features of an array. In C#, indexers are created using this keyword. Indexer can be overloaded. Indexer is an instance member so can't be static but property can be static.

Indexer vs Property

An indexer allows an instance of a class or struct to be indexed as an array.  If the user will define an indexer for a class, then the class will behave like a virtual array. Indexers are also known as the Smart Arrays or Parameterized Property in C#. Indexer can’t be a static member as it is an instance member of the class

Static:

The main features of a static class

* Contains only static members.
* Cannot be instantiated.
* Is sealed.
* Cannot contain Instance Constructor

static constructor called before the class is referenced for the first time in your program. A static constructor is only called one time, and a static class remains in memory for the lifetime of the application domain in which your program resides.

Static methods can be overloaded but not overridden, because they belong to the class, and not to any instance of the class.

Static class will have everything static

Static can access only static within class( by method name, field name), for any non static with in class it can accessed only through class object

Static can access static outside by their class name

For Inheritance base class has to be accessible(public or Protected)

For Object creation of class A in another class B, Class A has to be accessible.

If there is class inside a class : and method u are calling is non static

Example:

LearnOverloadIndexer.OverloadIndexer lo = new LearnOverloadIndexer.OverloadIndexer();

lo.OverloadIndexerMain();

if static:

LearnOverloadIndexer.OverloadIndexer.OverloadIndexerMain();

Delegate:

Delegate is a pointer to a method, which stores location of method.

Delegate vs Interface

A [**Delegate**](https://www.geeksforgeeks.org/c-sharp-delegates/) is an object which refers to a method or you can say it is a reference type variable that can hold a reference to the methods. It provides a way which tells which method is to be called when an event is triggered. It created at run time. It does not support inheritance. It can access anonymous methods.

Delegates allow methods to be passed as parameters. Delegates can be chained together; for example, multiple methods can be called on a single event.

Reflection

Reflection in C# is used to retrieve metadata on types at runtime. In other words, you can use reflection to inspect metadata of the types in your program dynamically -- you can retrieve information on the loaded assemblies and the types defined in them.