**Delegate in C#**

A delegate is like a pointer or reference to a method or function. It allows you to call that method indirectly through the delegate.

There are two main types of delegates in C#:

**Single-Cast Delegate**: This refers to a delegate that points to a single method or function.

**Multicast Delegate**: Unlike Single-Cast, a Multicast Delegate can point to and call multiple methods or functions.

**Key Points about Multicast Delegates**:

**Holding Multiple Methods**: Multicast Delegates can hold references to multiple methods.

**Invocation Order**: When you call a Multicast Delegate, it executes each referenced method in the order they were added.

**Adding and Removing Methods**: You can add methods to a Multicast Delegate using the + or += operator. Conversely, you can remove methods using the - or -= operator.

**Invocation Principle**: Methods within a Multicast Delegate are called following the FIFO (First In, First Out) principle. This means the methods are executed in the order they were added.

**Matching Method Signatures:** For a Multicast Delegate to call multiple methods, all those methods must have matching signatures (return type and parameters).

Imagine a delegate as a container holding references to different methods. When you invoke (call) the delegate, it triggers each method it holds, executing them one after another in the specified order.

For example, if you have a Multicast Delegate myDelegate, you can add multiple methods to it using myDelegate += SomeMethod; and then call myDelegate(); to execute all those methods one by one.

Multicast Delegates help streamline execution by allowing multiple methods to be called through a single delegate, following a predefined order and specific criteria for method signatures.