Delegates

- Delegates are similar to function pointers in c++
- It is a mechanism by which methods can be passed as method parameters instead of data.
- But unlike C's function pointer, delegates are type-safe and object oriented.
- Delegates are used to implement call-backs.
- A method can call another method through the delegate that is passed to it . This is known as an asynchronous callback.
- Like class and interface it is also a type and its references are also created and instantiated.
- A Delegate is an object that refer to a method.
- Delegates are general-purpose mechanism for indirectly calling methods at runtime.
- When we call a Delegate all the methods associate with the Delegate object will executes.

Delegates

- To implement delegate in our application we need to declare delegates, instantiate delegates and call delegates.
- We can declare delegates by using *delegate* keyword.
  - Declaration:

  access-modifier delegate return-type delegate-name(parameter-list);

  Ex: delegate void Mydelegate(string s);
  - Instantiation

  delegate-name object-name=new delegate-name(method-name)

  Or

  delegate-name object-name=method-name
  - Invocation

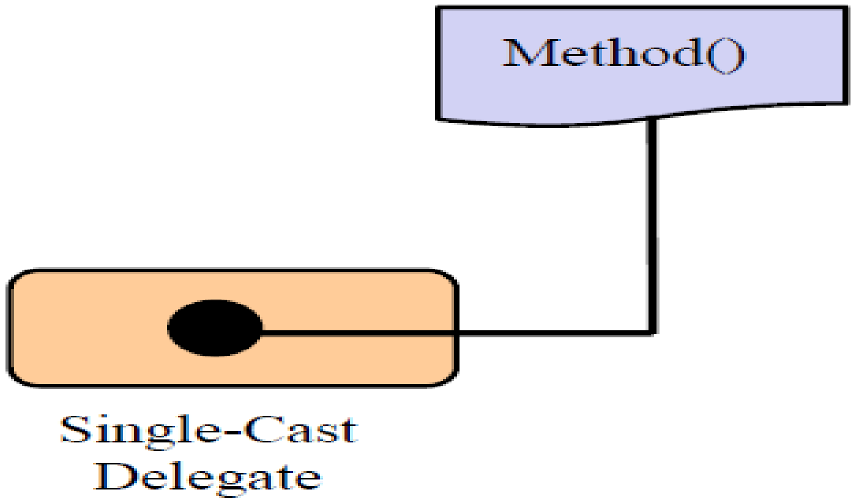
  object-name();
- Note: The delegate can refer to the methods, which have the same signature and return type of the delegate .

Types of Delegates

- Delegates are two types.
  - Single- cast Delegate
  - Multi cast Delegate

Types of Delegates[Contd.]

- Single cast Delegate:
- Single-cast delegate refer to one method at a time.
- A single-cast delegate derived from System.Delegate class



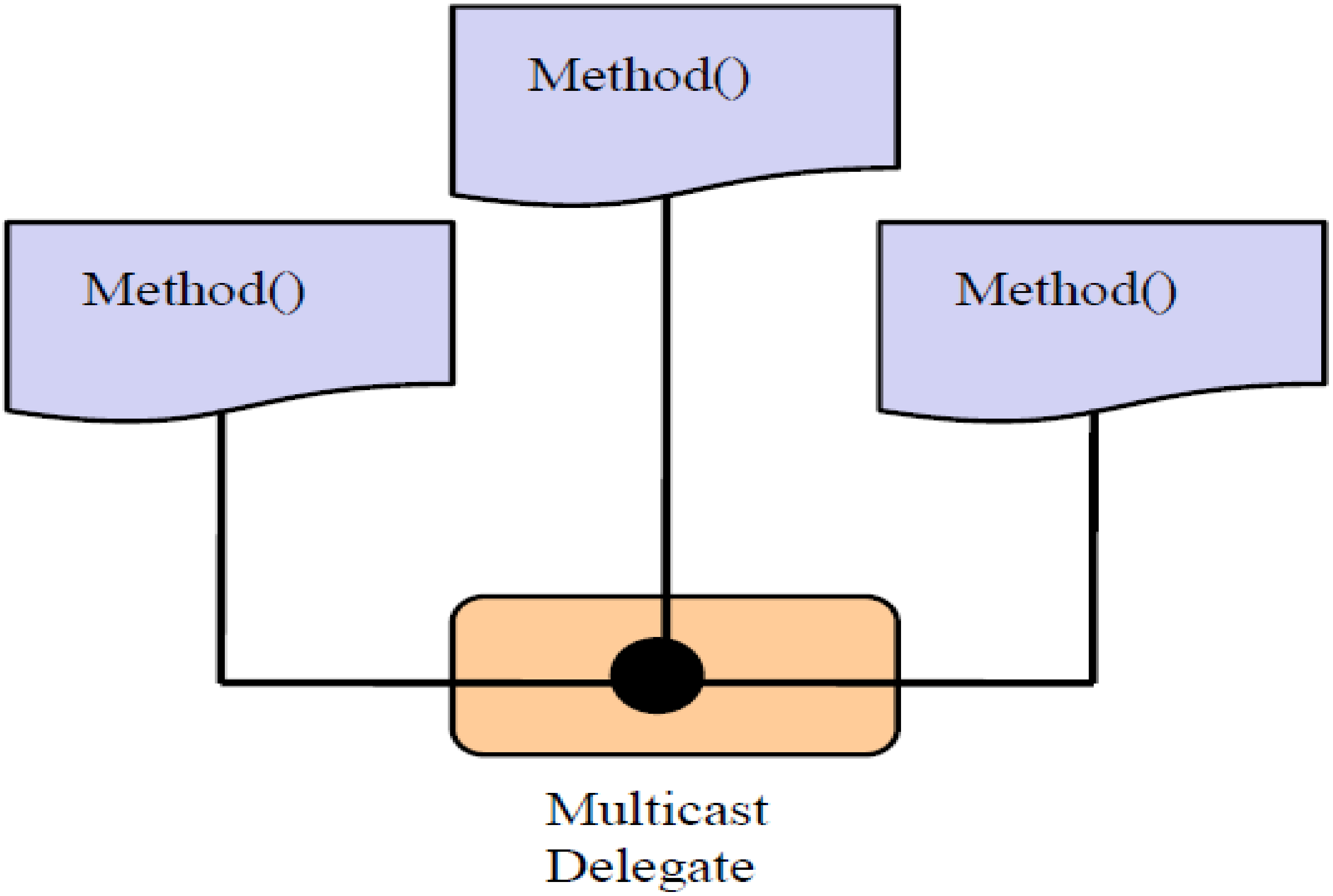
Multicast Delegate

- A Multicast delegate can invoke multiple methods at same time.
- A Multicast delegate refers from the System.MulticastDelegate Class
- When a multicast delegate is called ,it executes all the methods it wraps in the calling order.
- The methods called by the multicast delegate should not have a return value.

Multicast Delegate

- add a method to the delegate object , you simply make use of the overloaded += operator.
- remove a method from the delegate object you make use of the overloaded operator -=
- multicast delegates calls a sequence of methods in the specified order.
- If one of the methods in the sequence throws an exception, the iteration stops there!

Multicast Delegate



Delegates

- why delegates:
  - Delegates are type-safe
  - In Event handling mechanism to know which method to call when the event occurs.
  - In multithreaded programming to supply the starting point of the thread execution.

Anonymous Methods

- Anonymous method is a new feature added in C# 2.0
- An anonymous method is an unnamed block of code that is used as parameter for the delegate.

```
    Public  delegate void Mydelegate();
    Mydelegate d=delegate
        {
        ---------------------
        -------------------
        };
d()
```

Anonymous code

Lambda expression

- A lambda expression is an anonymous function that can be used to create delegates or expression tree types.
- => is the lambda operator, which is read as "goes to".
- many LINQ expressions can be written using Lambda expression.
- It is very easy to use aggregate functions with lambda expression.

Lambda expression example

```
using System;
class X
{
    delegate int cube(int i);
    static void Main(string[] args)
    {
        cube myDelegate = x => x*x*x;
        int j = myDelegate(5);
        Console.Write(j);
    }
}

//Prints 125
```