**Session 16: DelegatesAndEvents**

**Delegates: -** it’s an object which refers to method. Its reference type that can hold reference of method. Return type and parameter should be same. It is also called function pointer. Delegate keyword is used. Point to parameterised and non-parameterised method. No implementation (Body). Used for encapsulate method. Type safe pointer for any method. Delegates will be called using Invoke method. It decides at runtime. It contain general name for delegate. Increase the performance.

* **Types of Delegate: Single Cast Delegate:-**  point to single method. Cannot pass more method reference. Derived from system. Delegate Class.
* **Multiple Delegates:-** calling multiple methods using delegate. Depends on name, parameter, and type.
* **Multicast Delegate:-** its extension of normal delegates. It helps to point more than one method in single call. Derived from system.MulticastDelegate class. We can use += (Addion of method)and -= (Remove Method)operator to implement multicast delegates.

**Events:-**  its encapsulation over delegates. Just like properties encapsulate private field events encapsulate delegate. When delegate fire automatically then we use events. Delegate enable publisher subscriber pattern where delegate object is publisher and target method is subscriber. Using event we can formalise. All class subsriced to events notify at once.

S16\_\_Delegates.cs

using System;

using System.Collections.Generic;

using System.Reactive;

using System.Text;

namespace OOPS\_\_AllSession

{

public delegate void PrintNumbers();

public delegate void PrintNames(string[] names);

class S16\_\_Delegates

{

public static void PrintPositiveNumbers()

{

Console.Write("\n\nHow Many Numbers you want: ");

int userArray = Convert.ToInt32(Console.ReadLine());

int[] arrays = new int[userArray];

for (int i = 0; i < userArray; i++)

{

Console.Write("Enter Data Value: ");

int finalyArray = Convert.ToInt32(Console.ReadLine());

arrays[i] = finalyArray;

}

Console.WriteLine("\n\*\*\*\*\*\* Your Final Array is: \*\*\*\*\*\*\*\*\*\*\*\*");

Console.Write("Final Array is: ");

foreach (int data in arrays)

Console.Write(data + " ");

}

public void PrintCityName(string[] city)

{

Console.Write($"Length of Array is : {city.Length} \n\nCities Are: \t");

foreach (string cities in city)

Console.Write(cities + ", ");

Console.WriteLine();

}

public static void PrintPersonName(string[] name)

{

Console.Write($"Length of Array is : {name.Length} \nNames Are: \t");

foreach (string names in name)

Console.Write(names + ", ");

Console.WriteLine();

}

public static void PrintName(string[] fullName)

{

Console.Write($"Length of Array is : {fullName.Length} \nFull Name is: ");

foreach (string name in fullName)

Console.Write(name + ", ");

Console.WriteLine();

}

}

//Events

class EventNotifications

{

public event PrintNumbers notificationEvent;

public void NotificationMethod()

{

if (notificationEvent != null)

notificationEvent();

}

}

class User1

{

public static void User1\_Message()

{

Console.WriteLine("Hello This is Sample Message For User 1...");

Console.ReadLine();

}

}

class User2

{

public static void User2\_Message()

{

Console.WriteLine("Hello This is Sample Message For User 2...");

}

}

}

OopsSessions.cs

using OOPS\_\_AllSession;

using System;

using static OOPS\_\_AllSession.S11\_\_ClassAndTypes;

namespace Oops\_\_AllSession

{

class OopsSessions

{

static void Main(string[] args)

{

Console.WriteLine("\*\*\*\*\*\*\*\*Welcome To Main Method\*\*\*\*\*\*\*\*\*\*\*");

//S16\_\_Delegates

S16\_\_Delegates delegates = new S16\_\_Delegates();

//Single

PrintNumbers positiveNumber = new PrintNumbers(S16\_\_Delegates.PrintPositiveNumbers); //Single

positiveNumber.Invoke();

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*EVENTS \* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//EventNotifications eventNotify = new EventNotifications();

//eventNotify.notificationEvent += User1.User1\_Message;

//eventNotify.notificationEvent += User2.User2\_Message;

//eventNotify.NotificationMethod();

// Multiple 1

string[] city = { "Panvel", "Pune", "Nanded", "Latur", "Beed", "Mumbai", "Dubai" };

PrintNames printCity = new PrintNames(delegates.PrintCityName);

printCity.Invoke(city);

// Multiple 1

//Adding Refrence to Method or Pointing to method

string[] names = { "Amit", "Amir", "Sahiba", "Imran", "Kanak", "Prakash", "Sayali", "Akhil", "Shubham" };

printCity = S16\_\_Delegates.PrintPersonName;

printCity.Invoke(names);

//Multicast

string[] fullName = { "ImranShaikh" };

printCity += S16\_\_Delegates.PrintName;

printCity.Invoke(fullName);

}

}

}