**Session 14: ExtensionMethodAndInterface**

**Methods / Function:-** its block of code. Only run when it calls. It contain return type.parameter non parameter. Used to perform certain action. Reusing code. ( ) used to define method. Method can be static non static, parameterise non parameterise.

* **Extension Method:-** it allows u to add new method in class. Without modifying code. Deriving or recompiling th original class ,struct or,interface. Its always define as static method. Dosnt support overriding. Add methods in sealed class using extension method. This keyword is used to bind. Once they bind then this will become none static method. Same method is present in static class and extension method then extension method cant be called. Biniding parameter will ignore always.
  + PartialClass(this S14\_\_ExtensionMethodAndInterface ExtensionMethods >> Binding Parameter)
* **Interface:-** interface is a blueprint of class. Its contract between class and interface. Multiple inheritance possible using interface. It define propertis, method,and events which are members of interface. It dosent contain method body only contain method definition. **Bydefault members are public.** Members are abstract. Public modifier no need to put. : symbol is used to access. Define **what Part how part will define by derived class.** We cannt use abstract keyword. Cannot instantiate (not create object). Inherit by class or other interface. Interface Keyword will be use.

S14\_\_ExtensionMethodAndInterface.cs

using System;

using System.Collections.Generic;

using System.Text;

namespace OOPS\_\_AllSession

{

class S14\_\_ExtensionMethodAndInterface

{

public void DisplayCity()

{

string[] city = { "Pune", "AbuDabi", "Mumbai", "Dubai", "Jermany", "America", "Nashik", "Amrawati", "London", "UK", "US", "Japan", };

Console.Write("\nCities Are: ");

foreach (string cities in city)

Console.Write(cities);

}

public void MultiDimensionalArray()

{

int[][] numericalValues = new int[3][];

numericalValues[0] = new[] { 12, 24, 36, 48, 99, 88, 77 };

numericalValues[1] = new[] { 60, 72, 84, 96, 00 };

numericalValues[2] = new[] { 10, 20, 30, 44 };

Console.WriteLine("\n\*\*\*\*\*\*\*\* ForeachLoop \*\*\*\*\*\*\*\*\*\*");

foreach (int[] number in numericalValues)

{

foreach (int i in number)

{

Console.Write(i + " ");

}

Console.WriteLine();

}

Console.WriteLine($"\nValue of JaggedArray is: {numericalValues[1][2]}");

}

public void PrintMonth()

{

string[] month = { "Jan", "Feb", "March", "April", "May", "June", "July", "August", "Sepetember", "Octobor", "November", "December" };

Console.Write("\nMonths Are: \n");

foreach (string months in month)

Console.Write($"{months}"+" " + "\n");

}

}

static class Extension\_Methods

{

static string firstName { get; set; }

static int age { get; set; }

static char gender { get; set; }

static long phoneNumber { get; set; }

enum Months { Jan, Feb, March, April, May, June, July, August, Sepetember, Octobor, November, December };

public static void StudentDetails(this S14\_\_ExtensionMethodAndInterface extensionMethod)

{

Console.Write("Enter Student Name: ");

firstName = Console.ReadLine();

Console.Write("Enter Student Age: ");

age = Convert.ToInt32(Console.ReadLine());

Console.Write("Enter Student Gender: ");

gender = Convert.ToChar(Console.ReadLine());

Console.Write("Enter Student PhoneNumber: ");

phoneNumber = Convert.ToInt64(Console.ReadLine());

Console.WriteLine($"\n\*\*\*\*\*\*\*\*Student Detail Are:\*\*\*\*\*\*\*\*\*\*\n Student Name : {firstName}\n Student Age: {age}\n Student Gender: {gender}\n Student Phone: {phoneNumber}");

}

public static void PrintMonth(this S14\_\_ExtensionMethodAndInterface extensionMethods)

{

foreach (int month in Enum.GetValues(typeof(Months)))

{

Console.Write($"{Enum.GetName(typeof(Months), month)}");

Console.WriteLine($" {month}");

}

}

//Binding With Struct

public static int[] PrintArrayNumber(this int[] num)

{

int[] numbers = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };

Console.Write("\nNumbers Are: " +"\n");

foreach (int number in numbers)

Console.Write($" {number}" +" ");

return numbers;

}

interface ISampleData

{

void MarkChecker();

}

class SampleData\_Interface

{

public void MarkChecker() //S5

{

//IF...IF Then IF...else if

Console.Write("Enter Percentage value \t");

int percentage = int.Parse(Console.ReadLine());

if (percentage >= 80)

Console.WriteLine("Congrats! You Pass the Exam");

// Console.WriteLine("Wish You best Luck"); // Error

else if (percentage >= 60)

Console.WriteLine("Good! You Are Eligible For Commerce");

else if (percentage >= 50)

Console.WriteLine("All The Best");

else if (percentage < 35)

Console.WriteLine("Failed");

else

Console.WriteLine("Sorry! \tTry Next Time");

}

}

}

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\*\*\*\*\*\*\*\*\*\*\*");

//S14\_\_Extension Method And Interface

S14\_\_ExtensionMethodAndInterface extensionMethod = new S14\_\_ExtensionMethodAndInterface();

extensionMethod.StudentDetails();

extensionMethod.PrintMonth();

int[] number = { };

number.PrintArrayNumber();

//extensionMethod.MultiDimensionalArray();

} }}