**Session 13: MethodsAndTypesOfMethodProgram.**

**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.

**Return :** return keyword is used to return value to called method.

* **Types: Parameterised Method :** the method contain some parameter in it.at the time of calling we need to pass **argument** for that method. Variables passed to method called **parameter**.
  + **Eg:- public void Method(string name= “Imran”, int age) >> Optional Parameter** 
    - **Objname.Method(age:33,name:“Imran”)**
  + **Named Argument:** while passing the argument in method then name is specify.
* **Non Parameterised Method:** not contain any type of method.
* **Non Static Method :** the method which is define without static keyword. Object is required to call the method.
* **Abstract Method:** the method which will be used abstract keyword. And it only written in abstract class.
* **Static Method:** the method which is created using static keyword whenever want to call the call can be done using ClassName.MethodName.
* **Sealed Method:** used in method **overriding**. When derived class override base class method,variable,property,event etc.. Can be declared as sealed.
* **Method overloading:-**having multiple definition with same function name in scope(class body). The name should be same but parameter different. Cannot overload function declaration that differ only by return type. **Static Polymorphism: Compile Time** Polymorphism
* **Method Overriding**:- it allows us to have virtual and abstract methods in base using derived class with same name and same parameter. Method name same but implementation could be different. **Dynamic Polymorphism RunTime.**
  + **3types of keyword is used: - Virtual:** use within base class method. Its used to modify method in base class for overriding method.;
  + **Override:** used with derived class. Used to modify virtual or abstract method into derived class which present in base class.
  + **Base:** used to call parent(base) Method from Derrived(Child) Class.

**MethodOverLoading methodOverload\_Override in= new MethodOverRiding();**

Access child method.

S13\_\_MethodsAndTypes\_MthdOverlod\_Override.cs

using System;

using System.Collections.Generic;

using System.Text;

namespace OOPS\_\_AllSession

{

class S13\_\_MethodsAndTypes\_MthdOverlod\_Override

{

//Private Method\*\*\*\*\*\*\*\*\*\*\*\*

void Addition()

{

int num1, num2;

Console.Write("Enter Number 1: ");

num1 = int.Parse(Console.ReadLine());

Console.Write("Enter Number 2: ");

num2 = int.Parse(Console.ReadLine());

Console.Write($"Addition is: {num1 + num2}");

}

//Return Type Int With PArameterise --------------

public int Substraction(int num1, int num2)

{

int sub = num1 - num2;

Console.WriteLine("\n\nAddition Method Called\*\*\*\*\*\*\*\*\*\*\*\*");

this.Addition();

Console.WriteLine("\nReturn Value " + sub);

return sub;

}

//Non Static With PArameterised------------

public void StringManupulation(string name = "Imran")

{

Console.Write("\nName is: " + name);

S13\_\_MethodsAndTypes\_MthdOverlod\_Override.PrinitingName("Pune", 23);

S13\_\_MethodsAndTypes\_MthdOverlod\_Override.PrinitingName(age: 28, address: "Address"); //Named Argument

// S13\_\_MethodsAndTypes\_MthdOverlod\_Override.PrinitingName(23,"Address"); Error

}

//Static Method------------

public static void PrinitingName(string address, int age)

{

Console.Write("\n\nAddress Is: " + address + "\nAge is: " + age);

}

}

//Abstract Method without abstract class you cannot create abstract method

//public class BasicDetails // Base Class

//{

// public string firstName;

// public int age { get; set; }

// public char gender;

// public long phoneNumber;

// public abstract void AbstractClassAndDisplayDetail()

// {

// }

// // Abstract Method

//}

//Method Overloading And Overriding

class MethodOverLoading

{

public void MethodOverloading()

{

MethodOverLoading methodOverloading = new MethodOverLoading();

methodOverloading.Addition();

methodOverloading.Addition(30, 30);

}

public void Addition()

{

int num1 = 40, num2 = 40;

Console.WriteLine($"Addition is For Non Parameter Method: {num1 + num2}");

}

public void Addition(int num1, int num2)

{

Console.WriteLine($"Addition is For Parameter Int Method: {num1 + num2}");

}

public virtual void Addition\_MethodOverride() /// Method Overriding

{

Console.Write("Enter 1st No: ");

int no1 = int.Parse(Console.ReadLine());

Console.Write("Enter 2nd No: ");

int no2 = int.Parse(Console.ReadLine());

Console.Write($"Addition is : {no1 + no2} \n\n");

}

}

class MethodOverRiding : MethodOverLoading

{

public override void Addition\_MethodOverride() /// Method Overriding

{

Console.Write("Enter 1st No: ");

int no3 = int.Parse(Console.ReadLine());

Console.Write("Enter 2nd No: ");

int no4 = int.Parse(Console.ReadLine());

Console.Write($"Multiplication is : {no3 \* no4}\n\n");

Console.WriteLine("Calling Parent Method");

base.Addition\_MethodOverride();

}

}

}

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

//S13\_\_MethodAnd Types

S13\_\_MethodsAndTypes\_MthdOverlod\_Override methods = new S13\_\_MethodsAndTypes\_MthdOverlod\_Override();

//methods.StringManupulation();

//methods.Substraction(60,30);

MethodOverLoading methodOverload\_Override = new MethodOverRiding();

methodOverload\_Override.Addition\_MethodOverride();

}

}

}