**Session 8: Call by Value (Pass By Value) OR Call by Refrence.**

Function can be invoked(call) in two ways. And its used with method OR Function.

**Value Types:** allocate value in stack(LIFO). Integer and floating point based. Stack Memory. Less memory. Cannot change value in another instance. It create block again and again. Struct is value type. Cannot contain null value.Compile Time Memory Allocate.

a

10

Name

name

Heap

**Stack int a=10;**

**String name= “Imran”;**

**Reference Type: -** store value in heap: handle big value. it will allocate whenever it require. dosnt contain the actual stored value data stored in variable but they are havning refrence only. If the one variable change it will reflect automatically to other.more memory. Value can be change at the same time. Class is refrence type. Run Time Memory Allocate.

* **Call By (Pass By ) Value:** pass value of variable in method as parameter. Each variable in calling function copied into dummy varibales of called function. Changes cannot be impact on the actual values. We cannot alter values for actual variable. Bydefault all object pass by value.
* **Call By (Pass By) Reference:** - pass the address of variable (location of varibale). Address of actual variable in calling function are copied into dummy variable of called function. Having access to the actual variable so easy to manipulate. Can alter the values of actual variable. Pointer variable are used to store address of value. Ref keyword is used to pass refrence.
* **Out Keyword :-** out keyword is used to pass the argument to the method as refrence type and value type. Its used when method return multiple values. Its not pass the **properties.** Its work as refrence type.

S8\_\_CallBy\_ValuesAndRefrenceProgram.cs

using System;

using System.Collections.Generic;

using System.Text;

namespace AllSession

{

class S8\_\_CallBy\_ValuesAndRefrenceProgram

{

//Call By Value

public void PassByValue\_Addition(int num)

{

num = num + 10;

Console.WriteLine("Value is:" + num);

}

public void PassByValue()

{

int numMul = 20;

PassByValue\_Addition(numMul);

Console.WriteLine("Inside PasssByValue " + numMul +"\n\n");

}

//Refrence Type

public void PassByRefrence\_Substraction(ref int value)

{

value = value - 10;

Console.WriteLine("Substraction value is : "+value);

}

public void PassByRefrence()

{

int numDiv = 30;

PassByRefrence\_Substraction(ref numDiv);

Console.Write("Inside PasssByRefrecne: "+ numDiv+"\n");

}

//PassBy Out : Out Keyword

public void PassByOut\_Devide(out int value)

{

value = 60;

Console.WriteLine("\nOutValue is : " + value);

}

public void PassByOut()

{

int numDivide;

PassByOut\_Devide(out numDivide);

Console.Write("Inside PasssByOut: " + numDivide + "\n");

}

}

}

Program.cs

using AllSession;

using Session1.nestedNamespace; // Assembly Refrence OR Namespace OR PAckage

using System;

namespace Session1

{

class Program : S4\_AccessModifiereAndKeyword

{

static void Main(string[] args)

{

//Session1 8 Call by Value And Call By Refrence

S8\_\_CallBy\_ValuesAndRefrenceProgram callBy = new S8\_\_CallBy\_ValuesAndRefrenceProgram();

callBy.PassByValue();

callBy.PassByRefrence();

callBy.PassByOut();

}

}

}