**Section01 C# Top level Program**

**Notes:-**

**1-in the normal way in order to call method in any namespace, we have to using the namespace and then in call it we have to call class name. Method name**

**//we have to using the namespace**

**using System;**

**namespace CSharp9Pro{**

**class Program{**

**static void Main(string[] args){**

**//we have to call class name. Method name**

**Console.WriteLine("Hello World!");}}}**

**2-C# 9.0 introduce Top Level program which enable us to using static call of namespace name. Class name and call the method directly as below**

**//we using top level program which is using static namespace. Class name**

**using static System.Console;**

**namespace CSharp9Pro{**

**class Program{**

**static void Main(string[] args){**

**//we call the method directly as below**

**WriteLine("Hello World!");**

**var person = new Person() { FullName = "Mohammed Nabil Enbeh" };**

**WriteLine(person.FullName);}}**

**public class Person{public string FullName { get; set; }}}**

**Lesson02 Record Types**

**Notes:-**

**1-Record types is an easy way to define reference type that is immutable which means that each time you change value it will redirect to the new instance**

**2-Record types internally generate class with read-only properties**



**3-Record Type generated functionality**

**A-Value based equality**

**B-Get Hash Code**

**C-Copy and Clone**

**D-Print Member and string**

**E-Deconstruct : which deconstruct record into tuple**

**Example:-**

**using Microsoft.AspNetCore.Mvc;**

**using Microsoft.Extensions.Logging;**

**using System;**

**using System.Collections.Generic;**

**using System.Linq;**

**using System.Threading.Tasks;**

**namespace CSharp9API.Controllers{**

**[ApiController]**

**[Route("api/[controller]")]**

**public class EmployeesController : ControllerBase{**

**[HttpGet]**

**public ActionResult CheckUser(){**

**var ps1 = new Person("Ali Ahmad", 12);**

**var ps2 = new Person("Ali Ahmad ", 12);**

**//we can make copy of the instance with new variable by using with command**

**var ps3 = ps1 with { fullName = "Ali Ahmad" };**

**//because record is reference type each variable reference to separate instance**

**(result = false)**

**var isEqual01 = ps1 == ps3;**

**//the equal means that it equal to properties its belong so the result is equal true**

**var isEqual = ps1.Equals(ps2);**

**var details = ps1.GetDetails();**

**//we can create instance that inherit from the parent record but the equal is false becosue //the type is equal false**

**var dev1 = new Developer("Ali Ahmad", 12, 1200);**

**var isEqual02 = ps1 == dev1;**

**return Ok(new { Result = isEqual, Details = details, IsEqual01 = isEqual01, IsEqualType02 = isEqual02, ps1Result = ps1.ToString(), ps2Result = ps2.ToString(),shapeEqual = shapeEqual });}}**

**//record type is value type which is check for the type is equal or not , so the Person and developer is different type , we can implement custom method as below**

**public record Person(string fullName, int age){**

**public string GetDetails(){return $"fullName {fullName} , age : {age}";}}**

**//we can apply inheritance and set its specify property as below**

**public record Developer : Person{**

**public double Salary { get; }**

**public Developer(string name, int age, double salary) : base(name, age)**

**{Salary = salary;}}}**

**Lesson03 Pattern Matching**

**Notes:-**





