**S17: AnonymousFunctionAndLambda Expression**

**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.

* **Anonymous Function:-** its introduced in C# 2.0.method without name. But it has body. No need to define return type. Delegate is required to define anonymous function. Delegate keyword is use. Used to create inline method. Doesn’t contain jump statement. Doesn’t access unsafe code. Also use as an event handler. Don’t require access modifier. It’s not static or non-static. Cannot used ref out parameter with it. It must be assigned to delegate. It can be access outer variable. It can be pass as parameter.
* **Lambda Expression:-** its introduced in C# 3.0. it also work like anonymous methods. Its shorthand of anonymous function. Its easy as compare to anonymous method. Don’t need to specify type of value. Its also used in **LINQ.** The **=>** is used for lambda expression. Its devided in two parts **Left:** input or Parameter side **Right**: Expression Side.
* **Types of Lambda Expression: Expression Lambda:**- it contain only one line and within that line we can perform the operation. It dosnt contain { } brackets.
* **Input Expression:- Statement Lambda:-** consist more than one line. Or statement. It contain the body in { } bracket. Doesn’t return value implicitly (Automatically). For returning value u need to change return type in delegate.

S17\_\_AnonymousMethodAndLambdaExpression.cs

using System;

using System.Collections.Generic;

using System.Text;

namespace OOPS\_\_AllSession

{

public delegate int CubeNumbers(int numbers);

public delegate void PrintPositiveNumbers();

public delegate void PrintPersonNames();

public delegate void PrintCityNames(string[] cities);

class S17\_\_AnonymousMethodAndLambdaExpression

{

public static void AnonymousMethod()

{

PrintPositiveNumbers anonymousMethod = delegate ()

{

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 + " ");

};

anonymousMethod.Invoke();

//Lambda Expression 1. Expression Lambda

Console.WriteLine("\n\n\*\*\*\*\*\*\*\*\*\*\*\* Under the Expression Lambda Expression");

CubeNumbers expressionLambda = (numbers) => numbers \* numbers \* numbers;

Console.Write("Cube of Numbers: \t");

Console.Write(expressionLambda.Invoke(2));

//2. Statment Lambda

Console.WriteLine("\n\n\*\*\*\*\*\*\*\*\*\*\*\* Under the Lambda Expression");

PrintPersonNames lambdaExpression = () =>

{

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

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

foreach (string name in names)

Console.Write(name + ", ");

Console.WriteLine();

};

lambdaExpression.Invoke();

Console.WriteLine("\n\n\*\*\*\*\*\*\*\*\*\*\*\* Under the PArameterised Lambda Expression");

PrintCityNames parameters\_LambdaExpression = (cities) =>

{

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

foreach (string city in cities)

Console.Write(city + ", ");

Console.WriteLine();

};

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

parameters\_LambdaExpression.Invoke(city);

}

}

}

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

//S17\_\_Anonymous Function And LambdaExpression

S17\_\_AnonymousMethodAndLambdaExpression anonymousMethod = new S17\_\_AnonymousMethodAndLambdaExpression();

S17\_\_AnonymousMethodAndLambdaExpression.AnonymousMethod();

}

}

}