**S22: LinQ**

**Linq: -** language integrated query. Introduced in 3.0. It’s similar to SQL. Its uniform query to retrieve data from different data sources and format such as: SQL, Collection, Dataset, XMl etc. It return result as object it enable object oriented approach. It uses standered query operator and also it uses **IEnumerable** and **IQueryable**. There are 50 standard operator are available in LINQ : Filtering Operator, Join Operator, sorting, aggregation etc.. It will present in System.Linq namespace. We can use lambda expression also. Implicitly type variable **var** is used.

**Query Syntax:-** it begins with from keyword and end with select or Groupby. After from u can perform operations such as filter, sort, group.

**Method Syntax:-** also known as **fluent syntax** uses extension method include Enumarable and Quarable static class similar to how extension method call. Compiler convert query syntax into method syntax at compile time.

**IEnumerable** and **IQueryable used witht collection.**

**IEnumerable :-** System.Collection.Namespace. Linq to Object & Linq to xml. No lazy loading. In memory Query. Dont support custom query.

**IQueryable** are interfaces that allows you to manipulate and query collections of data. IEnumerable is inherited by IQueryable, that means IQueryable add features to IEnumerable interface. Linq to SQL. Lazy loading. Out Memory Suport custom query.

S22\_\_LinQ.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Reflection;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

namespace OOPS\_\_AllSession

{

class S22\_\_LinQ

{

public void PrintNames()

{

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

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

var studentNames = from name in names where name.Length <= 0 orderby name select name;

var studentName = from name in names where name.Contains('k') orderby name select name;

//var twoDatas = studentName.Zip(studentName, (names) => new { studentNames = names, studentName = names });

foreach (string name in studentNames)

Console.Write(name + ", ");

Console.WriteLine();

}

}

class ProductReview

{

public int productId { get; set; }

public int userId { get; set; }

public double rating { get; set; }

public string review { get; set; }

public bool isLike { get; set; }

public void CallingMethods()

{

List<ProductReview> list = new List<ProductReview>()

{

new ProductReview() {productId=1,userId=1,rating=5,review="Good",isLike=true},

new ProductReview() { productId = 1, userId = 100, rating = 5, review = "Good", isLike=true },

new ProductReview() { productId = 2, userId = 100, rating = 5, review = "Good", isLike=true },

new ProductReview() { productId = 3, userId = 101, rating = 2.5, review = "Bad", isLike=false },

new ProductReview() { productId = 4, userId = 101, rating = 2.8, review = "Bad", isLike=true },

new ProductReview() { productId = 5, userId = 102, rating = 8, review = "Nice", isLike=true },

new ProductReview() { productId = 6, userId = 103, rating = 8, review = "Nice", isLike=false },new ProductReview() { productId = 5, userId = 104, rating = 8, review = "Nice", isLike=true },

new ProductReview() { productId = 60, userId = 105, rating = 8, review = "Nice", isLike=false }

};

ProductReview productLinQ = new ProductReview();

productLinQ.TopRecords(list);

productLinQ.GetRecord\_ByUserID(list);

productLinQ.CountRecord(list);

}

public void TopRecords(List<ProductReview> listProductReview)

{

var recordedData = (from productReviews in listProductReview

orderby productReviews.rating descending

select productReviews).Take(3).ToList();

DisplayRecord((List<ProductReview>)recordedData);

}

public void CountRecord(List<ProductReview> listProductReview)

{

var recordData = listProductReview.GroupBy(x => x.productId).Select(x => new { productId = x.Key, Count = x.Count() });

Console.WriteLine("\nResult for Records Grouped By ProductID");

foreach (var list in recordData)

Console.WriteLine("Product ID :" + list.productId + " ---Count--- " + list.Count);

}

public void GetRecord\_ByUserID(List<ProductReview> listproductReview)

{

Console.WriteLine("\n\nUser ID 100 Record ");

var recordData = (from ProductReview in listproductReview

where (ProductReview.userId == 100)

orderby ProductReview.rating descending

select ProductReview).ToList();

DisplayRecord((List<ProductReview>)recordData);

}

public void DisplayRecord(List<ProductReview> record)

{

foreach (var lists in record)

Console.WriteLine($"Product Id: {lists.productId}\t User Id: {lists.userId}\t Ratings: {lists.rating}\t Review Is: {lists.review}\t Is Like: {lists.isLike}");

}

}

}

OopsSessions.cs

using OOPS\_\_AllSession;

using System;

using System.Threading;

using static OOPS\_\_AllSession.S11\_\_ClassAndTypes;

namespace Oops\_\_AllSession

{

class OopsSessions

{

static void Main(string[] args)

{

Console.WriteLine("\*\*\*\*\*\*\*\*Welcome To Main Method\*\*\*\*\*\*\*\*\*\*\*");

S22\_\_LinQ linq = new S22\_\_LinQ();

linq.PrintNames();

ProductReview productLinQ = new ProductReview();

productLinQ.CallingMethods();

}

}

}