

**МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ “КПІ”**

**Кафедра**

**Автоматизованих систем обробки інформації та управління**

**Звіт**

**до лабораторної роботи з дисципліни**

**“** **Технології розроблення програмного забезпечення -1. Основи розробки програмного забезпечення на платформі Microsoft.NET”**

**Комп’ютерний практикум № 4.**

**Варіант - 6**

**Перевірли:**

**доц. Ліщук К. І.**

**Виконала:**

**студентка групи ІС-83**

**Прийдун М.В.**

**м. Київ-2020**

**Комп‘ютерний практикум № 4. LINQ to Objects**

**Мета:** ознайомитися з обробкою даних з використанням бібліотеки LINQ to Objects

**Постановка задачі комп‘ютерного практикуму № 4**

При виконанні комп‘ютерного практикуму необхідно виконати наступні дії:

1. Розробити структуру даних для зберігання згідно варіантів, наведених нижче. У кожному з варіантів має бути як мінімум 3-4 класи. В рамках реалізації повинні бути продемонстровані зв'язки між класами: один-до-багатьох і багато-до-багатьох.
2. Розробити як мінімум 15 різних запитів, використовуючи різні дії над множинами: групування, сортування, фільтрацію, об'єднання результатів декількох запитів в один (join, concat) та інше. Крім того, необхідно використовувати обидва можливі варіанти реалізації LINQ-запитів (класичний варіант та з використанням методів розширення), причому запити не повинні повторюватись.
3. Створити програмне забезпечення, котре реалізує обробку даних з використання бібліотеки LINQ to Objects.
4. Програмне забезпечення необхідно розробити у вигляді консольного застосування на мові C#.
5. Коротко описати архітектуру проекту

**Завдання:**

Розробити структуру даних для зберігання інформації про проекти, що виконуються на підприємстві. По кожному проекту зберігається інформація: шифр проекту, найменування проекту, вартість робіт для виконання проекту, дата початку проекту і дата закінчення проекту, список осіб, які беруть участь в проекті.

**Опис архітектури проету:**

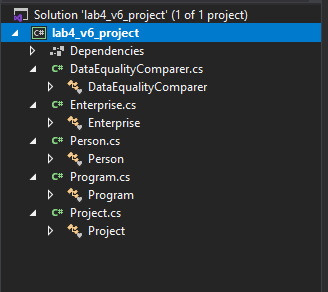


Рис.1 Структура проекту

## Файл DataEqualityComparer.cs

using System;

using System.Collections.Generic;

using System.Text;

namespace lab4\_v6\_project

{

class DataEqualityComparer: IEqualityComparer<Enterprise>

{

public bool Equals(Enterprise x, Enterprise y)

{

bool Result = false;

if (x.Name == y.Name && x.Code\_Projects == y.Code\_Projects)

Result = true;

return Result;

}

public int GetHashCode(Enterprise obj)

{

return obj.Code\_Projects.GetHashCode();

}

}

}

## Файл Enterprise.cs

using System;

using System.Collections.Generic;

using System.Diagnostics;

using System.Text;

namespace lab4\_v6\_project

{

class Enterprise

{

string name;

string code\_Projects;

public Enterprise(string n, string c)

{

name = n;

code\_Projects = c;

}

public override string ToString()

{

return "Name of enterprise - " + this.Name + ". Code of the project " + this.Code\_Projects ;

}

public string Name

{

get

{

return name;

}

set

{

name = value;

}

}

public string Code\_Projects

{

get

{

return code\_Projects;

}

set

{

code\_Projects = value;

}

}

}

}

## Файл Person.cs

using System;

using System.Collections.Generic;

using System.Text;

namespace lab4\_v6\_project

{

class Person

{

//int id;

private string name;

private string position;

public Person(/\*int num,\*/string n, string p)

{

//id = num;

name = n;

position = p;

}

public override string ToString()

{

return "Name - " + this.Name + ". Position - " + this.Position;

}

//public int Id

//{

// get

// {

// return id;

// }

// set

// {

// id = value;

// }

//}

public string Name

{

get

{

return name;

}

set

{

name = value;

}

}

public string Position

{

get

{

return position;

}

set

{

position = value;

}

}

}

}

## Файл Program.cs

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Net.Http.Headers;

namespace lab4\_v6\_project

{

class Program

{

static void Main()

{

List<Enterprise> enterprises2 = new List<Enterprise>

{

new Enterprise("Luxoft","LX18567"),

new Enterprise ("Ciklum","CK184545"),

new Enterprise("Netcracker","NC82245"),

new Enterprise("Genesis", "GN18275"),

new Enterprise("Luxoft","LX18568"),

};

List<Enterprise> enterprises = new List<Enterprise>();

enterprises.Add(new Enterprise("Epam", "EP83956"));

enterprises.Add(new Enterprise("GlobalLogic", "GL85253"));

enterprises.Add(new Enterprise("SoftServe", "SS84426"));

enterprises.Add(new Enterprise("Ubisoft", "US78535"));

enterprises.Add(new Enterprise("Innovecs","IN45026"));

enterprises.Add(new Enterprise("DataArt", "DA18275"));

enterprises.Add(new Enterprise("Genesis", "GN18275"));

List<Project> projects = new List<Project>();

projects.Add(new Project("EP83956", "Mobile application to support patients", 2000, new DateTime(2019, 7, 20), new DateTime(2019, 7, 20),

new List<Person>

{

new Person("Illyenko Makar","Project Manager"),

new Person("MSK","Customer"),

new Person("Makorov Volodymyr","Architect"),

new Person("Korylov Mykhayil","Developer"),

new Person("Adamenko Mariya","UI / UX designer"),

new Person("Kalynovskyy Andriy","QA Engineer"),

new Person("Zabudko Kateryna","Developer"),

new Person("Mykhaylochyk Illya","Developer")

}

)) ;

projects.Add(new Project("GL85253", "Fluid", 2500, new DateTime(2019, 9, 10), new DateTime(2020, 3, 29),

new List<Person>

{

new Person("Zenko Maryna","Project Manager"),

new Person("John Martinez","Customer"),

new Person("Rykorov Vlad","Architect"),

new Person("Zarylov Adam","Developer"),

new Person("Hrind Maksym","UI / UX designer"),

new Person("Rudyk Taras","QA/QC Engineer"),

new Person("Zabludko Roman","Developer"),

new Person("Zarinovych Artur","Developer")

}

));

projects.Add(new Project("SS84426", "Robo Advisor", 3000, new DateTime(2019, 12, 18), new DateTime(2020, 2, 20),

new List<Person>

{

new Person("Alushta Andrii","Project Manager"),

new Person("John Mayer","Customer"),

new Person("Borshchahivkа Volodymyr","Architect"),

new Person("Zghurskyi Bohdan","Developer"),

new Person("Haievych Dmytro","UI / UX designer"),

new Person("Kazymyrchuk Zhanna","QA/QC Engineer"),

new Person("Ivankiv Oleh","Developer"),

new Person("Ivashchenko Oleksii","Developer")

}

));

projects.Add(new Project("US78535", "Brawhalla", 30000, new DateTime(2019, 12, 18), new DateTime(2020, 2, 20),

new List<Person>

{

new Person("Kovalenko Petro","Project Manager"),

new Person("Rhine Horwood","Customer"),

new Person("Lebedyn Volodymyr","Architect"),

new Person("Marynych Bohdan","Developer"),

new Person("Onyshchenko Nataliіa","UI / UX designer"),

new Person("Reshetylivka Solomiia","QA/QC Engineer"),

new Person("Filipchuk Uliana","Developer"),

new Person("Stetsenko Oleksii","Developer")

}

));

projects.Add(new Project("IN45026", "Gelato", 800, new DateTime(2019, 8, 18), new DateTime(2019, 11, 30),

new List<Person>

{

new Person("Shevchenko Bohdan","Project Manager"),

new Person("William Smith","Customer"),

new Person("Harashchenko Volodymyr","Architect"),

new Person("Yaroshenko Bohdan","Developer"),

new Person("Trots Yurii","UI / UX designer"),

new Person("Reshetylivka Solomiia","QA/QC Engineer"),

new Person("Filipchuk Kostiantyn","Developer"),

new Person("Znamianka Oleksii","Developer")

}

));

projects.Add(new Project("DA18275", "Skillorton", 5000, new DateTime(2019, 9, 8), new DateTime(2019, 11, 1),

new List<Person>

{

new Person("Stetskiv Bohdan","Project Manager"),

new Person("Jacob Miller","Customer"),

new Person("Tymofiyiv Volodymyr","Architect"),

new Person("Todoriv Bohdan","Developer"),

new Person("Fedkov Yurii","UI / UX designer"),

new Person("Tsapiv Solomiia","QA/QC Engineer"),

new Person("Tsakhniv Kostiantyn","Developer"),

new Person("Yurtsiv Oleksii","Developer")

}

));

while(true)

{

Console.WriteLine("Menu");

Console.WriteLine("1. Display the list of enterprises.");

Console.WriteLine("2. Display the list of projects.");

Console.WriteLine("3. Display the list of completed projects by increasing their price.");

Console.WriteLine("4. Display the list of project participants.");

Console.WriteLine("5. Display the project codes that started earlier on 10/09/2019");

Console.WriteLine("6. Display the name of project and enterprise that created it(join)");

Console.WriteLine("7. Display the merging lists of enterprises");

Console.WriteLine("8. Display the list enterprises that are on both lists");

Console.WriteLine("9. Display the difference of lists of enterprises");

Console.WriteLine("10. Display the enterprisesand number of their project.");

Console.WriteLine("11. Display the project with the highest price.");

Console.WriteLine("12. Display the total cost of all projects.");

Console.WriteLine("13. Display the average cost of all projects.");

Console.WriteLine("14. Check that the cost of each project is more than $ 3000.");

Console.WriteLine("15. Check that there is at least one project with a price over 25,000.");

Console.WriteLine("16.Exit");

Console.WriteLine("Enter the menu item:");

string ch = Console.ReadLine();

switch(ch)

{

case "1":

Console.WriteLine("===================== Enterprises=======================");

var e = from x in enterprises

select x;

foreach (var x in e)

Console.WriteLine(x);

Console.ReadLine();

Console.Clear();

break;

case "2":

Console.WriteLine("========================= Projects=============================");

var p = from x in projects

select x;

foreach (var x in p)

Console.WriteLine(x);

Console.ReadLine();

Console.Clear();

break;

case "3":

var selectedProjects = from project in projects

where project.Date2 < DateTime.Now

orderby project.Price

select project;

Console.WriteLine("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

Console.WriteLine("Project - Price");

Console.WriteLine("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

foreach (Project finished in selectedProjects)

Console.WriteLine($"{finished.Name} - {finished.Price}$");

Console.WriteLine("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

Console.ReadLine();

Console.Clear();

break;

case "4":

Console.WriteLine("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

Console.WriteLine("Select project:");

Console.WriteLine("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

string np="";

int i = 1;

var pr = from x in projects

select x;

foreach (Project x in pr)

{

Console.WriteLine(i + ". " + x.Name);

i++;

}

Console.WriteLine("Enter number of project:");

int C = Convert.ToInt32(Console.ReadLine());

if (C <= i)

{

int t = 1;

foreach (var x in pr)

{

if (t == C)

np = x.Name;

t++;

}

}

else

{

Console.WriteLine("Error!There is no project with this number");

Console.ReadLine();

break;

}

var selected = from p1 in projects

from part in p1.Participants

where p1.Name == np

select p1;

foreach (Project p2 in selected)

{

foreach (Person person in p2.Participants)

{

Console.WriteLine(person);

}

break;

}

Console.ReadLine();

Console.Clear();

break;

case "5":

var sproject= projects.Where(proj1 => proj1.Date1 < new DateTime(2019,9,10));

Console.WriteLine("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

Console.WriteLine("Project Code - Date of started");

Console.WriteLine("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

foreach (Project x in sproject)

Console.WriteLine($"{x.Code} - {x.Date1.ToShortDateString()}");

Console.WriteLine("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

Console.ReadLine();

Console.Clear();

break;

case "6":

var eproj = enterprises.Join(projects,

e => e.Code\_Projects,

pr => pr.Code,

(e, pr) => new { Name = e.Name, Code\_Projects = e.Code\_Projects, NameProject = pr.Name });

foreach (var x in eproj)

Console.WriteLine(x);

Console.ReadLine();

Console.Clear();

break;

case "7":

var result = enterprises.Union(enterprises2, new DataEqualityComparer());

// var result2 = enterprises.Concat(enterprises2).Distinct(new DataEqualityComparer());

foreach (Enterprise s in result)

Console.WriteLine(s);

Console.ReadLine();

Console.Clear();

break;

case "8":

var interect = enterprises.Intersect(enterprises2, new DataEqualityComparer());

foreach (var IN in interect)

Console.WriteLine(IN);

Console.ReadLine();

Console.Clear();

break;

case "9":

var except = enterprises.Except(enterprises2, new DataEqualityComparer());

foreach (var IN in except)

Console.WriteLine(IN);

Console.ReadLine();

Console.Clear();

break;

case "10":

Console.WriteLine("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

Console.WriteLine("Enterprise Number of projects");

Console.WriteLine("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

var number = from m in enterprises2

group m by m.Name into g

select new { Name = g.Key, Count = g.Count()

};

foreach (var group in number)

Console.WriteLine($"{group.Name} : {group.Count}");

Console.WriteLine("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

Console.ReadLine();

Console.Clear();

break;

case "11":

var proj3 = projects.Where(one => one.Price == projects.Max(n => n.Price));

Console.WriteLine("The project whis the highest price: ");

foreach (Project one in proj3)

Console.WriteLine($"{one.Name} : {one.Price}");

Console.ReadLine();

Console.Clear();

break;

case "12":

float Sum = projects.Sum(proj4 => proj4.Price);

Console.WriteLine("The total cost of all projects:{0}", Sum);

Console.ReadLine();

Console.Clear();

break;

case "13":

float average = projects.Average(proj4 => proj4.Price);

Console.WriteLine("The average cost of all projects:{0}", average);

Console.ReadLine();

Console.Clear();

break;

case "14":

bool result1 = projects.All(u => u.Price > 3000);

if (result1)

Console.WriteLine("The cost of each project is more than 3000$");

else

Console.WriteLine("There are projects with a cost less than 3000$");

Console.ReadLine();

Console.Clear();

break;

case "15":

bool result2 = projects.Any(u => u.Price > 25000);

if (result2)

Console.WriteLine("There is at least one project costing over 25,000$");

else

Console.WriteLine("There are no projects with a cost more than 25000$");

Console.ReadLine();

Console.Clear();

break;

case "16":

var selectproject = projects.FirstOrDefault(proj4 => proj4.Price == 2001);

Console.WriteLine(selectproject.Name);

break;

case "17":

Environment.Exit(0);

break;

default:

{

Console.WriteLine("Error! Enter an existing menu item.");

Console.ReadLine();

break;

}

}

}

}

}

}

## Файл Project.cs

using System;

using System.Collections.Generic;

using System.Data;

using System.Reflection.Metadata;

using System.Text;

namespace lab4\_v6\_project

{

class Project

{

string code;

string name;

float price;

DateTime date1 /\*= new DateTime()\*/;

DateTime date2 /\*= new DateTime()\*/;

public List<Person> Participants { get; set; }

public Project(string c, string n, float p, DateTime d1,DateTime d2, List<Person> persons)

{

code = c;

name = n;

price = p;

date1 = d1;

date2 = d2;

Participants = persons;

}

public override string ToString()

{

return "Code of the project " + this.Code + ". Name - " + this.Name + ". Price:" + this.Price.ToString() + Environment.NewLine+

"Date of creation begining of project:" + this.Date1.ToShortDateString() + ". Date of creation finishing of project:" + this.Date2.ToShortDateString();

}

public string Code

{

get

{

return code;

}

set

{

code = value;

}

}

public string Name

{

get

{

return name;

}

set

{

name = value;

}

}

public float Price

{

get

{

return price;

}

set

{

price = value;

}

}

public DateTime Date1

{

get

{

return date1;

}

set

{

date1 = value;

}

}

public DateTime Date2

{

get

{

return date2;

}

set

{

date2 = value;

}

}

}

}

Проект містить 5 класів. Клас DataEqulityComparer наслідується від інтерфейсу IEqualityComparer. Клас Project містить список учасників проекту типу Person.

У класі Program я з’єдную список Enterprise і Project по коду проекту.

**Висновок:** на лабораторній роботі я ознайомитися з обробкою даних з використанням бібліотеки LINQ to Objects.