Міністерство освіти і науки України
Національний технічний університет України «КПІ ім. Ігоря Сікорського"
Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Звіт

З лабораторної роботи №1

з курсу

«Основи розробки програмного забезпечення на платформі Microsoft.NET»

Виконав студент *ІП-01 Галько Міла*
(шифр, ПІБ)

Перевірила *Ліщук К.І.*
(ПІБ)

Київ 2022

Комп‘ютерний практикум № 1. LINQ to Objects

Мета:

- ознайомитися з обробкою даних з використанням бібліотеки LINQ to Objects

Постановка задачі комп‘ютерного практикуму № 1

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. Наприклад (предметне середовище Кінофільми):
    a. Вивести перелік всіх кінофільмів
    b. Вивести перелік акторів, котрі грають у кінофільмах, котрі починаються з літери «А»
    c. Вивести перелік всіх акторів та кінофільмів, в яких вони грають
    d. Вивести перелік всіх акторів, згрупувавши дані по рокам народження
    e. Вивести перелік кінофільмів, в яких хоча б у одного актора прізвище починається на літеру "А"
    f. Вивести всі кінофільми, відсортувавши їх по роках
    g. Вивести всіх акторів, згрупувавши по амплуа та роком народження
    h. З'єднати джерела даних Кінофільм» і «Актор». Вивести назву фільму, прізвище автора, прізвище актора в головній ролі.
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 та створити звіт, котрий завантажити в moodle

Варіант №1

1) Розробити структуру даних для зберігання інформації про студентів дипломників та їх керівників. Про студентів необхідно зберігати щонайменше наступну інформацію: ПІБ, група, дата народження, середній бал. Про керівників: ПІБ, посада. У одного керівника може бути декілька студентів-дипломників.

    Опис класів:

    В ході розробки програмного забезпечення було створено 3 класи: Student, Supervisor, Subject.

    Клас Student реалізує студента, у якого є один керівник та багато предметів. Загалом має атрибути: ID, ПІБ, група, дата народження, керівник, предмети з оцінками та метод для підрахунку середнього балу.

    Клас Supervisor реалізує керівника, що може мати декілька студентів-дипломників. Керівник має атрибути: ID, ПІБ, посада.

    Клас Subject реалізує дисципліну. У кожної дисципліни є перелік студентів, що її вивчають та сама назва предмета у якості атрибутів.

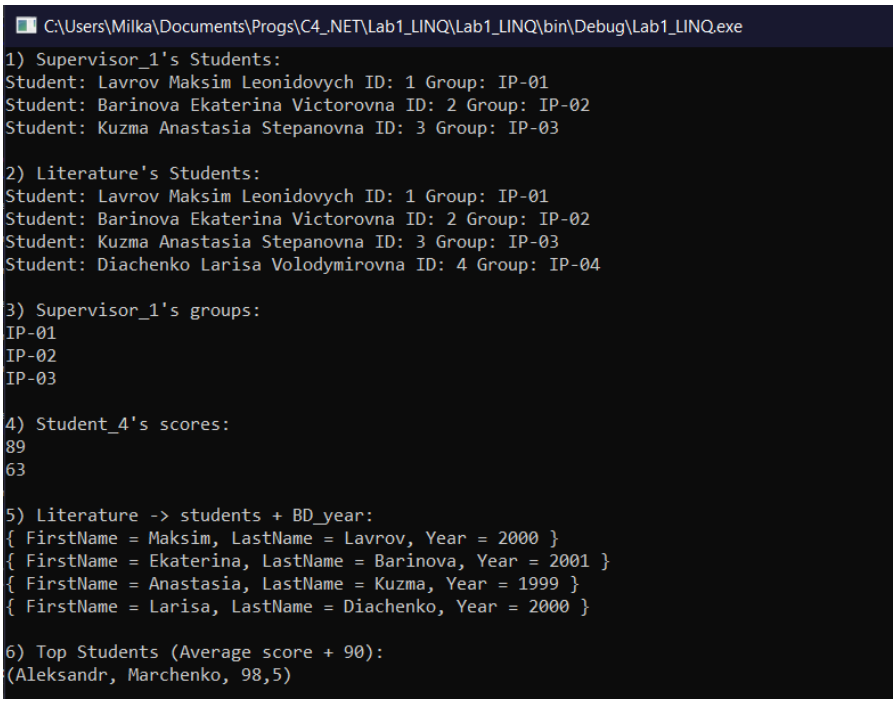
В результаті створення вищезазначених класів були відтворені зв’язки типу: один до багатьох (керівник до студента); багато до багатьох (студент до предмета).

Програмний код:

Посилання на github: https://github.com/MilaHalko/C4_.NET/tree/Lab1
Також код доданий в кінці документу.

Про програмне забезпечення:

У консолі відображаються результати роботи 26 різних запитів та опис цих дій над структурами даних для більшого розуміння. Приклад виводу у консоль:



Файл Student.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Lab1_LINQ
{
    internal class Student
    {
        public int Id { get; set; }
        public string FirstName { get; set; }
        public string LastName { get; set; }
        public string Patronymic { get; set; }
        public string Group { get; set; }
        public DateTime BirthDate { get; set; }

        public int SupervisorID { get; set; }


        public Dictionary<Subject, int> Scores = new Dictionary<Subject, int>();
        public float AverageScore {
            get
            {
                float sum = 0;
                foreach (var score in Scores)
                {
                    sum += score.Value;
                };
                return sum / Scores.Count;
            }
        }

        public override string ToString()
        {
            return String.Format(@"Student: {0} {1} {2} ID: {3} Group: {4}", LastName, FirstName, Patronymic, Id,
Group);
        }
    }
}
```

Файл Supervisor.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Lab1_LINQ
{
    internal class Supervisor
    {
        public int ID { get; set; }
        public string FirstName { get; set; }
        public string LastName { get; set; }
        public string Patronymic { get; set; }
        public string Post { get; set; }
        public ICollection<Student> Students { get; set; } = new List<Student>();

        public override string ToString()
        {
            string students = string.Join(" ", Students);
            return String.Format(@"
                Supervisor: {0} {1} {2}
                Post: {3}
                    Students: {4}", LastName, FirstName, Patronymic, Post, students);
        }

        internal object Join(List<Student> students, Func<object, object> p1, Func<Student, Student> p2, Func<obje
ct, object, object> p3)
        {
            throw new NotImplementedException();
        }
    }
}
```

Файл Subject.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Lab1_LINQ
{
    internal class Subject
    {
        public string Name { get; set; }
        public ICollection<Student> Students { get; set; } = new List<Student>();

        public override string ToString()
        {
            string students = string.Join(" ", Students);
            return String.Format(@"
                Subject: {0}
                    Students: {1}", Name, students);
        }
    }
}
```

Файл Program.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Lab1_LINQ
{
    internal class Program
    {
        static void Main(string[] args)
        {
            Console.OutputEncoding = Encoding.UTF8;

            // Students

            Student st1 = new Student { Id = 1, FirstName = "Maksim", LastName = "Lavrov", Patronymic = "Leonidovy
ch", Group = "IP-01", BirthDate = new DateTime(2000, 12,31)};
            Student st2 = new Student { Id = 2, FirstName = "Ekaterina", LastName = "Barinova", Patronymic = "Vict
orovna", Group = "IP-02", BirthDate = new DateTime(2001, 8, 14)};
            Student st3 = new Student { Id = 3, FirstName = "Anastasia", LastName = "Kuzma", Patronymic = "Stepano
vna", Group = "IP-03", BirthDate = new DateTime(1999, 4, 23)};
            Student st4 = new Student { Id = 4, FirstName = "Larisa", LastName = "Diachenko", Patronymic = "Volody
mirovna", Group = "IP-04", BirthDate = new DateTime(2000, 10, 11)};
            Student st5 = new Student { Id = 5, FirstName = "Elena", LastName = "Kuzma", Patronymic = "Vyloriivna"
, Group = "IT-01", BirthDate = new DateTime(2001, 2, 11)};
            Student st6 = new Student { Id = 6, FirstName = "Aleksandr", LastName = "Marchenko", Patronymic = "Ale
ksandrovych", Group = "IT-02", BirthDate = new DateTime(2002, 3, 15)};

            List<Student> students = new List<Student> { st1, st2, st3, st4, st5, st6 };

            // Supervisors

            List<Student> students1 = new List<Student> { st1, st2, st3};
            List<Student> students2 = new List<Student> { st4};
            List<Student> students3 = new List<Student> { st5, st6};

            Supervisor sup1 = new Supervisor { ID = 1, FirstName = "Victor", LastName = "Barinov", Patronymic = "P
etrovich", Post = "professor", Students = students1};
            Supervisor sup2 = new Supervisor { ID = 2, FirstName = "Victoria", LastName = "Honcharova", Patronymic
 = "Yevhenevna", Post = "lecturer", Students = students2};
            Supervisor sup3 = new Supervisor { ID = 3, FirstName = "Robert", LastName = "Dawny", Patronymic = "Sec
ond", Post = "philanthropist", Students = students3};

            List<Supervisor> allSupervisors = new List<Supervisor> { sup1, sup2, sup3 };

            // Subjects

            List<Student> students4 = new List<Student> { st1, st3, st4, st6 };
            List<Student> students5 = new List<Student> { st2, st5, st6 };
            List<Student> students6 = new List<Student> { st1, st2, st3, st4 };

            Subject sub1 = new Subject { Name = "Math", Students = students4};
            Subject sub2 = new Subject { Name = "Web", Students = students5};
            Subject sub3 = new Subject { Name = "Literature", Students = students6};

            List<Subject> allSubjects = new List<Subject> { sub1, sub2, sub3 };


            // Set Subjects' Scores & SupervisorIDs for Students

            st1.Scores[sub1] = 95;
            st1.Scores[sub3] = 73;
            st1.SupervisorID = 1;

            st2.Scores[sub2] = 85;
            st2.Scores[sub3] = 86;
            st2.SupervisorID = 1;

            st3.Scores[sub1] = 95;
            st3.Scores[sub3] = 76;
            st3.SupervisorID = 1;

            st4.Scores[sub1] = 89;
            st4.Scores[sub3] = 63;
            st4.SupervisorID = 2;

            st5.Scores[sub2] = 87;
            st5.SupervisorID = 3;

            st6.Scores[sub1] = 99;
            st6.Scores[sub3] = 98;
            st6.SupervisorID = 3;

            // 1. Simple select:
            Console.WriteLine("1) Supervisor_1's Students: ");
            var sup1_students = from student in sup1.Students
```

```
                    select student;
PrintLINQ(sup1_students);

Console.WriteLine("2) Literature's Students: ");
var sub3_students = sub3.Students.Select(student => student);
PrintLINQ(sub3_students);


// 2. Select list's attribute / projection:
Console.WriteLine("3) Supervisor_1's groups: ");
var sup1_groups = sup1.Students.Select(s => s.Group);
PrintLINQ(sup1_groups);

Console.WriteLine("4) Student_4's scores: ");
var student4_subjects = from s in st4.Scores
                        select s.Value;
PrintLINQ(student4_subjects);


// 3. New anonymous type object + .Year:
Console.WriteLine("5) Literature -> students + BD_year: ");
var Literature_students_scores = from s in sub3.Students
                                 select new { s.FirstName, s.LastName, s.BirthDate.Year };
PrintLINQ(Literature_students_scores);


// 4. Conditions:
Console.WriteLine("6) Top Students (Average score + 90): ");
var topStudents = from s in students
                  where s.AverageScore > 90
                  select ( s.FirstName, s.LastName, s.AverageScore );
PrintLINQ(topStudents);


// 5. Sorting
Console.WriteLine("7) Student_2's sorted subjects: ");
var st6_sortedSubjects = from s in st2.Scores
                         orderby (s.Key.Name)
                         select ( s.Key.Name, s.Value );
PrintLINQ(st6_sortedSubjects);


// 6. Order + Then
Console.WriteLine("8) Students's list by year of BD & Lastname: ");
var stsByYearLastname = students.OrderBy(s => s.BirthDate.Year).ThenBy(s => s.LastName);
foreach (var st in stsByYearLastname)
{
    Console.WriteLine("{0} - {1} {2}", st.BirthDate.Year, st.LastName, st.FirstName);
}
Console.WriteLine();


// 7. Hard Sorting + Count()
Console.WriteLine("9) Sorted Students & Subjects' quantity: ");
var sortedSupsStuds = from st in students
                      orderby st.LastName
                      orderby st.Scores.Count()
                      select ( st.LastName, st.Scores.Count());
PrintLINQ(sortedSupsStuds);


// 8. Skip & Take
Console.WriteLine("10) Skip Average < 60 and take Average < 90: ");
var students_skipTake = students.SkipWhile(s => s.AverageScore < 60)
                                .TakeWhile(s => s.AverageScore < 90)
                                .OrderBy(s => s.AverageScore)
                                .Select(s => (s.LastName, s.AverageScore));
PrintLINQ(students_skipTake);


// 9. Min & Max
Console.WriteLine("11) Students with their lowest score: ");
var lowestScore = from s in students
                  orderby s.Scores.Values.Min()
                  select (s.LastName, s.Scores.Values.Min());
PrintLINQ(lowestScore);

Console.WriteLine("12) Students with their highest score: ");
var highestScore = from s in students
                   select (s.LastName, s.Scores.Values.Max());
PrintLINQ(highestScore);


// 10. StartWith + ToUpper
Console.WriteLine("13) IP Students: ");
var IPstudents = from st in students
                 where st.Group.ToUpper().StartsWith("IP")
                 orderby st.Group
                 select st.LastName + " " + st.FirstName + ", group: " + st.Group;
```

```
            PrintLINQ(IPstudents);

            Console.WriteLine("14) IT Students: ");
            var ITstudents = students.Where(s => s.Group.ToUpper().StartsWith("IT")).OrderBy(s => s.Group);
            PrintLINQ(ITstudents);


            // 11. Date
            Console.WriteLine("15) Students by age: ");
            var stsByAge = from st in students
                           orderby st.BirthDate
                           select st.BirthDate.ToString("yyyy/MM/dd") + " " + st.LastName + " " + st.FirstName;
            PrintLINQ(stsByAge);

            Console.WriteLine("16) Students by age of #3 supervisor: ");
            var stsByAgeOf3Sup = students3.OrderBy(s => s.BirthDate).Select(s => s.LastName + " " + s.FirstName);
            PrintLINQ(stsByAgeOf3Sup);


            // 12. All
            Console.WriteLine("17) Check all students are older than 18:");
            var boolAge18 = students.All(s => DateTime.Now.Year - s.BirthDate.Year > 18);
            Console.WriteLine("All students are +18 age - {0}", boolAge18);
            Console.WriteLine();


            // 13. Concat & Distinct
            Console.WriteLine("18) Concated Students from 1&3 Subjects: ");
            var concatStsBySubs13 = (from s in sub1.Students
                                     select s.LastName + " " + s.FirstName)
                                     .Concat(from s in sub3.Students
                                             select s.LastName + " " + s.FirstName).Distinct();
            PrintLINQ(concatStsBySubs13);


            // 14. Intersect
            Console.WriteLine("19) Intersect of Subject 1 & 3 students: ");
            var sub13StsIntersect = sub1.Students.Intersect(sub3.Students);
            PrintLINQ(sub13StsIntersect);


            // 15. Except
            Console.WriteLine("20) Subject1 students except Subject2:");
            var sub12Except = sub1.Students.Except(sub2.Students);
            PrintLINQ(sub12Except);


            // 16. Union
            Console.WriteLine("21) Union of Supervisor 1 & 3's Students: ");
            var unionSup13Sts = (from s in sup1.Students
                                 select s)
                                .Union(from s in sup3.Students
                                       select s)
                                .OrderByDescending(s => s.LastName);
            PrintLINQ(unionSup13Sts);


            // 17.Join
            Console.WriteLine("22) Student_1 -> join Subjects: ");
            var st1JoinSubs = from sc in st1.Scores
                              join sub in allSubjects on sc.Key equals sub
                              select sc.Key.Name + ": " + sc.Value;
            PrintLINQ(st1JoinSubs);

            Console.WriteLine("23) Supervisor_1 -> join Students Average Score: ");
            var sup1JoinStScores = sup1.Students.Join(students, s1 => s1.Id, s2 => s2.Id,
                                                      (s1, s2) => new { Name = s1.LastName + " " + s1.FirstName, Sc
ore = s1.AverageScore });
            PrintLINQ(sup1JoinStScores);


            // 18. Group Join
            Console.WriteLine("24) AllSupervisors -> GroupJoin Students: ");
            var supsJoinSts = allSupervisors.GroupJoin(students,
                                                       sup => sup.ID,
                                                       st => st.SupervisorID,
                                                       (sups, sts) => new
                                                       {
                                                           Supervisor = sups.FirstName + " " + sups.Patronymic,
                                                           Students = sts.Select(s => s.LastName)
                                                       });
            foreach (var sups in supsJoinSts)
            {
                Console.WriteLine($"{sups.Supervisor}:");
                foreach (var student in sups.Students)
                {
                    Console.WriteLine($"\t{student}");
                }
            }
```

```
            Console.WriteLine();


            // 19. Group by
            Console.WriteLine("25) Students grouped by LastName: ");
            var stsGroupedLastName = from st in students
                                     group st by st.LastName;
            foreach (var sts in stsGroupedLastName)
            {
                Console.WriteLine(sts.Key + ":");
                foreach (var student in sts)
                {
                    Console.WriteLine($"\t{student.FirstName}");
                }
            }
            Console.WriteLine();

            Console.WriteLine("26) Students' level by Average score: ");
            var stsLevel = students.GroupBy(st =>
            {
                if (st.AverageScore < 80) { return "3 Low score"; }
                else if (st.AverageScore < 90) { return "2 Middle score"; }
                else { return "1 High score"; }
            }).OrderBy(st => st.Key);
            foreach (var sts in stsLevel)
            {
                Console.WriteLine(sts.Key + ": ");
                foreach (var st in sts)
                {
                    Console.WriteLine("\t{0} {1}: {2}", st.LastName, st.FirstName, st.AverageScore);
                }
            }
            Console.WriteLine();

            Console.ReadLine();
        }

        private static void PrintLINQ<T>(IEnumerable<T> list)
        {
            foreach (T item in list)
            {
                Console.WriteLine(item);
            }
            Console.WriteLine();
        }
    }
}
```