Автор: Ямпольский Дмитро КІТ-119а

Дата: 11.12.2021

**Лабораторна робота 7**

**Тема**. Об'єктно орієнтована декомпозиція. Рефакторинг – реорганізація програмного коду

Задачі:

1. Оптимізувати структуру класів, згрупувавши методи обробки відповідних класах згідно з призначенням.
2. Реалізувати можливість перегляду користувачем довідника студентів, продемонструвавши різні варіанти застосування LINQ:

* відкладене виконання запитів;
* примусове виконання запитів;
* використання статистичних функцій;
* використання лямбда виражень;
* змішаний синтаксис запиту та методу

Опис класів

Container – власний клас контейнера для реалізації колекції об'єктів;

ContainerEnumerator – клас, який реалізує інтерфейс IEnumerator;

DataProcessing – клас, який виконує обробку даних студента;

DataPrintService – клас, який виконує роботу з виводу даних;

FileService – клас, який виконує роботу із файлами;

Текст програми

Container.cs

using System;

using System.Collections;

using System.Linq;

using menshakov01;

namespace menshakov07

{

    /// <summary>

    /// Class Container

    /// class that implements class container

    /// for collection of students

    /// </summary>

    public class Container

    {

        /// <summary>

        /// Private field students

        /// </summary>

        private Student[] \_students;

        /// <summary>

        /// Constructor with one parameter

        /// </summary>

        /// <param name="students"></param>

        public Container(Student[] students)

        {

            \_students = new Student[students.Length];

            for (var i = 0; i < students.Length; i++)

            {

                \_students[i] = students[i];

            }

        }

        /// <summary>

        ///

        /// </summary>

        public Student[] Students => \_students;

        /// <summary>

        /// Method that adds student to collection

        /// </summary>

        /// <param name="student"></param>

        public void Add(Student student)

        {

            if (student == null)

            {

                throw new ArgumentNullException(nameof(student), "Student is null");

            }

            var newArr = new Student[\_students.Length + 1];

            for (var i = 0; i < \_students.Length; i++)

            {

                newArr[i] = \_students[i];

            }

            newArr[newArr.Length - 1] = student;

            \_students = newArr;

        }

        /// <summary>

        /// Method that removes student from collection

        /// </summary>

        /// <param name="student"></param>

        /// <returns>True if student was removed otherwise false</returns>

        public bool Remove(Student student)

        {

            if (student == null)

            {

                return false;

            }

            var pos = -1;

            for (var i = 0; i < \_students.Length; i++)

            {

                if (\_students[i].Equals(student))

                {

                    pos = i;

                    break;

                }

            }

            if (pos == -1)

            {

                return false;

            }

            var newArr = new Student[\_students.Length - 1];

            for (var i = 0; i < pos; i++)

            {

                newArr[i] = \_students[i];

            }

            for (var i = pos + 1; i < \_students.Length; i++)

            {

                newArr[i - 1] = \_students[i];

            }

            \_students = newArr;

            return true;

        }

        /// <summary>

        /// Method that clears the collection

        /// </summary>

        public void Clear()

        {

            \_students = null;

        }

        /// <summary>

        /// Method that removes student by chosen criteria

        /// </summary>

        /// <returns>True if student was removed otherwise false</returns>

        public bool RemoveByCriteria()

        {

            Console.WriteLine("Enter criteria of the deletion:");

            Console.WriteLine("1) group index");

            Console.WriteLine("2) specialty");

            Console.WriteLine("3) faculty\n");

            Student[] students = null;

            var input = Console.ReadLine();

            switch (input)

            {

                case "group index":

                    Console.WriteLine("Write group index:");

                    input = Console.ReadLine();

                    students = \_students.Where(s => s.GroupIndex.Equals(Convert.ToChar(input))).ToArray();

                    break;

                case "specialty":

                    Console.WriteLine("Write specialty:");

                    input = Console.ReadLine();

                    students = \_students.Where(s => s.Specialty.Equals(input)).ToArray();

                    break;

                case "faculty":

                    Console.WriteLine("Write faculty:");

                    input = Console.ReadLine();

                    students = \_students.Where(s => s.Faculty.Equals(input)).ToArray();

                    break;

                default:

                    input = string.Empty;

                    Console.WriteLine("Invalid option\n");

                    break;

            }

            if (!string.IsNullOrEmpty(input))

            {

                var previousSize = \_students.Length;

                foreach (var item in \_students.Intersect(students))

                {

                    Remove(item);

                }

                if (previousSize != \_students.Length)

                {

                    return true;

                }

            }

            return false;

        }

        /// <summary>

        /// Implemented GetEnumerator method

        /// </summary>

        /// <returns>ContainerEnum</returns>

        public IEnumerator GetEnumerator()

        {

            return new ContainerEnumerator(\_students);

        }

    }

}

ContainerEnumerator.cs

using menshakov01;

using System;

using System.Collections;

namespace menshakov02

{

    /// <summary>

    /// Class ContainerEnum

    /// class that implements IEnumerator for student class

    /// </summary>

    public sealed class ContainerEnumerator : IEnumerator

    {

        /// <summary>

        /// Private fields of a class

        /// </summary>

        private Student[] \_students;

        private int \_position = -1;

        /// <summary>

        /// Constructor with one parameter

        /// </summary>

        /// <param name="students"></param>

        public ContainerEnumerator(Student[] students)

        {

            \_students = students;

        }

        /// <summary>

        /// Implemented Current property

        /// </summary>

        public object Current

        {

            get

            {

                try

                {

                    return \_students[\_position];

                }

                catch (IndexOutOfRangeException)

                {

                    throw new InvalidOperationException();

                }

            }

        }

        /// <summary>

        /// Implemented MoveNext method

        /// </summary>

        /// <returns></returns>

        public bool MoveNext()

        {

            \_position++;

            return \_position < \_students.Length;

        }

        /// <summary>

        /// Implemented Reset method

        /// </summary>

        public void Reset()

        {

            \_position = -1;

        }

    }

}

DataProcessing.cs

using System;

using System.Collections;

using System.Linq;

using menshakov01;

namespace menshakov07

{

    public static class DataProcessing

    {

        delegate int IsEqual(Student[] student);

        /// <summary>

        /// Method that finds student in collection

        /// </summary>

        /// <param name="student"></param>

        /// <returns>If such student exists returns it otherwise null</returns>

        public static Student Find(this Student[] students, Student student)

        {

            for (var i = 0; i < students.Length; i++)

            {

                if (students[i].Equals(student))

                {

                    return students[i];

                }

            }

            return null;

        }

        public static void Sort(this Student[] students, IComparer comparer)

        {

            Array.Sort(students, comparer);

        }

        /// <summary>

        /// Method that allows to edit data of chosen student

        /// </summary>

        /// <param name="student"></param>

        public static void EditData(this Student[] students, Student student)

        {

            var pos = -1;

            for (var i = 0; i < students.Length; i++)

            {

                if (students[i].Equals(student))

                {

                    pos = i;

                    break;

                }

            }

            if (pos != -1)

            {

                Console.WriteLine("Enter what field you want to edit:\n1) Name\n2) Surname\n3) Patronymic\n4) Date of birth\n5) Date of admission\n" +

                    "6) Group index\n7) Faculty\n8) Specialty\n9) Academic performance\n");

                var option = Console.ReadLine();

                try

                {

                    switch (option)

                    {

                        case "Name":

                            students[pos].Name = Console.ReadLine();

                            break;

                        case "Surname":

                            students[pos].Surname = Console.ReadLine();

                            break;

                        case "Patronymic":

                            students[pos].Patronymic = Console.ReadLine();

                            break;

                        case "Date of birth":

                            students[pos].DateOfBirth = DateTime.Parse(Console.ReadLine());

                            break;

                        case "Date of admission":

                            students[pos].DateOfAdmission = DateTime.Parse(Console.ReadLine());

                            break;

                        case "Group index":

                            students[pos].GroupIndex = char.Parse(Console.ReadLine());

                            break;

                        case "Faculty":

                            students[pos].Faculty = Console.ReadLine();

                            break;

                        case "Specialty":

                            students[pos].Specialty = Console.ReadLine();

                            break;

                        case "Academic performance":

                            students[pos].AcademicPerformance = int.Parse(Console.ReadLine());

                            break;

                        default:

                            Console.WriteLine("Invalid option\n");

                            break;

                    }

                }

                catch (FormatException ex)

                {

                    Console.WriteLine(ex.Message);

                }

            }

            else

            {

                Console.WriteLine("There is no such student in collection\n");

            }

        }

        /// <summary>

        /// Method that counts chosen average value of a given collection

        /// </summary>

        /// <returns>Returns average value of a chosen field</returns>

        public static int CountAverage(this Student[] \_students)

        {

            IsEqual func = null;

            Console.WriteLine("Count avg age or academic performance:");

            Console.WriteLine("1) Age");

            Console.WriteLine("2) Performance");

            var input = Console.ReadLine();

            if (input == "Age")

            {

                func = CountAvgAge;

            }

            else if (input == "Performance")

            {

                func = CountAvgPerformance;

            }

            else

            {

                Console.WriteLine("Invalid option");

                return -1;

            }

            Console.WriteLine("Enter criteria of the counting:");

            Console.WriteLine("1) group index");

            Console.WriteLine("2) specialty");

            Console.WriteLine("3) faculty\n");

            Student[] students = null;

            input = Console.ReadLine();

            switch (input)

            {

                case "group index":

                    Console.WriteLine("Write group index:");

                    input = Console.ReadLine();

                    students = \_students.Where(x => x.GroupIndex.Equals(Convert.ToChar(input))).ToArray();

                    break;

                case "specialty":

                    Console.WriteLine("Write specialty:");

                    input = Console.ReadLine();

                    students = \_students.Where(x => x.Specialty.Equals(input)).ToArray();

                    break;

                case "faculty":

                    Console.WriteLine("Write faculty:");

                    input = Console.ReadLine();

                    students = \_students.Where(x => x.Faculty.Equals(input)).ToArray();

                    break;

                default:

                    input = string.Empty;

                    Console.WriteLine("Invalid option\n");

                    break;

            }

            return func(students);

        }

        /// <summary>

        /// Method that counts average students` age of a given collection

        /// </summary>

        /// <param name="students"></param>

        /// <returns>Returns average value of an age field</returns>

        private static int CountAvgAge(Student[] students)

        {

            var count = 0;

            foreach (var student in students)

            {

                count += DateTime.Now.Year - student.DateOfBirth.Year;

            }

            return count / students.Length;

        }

        /// <summary>

        /// Method that counts average students` performance of a given collection

        /// </summary>

        /// <param name="students"></param>

        /// <returns>Returns average value of an performance field</returns>

        private static int CountAvgPerformance(Student[] students)

        {

            var averagePerformance = (from student in students

                         select student.AcademicPerformance)

                        .Average();

            return (int)averagePerformance;

        }

    }

}

DataPrintService.cs

using menshakov01;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace menshakov07

{

    class DataPrintService

    {

        /// <summary>

        /// Method that prints chosen data about student

        /// </summary>

        /// <param name="student"></param>

        public void ShowData(Student student)

        {

            var dataForPrint = new StringBuilder();

            Console.WriteLine("Enter what data you want to get:\n1) group index\n2) course\n3) age\n");

            var option = Console.ReadLine();

            switch (option)

            {

                case "group index":

                    dataForPrint.AppendFormat("\nFaculty: {0}\nSpecialty: {1}\nDate of admission: {2}\nGroup index: {3}", student.Faculty,

                        student.Specialty, student.DateOfAdmission.Year, student.GroupIndex);

                    Console.WriteLine(dataForPrint.ToString());

                    dataForPrint.Clear();

                    break;

                case "course":

                    dataForPrint.AppendFormat("\nCourse: {0}\nSemester: {1}\n", (DateTime.Now.Year - student.DateOfAdmission.Year) + 1,

                        Math.Ceiling((double)((12 \* (DateTime.Now.Year - student.DateOfAdmission.Year) + DateTime.Now.Month - student.DateOfAdmission.Month)

                       - 2 \* (DateTime.Now.Year - student.DateOfAdmission.Year))) / 5);

                    Console.WriteLine(dataForPrint.ToString());

                    dataForPrint.Clear();

                    break;

                case "age":

                    dataForPrint.AppendFormat("\nYears: {0}\nMonth: {1}\nDays: {2}\n", DateTime.Now.Year - student.DateOfBirth.Year,

                        (Math.Abs(DateTime.Now.Month - student.DateOfBirth.Month)) - 1, DateTime.Now.Day);

                    Console.WriteLine(dataForPrint.ToString());

                    dataForPrint.Clear();

                    break;

                default:

                    Console.WriteLine("Invalid option\n");

                    break;

            }

        }

        /// <summary>

        /// Method that prints chosen data about student in table format

        /// </summary>

        public void ShowFormattedData(Student[] students)

        {

            var separator = new string('-', 76);

            var dataForPrint = new StringBuilder();

            dataForPrint.AppendFormat("|{0,-30}|{1,-12}|{2,-21}|{3,-8}|", "Full name", "Group index", "Specialty", "Faculty");

            Console.WriteLine(separator);

            Console.WriteLine(dataForPrint);

            Console.WriteLine(separator);

            foreach (var student in students)

            {

                dataForPrint.Clear();

                var fullName = new StringBuilder(student.Surname + " " + student.Name + " " + student.Patronymic);

                dataForPrint.AppendFormat("|{0,-30}|{1,-12}|{2,-21}|{3, -8}|", fullName, student.GroupIndex, student.Specialty, student.Faculty);

                Console.WriteLine(dataForPrint);

                Console.WriteLine(separator);

            }

        }

    }

}

FileService.cs

using menshakov01;

using System;

using System.IO;

using System.Runtime.Serialization.Json;

namespace menshakov07

{

    public class FileService

    {

        /// <summary>

        /// Method that writes students' data to JSON file

        /// </summary>

        public void WriteToFile(Student[] students)

        {

            var jsonFormatter = new DataContractJsonSerializer(typeof(Student[]));

            try

            {

                using (var file = new FileStream("students.json", FileMode.Create))

                {

                    try

                    {

                        jsonFormatter.WriteObject(file, students);

                    }

                    catch (System.Runtime.Serialization.SerializationException ex)

                    {

                        Console.WriteLine(ex.Message);

                    }

                }

            }

            catch (UnauthorizedAccessException ex)

            {

                Console.WriteLine(ex.Message);

            }

        }

        /// <summary>

        /// Method that reads students' data from JSON file

        /// </summary>

        public void ReadFromFile(Student[] students)

        {

            if (students != null)

            {

                var jsonFormatter = new DataContractJsonSerializer(typeof(Student[]));

                try

                {

                    using (var file = new FileStream("students.json", FileMode.Open))

                    {

                        try

                        {

                            students = jsonFormatter.ReadObject(file) as Student[];

                        }

                        catch (System.Runtime.Serialization.SerializationException ex)

                        {

                            Console.WriteLine(ex.Message);

                        }

                    }

                }

                catch (FileNotFoundException ex)

                {

                    Console.WriteLine(ex.Message);

                }

            }

            else

            {

                Console.WriteLine("There are no students in container\n");

            }

        }

    }

}

Program.cs

using menshakov01;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace menshakov07

{

    class Program

    {

        static void Main(string[] args)

        {

            var customStudent = new Student("Momot", "Roman", "Evegenievich", DateTime.Parse("10-8-2001"), DateTime.Parse("16-05-2019"), 'b', "CIT", "Computer engineering", 80);

            var students = new Student[] { new Student("Bily", "Vadim", "Ivanovich", DateTime.Parse("12-6-2001"), DateTime.Parse("16-05-2019"), 'a', "CIT", "Computer engineering", 100),

                new Student("Menshakov", "Dmytro", "Olegovich", DateTime.Parse("16-11-2000"), DateTime.Parse("23-8-2019"), 'b', "CIT", "Computer engineering", 90)};

            var list = new Container(students);

            list.Add(customStudent);

            list.Students.CountAverage();

            var query = from student in students

                        where student.GroupIndex == 'b'

                        select student;

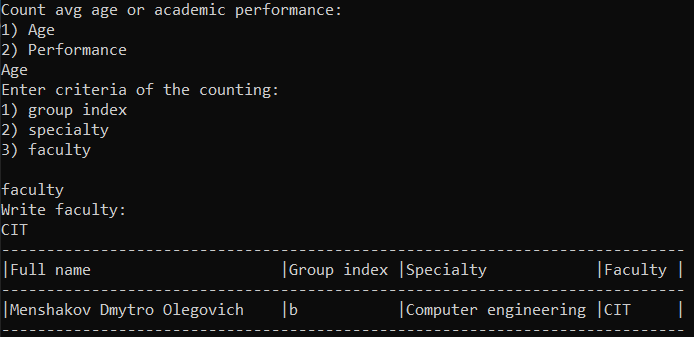
            var dataPrintService = new DataPrintService();

            dataPrintService.ShowFormattedData(query.ToArray());

        }

    }

}



Результати роботи програми

**Висновок**: у результаті виконання лабораторної роботи було оптимізовано структуру класів, згрупувавши методи обробки відповідних класах згідно з призначенням, реалізовано можливість перегляду користувачем довідника студентів, за використанням різних варіантів застосування LINQ: відкладене виконання, примусове, використання статистичних функцій та лямбда виражень, змішаний синтаксис запиту та методу.