Автор: Cиромятников Марк КІТ-119а

**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

DataProcessing.cs

using syromiatnikov01;

using System;

using System.Linq;

namespace syromiatnikov07

{

public static class DataWorkingService

{

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;

}

/// <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) First name\n2) Last name\n3) Patronymic\n4) Date of birth\n5) Date of admission\n" +

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

var option = Console.ReadLine();

try

{

switch (option)

{

case "Name":

\_students[pos].FirstName = Console.ReadLine();

break;

case "Surname":

\_students[pos].LastName = 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":

\_students[pos].Group = 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(s => s.Group.Equals(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;

}

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 sumOfyears = (from student in students

select student.DateOfBirth.Year).Sum();

var count = ((from student in students

select student).Count() \* DateTime.Now.Year) - sumOfyears;

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 count = 0;

foreach (var student in students)

{

count += student.AcademicPerformance;

}

return count / students.Length;

}

}

}

Program.cs

using System;

using System.Linq;

using syromiatnikov01;

namespace syromiatnikov07

{

class Program

{

static void Main(string[] args)

{

var customStudent = new Student("Momot", "Roman", "Evegenievich", DateTime.Parse("10-8-2001"), DateTime.Parse("16-05-2019"), "119b", "CIT", "Computer engineering", 80);

var students = new Student[] { new Student("Bily", "Vadim", "Ivanovich", DateTime.Parse("12-6-2001"), DateTime.Parse("16-05-2019"), "119a", "CIT", "Computer engineering", 100),

new Student("Menshakov", "Dmytro", "Olegovich", DateTime.Parse("16-11-2000"), DateTime.Parse("23-8-2019"), "119b", "CIT", "Computer engineering", 90)};

var list = new Container(students);

list.Students.CountAverage();

list.Add(customStudent);

var query = from element in list.Students

where element.Faculty.Equals("CIT")

select element;

var printService = new PrintService();

printService.ShowFormattedData(list.Students);

var stud = list.Students.Find(customStudent);

// list.RemoveByCriteria();

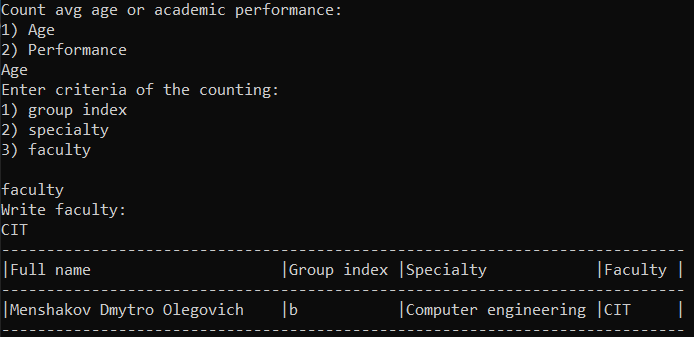
list.Clear();

Console.ReadLine();

}

}

}



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.