НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Кафедра «ОБЧИСЛЮВАЛЬНА ТЕХНІКА ТА ПРОГРАМУВАННЯ»

*Звiт з лабораторної роботи №7*

*Тема: «Об’єктно-орієнтована декомпозиція. Рефакторинг»*

Виконав:   
ст. гр. КІТ-119б  
Шарма О.Р.

Перевірив:  
Бартош М.В.

Харків – 2021

***Індивідуальне завдання:***

1. Оптимізувати структуру класів, згрупувавши методи обробки в відповідних класах згідно з призначенням. Рекомендується групувати методи обробки даних за наступними напрямками: призначеним для користувача інтерфейсом (консольне введення / виведення); угруповання / сортування / пошук; обчислення; генерація / перевірка / тестування.   
2. У звіті обґрунтувати обрану структуру класів описати реалізовані у вигляді класів суті, їх призначення і взаємозв'язок.

3. Реалізувати можливість перегляду користувачем довідника студентів, продемонструвавши різні варіанти застосування LINQ: відкладене виконання запитів; примусове виконання запитів; використання статистичних функцій; використання лямбдавираженій; змішаний синтаксис запиту і методу.

4. У звіті привести опис використовуваних способів застосування LINQ

ОПИС ПРОГРАМИ

Дана програма містить три класи з методами.

**Клас Main**

Головний клас програми, містить:

* Метод main

Послідовність виконання дій :

* 1. Створення контейнеру, який міститиме масив об’єктів типу Student.
  2. Запит у користувача чи бажає він додати студента.
  3. Якщо так, то запит користувача заповнити всі дані.
  4. Створення на основі даних об’єкта типу Student.
  5. Додання об’єкту до контейнеру.
  6. Виведення на екран вміст контейнера за допомогою foreach.
  7. Знаходження студента по індексу.
  8. Збереження введених даних у файлі для подальшого редагування.
  9. Відновлення раніше збережених даних з файлу.
  10. Пошук і редагування даних студента.
  11. Видалення даних особової справи студента.
  12. Виведення для обраного студента назви групи.
  13. Виведення для обраного студента номера курсу і семестру на поточний момент.
  14. Розрахунок і виведення для обраного студента віку на поточний момент (до дня).
  15. Розрахунок і виведення віку студентів обраного факультету на поточний момент, використовуючи LINQ.
  16. Розрахунок і виведення назв груп студентів обраного, використовуючи LINQ.
  17. Розрахунок і виведення номерів курса студентів обраного факультету на поточний момент, використовуючи LINQ.

**Клас Student**

Містить поля – інформація про студенів. Містить сеттери та геттери, а також метод toString та два конструктори.

**Клас Container**

Наслідується від IEnumerable та IEnumerator. Містить поля – масив студентів, розмір масиву. Також містить методи GetEnumerator, MoveNext, Reset, Current, Add, NewSize, Search, Remove, Edit.

ТЕКСТ ПРОГРАМИ

**Program.cs**

using System;

using System.Text;

using System.Linq;

using System.Collections.Generic;

namespace Sharma07

{

class Program

{

static void Main(string[] args)

{

string path = @"C:/Users/{Dean}/source/repos/Sharma07/Sharma07/saved.txt”;

StringBuilder sb = new StringBuilder();

var records = new Container();

bool loop = true;

int number;

int choice;

while (loop)

{

Console.WriteLine("Что Вы хотите сделать?\n 1 - Добавить данные про студента\n 2 - Вывести на экран данные\n 3 - Записать данные в файл" +

"\n 4 - Прочитать данные из файла\n 5 - Найти элемент по индексу\n 6 - Удалить данные о студенте\n 7 - Редактировать данные студента" +

"\n 8 - Вывести название группы студента\n 9 - Вывести текущий курс и семестр студента\n 10 - Вывести текущий возраст студента" +

"\n 11 - Возраст студентов факультета\n 12 - Группы студентов факультета\n 13 - Курс студентов факультета\n 14 - Вывести студентов старше 20 лет" +

"\n 15 - Найти минимальный возраст студента на факультете \n 16 - Выход");

choice = int.Parse(Console.ReadLine());

switch (choice)

{

case 1:

Helper.AddStud(records);

break;

case 2:

foreach (var student in records)

{

Console.WriteLine(student + " ");

}

break;

case 3:

Helper.WriteFile(records, path);

break;

case 4:

Helper.ReadFile(records, path);

break;

case 5:

Console.WriteLine("Номер студента, которого хотите найти: ");

number = int.Parse(Console.ReadLine());

records.Search(records, number);

break;

case 6:

Console.WriteLine("Номер студента, данные о котором хотите удалить: ");

number = int.Parse(Console.ReadLine());

records.Remove(records, number);

break;

case 7:

Console.WriteLine("Номер студента, данные о котором хотите отредактировать: ");

number = int.Parse(Console.ReadLine());

Console.WriteLine("Что хотите отредактировать? (1-имя, 2 - фамилию, 3 - отчество, 4 - день рождения, 5 - дата поступления, " +

"6 - индекс группы, 7 - факультет, 8 - специальность, 9 - успеваемость");

int n;

string str;

n = int.Parse(Console.ReadLine());

Console.WriteLine("Введите новые данные в соответствующем формате : ");

str = Console.ReadLine();

records.Edit(records, number, n, str);

break;

case 8:

Console.WriteLine("Номер студента, чью группу хотите узнать: ");

number = int.Parse(Console.ReadLine());

Helper.Group(records, number);

break;

case 9:

Console.WriteLine("Номер студента, чей номер курса и семестра на текущий момент хотите узнать: ");

number = int.Parse(Console.ReadLine());

Helper.Course(records, number);

break;

case 10:

Console.WriteLine("Номер студента, чей текущий возраст хотите узнать: ");

number = int.Parse(Console.ReadLine());

DateTime today = DateTime.Today;

DateTime b = records.students[number - 1].Birthday;

TimeSpan old = today.Subtract(b);

var d = new DateTime(old.Ticks);

Console.WriteLine($"Возраст: {d.Year - 1} лет, {d.Month - 1} месяцев, {d.Day - 1} дней");

break;

case 11:

Console.WriteLine("Введите факультет, возраст студентов которого хотите узнать: ");

string str3 = Console.ReadLine();

var selectedItems = from t in records.students

where t.Faculty.Equals(str3)

select t.Birthday.Year;

Console.WriteLine("Возраст студентов: ");

foreach (int y in selectedItems)

Console.Write(y + " ");

Console.WriteLine();

break;

case 12:

Console.WriteLine("Введите факультет, названия групп студентов которого хотите узнать: ");

string str4 = Console.ReadLine();

var selectedItems1 = from t in records.students

where t.Faculty.Equals(str4)

select t.Faculty + t.Specialty + "-" + t.Date.Year + t.Index;

Console.WriteLine("Группы студентов: ");

foreach (string s in selectedItems1)

Console.Write(s + " ");

Console.WriteLine();

break;

case 13:

Console.WriteLine("Введите факультет, номер курса студентов которого хотите узнать: ");

string str5 = Console.ReadLine();

List<int> selectedItems2 =

(from t in records.students

where t.Faculty.Equals(str5)

select DateTime.Today.Year - t.Date.Year).ToList();

Console.WriteLine("Курс студентов: ");

foreach (int s in selectedItems2)

Console.Write(s + " ");

Console.WriteLine();

break;

case 14:

var selectedStud = records.students.Where(student => student.Age(student) >= 20);

foreach (Student stud in selectedStud)

Console.WriteLine($"{stud.Name} {stud.Lastname} - {stud.Age(stud)}");

break;

case 15:

Console.WriteLine("Введите факультет, минимальный возраст студентов которого хотите узнать: ");

string str6 = Console.ReadLine();

var selectedStud1 = (from t in records.students

where t.Faculty.Equals(str6)

select t.Age(t)).Min();

Console.WriteLine($"Минимальный возраст студента: {selectedStud1}");

break;

case 16:

loop = false;

break;

}

}

}

}

}

**Container.cs**

using System;

using System.Collections;

namespace Sharma07

{

public class Container : IEnumerable, IEnumerator

{

static int size = 1;

public Student[] students = new Student[size];

int index = -1;

int currentSize = 0;

// Реализуем интерфейс IEnumerable

public IEnumerator GetEnumerator()

{

return this;

}

// Реализуем интерфейс IEnumerator

public bool MoveNext()

{

if (index == students.Length - 1)

{

Reset();

return false;

}

index++;

return true;

}

public void Reset()

{

index = -1;

}

public object Current

{

get

{

return students[index];

}

}

public void Add(Student student)

{

currentSize = NewSize();

if (currentSize < size)

{

students[currentSize] = student;

}

else

{

size = currentSize + 1;

Array.Resize(ref students, size);

students[size - 1] = student;

}

}

int NewSize()

{

currentSize = 0;

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

{

if (students[i] != null)

currentSize++;

}

return currentSize;

}

public void Search(Container records, int number)

{

Console.WriteLine("Студент по индексу :");

Console.Write(records.students[number - 1] + " ");

}

public void Remove(Container records, int number)

{

currentSize = NewSize();

students[number - 1] = null;

for (int i = number; i < currentSize; i++)

{

if (students[i] != null)

{

students[i - 1] = students[i];

}

}

Array.Resize(ref students, currentSize - 1);

Console.WriteLine("Измененный список студентов :");

foreach (var student in records)

{

Console.WriteLine(student + " ");

}

}

public void Edit(Container records, int number, int n, string str)

{

switch (n)

{

case 1:

records.students[number - 1].Name = str;

break;

case 2:

records.students[number - 1].Lastname = str;

break;

case 3:

records.students[number - 1].Patronym = str;

break;

case 4:

records.students[number - 1].Birthday = DateTime.Parse(str);

break;

case 5:

records.students[number - 1].Date = DateTime.Parse(str);

break;

case 6:

records.students[number - 1].Index = char.Parse(str);

break;

case 7:

records.students[number - 1].Faculty = str;

break;

case 8:

records.students[number - 1].Specialty = int.Parse(str);

break;

case 9:

records.students[number - 1].Progress = int.Parse(str);

break;

}

Console.WriteLine("Измененный список студентов :");

foreach (var student in records)

{

Console.WriteLine(student + " ");

}

}

public int Age(Container records, int i)

{

int year = records.students[i].Birthday.Year;

int age = DateTime.Now.Year - year;

return age;

}

}

}

**Helper.cs**

using System;

using System.IO;

using System.Text;

namespace Sharma07

{

class Helper

{

public static void AddStud(Container records)

{

string name, lastname, patronym, faculty;

DateTime birthday, date;

int progress, specialty;

char index;

Console.WriteLine("Имя: ");

name = Console.ReadLine();

Console.WriteLine("Фамилия: ");

lastname = Console.ReadLine();

Console.WriteLine("Отчество: ");

patronym = Console.ReadLine();

Console.WriteLine("День рождения: ");

birthday = DateTime.Parse(Console.ReadLine());

Console.WriteLine("Дата поступления: ");

date = DateTime.Parse(Console.ReadLine());

Console.WriteLine("Индекс группы: ");

index = char.Parse(Console.ReadLine());

Console.WriteLine("Факультет: ");

faculty = Console.ReadLine();

Console.WriteLine("Специальность: ");

specialty = int.Parse(Console.ReadLine());

Console.WriteLine("Успеваемость: ");

progress = int.Parse(Console.ReadLine());

var stud = new Student(name, lastname, patronym, birthday, date, index, faculty, specialty, progress);

records.Add(stud);

}

public static void ReadFile(Container records, string path)

{

string name, lastname, patronym, faculty,text;

DateTime birthday, date;

int progress, specialty;

char index;

try

{

using (StreamReader sr = new StreamReader(path))

{

text = sr.ReadToEnd();

string[] separatingStrings = { " ", "\r", "\n", ":", "00:00:00", "\t", "Студенты", "Имя", "Фамилия", "Отчество", "День рождения",

"Дата поступления", "Индекс группы", "Факультет", "Специальность", "Успеваемость"};

string[] words = text.Split(separatingStrings, System.StringSplitOptions.RemoveEmptyEntries);

for (int i = 0; i < words.Length / 9; i++)

{

name = words[9 \* i];

lastname = words[9 \* i + 1];

patronym = words[9 \* i + 2];

birthday = DateTime.Parse(words[9 \* i + 3]);

date = DateTime.Parse(words[9 \* i + 4]);

index = char.Parse(words[9 \* i + 5]);

faculty = words[9 \* i + 6];

specialty = int.Parse(words[9 \* i + 7]);

progress = int.Parse(words[9 \* i + 8]);

var stud = new Student(name, lastname, patronym, birthday, date, index, faculty, specialty, progress);

records.Add(stud);

}

}

}

catch (Exception e)

{

Console.WriteLine(e.Message);

}

}

public static void WriteFile(Container records, string path)

{

try

{

using (StreamWriter sw = new StreamWriter(path, false, System.Text.Encoding.Default))

{

sw.WriteLine("Студенты: ");

}

foreach (var student in records)

{

using StreamWriter sw = new StreamWriter(path, true, System.Text.Encoding.Default);

sw.WriteLine(student);

}

Console.WriteLine("Запись выполнена");

}

catch (Exception e)

{

Console.WriteLine(e.Message);

}

}

public static void Course(Container records, int number)

{

int course, semester;

course = DateTime.Today.Year - records.students[number - 1].Date.Year;

if (DateTime.Today.Month >= 7 && DateTime.Today.Month <= 12)

{

semester = course \* 2 - 1;

}

else

{

semester = course \* 2;

}

Console.WriteLine($"Курс : {course}, семестр : {semester}");

}

public static void Group(Container records, int number)

{

StringBuilder sb = new StringBuilder();

sb.Append(records.students[number - 1].Faculty);

sb.Append(records.students[number - 1].Specialty);

sb.Append("-");

sb.Append(records.students[number - 1].Date.Year);

sb.Append(records.students[number - 1].Index);

sb.AppendLine();

Console.WriteLine(sb.ToString());

}

}

}

**Student.cs**

using System;

namespace Sharma07

{

[Serializable]

public class Student

{

public string Name { get; set; }

public string Lastname { get; set; }

public string Patronym { get; set; }

public DateTime Birthday { get; set; }

public DateTime Date { get; set; }

public char Index { get; set; }

public string Faculty { get; set; }

public int Specialty { get; set; }

public int Progress { get; set; }

public Student() { }

public Student(string name, string lastname, string patronym, DateTime birthday, DateTime date, char index, string faculty, int specialty, int progress)

{

this.Name = name;

this.Lastname = lastname;

this.Patronym = patronym;

this.Birthday = birthday;

this.Date = date;

this.Index = index;

this.Faculty = faculty;

this.Specialty = specialty;

this.Progress = progress;

}

public override string ToString()

{

return $"{Name}\t {Lastname}\t {Patronym}\t {Birthday}\t {Date}\t {Index}\t {Faculty}\t {Specialty}\t {Progress} \n";

}

public int Age(Student stud)

{

int year = stud.Birthday.Year;

int age = DateTime.Now.Year - year;

return age;

}

}

}РЕЗУЛЬТАТ РОБОТИ ПРОГРАМИ

Результат роботи програми показано на рис. 1.

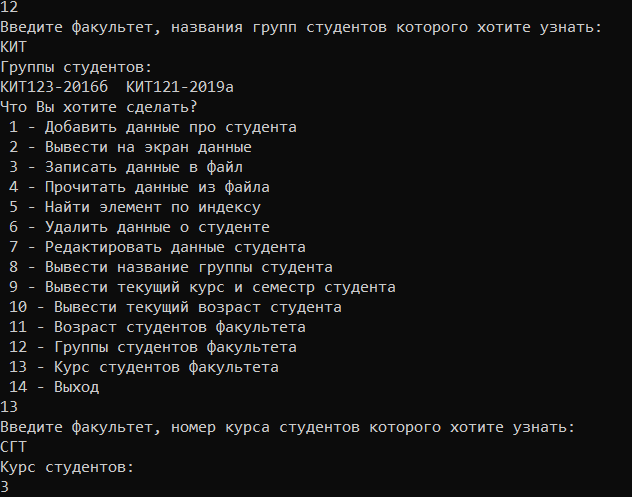


Рисунок 1 – Результат роботи програми

У даній лабораторній роботі був проведений рефакторинг, суть якого полягає у покращенні коду для його полегшеного сприйняття. Були видалені зайві, невикористовувані бібліотеки, змінні. Деякі функції були спрощені, але без зміни функціональностей коду.

У даній лабораторній роботі використовувалися різні варіанти застосування LINQ. Були використані варіанти відкладеного виконання запитів, примушеного виконання запитів.

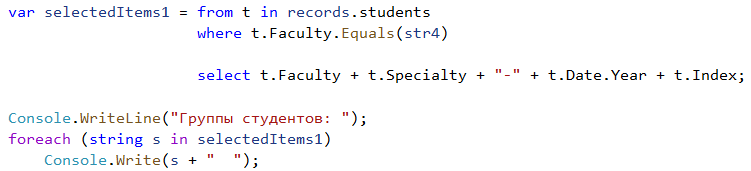


Рисунок 2 – Відкладене виконання запита LINQ

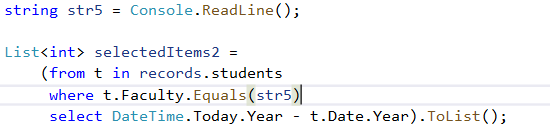


Рисунок 3 – Примушене виконання запита LINQ

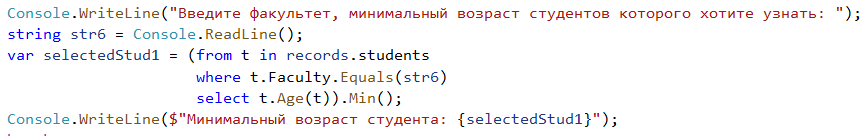


Рисунок 4 – Використання статистичних функцій LINQ

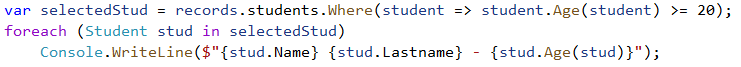


Рисунок 5 – Використання лямбда-виразів та змішаний синтаксис запросу та методу(Age)

При відкладеному виконанні LINQ-вираз не виконується, поки не буде проведена ітерація або перебір по вибірці. За допомогою ряду методів ми можемо застосувати негайне виконання запиту. Це методи, які повертають одне атомарному значення або один елемент. Наприклад, Count (), Average (), First () / FirstOrDefault (), Min (), Max () і т.д. Також для негайного виконання LINQ-запиту і кешування його результатів ми можемо застосовувати методи перетворення ToArray <T> (), ToList <T> (), ToDictionary () і т.д .. Ці методи отримують результат запиту у вигляді об'єктів Array, List і Dictionary відповідно.

ВИСНОВОК

Під час виконання даної лабораторної роботи було набуто навичок рефакторингу та використовування різних варіантів застосування LINQ.