НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Кафедра «ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ ТА ПРОГРАМУВАННЯ»

«DotNet»

*Звiт з лабораторної роботи №3*

*Тема: «Обробка колекцій об'єктів. Робота з файлами»*

Виконав:

ст. гр. КІТ-119а

Капелька Я. І.

Перевірив:

Бартош М. В.

Харків – 2021

**Тема:** Обробка колекцій об'єктів. Робота з файлами.

**Мета:** Оволодіння навичками обробки об’єктів колекцій та роботи з файлами на мові програмування C#.

**Розробник**: Капелька Ярослав, КІТ-119а, варіант №11.

**Опис програми:**

**Засоби ООП**: клас, метод класу, поле класу.

**Ієрархія та структура класів:** один публічний клас Main, публічний клас Student, у якого є поля: Firstname, Surname, GroupIndex, Faculty, Specialization, AcademicPerformance, DateOFBirth, DateOfEnter, Printer; гетери, сетери, конструктор класу та метод виведення даних класу. Інтерфейс IPrinter. Класс StudCont для створення та обробки колекції студентів.

**Важливі фрагменти програми:**

using System;

using System.IO;

namespace lab3

{

public class Program

{

static void Main(string[] args)

{

var studList = new StudCont();

studList.Add(new Student("Каркуша Дмитрий Евгениевич", new DateTime(2002, 4, 2), new DateTime(2019, 9, 1), "Б", 119, "КИТ", "Компьютерные игры", 95));

studList.Add(new Student("Черняева Влада Станислвовна", new DateTime(2001, 12, 8), new DateTime(2019, 9, 1), "А", 119, "КИТ", "Компьютерные игры", 94));

studList.Add(new Student("Капелька Ярослав Иванович", new DateTime(2002, 6, 7), new DateTime(2019, 9, 1), "А", 119, "КИТ", "Компьютерные игры", 67));

studList.Add(new Student("Аксьонов Александр Юрьевич", new DateTime(2002, 3, 10), new DateTime(2019, 9, 1), "В", 419, "КИТ", "АТМ", 75));

studList.Add(new Student("Момот Роман Анатолиевич", new DateTime(2002, 12, 24), new DateTime(2019, 9, 1), "В", 419, "КИТ", "АТМ", 71));

StudContHelper helper = new StudContHelper(studList);

helper.Write();

var studList2 = helper.Read();

studList2.PrintAll(false);

var name = "Капелька Ярослав Иванович";

var stud = studList2.GetStudent(name);

stud.Performance += 5;

studList2.ChangeStudent(name, stud);

Console.WriteLine("\nВывод изменненых данных: \n\n");

studList2.PrintAll(false);

File.WriteAllText("C:\\Users\\Admin\\source\\repos\\lab3\\output.txt", stud.ToString() + "\n");

Console.WriteLine("\n\nСчитывание данных с файла: ");

File.ReadAllText("C:\\Users\\Admin\\source\\repos\\lab3\\output.txt");

Console.ReadLine();

}

}

}

using System;

using System.Linq;

namespace lab3

{

public class Student

{

private readonly string \_name;

private readonly DateTime \_dateOfBirth;

private readonly DateTime \_dateOfAdmission;

private string \_groupIndex;

private int \_groupNum;

private string \_faculty;

private string \_specialty;

private int \_performance;

private string[] \_faculties = { "Э", "МИТ", "И", "ХТ", "БЭМ", "МО", "СГТ", "КН", "КИТ" };

public Student(string name, DateTime dateOfBirth, DateTime dateOfAdmission, string groupIndex, int groupNum, string faculty, string specialty, int performance)

{

\_name = name ?? throw new ArgumentNullException(nameof(name));

\_dateOfBirth = dateOfBirth;

\_dateOfAdmission = dateOfAdmission;

\_groupIndex = groupIndex ?? throw new ArgumentNullException(nameof(groupIndex));

\_groupNum = groupNum;

\_faculty = faculty ?? throw new ArgumentNullException(nameof(faculty));

\_specialty = specialty ?? throw new ArgumentNullException(nameof(specialty));

\_performance = performance;

}

public string Name { get { return \_name; } }

public DateTime DateOfBith { get { return \_dateOfBirth; } }

public DateTime DateOfAdmission { get { return \_dateOfAdmission; } }

public string GroupIndex

{

get { return \_groupIndex; }

private set

{

if (value.Length <= 2 && value.Length != 0)

{

\_groupIndex = value;

}

}

}

public int GroupNum { get { return \_groupNum; } private set { \_groupNum = value; } }

public string Faculty

{

get { return \_faculty; }

private set

{

if (\_faculties.Contains(value.ToUpper()))

{

\_faculty = value.ToUpper();

}

}

}

public string Specialty { get { return \_specialty; } private set { \_specialty = value; } }

public int Performance

{

get { return \_performance; }

set

{

if (value <= 100 && value >= 0)

{

\_performance = value;

}

}

}

public override string ToString()

{

string output = \_name + ";" + \_dateOfBirth.ToString() + ";" + \_dateOfAdmission + ";" + \_faculty + ";" + \_groupNum + ";" + \_groupIndex + ";" + \_specialty + ";" + \_performance;

return output;

}

public static Student ParseString(string line)

{

string[] arr = line.Split(";");

Student stud = new Student(arr[0], DateTime.Parse(arr[1]), DateTime.Parse(arr[2]), arr[5], Int32.Parse(arr[4]), arr[3], arr[6], Int32.Parse(arr[7]));

return stud;

}

public string ToText()

{

string output = "Ф.И.О.: " + \_name + "\nДата рождения: " + \_dateOfBirth.ToString() + "\nДата поступления: " + \_dateOfAdmission + "\nГруппа: " + \_faculty + "-" + \_groupNum + \_groupIndex + "\nСпециальность: " + \_specialty + "\nУспеваемость:" + \_performance + "%";

return output;

}

}

}

using System.IO;

using System.Linq;

namespace lab3

{

class StudContHelper

{

public string Path { get; set; }

public StudCont Container { get; set; }

public StudContHelper(StudCont cont = null, string path = "output.txt")

{

Path = path;

Container = cont;

}

public void Write(StudCont cont = null, string path = null)

{

StudCont rescont = cont ?? Container;

string respath = path ?? Path;

if (rescont != null)

{

File.WriteAllLines(respath, rescont.ToList());

}

}

public StudCont Read(string path = null)

{

string respath = path ?? Path;

string[] lines = File.ReadAllLines(respath);

StudCont rescont = new StudCont();

foreach (var line in lines)

{

rescont.Add(Student.ParseString(line));

}

return rescont;

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Collections;

namespace lab3

{

public class StudCont : IEnumerator, IEnumerable

{

private List<Student> list;

private int position = -1;

public StudCont()

{

list = new List<Student>();

}

public void Add(Student stud)

{

list.Add(stud);

}

public bool Delete(int index)

{

if (index < list.Count && index >= 0)

{

list.RemoveAt(index);

return true;

}

return false;

}

public void PrintAll(bool brief = true)

{

int i = 1;

foreach (Student stud in list)

{

if (!brief)

{

Console.WriteLine(i + "." + stud.ToText() + "\n");

}

else Console.WriteLine(i + "." + stud.Name + "\n");

i++;

}

}

public Student GetStudent(int index)

{

if (index < list.Count && index >= 0)

{

return list.ElementAt<Student>(index);

}

else throw new IndexOutOfRangeException();

}

public bool ChangeStudent(int index, Student stud)

{

if (index < list.Count && index >= 0)

{

list[index] = stud;

return true;

}

else return false;

}

public bool ChangeStudent(string name, Student stud)

{

int i = 0;

foreach (Student stu in list)

{

if (stu.Name == name)

{

list[i] = stud;

return true;

}

i++;

}

return false;

}

public Student GetStudent(string name)

{

foreach (Student stud in list)

{

if (stud.Name == name)

{

return stud;

}

}

return null;

}

public int Count()

{

return list.Count;

}

public IEnumerator GetEnumerator()

{

return (IEnumerator)this;

}

//IEnumerator

public bool MoveNext()

{

position++;

return (position < list.Count());

}

//IEnumerable

public void Reset()

{

position = 0;

}

//IEnumerable

public object Current

{

get { return list.ElementAt<Student>(position); }

}

public string[] ToList()

{

string[] lines = new string[Count()];

var i = 0;

foreach (Student stud in list)

{

lines[i] = stud.ToString();

i++;

}

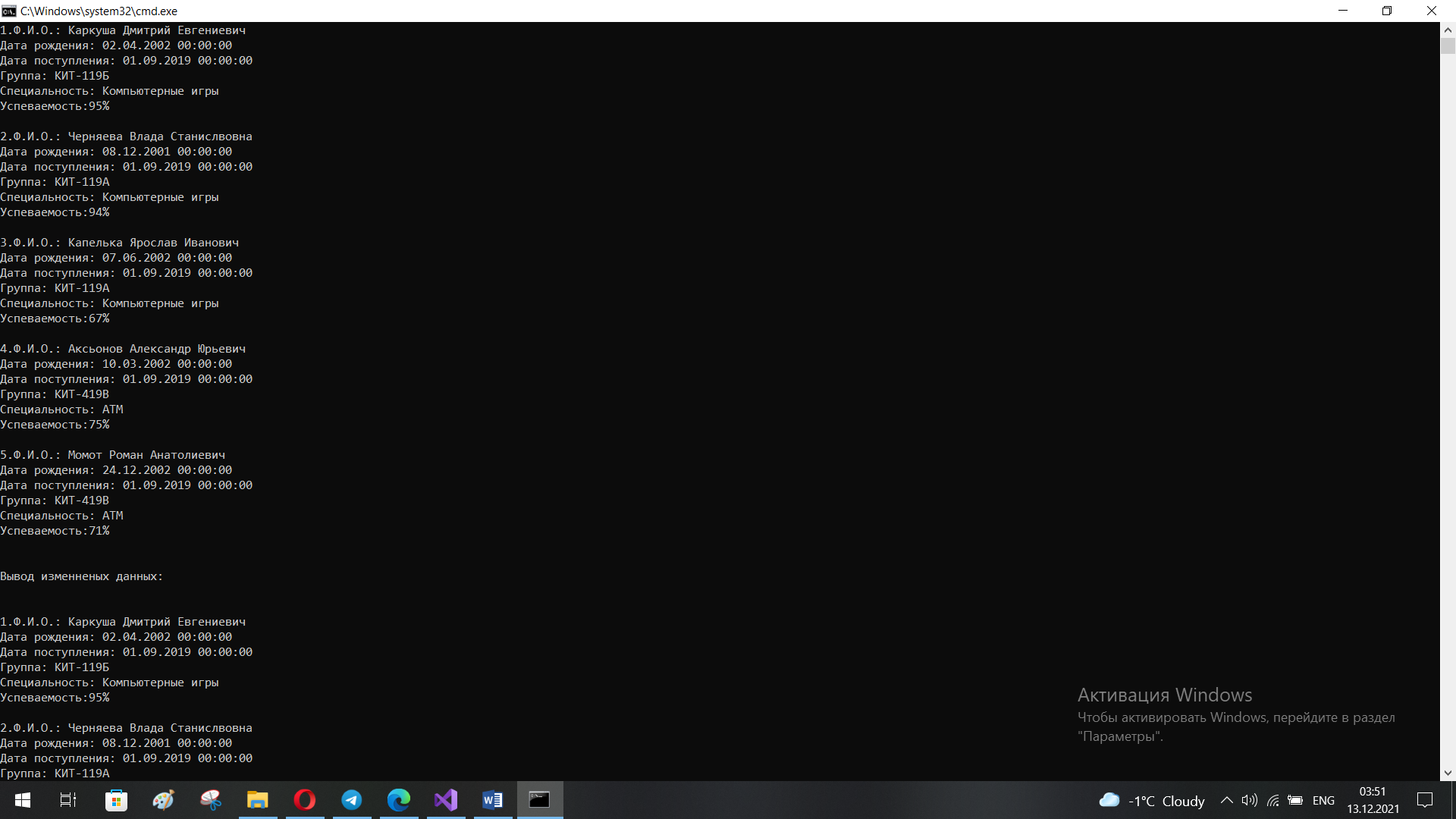
return lines;

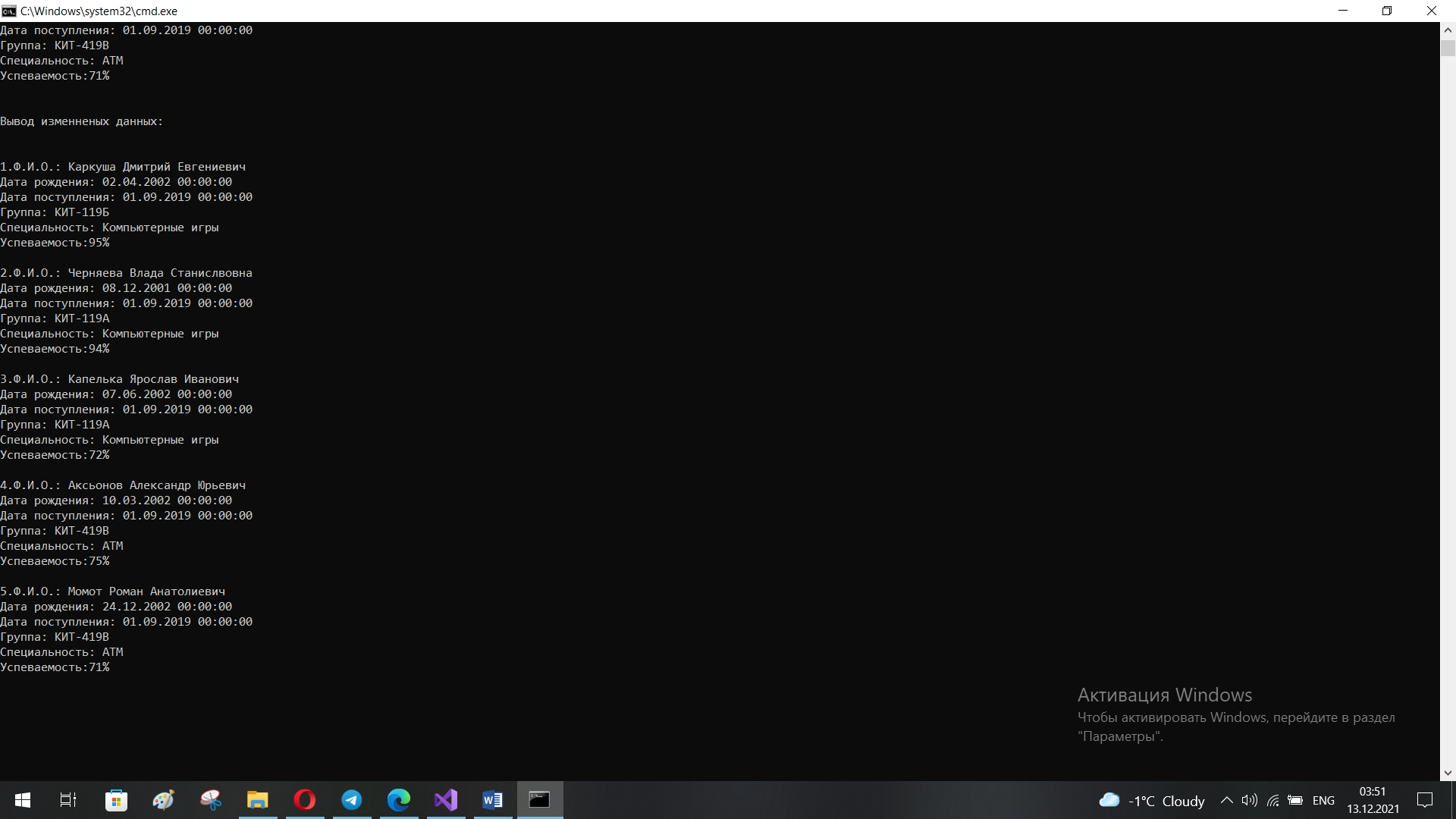
}

}

}

**Результат роботи програми:**





**Висновок:**

При виконанні даної лабораторної роботи було набуто навичок обробки об’єктів колекцій та роботи з файлами на мові програмування C#.