НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Кафедра «ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ ТА ПРОГРАМУВАННЯ»

«DotNet»

*Звiт з лабораторної роботи №8*

*Тема: «ООП. Реінжиніринг програм C»*

Виконав:

ст. гр. КІТ-119а

Капелька Я. І.

Перевірив:

Бартош М. В.

Харків – 2021

**Тема:** ООП. Реінжиніринг програм C.

**Мета:** Оволодіння навичками використання ООП та реінжинірингу програм 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.

Доданий StringBuilder для обробки класу Student. Також додано класс StudContCalculator, який містить методи рефакторингу коду. Додано клас Group, в який винесено поле faculties.

**Важливі фрагменти програми:**

using System;

using System.Collections.Generic;

using System.Text.Json;

namespace lab8

{

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));

Console.WriteLine(StudContHelper.ToTable(studList));

studList.Reset();

studList.Sort(StudCont.SortType.Performance);

Console.WriteLine("Сортировка по успеваемости.\n");

studList.Reset();

Console.WriteLine(StudContHelper.ToTable(studList));

Console.ReadLine();

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace lab8

{

public class Group

{

public readonly int \_groupNum;

public readonly string \_faculty;

public readonly string \_groupIndex;

public readonly string[] faculties = { "Э", "МИТ", "И", "ХТ", "БЭМ", "МО", "СГТ", "КН", "КИТ", "CIT" };

public string FullName

{

get

{

return $"{\_faculty}-{\_groupNum}{\_groupIndex}";

}

}

public Group(int groupNum, string faculty, string groupIndex)

{

if (faculties.Contains(faculty.ToUpper()))

{

\_groupIndex = groupIndex;

\_faculty = faculty;

\_groupNum = groupNum;

}

else throw new Exception("faculty value is not in facultie names list");

}

public override bool Equals(object obj)

{

if (obj.GetType() == this.GetType())

{

Group other = (Group)obj;

return other.FullName == FullName;

}

return false;

}

public override string ToString()

{

return $"{\_faculty};{\_groupNum};{\_groupIndex}";

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Collections;

namespace lab8

{

public class StudCont : IEnumerator, IEnumerable

{

private List<Student> list;

private int position = -1;

public enum SortType

{

Name,

Age,

Group,

Year,

Performance

}

public StudCont()

{

list = new List<Student>();

}

public StudCont(List<Student> studList)

{

list = studList;

}

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;

IEnumerable<string> query =

from Student stud in list

select brief ? stud.Name : stud.ToText();

foreach (string line in query)

{

//if (!brief) Console.WriteLine(i + "." + stud.ToText() + "\n");

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

Console.WriteLine(i + "." + line + "\n");

i++;

}

}

public Student GetStudent(int index)

{

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

//{

// return list.ElementAt<Student>(index);

//}

//else throw new IndexOutOfRangeException();

IEnumerable<Student> query =

from Student stud in list

where list.IndexOf(stud) == index

select stud;

return query.First();

}

public Student GetStudent(string name)

{

IEnumerable<Student> query =

from Student stud in list

where stud.Name == name

select stud;

return query.First();

}

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 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 = -1;

}

//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;

}

public void RemoveByComp(Student.Compare comp, string line)

{

//foreach(Student stud in list)

//{

// if(comp(stud, line))

// {

// list.Remove(stud);

// }

//}

IEnumerable<Student> query =

from Student stud in list

where comp(stud, line)

select stud;

foreach (Student stud in query)

{

list.Remove(stud);

}

}

public void Sort(SortType type)

{

IOrderedEnumerable<Student> result;

switch (type)

{

case SortType.Name:

result = from s in list

orderby s.Name

select s;

list = result.ToList();

break;

case SortType.Age:

result = from s in list

orderby s.Age

select s;

list = result.ToList();

break;

case SortType.Performance:

result = from s in list

orderby s.Performance

select s;

list = result.ToList();

break;

case SortType.Group:

result = from s in list

orderby s.\_group

select s;

list = result.ToList();

break;

case SortType.Year:

result = from s in list

orderby s.Year

select s;

list = result.ToList();

break;

}

}

public void Unite(StudCont cont)

{

var res = from Student s in cont

select s;

foreach (Student s in res)

{

Add(s);

}

}

}

}

using System;

using System.IO;

using System.Linq;

namespace lab8

{

public class StudContHelper

{

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

{

}

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

{

string respath = path;

if (cont != null)

{

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

}

}

public static StudCont Read(string path = null)

{

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

StudCont rescont = new StudCont();

foreach (var line in lines)

{

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

}

return rescont;

}

public static string ToTable(StudCont cont)

{

string result = $"|{"Ф.И.О",-35}|{"Группа",-10}|{"Успеваемость",-15}|\n";

result += "|--------------------------------------------------------------|\n";

foreach (Student stud in cont)

{

result += $"|{stud.Name,-35}|{stud.\_group,-10}|{stud.Performance,-15}|\n";

result += "|--------------------------------------------------------------|\n";

}

return result;

}

public static string ToTable(StudCont cont, Student.Compare comp, string line)

{

string result = $"|{"Ф.И.О",-35}|{"Группа",-10}|{"Успеваемость",-15}|\n";

result += "|--------------------------------------------------------------|\n";

foreach (Student stud in cont)

{

if (comp(stud, line))

{

result += $"|{stud.Name,-35}|{stud.\_group,-10}|{stud.Performance,-15}|\n";

result += "|--------------------------------------------------------------|\n";

};

}

return result;

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace lab8

{

class StudContCalculator

{

public StudCont Container { get; set; }

public StudContCalculator(StudCont cont = null)

{

Container = cont;

}

public float AverageAge(Student.Compare comp, string line)

{

float total = 0;

int count = 0;

//foreach(Student stud in Container)

//{

// if(comp(stud, line))

// {

// total += stud.Age;

// count++;

// }

//}

//return total / count;

IEnumerable<int> query =

from Student stud in Container

where comp(stud, line)

select stud.Age;

foreach (var age in query)

{

total += age;

count++;

}

return total / count;

}

public float AveragePerformance(Student.Compare comp, string line)

{

float total = 0;

int count = 0;

//foreach (Student stud in Container)

//{

// if (comp(stud, line))

// {

// total += stud.Performance;

// count++;

// }

//}

IEnumerable<int> query =

from Student stud in Container

where comp(stud, line)

select stud.Performance;

foreach (var age in query)

{

total += age;

count++;

}

return total / count;

}

public int CountByComp(Student.Compare comp, string line)

{

int result =

(from Student stud in Container

where comp(stud, line)

select stud).Count();

return result;

}

public static int CountByComp(StudCont Container, Student.Compare comp, string line)

{

int result =

(from Student stud in Container

where comp(stud, line)

select stud).Count();

return result;

}

public static float AveragePerformance(StudCont Container, Student.Compare comp, string line)

{

float total = 0;

int count = 0;

//foreach (Student stud in Container)

//{

// if (comp(stud, line))

// {

// total += stud.Performance;

// count++;

// }

//}

IEnumerable<int> query =

from Student stud in Container

where comp(stud, line)

select stud.Performance;

foreach (var age in query)

{

total += age;

count++;

}

return total / count;

}

public static float AverageAge(StudCont Container, Student.Compare comp, string line)

{

float total = 0;

int count = 0;

//foreach(Student stud in Container)

//{

// if(comp(stud, line))

// {

// total += stud.Age;

// count++;

// }

//}

//return total / count;

IEnumerable<int> query =

from Student stud in Container

where comp(stud, line)

select stud.Age;

foreach (var age in query)

{

total += age;

count++;

}

return total / count;

}

}

}

using System;

using System.Linq;

using System.Collections;

using System.Text;

namespace lab8

{

public class Student

{

private string \_name;

private DateTime \_dateOfBirth;

private DateTime \_dateOfAdmission;

private string \_specialty;

private int \_performance;

public Group \_group;

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

public delegate bool Compare(Student stud, string line);

public static bool CompareGroup(Student stud, string line)

{

if (stud.\_group.FullName == line) return true;

else return false;

}

public static bool CompareSpecialty(Student stud, string line)

{

if (stud.Specialty == line) return true;

else return false;

}

public static bool CompareFaculty(Student stud, string line)

{

if (stud.\_group.\_faculty == line) return true;

else return false;

}

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;

\_group = new Group(groupNum, faculty, groupIndex);

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

\_performance = performance;

}

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

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

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

public int Year

{

get

{

int year = (int)((DateTime.Now - \_dateOfAdmission).TotalDays / 365.2425) + 1;

if (year > 6) year = 6;

else if (year < 0)

{

year = 0;

}

return year;

}

}

public int Semester

{

get

{

int day = (int)((DateTime.Now - \_dateOfAdmission).TotalDays % 365.2425);

if (day < 150)

{

return 1;

}

else return 2;

}

}

public int Age

{

get

{

return (int)((DateTime.Now - \_dateOfBirth).TotalDays / 365.2425);

}

}

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};{\_group.ToString()};{\_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: {\_group.FullName}\nСпециальность: {\_specialty}\nУспеваемость: {\_performance}%";

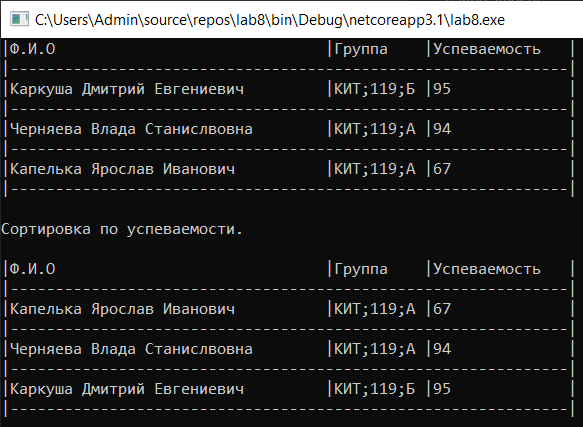
return output;

}

}

}

**Результат роботи програми:**



**Висновок:**

При виконанні даної лабораторної роботи було набуто навичок використання ООП та реінжинірингу програм C.