|  |  |  |
| --- | --- | --- |
| Министерство науки и высшего образования  Российской Федерации | | |
| Федеральное государственное бюджетное  образовательное учреждение высшего образования | | |
| «Новосибирский государственный технический университет» | | |
|  | | |
| Кафедра прикладной математики | | |
|  | | |
| Практическое задание № 2 | | |
| по дисциплине «объектно-ориентированное программирование с | | |
| использованием С++/C#» | | |
|  | | |
|  | | |
|  | Факультет: | ПМИ |
| Группа: | ПМ-71 |
| Студент: | Востриков Вячеслав |
| Преподаватель: | Ступаков Илья Михайлович |
|  |  |
|  |  |
|  |  |
|  |  |
|  | | |
| Новосибирск | | |
| 2019 | | |

1. Условие задачи
2. Сделать интерфейс IPerson

interface IPerson

{

   string Name { get; }

   string Patronomic { get; }

   string Lastname { get; }

   DateTime Date { get; }

   int Age { get; }

}

1. Сделать классы Student и Teacher реализующие этот интерфейс. Они должны содержать конструктор, принимающий значения свойств, статическую функцию создания из строки и переопределение функции ToString.
   1. У класса Student дополнительно должны быть свойства: курс, группа и средний балл.
   2. У класса Teacher дополнительно должны быть свойства: кафедра, стаж и должность, причем должность должна быть типом enum.
2. Сделать класс University, реализующий свойства и методы из следующего интерфейса.

interface IUniversity

{

   IEnumerable<IPerson> Persons { get; }   // отсортировать в соответствии с вариантом 1-й лабы

   IEnumerable<Student> Students { get; }  // отсортировать в соответствии с вариантом 1-й лабы

   IEnumerable<Teacher> Teachers { get; }  // отсортировать в соответствии с вариантом 1-й лабы

   void Add(IPerson person);

   void Remove(IPerson person);

   IEnumerable<IPerson> FindByLastName(string lastName);

   // Для нечетных вариантов. Выдать всех студентов, чей средний балл выше заданного.

   // Отсортировать по среднему баллу

   IEnumerable<Student > FindByAvrPoint(float avrPoint);

   // Для четных вариантов. Выдать всех преподавателей, название кафедры которых содержит

   // заданный текст. Отсортировать по должности.

   IEnumerable<Teacher> FindByDepartment(string text);

}

Сделать головную программу с консольным меню, которая бы демонстрировала работу всего вышеперечисленного.

1. Код программы

using System;

using System.IO;

using System.Collections.Generic;

using System.Linq;

namespace Task\_2

{

interface IPerson

{

string Name { get; }

string Surname { get; }

string Lastname { get; }

DateTime Date { get; }

int Age { get; }

}

class University

{

List<IPerson> persons = new List<IPerson>();

public IEnumerable<Student> Students => persons.OfType<Student>().OrderByDescending(s => s.Surname);

public IEnumerable<Teacher> Teachers => persons.OfType<Teacher>().OrderByDescending(y => y.Surname);

public IEnumerable<IPerson> Persons => persons.OrderByDescending(x => x.Surname);

public IEnumerable<IPerson> Find(string str)

{

return persons.Where(s => (s.Surname == str));

}

public List<IPerson> Delete(IPerson peop)

{

persons.Remove(peop);

return persons;

}

public List<IPerson> Add(IPerson pers)

{

persons.Add(pers);

return persons;

}

public IEnumerable<Student> FindSt(float t)

{

return persons.OfType<Student>().Where(s => s.Avr\_score >= t).OrderByDescending(s => s.Avr\_score);

}

}

public class Student : IPerson

{

public int Age => Calc.CalcAge(Date);

public int Course { get; }

public string Name { get; }

public string Surname { get; }

public string Lastname { get; }

public float Avr\_score { get; }

public string Group { get; }

public DateTime Date { get; }

public Student(string surname, string name, string lastname, DateTime date, int course, string group, float avr\_score)

{

this.Name = name;

this.Surname = surname;

this.Lastname = lastname;

this.Course = course;

this.Group = group;

this.Avr\_score = avr\_score;

this.Date = date;

}

public static Student Create(string i)

{

string[] things = i.Split(new char[] { ' ' }, StringSplitOptions.RemoveEmptyEntries);

int course = Convert.ToInt32(things[4], 10);

float avr\_score = (float)Convert.ToDouble(things[6]);

DateTime date = Convert.ToDateTime(things[3]);

Student student = new Student(things[0], things[1], things[2], date, course, things[5], avr\_score);

return student;

}

override public String ToString()

{

return $"student Name: {Surname, 15} {Name, 10} {Lastname, 15}, BirthDate:{Date: dd.MM.yyyy}, Age:{Age, 4}, Course:{Course, 2}, Group:{Group, 7}, Avr\_score:{Avr\_score, 5:f2}";

}

}

public class Teacher : IPerson

{

public int Age => Calc.CalcAge(Date);

public float Years { get; set; }

public string Name { get; }

public string Surname { get; }

public string Lastname { get; }

public enum Dolgnost { Assist, Docent, Professor, St\_Prepod, Zav\_kaf } //

public Dolgnost Dolg { get; }

public string Department { get; }

public DateTime Date { get; }

public Teacher(string surname, string name, string lastname, DateTime date, string department, float years, Dolgnost dolg)

{

this.Name = name;

this.Surname = surname;

this.Lastname = lastname;

this.Years = years;

this.Department = department;

this.Dolg = dolg;

this.Date = date;

}

public static Teacher Create(string i)

{

string[] things = i.Split(new char[] { ' ' }, StringSplitOptions.RemoveEmptyEntries);

float years = (float)Convert.ToDouble(things[5]);

Dolgnost dolg = (Dolgnost)Enum.Parse(typeof(Dolgnost), things[6]);

DateTime date = Convert.ToDateTime(things[3]);

Teacher teacher = new Teacher(things[0], things[1], things[2], date, things[4], years, dolg);

return teacher;

}

override public String ToString()

{

return $"teacher Name: {Surname,15} {Name,10} {Lastname,15}, BirthDate:{Date: dd.MM.yyyy}, Age:{Age,4}, Department:{Department,7}, Years:{Years,7:f1}, Dolg:{Dolg,-3}";

}

}

static class Calc

{

public static int CalcAge(DateTime Date)

{

DateTime dn = DateTime.Now;

if (dn.Month > Date.Month)

return (dn.Year - Date.Year);

else if (dn.Month == Date.Month)

{

if (dn.Day < Date.Day)

return (dn.Year - Date.Year - 1);

else

return (dn.Year - Date.Year);

}

else return (dn.Year - Date.Year - 1);

}

}

class Program

{

public static IPerson ReadPerson(string i)

{

string

flag, k;

IPerson t;

flag = i.Substring(0, 1);

k = i.Substring(2);

if (flag == "1")

t = Student.Create(k);

else

t = Teacher.Create(k);

return t;

}

public static void write\_stud(IEnumerable<Student> students)

{

foreach (var a in students)

Console.WriteLine(a);

}

public static void write\_teach(IEnumerable<Teacher> teachers)

{

foreach (var a in teachers)

Console.WriteLine(a);

}

public static void write\_all(IEnumerable<IPerson> persons)

{

foreach (var a in persons)

Console.WriteLine(a);

}

static void Main(string[] args)

{

string[] text = File.ReadAllLines(@"info.txt");

University Univer = new University();

foreach (var b in text)

Univer.Add(ReadPerson(b));

Console.WriteLine("Выберите операцию.");

Console.WriteLine("0 - Выход, 1 - Вывести студентов, 2 - Вывести преподавателей, 3 - Вывести всех, Вывести по среднему баллу, 5 - Поиск по фамилии.");

int a = Convert.ToInt32(Console.ReadLine());

while (a != 0)

{

switch (a)

{

case 1:

{

write\_stud(Univer.Students);

break;

}

case 2:

{

write\_teach(Univer.Teachers);

break;

}

case 3:

{

write\_all(Univer.Persons);

break;

}

case 4:

{

Console.WriteLine("Введите бал: ");

float t = (float)Convert.ToDouble(Console.ReadLine());

IEnumerable<Student> B = Univer.FindSt(t);

write\_all(B);

break;

}

case 5:

{

Console.WriteLine("Введите фамилию: ");

string str = Console.ReadLine();

IEnumerable<IPerson> sup = Univer.Find(str);

if (sup.Any())

{

write\_all(sup);

Console.WriteLine("Удалить из списка? (1 - да, 0 - нет)");

int k = Convert.ToInt32(Console.ReadLine());

if (k == 1)

Univer.Delete(sup.First());

write\_all(Univer.Persons);

}

else

Console.WriteLine("Не найдено.\n");

break;

}

default: { break; }

}

Console.WriteLine("Выберите операцию.");

Console.WriteLine("0 - Выход, 1 - Вывести студентов, 2 - Вывести преподавателей, 3 - Вывести всех, Вывести по среднему баллу, 5 - Поиск по фамилии.");

a = Convert.ToInt32(Console.ReadLine());

}

}

}

}

1. Результаты работы программы

Исходные данные в info.txt:



Работа программы:

