МИНИСТЕРСТВО НАУКИ
И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

НГТУ НЭТИ

Кафедра прикладной математики

Практическое задание № 2

по дисциплине «Разработка объектно-ориентированных программ с использованием C#/C++»

ПРИВЕТ, ИНТЕРФЕЙСЫ!

ФПМИ

Бригада №8	ПМ-92 БЕГИЧЕВ АЛЕКСАНДР
	ПМ-92 ШИШКИН НИКИТА
Преподаватель	СТУПАКОВ ИЛЬЯ МИХАЙЛОВИЧ
Дата	17.10.2020

Новосибирск

Вариант №8

Программа была протестирована на вшивость. Она работает.
В качестве обрабатываемых данных на вход программе подаётся два файла: `students.txt` и `teachers.txt`.

`students.txt`

```
Александр Вячеславович Бегичев 2 пм92 3.9 01/01/2000
Никита КомуКакоеДело Шишкин 2 пм92 4.1 01/01/2000
Андрей Андреевич Олегов 3 аниме 4.8 01/01/1999
```

`teachers.txt`

```
Иван Васильевич Пупкин ЛОЛ Dean 10/05/1985 20/10/1965
Андрей Васильевич Пупкин ЛОЛ Provost 01/07/1986 18/10/1965
Саша Андреевич Засранцев АНИМЕ Teacher 23/03/1990 18/10/1970
```

Код программы

```
using System;
using System.IO;
using System.Linq;
using System.Collections.Generic;
using System.Globalization;

namespace pz2
{
    class Program
    {
        static void Main(string[] args)
        {
            University university = new University();

            // Считываем из teachers.txt строки
            // и преобразует их в экземпляры класса Teacher
            // и запихивает их в университет
            foreach (var teacher in File.ReadAllLines("teachers.txt")
                .Select(teacherString => Teacher.Parse(teacherString)))
                university.Add(teacher);

            // Считываем из students.txt строки
            // и преобразует их в экземпляры класса Student
            // и призывает их в университет
            foreach (var student in File.ReadAllLines("students.txt")
                .Select(studentString => Student.Parse(studentString)))
                university.Add(student);

            // Выводим по очереди сначала всех персон,
            // а затем студентов и учителей
```

```
            Console.WriteLine("\nВсе персоны:");
            foreach(var person in university.Persons)
                Console.WriteLine(person.ToString());

            Console.WriteLine("\nВсе студенты:");
            foreach(var student in university.Students)
                Console.WriteLine(student.ToString());

            Console.WriteLine("\nВсе учителя:");
            foreach(var teacher in university.Teachers)
                Console.WriteLine(teacher.ToString());

            //university.Remove(university.Teachers);
            //university.Remove((Student) university.Students);

            Console.WriteLine("\nИщем всех Пупкиных");
            foreach (var person in university.FindByLastName("Пупкин"))
                Console.WriteLine(person.ToString());

            // Пускаем волну отчислений
            Console.WriteLine("\nПускаем волну отчислений");

            foreach(var student in university.Students)
                university.Remove(student);


            Console.WriteLine("Студентов как не было:");
            foreach(var person in university.Persons)
                Console.WriteLine(person.ToString());
        }
    }


    interface IPerson
    {
        string Name { get; }
        string Patronomic { get; }
        string LastName { get; }
        DateTime Date { get; }
        int Age { get; }
    }


    class Student : IPerson
    {
        public Student(string name, string patronomic, string lastName,
                ushort course, string group, decimal avarangeGrade, DateTime date)
        {
            Name = name;
            Patronomic = patronomic;
            LastName = lastName;
            Course = course;
```

```
            Group = group;
            AvarangeGrade = avarangeGrade;
            Date = date;
        }

        public static Student Parse(string studentString)
        {
            var data = studentString.Split();
            Student student = new Student(
                    data[0], data[1], data[2], ushort.Parse(data[3]),
                    data[4], decimal.Parse(data[5]),
                    DateTime.ParseExact(data[6], @"dd/MM/yyyy",
CultureInfo.InvariantCulture)
                    );

            return student;
        }

        public override string ToString()
        {
            string result = $"{Name} {Patronomic} {LastName}, {Course} c., {Group} gr.,
av. grade {AvarangeGrade}, birhday: {Date.ToString(@"dd\/MM\/yyyy")}";

            return result;
        }
        public string Name { set; get; }
        public string Patronomic { set; get; }
        public string LastName { set; get; }
        public DateTime Date { set; get; }
        public ushort Course { set; get; }
        public string Group { set; get; }
        public decimal AvarangeGrade { set; get; }
        public int Age
        {
            get
            {
                DateTime today = DateTime.Now;
                return today.Year - Date.Year +
                    ((Date.Month >= today.Month && Date.Day >= today.Day) ? -1 : 0);
            }
        }
    }



    class Teacher: IPerson
    {
        public enum Positions : ushort
        {
            None,
            Provost,
            Dean,
            DeputyDean,
```

```
            Teacher,
            GraduateStudent,
        }

        public Teacher(string name, string patronomic, string lastName,
                string department, Positions position,
                DateTime jobPlacement, DateTime date)
        {
            Name = name;
            LastName = lastName;
            Patronomic = patronomic;
            Department = department;
            Position = position;
            JobPlacement = jobPlacement;
            Date = date;
        }

        public static Teacher Parse(string teacherString)
        {
            var data = teacherString.Split();
            Teacher teacher = new Teacher(
                    data[0], data[1], data[2], data[3],
                    (Positions) Enum.Parse(typeof(Positions), data[4]),
                    DateTime.ParseExact(data[5], @"dd/MM/yyyy",
CultureInfo.InvariantCulture),
                    DateTime.ParseExact(data[6], @"dd/MM/yyyy",
CultureInfo.InvariantCulture)
                    );

            return teacher;
        }

        public override string ToString()
        {
            string result = $"{Name} {Patronomic} {LastName}, {Position}{Position ==
Positions.Provost ? "" : " of " + Department}, exp.: {Experience}, birhday:
{Date.ToString(@"dd/MM/yyyy")}";

            return result;
        }
        public string Name { set; get; }
        public string Patronomic { set; get; }
        public string LastName { set; get; }
        public DateTime Date { set; get; }
        public string Department { set; get; }
        public Positions Position { set; get; }
        public DateTime JobPlacement {set; get; }
        public int Experience
        {
            get
            {
                DateTime today = DateTime.Now;
                return today.Year - JobPlacement.Year +
```

```
                    ((JobPlacement.Month >= today.Month && JobPlacement.Day >=
today.Day) ? -1 : 0);
            }
        }
        public int Age
        {
            get
            {
                DateTime today = DateTime.Now;
                return today.Year - Date.Year +
                    ((Date.Month >= today.Month && Date.Day >= today.Day) ? -1 : 0);
            }
        }
    }


    interface IUniversity
    {
        void Add(IPerson person);
        void Remove(IPerson person);

        IEnumerable<IPerson> FindByLastName(string lastName);
        IEnumerable<Teacher> FindByDepartment(string text);

        IEnumerable<IPerson> Persons { get; }
        IEnumerable<Student> Students { get; }
        IEnumerable<Teacher> Teachers { get; }
    }


    class University
    {
        private List<IPerson> persons = new List<IPerson>();

        public void Add(IPerson person)
        {
            persons.Add(person);
        }

        public void Remove(IPerson person)
        {
            persons.Remove(person);
        }

        public IEnumerable<IPerson> FindByLastName(string lastName)
        {
            foreach (var person in persons.Where(person => person.LastName == lastName)
                    .OrderBy(person => person.LastName))
                yield return person;
            yield break;
        }

        public IEnumerable<Teacher> FindByDepartment(string text)
```

```
        {
            foreach (var teacher in persons.Where(person => person is Teacher)
                    .Select(person => (Teacher) person)
                    .OrderBy(teacher => teacher.Position))
                yield return teacher;
            yield break;
        }


        public IEnumerable<IPerson> Persons
        {
            get
            {
                foreach (var person in persons.OrderBy(person => person.Date))
                    yield return person;
                yield break;
            }
        }

        public IEnumerable<Student> Students
        {
            get
            {
                foreach (var student in persons.Where(person => person is Student)
                        .Select(person => (Student) person)
                        .OrderBy(student => student.Date))
                    yield return student;
                yield break;
            }
        }

        public IEnumerable<Teacher> Teachers
        {
            get
            {
                foreach (var teacher in persons.Where(person => person is Teacher)
                        .Select(person => (Teacher) person)
                        .OrderBy(teacher => teacher.Position))
                    yield return teacher;
                yield break;
            }
        }
    }}
```