Отчёт ЛР№1 Понкращенков Д.Б. ПИН-21Д.

Вариант №1.

Оглавление

1. **Задание** 2
2. **Результат работы программы** 4
3. **Листинг** 5

`Файл Program.cs` 5

`Файл Types.cs` 7

`Файл Exam.cs` 8

`Файл Person.cs` 9

`Файл Student.cs` 10

1. Задание

Определить тип **Education** – перечисление (enum) со значениями Specialist, Bachelor, SecondEducation.

Определить класс **Exam**, который имеет три открытых автореализуемых свойства, доступных для чтения и записи:

- свойство типа string, в котором хранится название предмета;
- свойство типа int, в котором хранится оценка;
- свойство типа System.DateTime для даты экзамена.

В классе **Exam** определить:

- конструктор с параметрами типа string, int и DateTime для инициализации всех свойств класса;
- конструктор без параметров, инициализирующий все свойства класса некоторыми значениями по умолчанию;
- перегруженную(override) версию виртуального метода string ToString() для формирования строки со значениями всех свойств класса.

Определить класс **Student**, который имеет

- закрытое поле типа Person, в котором хранятся данные студента;
- закрытое поле типа Education для информации о форме обучения;
- закрытое поле типа int для номера группы;
- закрытое поле типа Exam [] для информации об экзаменах, которые сдал студент.

В классе **Student** определить конструкторы:

- конструктор с параметрами типа Person, Education, int для инициализации соответствующих полей класса;
- конструктор без параметров, инициализирующий поля класса значениями по умолчанию.

В классе **Student** определить свойства с методами get и set:

- свойство типа Person для доступа к полю с данными студента;
- свойство типа Education для доступа к полю с формой обучения;
- свойство типа int для доступа к полю с номером группы;
- свойство типа Exam [] для доступа к полю со списком экзаменов.

В классе **Student** определить

- свойство типа double (только с методом get), в котором вычисляется средний балл как среднее значение оценок в списке сданных экзаменов;
- индексатор булевского типа (только с методом get) с одним параметром типа Education; значение индексатора равно true, если значение поля с формой обучения студента совпадает со значением индекса, и false в противном случае;
- метод void AddExams (params Exam []) для добавления элементов в список экзаменов;
- перегруженную версию виртуального метода string ToString() для формирования строки со значениями всех полей класса, включая список экзаменов;
- виртуальный метод string ToShortString(), который формирует строку со значениями всех полей класса без списка экзаменов, но со значением среднего балла.

В методе **Main()**

1. Создать один объект типа Student, преобразовать данные в текстовый вид с помощью метода ToShortString() и вывести данные.
2. Вывести значения индексатора для значений индекса Education.Specialist, Education.Bachelor и Education.SecondEducation.
3. Присвоить значения всем определенным в типе Student свойствам, преобразовать данные в текстовый вид с помощью метода ToString() и вывести данные.
4. С помощью метода AddExams(params Exam[]) добавить элементы в список экзаменов и вывести данные объекта Student, используя метод ToString().
5. Сравнить время выполнения операций с элементами одномерного, двумерного прямоугольного и двумерного ступенчатого массивов с одинаковым числом элементов типа Exam.

2. Результат работы программы

Выполнение задания по вариантам:

```
Example working:

ToShortString:   01.01.0001 0:00:00 Specialist 0, AvgRat: 0

isSpecialist: True
isBachelor: False
isSecondEducation: False

ToString: name surname 11.05.2024 19:52:50 Specialist 1, AvgRat: 0 Exams:


ToString after added exams:
name surname 11.05.2024 19:52:50 Specialist 1, AvgRat: 2 Exams:
- noData 0, Date:01.01.0001 0:00:00
- Subject1 4, Date:11.05.2024 0:00:00
```

Выполнение общего задания для всех вариантов:

```
Введите кол-во строк и столбцов массива, через',':
10,5

Диагностика времени выполнения доступа к эл-там массива с изменением номера группы студента:

Время выполнения foreach Array: 67 Ticks.
Время выполнения for Array: 6 Ticks.

Время выполнения foreach Array2D: 2264 Ticks.
Время выполнения for Array2D: 9 Ticks.

Время выполнения foreach Array2DStep: 7 Ticks.
Время выполнения for Array2DStep: 8 Ticks.
```

3. Листинг

```
//--------------------------------------------------------------------------------------||
// Файл Program.cs
//--------------------------------------------------------------------------------------||

namespace LR1
{
    internal class Program
    {
        static void Main(string[] args)
        {
            HelloUser();

            Console.WriteLine(" Введите кол-во строк и столбцов массива, через\',\':");
            var input = Console.ReadLine();
            var tmp = input.Split(',');

            int nRow = int.Parse(tmp[0]);
            int nCol = int.Parse(tmp[1]);

            //int nRow = 7;
            //int nCol = 3;

            #region --- Init arrays ---

            int AllElements = nRow * nCol;

            Student[] Array = new Student[AllElements];
            for (int i = 0; i < AllElements; i++)
            {
                Array[i] = new Student();
            }

            Student[,] Array2D = new Student[nCol, nRow];
            for (int i = 0; i < nCol; i++)
            {
                for (int j = 0; j < nRow; j++)
                {
                    Array2D[i, j] = new Student();
                }
            }

            Student[][] Array2DStep = new Student[nCol][];
            for (int i = 0; i < nCol; i++)
            {
                Array2DStep[i] = new Student[nRow];
                for (int j = 0; j < nRow; j++)
                {
                    Array2DStep[i][j] = new Student();
                }
            }

            #endregion

            System.Diagnostics.Stopwatch stopwatch = new System.Diagnostics.Stopwatch();

            //------------------------------------------------------------------------- ||
            Console.WriteLine("\n Диагностика времени выполнения доступа к эл-там массива с " +
                "изменением номера группы студента: \n");
            //------------------------------------------------------------------------- ||

            Student student0;

            stopwatch.Reset();
            stopwatch.Start();
            foreach (Student student in Array)
            {
                student.NumberGroup = 1;
            }
            stopwatch.Stop();
```

```
            Console.WriteLine(" Время выполнения foreach Array: " + stopwatch.ElapsedTicks + "
Ticks.");

            stopwatch.Reset();
            stopwatch.Start();
            for (int i = 0; i < AllElements; i++)
            {
                Array[i].NumberGroup = 1;
            }
            stopwatch.Stop();
            Console.WriteLine(" Время выполнения for Array: " + stopwatch.ElapsedTicks + "
Ticks.\n");

            //------------------------------------------------------------------------- ||

            stopwatch.Reset();
            stopwatch.Start();
            foreach (Student student in Array2D)
            {
                student.NumberGroup = 1;
            }
            stopwatch.Stop();
            Console.WriteLine(" Время выполнения foreach Array2D: " + stopwatch.ElapsedTicks + "
Ticks.");

            stopwatch.Reset();
            stopwatch.Start();
            for (int i = 0; i < nCol; i++)
            {
                for (int j = 0; j < nRow; j++)
                {
                    Array2D[i,j].NumberGroup = 1;
                }
            }
            stopwatch.Stop();
            Console.WriteLine(" Время выполнения for Array2D: " + stopwatch.ElapsedTicks + "
Ticks.\n");

            //------------------------------------------------------------------------- ||

            stopwatch.Reset();
            stopwatch.Start();
            foreach (var arr in Array2DStep)
            {
                foreach (Student student in arr)
                {
                    student.NumberGroup = 1;
                }
            }
            stopwatch.Stop();
            Console.WriteLine(" Время выполнения foreach Array2DStep: " + stopwatch.ElapsedTicks
+ " Ticks.");

            stopwatch.Reset();
            stopwatch.Start();
            for (int i = 0; i < nCol; i++)
            {
                for (int j = 0; j < nRow; j++)
                {
                    Array2DStep[i][j].NumberGroup = 1;
                }
            }
            stopwatch.Stop();
            Console.WriteLine(" Время выполнения for Array2DStep: " + stopwatch.ElapsedTicks + "
Ticks.");

            //------------------------------------------------------------------------- ||

            Console.ReadKey();
        }
```

```
        static void HelloUser()
        {
            Console.WriteLine(" Example working: \n");

            var stud = new Student();
            Console.WriteLine(" ToShortString: " + stud.ToShortString() + "\n");
            Console.WriteLine(" isSpecialist: " + stud[Education.Specialist]);
            Console.WriteLine(" isBachelor: " + stud[Education.Bachelor]);
            Console.WriteLine(" isSecondEducation: " + stud[Education.SecondEducation]);

            stud.Info = new Person("name", "surname", DateTime.Now);
            stud.NumberGroup = 1;
            stud.Education = Education.Specialist;

            Console.WriteLine("\n ToString: " + stud.ToString() + "\n");

            stud.AddExams([new Exam(), new Exam("Subject1", 4, DateTime.Today)]);

            Console.WriteLine(" ToString after added exams: \n " + stud.ToString() + "\n\n");
        }
    }
}


//-----------------------------------------------------------------------------------------||
// Файл Types.cs
//-----------------------------------------------------------------------------------------||

namespace LR1
{
    internal enum Education
    {
        Specialist,
        Bachelor,
        SecondEducation
    }
}
```

```
//-----------------------------------------------------------------------------------------||
// Файл Exam.cs
//-----------------------------------------------------------------------------------------||

namespace LR1
{
    internal class Exam
    {
        /// <summary>
        /// Название предмета
        /// </summary>
        internal string NameSubject{ get; set; }

        /// <summary>
        /// Оценка
        /// </summary>
        internal int Rating { get; set; }

        /// <summary>
        /// Дата экзамена
        /// </summary>
        internal DateTime DateExam { get; set; }

        /// <summary>
        /// Конструктор по умолчанию
        /// </summary>
        internal Exam() {
            NameSubject = "noData";
            Rating = 0;
            DateExam = DateTime.MinValue;
        }

        /// <summary>
        ///
        /// </summary>
        /// <param name="nameSubject"></param>
        /// <param name="rating"></param>
        /// <param name="dateExam"></param>
        internal Exam(string nameSubject, int rating, DateTime dateExam) {
            NameSubject= nameSubject;
            Rating= rating;
            DateExam= dateExam;
        }

        /// <summary>
        ///
        /// </summary>
        /// <returns></returns>
        public override string ToString()
        {
            return NameSubject + " " + Rating + ", Date:" + DateExam.ToString();
        }

    }
}
```

```
//-----------------------------------------------------------------------------------------||
// Файл Person.cs
//-----------------------------------------------------------------------------------------||

namespace LR1
{
    internal class Person
    {
        /// <summary>
        /// Имя
        /// </summary>
        string _Forename;
        /// <summary>
        /// Фамилия
        /// </summary>
        string _Surname;
        /// <summary>
        /// День рождения
        /// </summary>
        DateTime _BirthData;

        internal string Forename { get { return _Forename; } }
        internal string Surname { get { return _Surname; } }
        internal DateTime BirthData {
            get { return _BirthData; }
            set { _BirthData = value; }
        }


        /// <summary>
        /// Конструктор по умолчанию
        /// </summary>
        internal Person()
        {
            _Forename = "";
            _Surname = "";
            _BirthData = DateTime.MinValue;
        }

        /// <summary>
        ///
        /// </summary>
        /// <param name="forename"></param>
        /// <param name="surname"></param>
        /// <param name="birthData"></param>
        internal Person(string forename, string surname, DateTime birthData)
        {
            _Forename = forename;
            _Surname = surname;
            _BirthData = birthData;
        }

        /// <summary>
        ///
        /// </summary>
        /// <returns></returns>
        public override string ToString()
        {
            return _Forename + " " + _Surname + " " + _BirthData.ToString();
        }

        /// <summary>
        ///
        /// </summary>
        /// <returns></returns>
        public virtual string ToShortString()
        {
            return _Forename + " " + _Surname + " ";
        }
    }
}
```

```
//-----------------------------------------------------------------------------------------||
// Файл Student.cs
//-----------------------------------------------------------------------------------------||

namespace LR1
{
    internal class Student
    {
        /// <summary>
        /// Данные студента
        /// </summary>
        Person info;

        /// <summary>
        /// Форма обучения
        /// </summary>
        Education education;

        /// <summary>
        /// Номер группы
        /// </summary>
        int numberGroup;

        /// <summary>
        /// Закрытые экзамены
        /// </summary>
        Exam[] closedExams;

        /// <summary>
        /// Средний балл
        /// </summary>
        double AvgRating {
            get {
                if (closedExams == null || closedExams.Length == 0) return 0;

                return closedExams.Average((ex) => { return ex.Rating; });
            }
        }

        internal Person Info
        {
            get { return info; }
            set { info = value; }
        }
        internal Education Education
        {
            get { return education; }
            set { education = value; }
        }
        internal int NumberGroup
        {
            get { return numberGroup; }
            set { numberGroup = value; }
        }
        internal Exam[] ClosedExams
        {
            get { return closedExams; }
            set { closedExams = value; }
        }
        internal bool this[Education education]
        {
            get => education == this.education;
        }

        /// <summary>
        ///
        /// </summary>
        internal Student() {

            info = new();
            education = new ();
            numberGroup = 0;
```

```
            closedExams = new Exam[0];
        }

        /// <summary>
        ///
        /// </summary>
        /// <param name="info"></param>
        /// <param name="education"></param>
        /// <param name="numberGroup"></param>
        internal Student( Person info,  Education education, int numberGroup )
        {
            this.info = info;
            this.education = education;
            this.numberGroup = numberGroup;
        }

        /// <summary>
        /// Добавляет перечень экзаменов в коллекцию закрытых экзаменов
        /// </summary>
        /// <param name="exams">Закрытые экзамены</param>
        internal void AddExams(Exam[] exams)
        {
            if (closedExams != null)
            {
                if (exams != null)
                    closedExams = closedExams.Concat(exams).ToArray();
            }
            else
                closedExams = exams;
        }

        public override string ToString()
        {
            string exams = "";

            if (closedExams != null)
                foreach (var ex in ClosedExams)
                {
                    exams += " - " + ex.ToString() + "\n";
                }

            return ToShortString() + " Exams:\n" + exams;
        }

        public virtual string ToShortString()
        {
            return info?.ToString() + " " + education.ToString() + " " + numberGroup + ",
AvgRat: " + AvgRating;
        }
    }
}
```