Отчёт ЛР№5 Понкращенков Д.Б. ПИН-21Д.

Вариант №1.

Оглавление

1. Задание

Определить новые версии классов **Student** (вариант **1**), **Magazine** (вариант **2**) и **ResearchTeam** (вариант **3**). В сформулированных ниже требованиях для этих классов использовано общее обозначение **T**.

В новые версии классов добавить **экземплярные** методы

- T DeepCopy() для создания полной копии объекта с использованием сериализации;
- bool Save(string filename) для сохранения данных объекта в файле с помощью сериализации;
- bool Load(string filename) для инициализации объекта данными из файла с помощью десериализации;
- bool AddFromConsole() для добавления в один из списков класса нового элемента, данные для которого вводятся с консоли;

и статические методы

- static bool Save(string filename, T obj) для сохранения объекта в файле с помощью сериализации;
- static bool Load(string filename, T obj) для восстановления объекта из файла с помощью десериализации.

В экземплярном методе **T DeepCopy()** вызывающий объект сериализуется в поток MemoryStream. Метод возвращает восстановленный при десериализации объект, который представляет собой полную копию исходного объекта.

Экземплярный метод **bool Save(string filename)** сериализует все данные вызывающего объекта в файл с именем filename. Если файл с именем filename существует, приложение его перезаписывает. Если такого файла нет, приложение его создает. Метод возвращает значение true, если сериализация завершилась успешно, и значение false в противном случае.

Экземплярный метод **bool Load(string filename)** десериализует данные из файла с именем filename и использует их для инициализации вызывающего объекта. Метод возвращает значение true, если инициализация завершилась успешно. Если полностью выполнить инициализацию объекта не удалось, исходные данные объекта должны остаться без изменения. В этом случае метод возвращает значение false.

Статические методы **bool Save(string filename, T obj)** и **bool Load(string filename, T obj)** получают через параметры имя файла и ссылку на объект, для

которого выполняется сериализация или восстановление. Методы возвращают значение true, если сериализация/инициализация завершилась успешно, и значение false в противном случае. Если полностью выполнить инициализацию объекта не удалось, исходные данные объекта должны остаться без изменения.

Во всех реализациях методов сохранения/восстановления данных из файла операции открытия файла, сериализации и десериализации данных должны находиться в блоках try-catch-finally.

В методе **bool AddFromConsole()** для добавления нового элемента в один из списков класса T

- пользователь получает приглашение ввести данные в виде одной строки символов с разделителями; приглашение содержит описание формата строки ввода, в том числе информацию о том, какие символы можно использовать в качестве разделителей;
- выполняется разбор данных; операции преобразования данных, которые могут бросить исключение, должны находиться в блоке try-catch;
- если разбор введенных данных был завершен успешно, в список добавляется новый элемент и метод возвращает значение true; в противном случае пользователь получает сообщение о том, что при вводе были допущены ошибки и возвращаемое значение метода равно false.

В **варианте 1** элементы, данные для которых вводятся с консоли, добавляются в список экзаменов System.Collections.Generic.List<Exam>. Вводятся название предмета, оценка и дата экзамена.

В **варианте 2** элементы добавляются в список статей в журнале System.Collections.Generic.List<Article>. Вводятся название статьи, данные автора статьи для объекта типа Person и рейтинг статьи.

В **варианте 3** элементы добавляются в список публикаций System.Collections.Generic.List<Paper>. Вводятся название публикации, данные автора статьи для объекта типа Person и дата публикации.

В методе **Main()**

1. Создать объект типа T с непустым списком элементов, для которого предусмотрен ввод данных с консоли. Создать полную копию объекта с помощью метода, использующего сериализацию, и вывести исходный объект и его копию.
2. Предложить пользователю ввести имя файла:

- если файла с введенным именем нет, приложение должно сообщить об этом и создать файл;
- если файл существует, вызвать метод Load(string filename) для инициализации объекта T данными из файла.

3. Вывести объект T.
4. Для этого же объекта T сначала вызвать метод AddFromConsole(), затем метод Save(string filename). Вывести объект T.
5. Вызвать последовательно
   - статический метод Load(string filename, T obj), передав как параметры ссылку на тот же самый объект T и введенное ранее имя файла;
   - метод AddFromConsole();
   - статический метод Save (string filename, T obj).
6. Вывести объект T.

Приложение должно работать в режиме накопления. Если выбирается один и тот же файл для записи, и пользователь вводит данные без ошибок, при каждом следующем выполнении приложения к списку добавляются два новых элемента. Приложение должно обрабатывать все исключения, которые могут возникнуть из-за ошибок при вводе данных. Независимо от того, корректно были введены данные или при вводе были допущены ошибки, все файловые потоки должны быть закрыты.

2. Результат работы программы

Выполнение задания:

```
1. ----------------------------------------------------------

asdas asdasda 08.06.2024 14:18:46 Bachelor 1000, AvgRat: 2,4
 Exams:
 - ESIHVB 3, Date:01.01.1601 3:00:16
 - TB6KDU 3, Date:01.01.1601 3:01:35
 - 3AT2RP 4, Date:01.01.1601 3:00:48
 - ZDNZBF 0, Date:01.01.1601 3:02:49
 - INVFWV 2, Date:01.01.1601 3:01:17

 Tests:


   ----------------------------------------------------------

asdas asdasda 08.06.2024 14:18:46 Bachelor 1000, AvgRat: 2,4
 Exams:
 - ESIHVB 3, Date:01.01.1601 3:00:16
 - TB6KDU 3, Date:01.01.1601 3:01:35
 - 3AT2RP 4, Date:01.01.1601 3:00:48
 - ZDNZBF 0, Date:01.01.1601 3:02:49
 - INVFWV 2, Date:01.01.1601 3:01:17

 Tests:


   ----------------------------------------------------------


2. ----------------------------------------------------------

Введите имя файла для сериализации:
f2
Десериализация объекта...
asdas asdasda 08.06.2024 14:18:21 SecondEducation 1000, AvgRat: 3
 Exams:
 - RTYR6W 4, Date:01.01.1601 3:01:11
 - BCZXP3 2, Date:01.01.1601 3:02:33
 - Y7NSNT 1, Date:01.01.1601 3:01:57
 - 91GQ0I 4, Date:01.01.1601 3:03:16
 - Z5PX92 4, Date:01.01.1601 3:03:03

 Tests:


   ----------------------------------------------------------

4. ----------------------------------------------------------

Введите экзамен: [Названия дисциплины], [Оценка], [Дата экзамена] без использования []
aaaa 5 2024.06.24
Введены не все данные!
Введите экзамен: [Названия дисциплины], [Оценка], [Дата экзамена] без использования []
aaaa, 5, 2024.06.24
asdas asdasda 08.06.2024 14:18:21 SecondEducation 1000, AvgRat: 3,3333333333333335
 Exams:
 - RTYR6W 4, Date:01.01.1601 3:01:11
 - BCZXP3 2, Date:01.01.1601 3:02:33
 - Y7NSNT 1, Date:01.01.1601 3:01:57
 - 91GQ0I 4, Date:01.01.1601 3:03:16
 - Z5PX92 4, Date:01.01.1601 3:03:03
 - aaaa 5, Date:24.06.2024 0:00:00

 Tests:


   ----------------------------------------------------------


5. ----------------------------------------------------------

Введите экзамен: [Названия дисциплины], [Оценка], [Дата экзамена] без использования []
bbb, [4], 2024.06.20 13:00
asdas asdasda 08.06.2024 14:18:21 SecondEducation 1000, AvgRat: 3,4285714285714284
 Exams:
 - RTYR6W 4, Date:01.01.1601 3:01:11
 - BCZXP3 2, Date:01.01.1601 3:02:33
 - Y7NSNT 1, Date:01.01.1601 3:01:57
 - 91GQ0I 4, Date:01.01.1601 3:03:16
 - Z5PX92 4, Date:01.01.1601 3:03:03
 - aaaa 5, Date:24.06.2024 0:00:00
 - bbb 4, Date:20.06.2024 13:00:00

 Tests:


   ----------------------------------------------------------
```

Сформированный файл:


```
{
  "Education": 2,
  "NumberGroup": 1000,
  "AvgRating": 3.4285714285714284,
  "ClosedExams": [
    {
      "NameSubject": "RTYR6W",
      "Rating": 4,
      "DateExam": "1601-01-01T03:01:11.3661131+03:00",
      "Date": "0001-01-01T00:00:00"
    },
    {
      "NameSubject": "BCZXP3",
      "Rating": 2,
      "DateExam": "1601-01-01T03:02:33.9860839+03:00",
      "Date": "0001-01-01T00:00:00"
    },
    {
      "NameSubject": "Y7NSNT",
      "Rating": 1,
      "DateExam": "1601-01-01T03:01:57.0147633+03:00",
      "Date": "0001-01-01T00:00:00"
    },
    {
      "NameSubject": "91GQ0I",
      "Rating": 4,
      "DateExam": "1601-01-01T03:03:16.8253957+03:00",
      "Date": "0001-01-01T00:00:00"
    },
    {
      "NameSubject": "Z5PX92",
      "Rating": 4,
      "DateExam": "1601-01-01T03:03:03.9278651+03:00",
      "Date": "0001-01-01T00:00:00"
    },
    {
      "NameSubject": "aaaa",
      "Rating": 5,
      "DateExam": "2024-06-24T00:00:00",
      "Date": "0001-01-01T00:00:00"
    },
    {
      "NameSubject": "bbb",
      "Rating": 4,
      "DateExam": "2024-06-20T13:00:00",
      "Date": "0001-01-01T00:00:00"
    }
  ],
  "Subjects": [],
  "Forename": "asdas",
  "Surname": "asdasda",
  "BirthData": "2024-06-08T14:18:21.3648019+03:00",
  "Date": "2024-06-08T14:18:21.3648019+03:00"
}
```

В своём рабочем проекте я использовал сериализацию в виде бинарного файла при помощи класса ***BinaryFormatter***, на платформе .Net Framework 4.8.

Как оказалось, на .Net Core его пометили устаревшим в связи уязвимости в безопасности, поэтому для выполнения данной работы я испробовал ***XMLSerialization*** и ***JsonSerializer***. И сразу же столкнулся с непонятными ошибками, для JsonSerialization это оказалась ошибка –

System.NullReferenceException: "Object reference not set to an instance of an object.",

Якобы сериализуемый объект или одно из его полей было null, но все поля были инициализированы. Камнем преткновения оказалось наследование от интерфейса ***IEnumerable***, после отказа от его реализации - оба класса сериализации начали работать.

Выяснить такую особенность ***JsonSerialization*** помог ***XMLSerialization*** – он выдавал ошибку при попытке сериализации:

«"System.InvalidOperationException: "To be XML serializable, types which inherit from IEnumerable must have an implementation of Add(System.Object) at all levels of their inheritance hierarchy. LR5.Student does not implement Add(System.Object)."»

Класс ***XMLSerialization*** принимал Student за коллекцию и требовал реализации функции добавления элемента. Вероятно, я по ошибке когда-то добавил наследование от ***IEnumerable,*** что теперь мне подарило несколько часов на выяснение причин и объяснение неочевидного.

3. Листинг

```
1. //-----------------------------------------------------------------------------------------------||
2. // Файл Student.cs
3. //-----------------------------------------------------------------------------------------------||
4.
5. using LR2;
6. using Exam = LR3.Exam;
7.
8. using AutoMapper;
9. using System.Text.Json;
10. using System.Xml.Serialization;
11.
12. namespace LR5
13. {
14.     [Serializable]
15.     public class Student : LR4.Student
16.     {
17.         #region --- Static ---
18.
19.         /// <summary>
20.         /// для сохранения объекта в файле с помощью сериализации;
21.         /// </summary>
22.         /// <param name="filename"></param>
23.         /// <param name="obj"></param>
24.         /// <returns></returns>
25.         static public bool Save(string filename, Student obj)
26.         {
27.             return obj.Save(filename);
28.         }
29.
30.         /// <summary>
31.         /// для восстановления объекта из файла с помощью десериализации.
32.         /// </summary>
33.         /// <param name="filename"></param>
34.         /// <param name="obj"></param>
35.         /// <returns></returns>
36.         static public bool Load(string filename, Student obj)
37.         {
38.             return obj.Load(filename);
39.         }
40.
41.         #endregion
42.
43.         public Student() : base() { }
44.
45.         public Student(Person p, Education e, int n) : base(p, e, n) { }
46.
47.         #region --- LR 5 ---
48.
49.         /// <summary>
50.         /// для сохранения данных объекта в файле с помощью сериализации
51.         /// </summary>
52.         /// <param name="filename"></param>
53.         /// <returns></returns>
54.         public bool Save(string filename)
55.         {
56.             try
57.             {
58.                 // Реализация №1
59.                 var options = new JsonSerializerOptions
60.                 {
61.                     WriteIndented = true                                // Для читаемого форматирования
62.                 };
63.
64.                 using (Stream stream = new FileStream(filename, FileMode.Create))
65.                 {
66.                     JsonSerializer.Serialize(stream, this, options);
67.                 }
68.
69.                 // Реализация №2
70.                 //XmlSerializer xmlSerializer = new XmlSerializer(typeof(Student));
71.                 //using (Stream stream = new FileStream(filename, FileMode.Create))
72.                 //{
73.                 //    xmlSerializer.Serialize(stream, this);
74.                 //}
75.             }
```

```
76.             catch (Exception)
77.             {
78.                 return false;
79.             }
80.             return true;
81.         }
82.
83.         /// <summary>
84.         /// для инициализации объекта данными из файла с помощью десериализации;
85.         /// </summary>
86.         /// <param name="filename"></param>
87.         /// <returns></returns>
88.         public bool Load(string filename)
89.         {
90.             try
91.             {
92.                 using (Stream stream = new FileStream(filename, FileMode.Open))
93.                 {
94.                     if (stream.Length == 0)
95.                     {
96.                         throw new InvalidOperationException("Файл пуст.");
97.                     }
98.
99.                     var tmp = JsonSerializer.Deserialize<Student>(stream);
100.
101.                    //XmlSerializer xmlSerializer = new XmlSerializer(typeof(Student));
102.                    //var tmp = xmlSerializer.Deserialize(stream) as Student;
103.
104.                    // Настройка AutoMapper
105.                    var config = new MapperConfiguration(cfg => cfg.CreateMap<Student, Student>());
106.                    IMapper mapper = config.CreateMapper();
107.
108.                    // Создание копии объекта
109.                    var copyObject = mapper.Map<Student, Student>(tmp, this);
110.                }
111.                //var tmp = new Student((Person)info.DeepCopy(), education, numberGroup);
112.                //tmp.closedExams = new List<Exam>();
113.                //tmp.subjects = new List<Test>();
114.
115.                //foreach (Exam exam in closedExams)
116.                //{
117.                //    tmp.closedExams.Add(exam.DeepCopy() as Exam);
118.                //}
119.
120.                //foreach (Test subject in subjects)
121.                //{
122.                //    tmp.subjects.Add(subject);
123.                //}
124.
125.            }
126.            catch (Exception)
127.            {
128.                return false;
129.            }
130.            return true;
131.        }
132.
133.        /// <summary>
134.        /// для добавления в один из списков класса нового элемента, данные для которого вводятся с консоли;
135.        /// </summary>
136.        /// <returns></returns>
137.        public bool AddFromConsole()
138.        {
139.            try
140.            {
141.                Console.WriteLine("Введите экзамен: [Названия дисциплины], [Оценка], [Дата экзамена] без
использования []");
142.                var input = Console.ReadLine();
143.
144.                var paramss = input.Split(", ");
145.                if (paramss.Count() < 3)
146.                    throw new Exception("Введены не все данные!");
147.
148.                for (int i = 0; i < paramss.Count(); i++)
149.                {
150.                    paramss[i] = paramss[i].Trim('[', ']');
151.                }
152.
153.                var exam = new Exam(paramss[0], int.Parse(paramss[1]), DateTime.Parse(paramss[2]));
154.
```

```
                closedExams.Add(exam);
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex.Message);
                return false;
            }
            return true;
        }
        /// <summary>
        /// Нельзя прегрузить, ошибка CS0506
        /// Копирование объекта через сериализцию
        /// </summary>
        /// <returns></returns>
        public Student DeepCopy()
        {
            // Сериализация вызывающего объекта в MemoryStream
            // Реализация №1
            using (var memoryStream = new MemoryStream())
            {
                // По какой-то приниче, если класс наследуется от интерфейса IEnumerable - ошибка:
                // Неизлечимая ошибка  - System.NullReferenceException: "Object reference not set to an
instance of an object."
                JsonSerializer.Serialize(memoryStream, this);

                // Перемещение указателя в начало потока перед десериализацией
                memoryStream.Seek(0, SeekOrigin.Begin);

                // Десериализация потока обратно в объект
                return JsonSerializer.Deserialize<Student>(memoryStream);
            }

            // Реализация №2
            //XmlSerializer xmlSerializer = new XmlSerializer(typeof(Student));
            // Сериализация вызывающего объекта в MemoryStream
            //using (var memoryStream = new MemoryStream())
            //{
            //    xmlSerializer.Serialize(memoryStream, this);
            //    memoryStream.Seek(0, SeekOrigin.Begin);
            //    // Десериализация потока обратно в объект
            //    return xmlSerializer.Deserialize(memoryStream) as Student;
            //}
        }

        #endregion

    }
}

```

Файл Student.cs

```
using LR2;
using LR3;
using System;
using System.Security.Cryptography;

using Exam = LR3.Exam;

namespace LR5
{
    internal class Program
    {
        private static Random random = new Random();

        static void Main(string[] args)
        {
            var student = getStudentRND();
            var copyStudent = student.DeepCopy();

            Console.WriteLine("\n1. ----------------------------------------------------------\n");
            Console.WriteLine(student.ToString());
            Console.WriteLine("\n   ----------------------------------------------------------\n");
            Console.WriteLine(copyStudent.ToString());
            Console.WriteLine("\n   ----------------------------------------------------------\n");


            Console.WriteLine("\n2. ----------------------------------------------------------\n");

            Student newstudent = null;

            Console.WriteLine("Введите имя файла для сериализации:");
            var path = Console.ReadLine();
            if (!File.Exists(path)) {
                Console.WriteLine(path + " не существует и будет создан.");
                Console.WriteLine("Сериализация объекта...");
                student.Save(path);
            }
            else
            {
                newstudent = new Student();
                Console.WriteLine("Десериализация объекта...");
                newstudent.Load(path);
                Console.WriteLine(newstudent.ToString());
            }
            Console.WriteLine("\n   ----------------------------------------------------------\n");

            Console.WriteLine("\n4. ----------------------------------------------------------\n");
            if(newstudent != null)
            {
                newstudent.AddFromConsole();
                newstudent.Save(path);
                Console.WriteLine(newstudent.ToString());
            }
            Console.WriteLine("\n   ----------------------------------------------------------\n");

            Console.WriteLine("\n5. ----------------------------------------------------------\n");
            var newStudent2 = new Student();
            Student.Load(path, newStudent2);
            newStudent2.AddFromConsole();
            Student.Save(path, newStudent2);
            Console.WriteLine(newStudent2.ToString());
            Console.WriteLine("\n   ----------------------------------------------------------\n");
        }

        #region --- Рандомайзеры ---

        static Student getStudentRND()
        {
            var student = new Student(new Person("asdas", "asdasda",DateTime.Now), Education.Bachelor, 1000);
            student.AddExams(new List<Exam>{
                    new Exam(getStringRND(), getRatRND(), DateTime.FromFileTime(getDateRND())),
                    new Exam(getStringRND(), getRatRND(), DateTime.FromFileTime(getDateRND())),
                    new Exam(getStringRND(), getRatRND(), DateTime.FromFileTime(getDateRND())),
                    new Exam(getStringRND(), getRatRND(), DateTime.FromFileTime(getDateRND())),
                    new Exam(getStringRND(), getRatRND(), DateTime.FromFileTime(getDateRND()))
            });
            student.Education = (LR2.Education)RandomNumberGenerator.GetInt32(0, 3);

```

```
78.             return student;
79.         }
80.
81.         static int getRatRND()
82.         {
83.             return RandomNumberGenerator.GetInt32(0, 6);
84.         }
85.
86.         static int getDateRND()
87.         {
88.             return RandomNumberGenerator.GetInt32(0, int.MaxValue);
89.         }
90.
91.         static string getStringRND()
92.         {
93.             int length = 6;
94.             const string chars = "ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789";
95.             return new string(Enumerable.Repeat(chars, length)
96.                 .Select(s => s[random.Next(s.Length)]).ToArray());
97.         }
98.
99.         #endregion
100.
101.     }
102. }
103.
```