Лабораторная работа № 15.

Тема: Сериализазия, десериализация объектов

Цель работы: Формирование умений и навыков использования сериализованных объектов.

Общие теоретические сведения:

Сериализация - процесс перевода какой-либо структуры данных в последовательность битов. Обратной к операции сериализации является операция десериализации (структуризации) — восстановление начального состояния структуры данных из битовой последовательности.

Зачем нужна сериализация?

Для хранения и удобного получения данных из файла была придумана сериализация. Все объекты записываются в определенном формате. Таким образом, в одном приложении мы сериализируем данные, в другом десериализируем и получаем готовые объекты.

Рассмотрим несколько способов сериализовать объект.

Binary — представляет из себя преобразование объектов в поток байтов, который затем записывается в поток данных. После десериализации объекта создается точная копия исходного объекта.

SOAP — протокол обмена структурированными сообщениями в распределённой вычислительной среде. Первоначально SOAP предназначался в основном для реализации удалённого вызова процедур. Сейчас протокол используется для обмена произвольными сообщениями в формате XML, а не только для вызова процедур.

REST — архитектурный стиль взаимодействия компонентов распределённого приложения в сети. REST представляет собой согласованный набор ограничений, учитываемых при проектировании распределённой гипермедиа-системы. В определённых случаях это приводит к повышению производительности и упрощению архитектуры. Представлена в виде JSON.

Рассмотрим пример. Даны два класса. Student с полями name строкового типа, age — тип int, group — тип Group. Класс Group имеет поля name (название группы) — тип string, номер группы — тип int. Задание создать 10 групп, создать 300 студентов, и записать их в случайную группу, а далее объекты сериализовать массив студентов и массив групп, различными способами: binary, soap, xml, json.

```
[Serializable]
Ссылок: 9
public class Student
{
    [NonSerialized]
    int age;
    Ссылок: 2
    public string Name { get; set; }

    Ссылок: 2
    public int Age
    {
        get { return age; }
        set
        {
            if (value <= 13)
                age = value + 5;
            else if (value <= 17)
                age = value + 3;
            else
                age = value;
        }
    }
```

```
Ссылок: 2
public Group Group { get; set; }

ссылка: 1
public Student(string name, int age)
{
    Name = name;
    Age = age;
}

Ссылок: 0
public Student(){}

Ссылок: 7
public override string ToString()
{
    return $"Студент {Name} возраст {Age} учится в группе {Group}";
}
```

```
[DataContract]
Ссылок: 9
public class Group
{
    [DataMember]
    Ссылок: 2
    public int Number { get; set; }

    Ссылок: 2
    public string Name { get; set; }

    Ссылок: 0
    public Group(){}

    ссылка: 1
    public Group(int number)
    {
        Name = "BEST";
        Number = number;
    }

    Ссылок: 7
    public override string ToString()
    {
        return $"{Name}--{Number}";
    }
}
```

```
class Runner
{
    static Group[] groups = new Group[10];
    static Student[] students = new Student[300];
    static Random random = new Random();

    Ссылок: 0
    static void Main(string[] args)
    {
        GreateGroups();
        CreateStudents();
        PrintStudents();

        var binarySerialiazable = new BinaryFormatter();
        using (var file = new FileStream("students.bin", FileMode.OpenOrCreate))
        {
            binarySerialiazable.Serialize(file, students);
        }

        Console.WriteLine("----------------------------------------");
        using (var file = new FileStream("students.bin", FileMode.OpenOrCreate))
        {
            var desirializableBinary = binarySerialiazable.Deserialize(file) as Student[];
            if (desirializableBinary != null)
            {
                foreach (var item in desirializableBinary)
                {
                    Console.WriteLine(item);
                }
            }
        }
```

```
var soap = new SoapFormatter();
using (var file = new FileStream("GROUPS.soap", FileMode.OpenOrCreate))
{
    soap.Serialize(file, groups);
}

Console.WriteLine("----------------------------------------");
using (var file = new FileStream("GROUPS.soap", FileMode.OpenOrCreate))
{
    var desirializableBinary = soap.Deserialize(file) as Group[];
    if (desirializableBinary != null)
    {
        foreach (var item in desirializableBinary)
        {
            Console.WriteLine(item);
        }
    }
}
```

```
XmlSerializer xmlformatter = new XmlSerializer(typeof(Student[]));
using (var file = new FileStream("students.xml", FileMode.OpenOrCreate))
{
    xmlformatter.Serialize(file, students);
    Console.WriteLine("XML created");
}

Console.WriteLine("----------------------------------");

using (var file = new FileStream("students.xml", FileMode.OpenOrCreate))
{

    var desXmlFormat = xmlformatter.Deserialize(file) as Student[];
    if (desXmlFormat != null)
    {
        Console.WriteLine(desXmlFormat[15]);
    }

}
```

```
DataContractJsonSerializer jsonformatter = new DataContractJsonSerializer(typeof(Group[]));
using (var file = new FileStream("grougs.json", FileMode.OpenOrCreate))
{
    jsonformatter.WriteObject(file, groups);
    Console.WriteLine("json created");

}

Console.WriteLine("---------------------------------");

using (var file = new FileStream("grougs.json", FileMode.OpenOrCreate))
{
    var desjson = jsonformatter.ReadObject(file) as Group[];
    if (desjson != null)
    {
        Console.WriteLine(desjson[2]);
    }
}
```

Практическая часть

Общее задание: Написать два любых класса (которые как-нибудь, между собой связаны, хотя бы как в примере), написать у каждого класса поля и свойства (если необходимы). Найти себе напарника, и договорится, кто будет сериализовать объекты, а кто будет десериализовать объект. (**напарник работает на отдельном компьютере в отдельном проекте, но есть один нюанс, заключенный в пространстве имен проекта**). Создать массивы объектов, ну или просто объекты. Реализовать сериализацию и десериализацию объекта согласно варианту (вариант у преподавателя).

Создать массив объектов более развернутого класса (минимум 4 объекта). Вывести массив на экран. Сериализавать его в один из файлов (*.bin, *.soap, *.xml, *.json), а напарник с вашего файла должен получить опять массив объектов и вывести его на экран.

Вариант	Задание
1	Создать два класса. Коробка и кот. Коробка имеет номер и материал коробки. Кот имеет поля имя, цвет шерсти и номер коробки в которой он лежит.
2	Создать два класса. Человек и адрес. Класс адрес содержит улицу номер дома и человека, который живет по данному адресу. Класс человек имеет имя, фамилия и возраст человека.
3	Создать два класса автомобиль и двигатель. Двигатель имеет поле объем, тип топлива. Автомобиль содержит поле бренд авто, модель авто, двигатель.
4	Создать два класса телефон и процессор. Процессор имеет поле частота, и компания производитель. телефон содержит поле

	бренд телефона, модель телефона, различные характеристики (минимум 2), процессор
5	Создать три класса человек, родители, дети дети имеют поля имя, возраст, мама, папа родители поля имя, возраст, состояние брака (брак, развод, сожительство) человек — имя возраст