ЛАБОРАТОРНАЯ РАБОТА №10

Сериализация объектов

Цель работы:

Изучить различные способы сохранения информации об объектах классов.

Постановка задачи:

Создать класс, содержащий несколько свойств и методов. Создать коллекцию объектов класса.

1. Использовать разные способы сериализации объектов для сохранения коллекции в файл и чтения из файла. Проанализировать полученные сохраненные файлы.
2. Доработать код лабораторной работы №5 для сохранения координат и параметров геометрических фигур в XML-файл.

Пример выполнения:

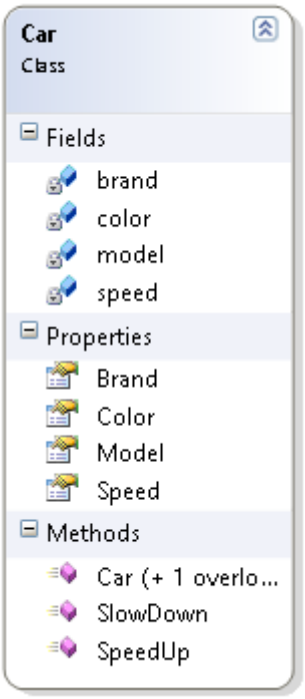
Имеется класс Car:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Xml;
using System.Xml.Serialization;
using System.Runtime.Serialization;
using System.Runtime.Serialization.Formatters.Binary;

namespace Serialization
{
    [Serializable]
    public class Car
    {
        private string brand;

        [XmlAttribute]
        public string Brand
        {
            get { return brand; }
            set { brand = value; }
        }
        private string model;

        public string Model
        {
            get { return model; }
            set { model = value; }
        }
        private string color;

        public string Color
        {
            get { return color; }
            set { color = value; }
        }

        private int speed;

        public int Speed
        {
            get { return speed; }
        }

        public void SpeedUp()
        {
            if (speed < 200) speed += 10;
        }

        public void SlowDown()
        {
            speed=speed==0? 0: speed-10;
        }

        public Car()
        { }

        public Car(string brand, string model, string color)
        {
            this.brand = brand;
```

Обязательно для `BinaryFormatter` и `XmlSerializer`

Поле доступно только для чтения

```
            this.model = model;
            this.color = color;
            speed = 0;
        }
    }
}
```



Сериализация объектов Car

Для использования стандартных средств сериализации необходимо подключить пространства имен:

```
using System.Xml.Serialization;
using System.Xml;
using System.Xml.Linq;
using System.Runtime.Serialization;
using System.Runtime.Serialization.Formatters.Binary;
```

Объекты класса Car помещены в коллекцию:

```
            List <Car> carList=new List<Car>();

            carList.AddRange(new Car[] {
                new Car("ford", "mondeo", "white"),
                new Car("volkswagen", "beetle", "green"),
                new Car("ford", "focus", "black"),
                new Car("opel", "corsa", "grey"),
                new Car("renault", "megane", "blue"),
                new Car("lada", "2011", "white")
            });
```

Использование BinaryFormatter для сериализации:

```
            BinaryFormatter bf = new BinaryFormatter();
```

```
using (FileStream fs=new FileStream("bin.dat",FileMode.OpenOrCreate))
{
    bf.Serialize(fs,carList);
}
```

Использование XmlSerializer

- *Сериализация в файл:*

```
// обратите внимание
// поля privte не сохраняются в XML файле
XmlSerializer xs = new XmlSerializer(typeof(List<Car>));
using (FileStream fs = new FileStream("xml.xml", FileMode.CreateNew))
{
    xs.Serialize(fs, carList);
}
```

- - *Чтение из файла:*

```
List<Car> newList = new List<Car>();

using (FileStream fs = new FileStream("xml.xml", FileMode.Open))
{
    newList = (List<Car>)xs.Deserialize(fs);
}
```

Использование объектной модели документа:

```
XmlDocument doc = new XmlDocument();
XmlElement root = doc.CreateElement("root");
foreach (Car c in carList)
{

    // Создать элемент Car
    XmlElement car = doc.CreateElement("Car");
    car.SetAttribute("Speed", c.Speed.ToString());

    //Создать подэлементы
    XmlElement brand = doc.CreateElement("Brand");
    brand.InnerText = c.Brand;
    XmlElement model = doc.CreateElement("Model");
    model.InnerText = c.Model;
    XmlElement color = doc.CreateElement("Color");
    color.InnerText = c.Color;

    // Добавить подэлементы в элемент Car
    car.AppendChild(brand);
    car.AppendChild(model);
    car.AppendChild(color);

    // Добавить элемент Car в документ

    try
    { root.AppendChild(car); }
    catch (InvalidOperationException) { };
}
doc.AppendChild(root);
doc.Save("DOM.xml");
```

Использование LINQ-toXML

Запись в файл:

```
XDocument doc = new XDocument();
// Создание корня документа
XElement root = new XElement("Cars");
foreach(var car in carList)
{
    // Создание одного элемента
    XElement carElement = new XElement("car",
        new XAttribute("brand", car.Brand),
        new XElement("model", car.Model),
        new XElement("color", car.Color));
    root.Add(carElement);
}
// Поместить корень в документ
doc.Add(root);
// Сохранить документ
doc.Save("linq.xml");
```

Содержимое файла:

```
<?xml version="1.0" encoding="utf-8"?>
<Cars>
  <car brand="ford">
    <model>mondeo</model>
    <color>white</color>
    <speed>20</speed>
  </car>
  <car brand="volkswagen">
    <model>beetle</model>
    <color>green</color>
    <speed>0</speed>
  </car>
  <car brand="ford">
    <model>focus</model>
    <color>black</color>
    <speed>10</speed>
  </car>
  <car brand="opel">
    <model>corsa</model>
    <color>grey</color>
    <speed>0</speed>
  </car>
  <car brand="renault">
    <model>megane</model>
    <color>blue</color>
    <speed>0</speed>
  </car>
  <car brand="lada">
    <model>2011</model>
    <color>white</color>
    <speed>0</speed>
  </car>
</Cars>
```

Чтение из файла:

```
List<Car> newListOfCars = new List<Car>();
// Открыть документ
XDocument doc = XDocument.Load("linq.xml");
// Выделить root
var root = doc.Root;
// Обойти коллекцию элементов внутри root
foreach(var element in root.Elements())
{
    var car = new Car();
    car.Brand = element.Attribute("brand").Value;
    car.Model = element.Element("model").Value;
    car.Color = element.Element("color").Value;
    car.Speed = int.Parse(element.Element("speed").Value);
    // Сохранить объект Car
    newListOfCars.Add(car);
}
```