Serelization

```
using System;
using System.IO;
using System.Runtime.Serialization.Formatters.Binary;
using System.Runtime.Serialization.Formatters.Soap;
using System.Xml.Serialization;
using System.Text.Json;
using System.Collections.Generic;
using System.Xml;
using Newtonsoft.Json.Linq;
using System.Linq;

namespace OOP13
{
    interface ISerelizator<T>
    {
        void Serializate(string filename, T formatter);
        void Deserializate(string filename, ref T formatter);
    }

    class Serelizator<M>: ISerelizator<M>
    {
        public void Serializate(string filename, M question)
        {
            var extension = filename.Split('.')[1];
           /* FileStream createFile = new FileStream(filename, FileMode.Create);*/
            if (extension.ToLower() == "json")
            {
                var options = new JsonSerializerOptions
                {
                    WriteIndented = true,
                    AllowTrailingCommas = true
                };
                using (FileStream fs = new FileStream(filename, FileMode.Create))
                {
                    JsonSerializer.Serialize<M>(fs, question, options);
                }
            }
            else if (extension.ToLower() == "soap")
            {
                SoapFormatter formatterSoap = new SoapFormatter();
                using (FileStream fs = new FileStream(filename, FileMode.Create))
                {
                    formatterSoap.Serialize(fs, question);
                }
            }
            else if (extension.ToLower() == "xml")
            {
                XmlSerializer xmlSerializer = new XmlSerializer(question.GetType());
                using (FileStream fs = new FileStream(filename, FileMode.Create))
                {
                    xmlSerializer.Serialize(fs, question);
                }
            }
            else if (extension.ToLower() == "bit")
            {
                BinaryFormatter formater = new BinaryFormatter();
                using (FileStream fs = new FileStream(filename, FileMode.Create))
                {
                    formater.Serialize(fs, question);
                }
            }
            else
            {
                Console.WriteLine("Объект не сериализован");
                return;
            }
            Console.WriteLine("Объект сериализован");
        }
        public void Deserializate(string filename, ref M question)
        {
            var extension = filename.Split('.')[1];
            if (extension.ToLower() == "json")
```

```
            {
                using (FileStream fs = new FileStream(filename, FileMode.Open))
                {
                    question = JsonSerializer.Deserialize<M>(fs);
                }
            }
            else if (extension.ToLower() == "bit")
            {
                using (FileStream fs = new FileStream(filename, FileMode.Open))
                {
                    BinaryFormatter formater = new BinaryFormatter();
                    question = (M)formater.Deserialize(fs);
                }
            }
            else if (extension.ToLower() == "soap")
            {
                using (FileStream fs = new FileStream(filename, FileMode.Open))
                {
                    SoapFormatter formatterSoap = new SoapFormatter();
                    question = (M)formatterSoap.Deserialize(fs);
                }
            }
            else if (extension.ToLower() == "xml")
            {
                using (FileStream fs = new FileStream(filename, FileMode.Open))
                {
                    XmlSerializer xmlSerializer = new XmlSerializer(question.GetType());
                    question = (M)xmlSerializer.Deserialize(fs);
                }
            }
            else
            {
                Console.WriteLine("Объект не десериализован");
                return;
            }
            Console.WriteLine("Объект десериализован");
        }
    }

    [Serializable]
    public class Question
    {
        [NonSerialized]
        public int id = 1;
        public string question { get; set; }
        public string questionAnswer { get; set; }

        public Question()
        {
            question = questionAnswer = null;
        }
        public Question(string question, string questionAnswer)
        {
            this.question = question;
            this.questionAnswer = questionAnswer;
        }
        public override string ToString()
        {
            return question;
        }
    }

    internal class Program
    {
        static void Main(string[] args)
        {
            Question question = new Question("Как дела?", "Норм");
            Question question2 = new Question("А у тебя?", "Норм");

            Serelizator<Question> serel = new Serelizator<Question>();
            serel.Serializate("test.json", question);
            serel.Deserializate("test.json", ref question);

            //-------------------------Task3----------------------------------

            List<Question> list = new List<Question> { question, question2 };

            Serelizator<List<Question>> serelForList = new Serelizator<List<Question>>();
```

```
            serelForList.Serialize("test.json", list);
            serelForList.Serialize("test.xml", list);

            List<Question> list2 = new List<Question>();
            List<Question> list3 = new List<Question>();
            serelForList.Deserialize("test.json", ref list2);
            serelForList.Deserialize("test.xml", ref list3);

            foreach (Question item in list2)
            {
                Console.WriteLine($"Вопрос: {item.question}\nОтвет: {item.questionAnswer}");
            }

            foreach (Question item in list3)
            {
                Console.WriteLine($"Вопрос: {item.question}\nОтвет: {item.questionAnswer}");
            }

            //-------------------------Task4---------------------------------
            XmlDocument xDoc = new XmlDocument();
            xDoc.Load("test.xml");
            XmlElement xRoot = xDoc.DocumentElement;
            XmlNodeList personNodes = xRoot.SelectNodes("Question");
            if (personNodes.Count > 0)
            {
                foreach (XmlNode node in personNodes)
                    Console.WriteLine(node.InnerText);
            }

            //-------------------------Task5---------------------------------
            string json = @"{
              'channel': {
                'title': 'James Newton-King',
                'link': 'http://james.newtonking.com',
                'description': 'James Newton-King\'s blog.',
                'item': [
                  {
                    'title': 'Json.NET 1.3 + New license + Now on CodePlex',
                    'description': 'Announcing the release of Json.NET 1.3, the MIT license and the source on CodePlex',
                    'link': 'http://james.newtonking.com/projects/json-net.aspx',
                    'categories': [
                      'Json.NET',
                      'CodePlex'
                    ]
                  },
                  {
                    'title': 'LINQ to JSON beta',
                    'description': 'Announcing LINQ to JSON',
                    'link': 'http://james.newtonking.com/projects/json-net.aspx',
                    'categories': [
                      'Json.NET',
                      'LINQ'
                    ]
                  }
                ]
              }
            }";

            JObject rss = JObject.Parse(json);

            var postTitles =
                from p in rss["channel"]["item"]
                select (string)p["title"];

            foreach (var item in postTitles)
            {
                Console.WriteLine(item);
            }
            //----------------------------------------------------------------
            Console.ReadLine();

        }
    }
}
```