**Лабораторная работа № 9 – Сериализация объектов**

**1 Цель работы**

Познакомиться с понятием и методами сериализации и десериализации объектов.

1. **Порядок выполнения работы**

**–** прочесть краткие теоретические сведения;

**–** выполнить задания раздела;

**–** составить отчет о лабораторной работе и защитить его у преподавателя.

**3 Теоретическая часть**

В пространстве имен **System.IO** предусмотрено множество типов, которые позволяют записывать двоичные и символьные данные в какое-то место для хранения (например, файл на диске или буфер в памяти) и считывать их. Однако среди них не существует типов, которые позволяют сохранять объекты классов целиком и затем восстанавливать их. Об этом процессе, который называется сериализацией, и пойдет речь далее.

**Сериализация** (serialization)– это термин, описывающий процесс преобразования объекта в линейную последовательность байтов. Обратный процесс, когда из потока байтов, содержащего всю необходимую информацию, объект восстанавливается в исходном виде, называется **десериализацией** (deserialization).

Чтобы можно было провести сериализацию объекта, каждый класс, который будет участвовать в сериализации, должен обладать атрибутом **[Serializable]**. Если же необходимо, чтобы какие-либо переменные данного класса были исключены из сериализации, достаточно просто пометить их как **[NonSerialized]**. Обычно так отмечаются те данные класса, которые «запоминать» не нужно (к примеру, те, которые на самом деле являются константами). Например:

**[Serializable]**

**class MyClass**

**{**

**public int x;**

**[NonSerialized]**

**private double pi = 3.14;**

**...**

**}**

После этого необходимо выбрать формат, в котором будет сохранен объектный граф. По умолчанию можно использовать один из двух форматов. Класс **BinaryFormatter** сериализует объект в компактном потоке двоичного формата, в то время как класс **SoapFormatter** представляет объект как сообщение протокола SOAP (Simple Object Access Protocol – простого протокола доступа к объектам) в формате XML. При необходимости можно создать свой собственный формат сериализации (и соответствующий ему объект **Formatter**).

Здесь рассматривается сериализация объектов в двоичном формате. Она производится при помощи класса **BinaryFormatter**, который определен в пространстве имен **System.Runtime.Serialization.Format-ters.Binary**. Для записи объекта в место хранения и восстановления объекта по сохраненной информации используются два главных метода этого класса:

|  |  |
| --- | --- |
| **Методы** | **Описание** |
| **virtual object Deserialize (Stream stream)** | Десериализует поток байтов **stream** в возвращаемый объект |
| **virtual void Serialize (Stream stream, object obj)** | Сериализует объект **obj** в поток **stream** |

**3.1 Пример выполнения работы**

Задание: разработать Windows-приложение, предназначенное для сериализации и десериализации текстовых файлов.

**3.2** **Визуальное проектирование главного окна приложения**

Внешний вид работающего приложения приведен на рисунке 1.

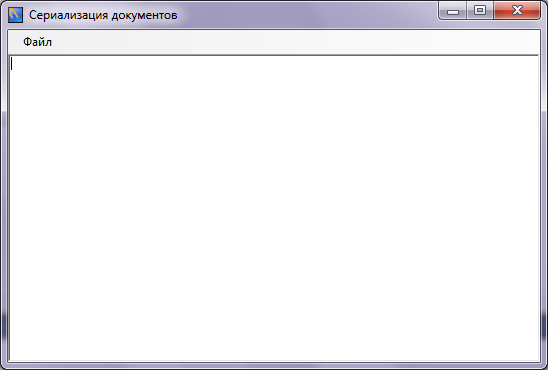


Рисунок 1 – Главное окно приложения

**3.3 Проектирование программного кода**

Прежде чем начать разрабатывать данное приложение, необходимо подключить следующие пространства имен:

using System.IO;

using System.Runtime.Serialization.Formatters.Binary;

using System.Runtime.Serialization;

**Открытие текстового документа**

Для открытия текстового документа необходимо выбрать команду Файл→Документ→Открыть. Открывается диалоговое окно, представленное на рисунке 2.

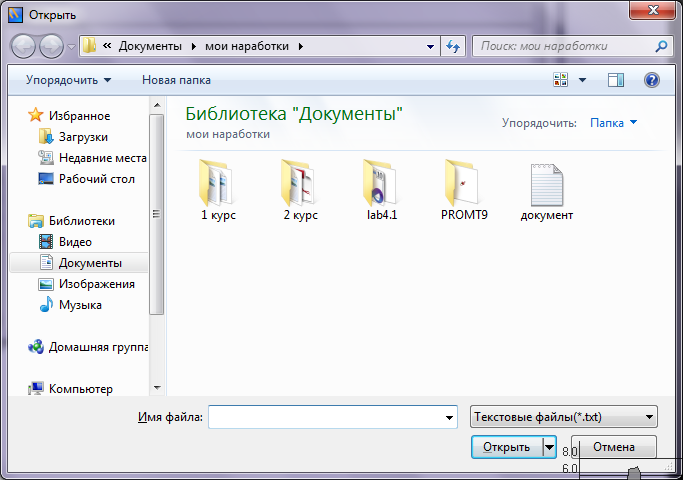


Рисунок 2 – Открытие документа

Программный код пункта меню Файл→Документ→Открыть:

private void открытьToolStripMenuItem\_Click(object sender, EventArgs e)

{

OpenFileDialog open = new OpenFileDialog();

open.Filter = "Текстовые файлы(\*.txt)|\*.txt|Все файлы(\*.\*)|\*.\*";

if (open.ShowDialog() != System.Windows.Forms.DialogResult.OK) { return; }

StreamReader read = new StreamReader(open.FileName);

List<string> list = new List<string>();

string s = "";

while ((s = read.ReadLine()) != null)

{

list.Add(s);

}

read.Close();

richTextBox1.Lines = list.ToArray();

}

**Сохранение текстового документа**

Для сохранения документа необходимо выбрать пункт меню Файл→Документ→Сохранить. Откроется диалоговое окно, приведенное на рисунке 3.

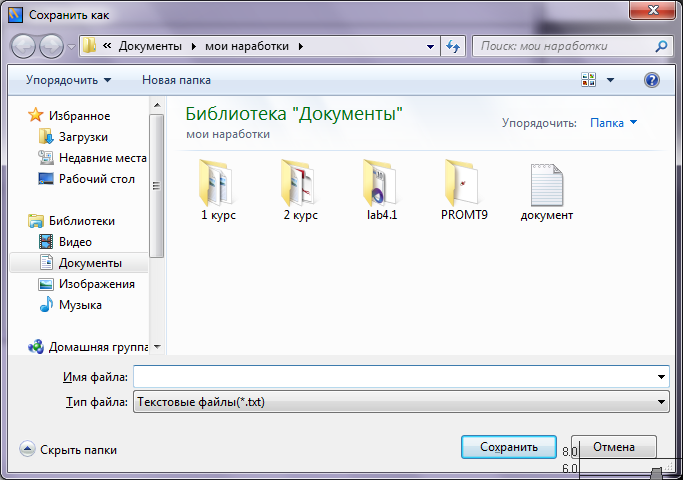


Рисунок 3 – Сохранение документа

Код пункта меню Файл→Документ→Сохранить:

private void сохранитьToolStripMenuItem\_Click(object sender, EventArgs e)

{

SaveFileDialog save = new SaveFileDialog();

save.Filter = "Текстовые файлы(\*.txt)|\*.txt|Все файлы(\*.\*)|\*.\*";

if (save.ShowDialog() != System.Windows.Forms.DialogResult.OK) { return; }

List<string> list = new List<string>();

StreamWriter write = new StreamWriter(save.FileName);

list.Add(richTextBox1.Text);

foreach (var item in list)

{

write.WriteLine(item);

}

write.Close();

richTextBox1.Clear();

}

**Сериализация**

Для выполнения сериализации (сохранения состояния объекта в потоке) необходимо выбрать команду Сериализация→Сериализация. Откроется диалоговое окно, приведенное на рисунке 4.

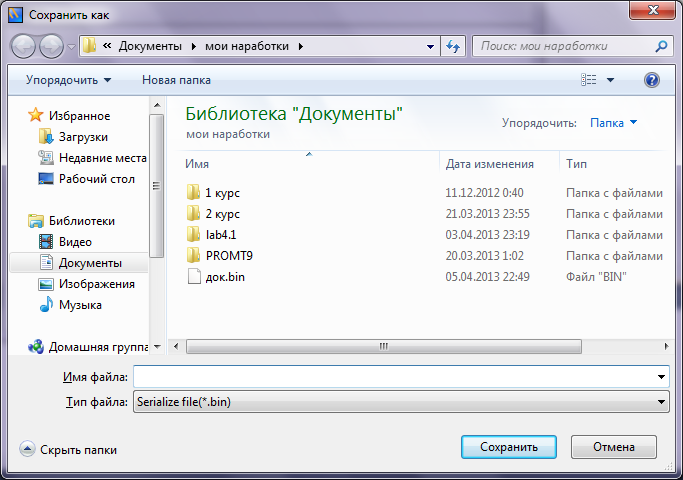


Рисунок 4 – Сериализация документа

Код команды Сериализация→Сериализация приведен ниже:

private void сериализацияToolStripMenuItem1\_Click(object sender, EventArgs e)

{

List<string> list = new List<string>();

for (int i = 0; i < richTextBox1.Lines.Count(); i++)

{

list.Add(richTextBox1.Lines[i].ToString());

}

SaveFileDialog save = new SaveFileDialog();

save.Filter = "Serialize file(\*.bin)|\*.bin|Все файлы(\*.\*)|\*.\*";

if (save.ShowDialog() != System.Windows.Forms.DialogResult.OK) { return; }

FileStream stream = new FileStream(save.FileName, FileMode.Create, FileAccess.Write, FileShare.ReadWrite);

BinaryFormatter bf = new BinaryFormatter();

bf.Serialize(stream, list);

stream.Close();

}

**Десериализация**

Для выполнения десериализации следует выбрать пункт меню Сериализация→Десериализация. Откроется диалоговое окно, представленное на рисунке 5.

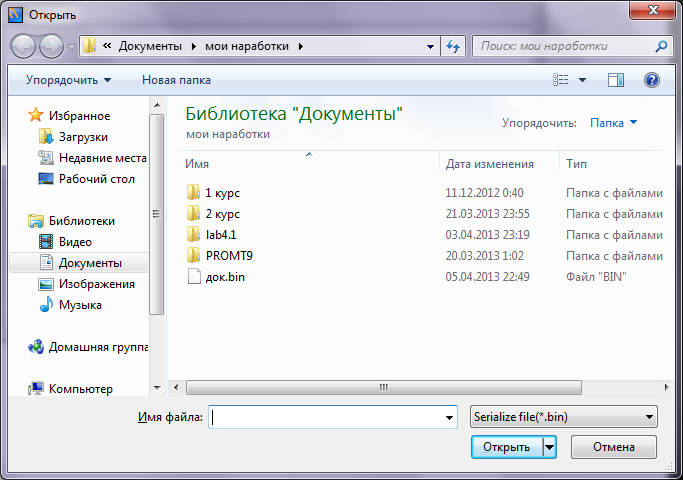


Рисунок 5 – Десериализация документа

Программный код пункта меню Сериализация→Десериализация:

private void десериализацияToolStripMenuItem\_Click(object sender, EventArgs e)

{

richTextBox1.Clear();

List<string> list = new List<string>();

OpenFileDialog open = new OpenFileDialog();

open.Filter = "Serialize file(\*.bin)|\*.bin|Все файлы(\*.\*)|\*.\*";

if (open.ShowDialog() != System.Windows.Forms.DialogResult.OK) { return; }

FileStream stream = new FileStream(open.FileName, FileMode.Open);

BinaryFormatter bf = new BinaryFormatter();

list = (List<string>)bf.Deserialize(stream);

stream.Close();

foreach (string words in list)

{

richTextBox1.Text += words + "\n";

}

}

**4. Задание для самостоятельной работы**

Добавить в программу предыдущей лабораторной работы возможность сохранения данных в файл и чтения из файла. Исходную коллекцию данных необходимо сохранять в файл в двоичном виде, а также предусмотреть ее сериализацию. Коллекцию-результат сохранять в текстовый файл.