Лабораторная работа №10

Работа с файловым потоком

Цель работы: получить навыки работы с потоковым чтением и записью на языке C#.

Теоретическая часть

При работе с файлом можно создать один единый поток, в котором будет производиться чтение из файла и запись в файл.

Для создания потока можно использовать класс `FileStream`. У этого класса есть множество конструкторов, которые позволяют открыть файл в различных режимах.

```
public class FileStream : Stream

    public FileStream(SafeFileHandle handle, FileAccess access);
    public FileStream(IntPtr handle, FileAccess access);
    public FileStream(string path, FileMode mode);
    public FileStream(SafeFileHandle handle, FileAccess access, int bufferSize);
    public FileStream(IntPtr handle, FileAccess access, bool ownsHandle);
    public FileStream(string path, FileMode mode, FileAccess access);
    public FileStream(SafeFileHandle handle, FileAccess access, int bufferSize, bool isAsync);
    public FileStream(IntPtr handle, FileAccess access, bool ownsHandle, int bufferSize);
    public FileStream(string path, FileMode mode, FileAccess access, FileShare share);
    public FileStream(string path, FileMode mode, FileAccess access, FileShare share, int bufferSize);
    public FileStream(IntPtr handle, FileAccess access, bool ownsHandle, int bufferSize, bool isAsync);
    public FileStream(string path, FileMode mode, FileAccess access, FileShare share, int bufferSize, bool useAsync);
    public FileStream(string path, FileMode mode, FileAccess access, FileShare share, int bufferSize, FileOptions options);
```

Рисунок 1 – Конструкторы класса FileStream

При создании потока можно указать режим работы с файлом, в котором будет создаваться поток. Для этого используется перечисление `FileMode`.

```
public enum FileMode

    CreateNew = 1,
    Create = 2,
    Open = 3,
    OpenOrCreate = 4,
    Truncate = 5,
    Append = 6
```

Рисунок 2 – Перечисление FileMode

Также можно указать режим доступа к файлу из этого потока с помощью перечисления `FileAccess`.

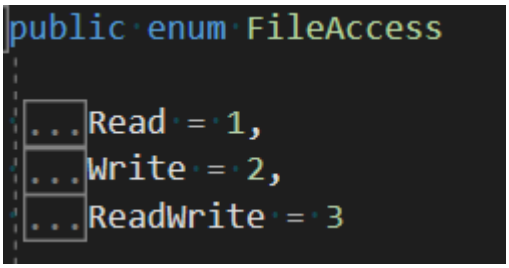

Рисунок 3 – Перечисление FileAccess

Попробуем создать новый поток для чтения и записи.

```
FileStream stream = new FileStream(
    @"D:\User\Source\NewDir\file.txt",
    FileMode.OpenOrCreate,
    FileAccess.ReadWrite,
    FileShare.Read);
```

Рисунок 4 – Создание потока

В классах `StreamReader` и `StreamWriter` есть множество конструкторов, которые принимают поток (класс `Stream`), по желанию можно указать нужную кодировку.

```
public StreamReader(Stream stream);
public StreamReader(string path);
public StreamReader(Stream stream, bool detectEncodingFromByteOrderMarks);
public StreamReader(Stream stream, Encoding encoding);
public StreamReader(string path, bool detectEncodingFromByteOrderMarks);
public StreamReader(string path, Encoding encoding);
public StreamReader(Stream stream, Encoding encoding, bool detectEncodingFromByteOrderMarks);
public StreamReader(string path, Encoding encoding, bool detectEncodingFromByteOrderMarks);
public StreamReader(Stream stream, Encoding encoding, bool detectEncodingFromByteOrderMarks, int bufferSize);
public StreamReader(string path, Encoding encoding, bool detectEncodingFromByteOrderMarks, int bufferSize);
public StreamReader(Stream stream, Encoding? encoding = null, bool detectEncodingFromByteOrderMarks = true, int bufferSize = -1, bool leaveOpen = false);
```

Рисунок 5 – Конструкторы класса StreamReader

```
public StreamWriter(Stream stream);
public StreamWriter(string path);
public StreamWriter(Stream stream, Encoding encoding);
public StreamWriter(string path, bool append);
public StreamWriter(Stream stream, Encoding encoding, int bufferSize);
public StreamWriter(string path, bool append, Encoding encoding);
public StreamWriter(Stream stream, Encoding? encoding = null, int bufferSize = -1, bool leaveOpen = false);
public StreamWriter(string path, bool append, Encoding encoding, int bufferSize);
```

Рисунок 6 – Конструкторы класса StreamWriter

Теперь создадим объекты `reader` и `writer` для работы с файлом.

```
StreamReader reader = new StreamReader(stream);

StreamWriter writer = new StreamWriter(stream);
```

Рисунок 7 – Создание объектов для работы с файлом

Не забываем сохранить изменения и очистить поток по окончании работы с файлом.

```
stream.Flush();
stream.Dispose();
```

Рисунок 8 – Конец работы с файлом

Практическая часть

1) Разработать класс `FileClass` для работы с текстовым файлом.
2) В классе должен быть определен закрытый конструктор, принимающий путь, по которому будет создан новый файл.
3) Статический метод `Create` принимает путь для нового файла и возвращает объект, созданный конструктором.
4) В классе должен быть определен закрытый конструктор, принимающий путь, по которому будет открыт существующий файл.
5) Статический метод `Open` принимает путь существующего файла и возвращает объект, созданный конструктором.
6) Метод `ReadLine` для чтения строки из файла.
7) Метод `Read` для чтения `count` символов из файла.
8) Метод `Writeline` для записи строки в файл.
9) Метод `Write` для записи `count` символов в файл.

10) Для чтения и записи использовать классы `StreamReader` и `StreamWriter` (поля `reader` и `writer`). Эти классы должны работать с одним и тем же потоком `FileStream`. Они создаются один раз в закрытом методе `CreateStream`. Этот метод используется в конструкторах.

11) Файл открывается только один раз. `reader` и `writer` должны создаваться только один раз, а не при каждом обращении к файлу. Поток, `reader` и `writer` должны закрываться тоже только один раз.

12) Конструкцию `using` использовать не нужно. Разработайте метод `Close` для сохранения изменений и освобождения ресурсов.

13) Продемонстрируйте работу с созданным классом.

Содержание отчета

1. Титульный лист
2. Цель работы
3. Задание
4. Код программы
5. Результат выполнения