Классы-адаптеры и стандартные потоки

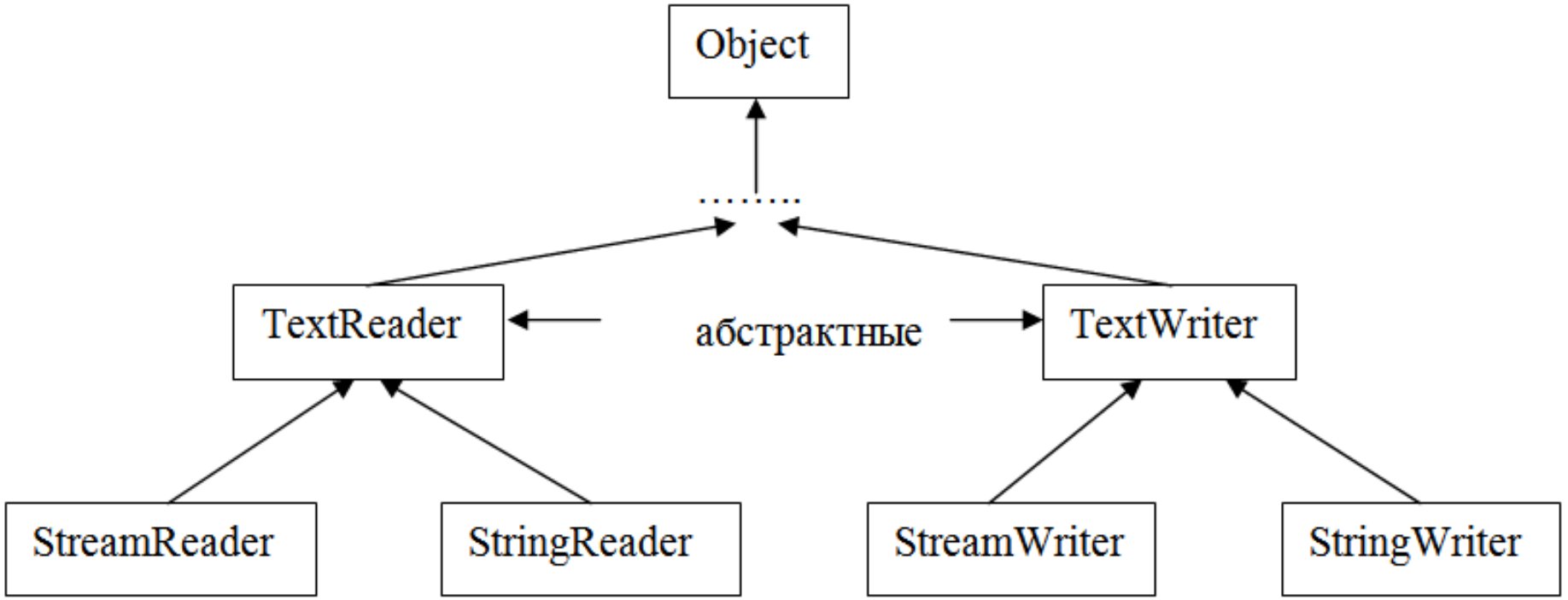
Классы-адаптеры

Текстовые адаптеры:
TextReader, TextWriter, StreamReader, StreamWriter, StringReader и StringWriter (предназначены для строк и символов, то есть для текстовой информации).

Двоичные адаптеры:
BinaryReader и BinaryWriter (предназначены для базовых типов **int, long, float**,. . .).

Адаптеры XML:
XmlReader и XmlWriter (предназначены для XML файлов).

Классы-адаптеры для чтения и записи текстовых файлов



StreamWriter/Reader использует в качестве источника данных поток (Stream) и кодировку (Encoding).
StringWriter/Reader использует StringBuilder / string в качестве источника.

Конструкторы класса StreamWriter

Конструкторы на основе потока:

StreamWriter(Stream поток [, Encoding кодировка[, int буфер]])

Конструкторы на основе имени файла:

StreamWriter(String путь [, Boolean режим[, Encoding кодировка[, int буфер]]])

режим – добавлять ли данные в существующий файл (append)

Примеры:

```
StreamWriter sw = new StreamWriter("Text.txt");// utf-8

StreamWriter dos = new StreamWriter("MS_Dos.txt", true,
        Encoding.GetEncoding(866));

FileStream fs = new FileStream(@"my_Doc.txt", FileMode.CreateNew,
        FileAccess.Write);
StreamWriter sw2 = new StreamWriter(fs);
```

https://source.dot.net/#System.Private.CoreLib/StreamWriter.cs

Запись в поток с помощью объекта класса StreamWriter

*Write(**string** строка)*
*Write(**char** символ)*
*Write(**char** массив_символов [ ])*
*Write(**char** [ ] array, **int** начало, **int** длина)*
*Write(**double** значение)*
*Write(**int** значение)*
*Write(**long** значение)*
*Write(**string** формат, **object** значение)*

Аналогично для WriteLine()
Конец строки: ‘\r’ + ‘\n’ (13+10 или 0d+0a)

Члены класса StreamWriter

Close() – закрывает поток StreamWriter.
Flush() – очищает буферы.

В классах **StreamWriter** и **StreamReader**

отсутствуют реализации:

1. метод **Seek**(),
2. свойство **Position**,
3. свойство **Length**.

Конструкторы класса StreamReader

Конструкторы на основе потока:

StreamReader(Stream** поток **[**, **Encoding** кодировка**[**, **Boolean** флаг**]])

Конструкторы на основе имени файла:

StreamReader(String** путь **[**, **Encoding** кодировка**[**, **Boolean** флаг**]])

флаг – detectEncodingFromByteOrderMarks

Кодировка	BOM (hex)	BOM (dec)
UTF-8	EF BB BF	239 187 191

Чтение из потока с помощью объекта класса StreamReader

***int** Read()*

***int** Read(**char[ ]** массив, **int** начало, **int** количество)*

***string** ReadLine()*

***int** Peek()*

Класс с информацией о файле

```
class Information {    // Сведения о файле

    public string FileName { get; set; }
    public long Length { get; set; }

    public Information(string n, long d)
    { FileName = n; Length = d; }

    public override string ToString()
        => $"fileName={FileName} length={Length}";

    public Information(string st) {
        try {
            string[] res = st.Split(new char[] { ' ', '=' },
StringSplitOptions.RemoveEmptyEntries);
            FileName = res[1];
            Length = long.Parse(res[3]);
        }
        catch { throw new FormatException("Не верен формат
аргумента!"); }
    }
}
```

Пример со StreamWriter

```
// Первый проект: запись сведений о файлах
public static void Main()
{
    string path = ".";
    DirectoryInfo dirf = new DirectoryInfo(path);
    Console.WriteLine("*** Имя каталога: " + dirf.Name);
    FileInfo[] files = dirf.GetFiles();
    using (StreamWriter sw = new
StreamWriter(@"..\..\..\FilesList.txt"))
    {
        foreach (FileInfo fi in files)
        {
            Information data = new Information(fi.Name, fi.Length);
            sw.WriteLine(data.ToString());
        } // foreach
    }  // using
}
```

Пример со StreamWriter / StreamReader

```
// Второй проект: чтение сведений о файлах
public static void Main()
{
    string path = @"..\..\..\FilesList.txt";
    if (!File.Exists(path)) {
        Console.WriteLine("Файл не найден!");
        return;
    }
    using (StreamReader sr = new StreamReader(path))
    {
        string text;
        Information inf = null;
        while ((text = sr.ReadLine()) != null)
        {
            inf = new Information(text);
            Console.WriteLine(inf.ToString());
        }
    }  // using
}  // Main( )
```

Результаты в FilesList.txt

fileName=WriteInform.exe length=6144

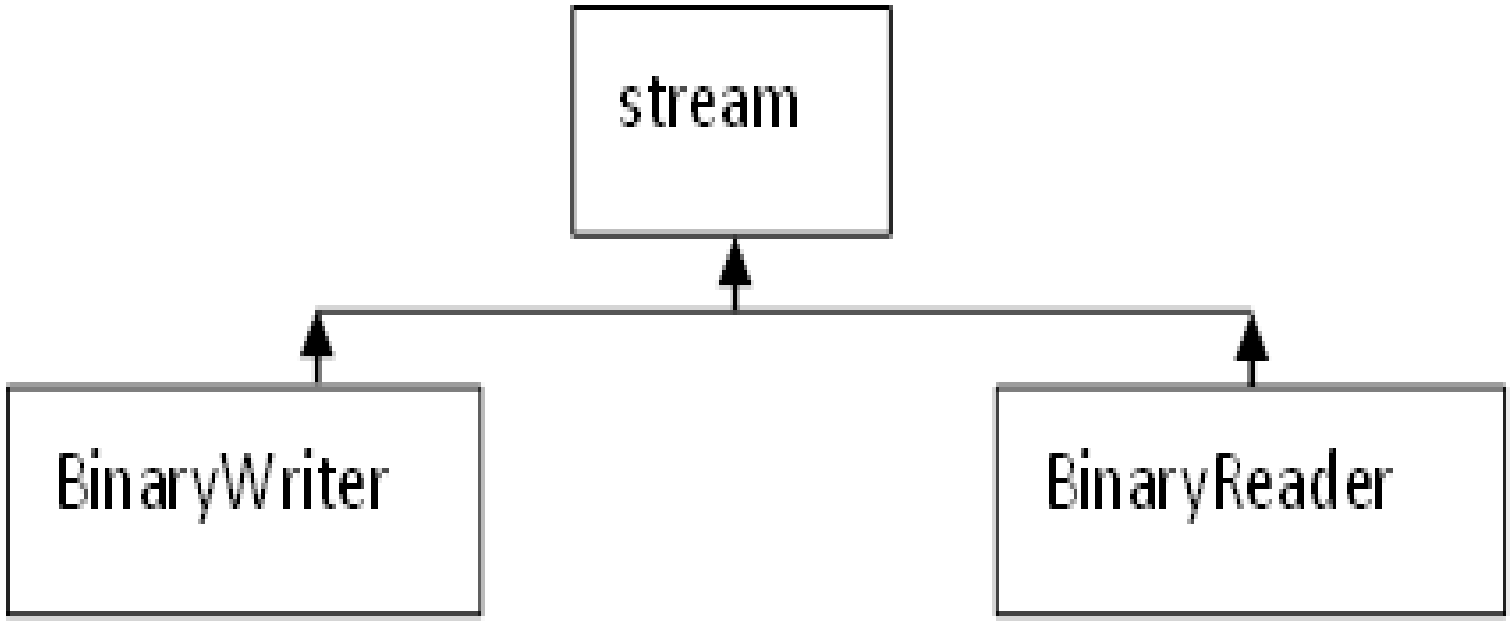
fileName=WriteInform.exe.config length=187

fileName=WriteInform.pdb length=13824

fileName=WriteInform.vshost.exe length=22984

fileName=WriteInform.vshost.exe.config length=187

Двоичные адаптеры



Конструкторы классов BinaryWriter и BinaryReader

BinaryWriter (Stream *поток*)
BinaryWriter (Stream *поток*, Encoding *кодировка*)
BinaryWriter (Stream *поток*, Encoding *кодировка*,
Bool *открыт*)

BinaryReader (Stream *поток*)
BinaryReader (Stream *поток*, Encoding *кодировка*)
BinaryReader (Stream *поток*, Encoding *кодировка*,
Bool *открыт*)

Средства записи BinaryWriter

public virtual void Write(**byte**[ ] *буфер*)

public virtual void Write(**byte**[ ] *буфер*, **int** *индекс-начало*, **int** *счетчик*)

public virtual void Write(**char**[] *буфер*, **int** *индекс-начало*, **int** *счетчик*)

protected void Write7BitEncodedInt(**int** *value*)

public virtual void Write(. . .)

Класс BinaryWriter

реализует интерфейс IDisposable

Члены класса BinaryWriter

Методы:

public virtual void Close() – Закрывает бинарный и базовый потоки.

public void Dispose() – Освобождает ресурсы.

public virtual void Flush() – Очищает буферы.

public virtual long Seek(int смещение,

SeekOrigin точка_отсчета) – устанавливает позицию записи.

BaseStream – свойство класса BinaryWriter

BinaryReader

Конструкторы:

- BinaryReader (Stream *поток*)
- BinaryReader (Stream *поток*, Encoding *кодировка*)
- BinaryReader (Stream *поток*, Encoding *кодировка*, Bool *открыт*)

открыт == leaveOpen

BaseStream – свойство

Методы:

- Close( ) – …
- PeekChar( ) – “подсмотреть” символ (возвращает следующий доступный для чтения символ, не перемещая позицию байта или символа вперед).
- Dispose() – …

Чтение из BinaryReader

int Read( ) – чтение отдельного символа

void Read(**byte**[] *буфер*, **int** *индекс-начало*, **int** *счётчик*)

void Read(**char**[] *буфер*, **int** *индекс-начало*, **int** *счётчик*)

int Read7BitEncodedInt() – читает упакованное целое число.

Если целое число будет помещаться в семь бит, целое число займет только один байт. (ожидается, что целое число записали через BinaryWriter.Write7BitEncodedInt()).

Исключение IOException

Чтение из BinaryReader значений базовых типов

bool ReadBoolean()
byte ReadByte()
byte [] ReadBytes(Int32)
char ReadChar()
char [] ReadChars(Int32)
decimal ReadDecimal()
double ReadDouble()
short ReadInt16()
int ReadInt32()
long ReadInt64()
sbyte ReadSByte()
float ReadSingle()
string ReadString()
ushort ReadUInt16()
uint ReadUInt32()
ulong ReadUInt64()

Класс для иллюстрации двоичной записи

```
class Discovery {
    public string Name { get; set; }
    public int Date { get; set; }
}
```

Пример с BinaryWriter

```
static void Main( )    {
  Discovery[] discoveries =  {
    new Discovery { Name = "Радиоприемник", Date = 1895 },
    new Discovery { Name = "Мазер", Date = 1954 } ,
    new Discovery { Name = "Парашют", Date = 1911 },
    new Discovery { Name = "Гальванопластика", Date = 1840 },
    new Discovery { Name = "Коллайдер", Date = 1960 },
    new Discovery { Name = "Иконоскоп", Date = 1929 }
  };
  using (FileStream fs = File.Create(@"..\..\..\data.bin"))
   using (BinaryWriter bw = new BinaryWriter(fs))
    foreach(Discovery dis in discoveries)  {
          bw.Write(dis.Name);
          bw.Write(dis.Date);
    }
}
```

Пример с BinaryReader

```
static void Main( )  {
    using(FileStream fs = new
FileStream(@"..\..\..\data.bin",FileMode.Open))
    using(BinaryReader br = new BinaryReader(fs)) {
        Console.WriteLine("Из файла \"data.bin\" сведения:");
        while(true)
            try {
                string name = br.ReadString();
                int date = br.ReadInt32();
                Console.WriteLine("Name={0}, Date={1}",name, date);
            }
        catch (EndOfStreamException) { break; }
    }
}
```

Пример с BinaryReader. Результаты

Из файла "data.bin" сведения:

Name=Радиоприемник, Date=1895

Name=Мазер, Date=1954

Name=Парашют, Date=1911

Name=Гальванопластика, Date=1840

Name=Коллайдер, Date=1960

Name=Иконоскоп, Date=1929

Пример с BinaryWriter

```
// Запись целых в двоичный поток
using System;
using System.IO;
class Program    {
        static void Main( )        {
            BinaryWriter fOut = new BinaryWriter(
                new FileStream("t.dat", FileMode.Create) );
            for (int i = 0; i <= 10; i += 2)
                fOut.Write(i);
            fOut.Close();
        }
}
```

Пример с BinaryReader

```
//.. Чтение целых из двоичного потока
using System;
using System.IO;
    class Program    {
        static void Main( )   {
            FileStream f = new FileStream("t.dat", FileMode.Open);
            BinaryReader fIn = new BinaryReader(f);
            long n = f.Length / 4;        int a;
            for (int i = 0; i < n; i++)
            {  a = fIn.ReadInt32(); Console.Write(a + " ");  }
            fIn.Close();
            f.Close();
        }
}
```

Результаты чтения: 0 2 4 6 8 10

Пример с BinaryReader и позиционированием

```
// позиционирование при чтении из двоичного потока:
FileStream f = new FileStream("../../../t.dat", FileMode.Open);
BinaryReader fIn = new BinaryReader(f);
        long n = f.Length;
        int a;
        for (int i = 0; i < n; i+=8)    {
            f.Seek(i, SeekOrigin.Begin);
            a = fIn.ReadInt32(); Console.Write(a + " "); }
        fIn.Close();
        f.Close();
    }
```

Результаты чтения: 0 4 8

Write7BitEncodedInt() и Read7BitEncodedInt()

```
protected void Write7BitEncodedInt(int value)
protected int Read7BitEncodedInt()
```

Вспомогательные классы

```
public class MyBinaryWriter : BinaryWriter {
public MyBinaryWriter(Stream stream) : base(stream) { }
public new void Write7BitEncodedInt(int i)
    { base.Write7BitEncodedInt(i); }
}
public class MyBinaryReader : BinaryReader {
public MyBinaryReader(Stream stream) : base(stream) { }
public new int Read7BitEncodedInt() {
    return base.Read7BitEncodedInt(); }
}
```

```
static void Main()  {  // Запись в двоичный поток
    MemoryStream stream = new MemoryStream();
    MyBinaryWriter writer = new MyBinaryWriter(stream);
    writer.Write7BitEncodedInt(127);
Console.WriteLine("BaseStream.Length = " + stream.Length);
    writer.Write7BitEncodedInt(127);
Console.WriteLine("BaseStream.Length = " + stream.Length);
    writer.Write7BitEncodedInt(256);
Console.WriteLine("BaseStream.Length = " + stream.Length);
    writer.Write7BitEncodedInt(4096);
Console.WriteLine("BaseStream.Length = " + stream.Length);
    writer.Write7BitEncodedInt(-4096);
Console.WriteLine("BaseStream.Length = " + stream.Length);
```

```
stream.Position = 0; // Чтение из битового потока
MyBinaryReader reader = new MyBinaryReader(stream);
    Console.WriteLine(reader.Read7BitEncodedInt());
    Console.WriteLine(reader.Read7BitEncodedInt());
    Console.WriteLine(reader.Read7BitEncodedInt());
    Console.WriteLine(reader.Read7BitEncodedInt());
    Console.WriteLine(reader.Read7BitEncodedInt());
}
```

Результаты выполнения программы

BaseStream.Length = 1

BaseStream.Length = 2

BaseStream.Length = 4

BaseStream.Length = 6

BaseStream.Length = 11

127

127

256

4096

-4096

Стандартные потоки

Класс **Console** не принадлежит пространству System.IO

Символьные потоки – свойства класса **Console**:

Console.In – объект класса **TextReader**

Console.Out - объект класса **TextWriter**

Console.Error - объект класса **TextWriter**

Свойства класса Console

- **public static TextReader In {get; }**
- **public static TextWriter Out {get; }**
- **public static TextWriter Error {get; }**

Применение потока Console.Error

```
using System;
class Program    {
static void Main()   {
    string[] lines = { "45","1.6","2,45","11" };
    int sum =0;
    try    {
       foreach(string st in lines)
          sum += int.Parse(st);
    }
    catch (Exception exc)     {
       Console.Error.WriteLine(exc.Message); }
    }
}
```

Результат выполнения программы:
Входная строка имела неверный формат.

Перенаправление стандартных потоков средствами операционной системы

1) Для потока Console.Out:

example.exe > NewFile.txt **// перезапись**

example.exe >> OldFile.txt **// добавление**

2) Для потока Console.In

example.exe < OldFile.txt

3) Для двух потоков (Console.Out, Console.In):

example.exe > NewFile.txt < OldFile.txt

Перенаправление стандартных потоков программными средствами

Статические методы класса Console:

static void SetIn(TextReader input)

static void SetOut(TextWriter output)

static void SetError(TextWriter output)

Перенаправление стандартных потоков

```
using System;
using System.IO;
class Program    {
static void Main()   {
    StreamWriter system_log = new StreamWriter(@"D:\system_log.txt");
    Console.SetOut(system_log);
    DateTime dt = DateTime.Now;
    Console.WriteLine("Начало системного журнала.");
    Console.WriteLine(dt + "; " + dt.Millisecond + " Milliseconds");
    for (int k = 0; k < 100000; k++)
        if (k % 10000 == 0) Console.WriteLine(k);
    Console.WriteLine("Конец системного журнала.");
    dt = DateTime.Now;
    Console.WriteLine(dt + "; " + dt.Millisecond + " Milliseconds");
    system_log.Close();
    }
}
```

Результат в файле "D:\system_log.txt":

Начало системного журнала.

09.03.2019 9:26:25; 390 Milliseconds

0

10000

20000

30000

40000

50000

60000

70000

80000

90000

Конец системного журнала.

09.03.2019 9:26:25; 420 Milliseconds

Методы получения ссылок на стандартные потоки

public static Stream OpenStandardError()

public static Stream OpenStandardInput()

public static Stream OpenStandardOutput()

«Восстановление» стандартных потоков

Для Console.In:

```
TextReader tr = Console.In;
. . . .
Console.SetIn(tr);
```

Для Console.Out:

```
TextWriter tw = Console.Out;
. . . .
Console.SetOut(tw);
```

Переназначение стандартного входного потока

```
using System;
using System.IO;
class Ввод_из_файла  {
static void Main()   {
    StreamReader input =
        new StreamReader(@"..\..\Ввод_из_файла.cs");
    Console.SetIn(input);
    string line;
    do {
         line = Console.ReadLine();
         Console.WriteLine(line);
       } while(line != null);
    }
}
```

Чтение данных из файла или консоли

```
static void Main(string [] args)   {
if(args.Length > 0)  {
        StreamReader sr = new StreamReader(args[0]);
        Console.SetIn(sr);
        Console.WriteLine("Данные прочитаны из файла "+args[0]);
        }
   else Console.WriteLine("Вводите числа с клавиатуры:");
   int x = 0, sum=0;
   while(true) {
      try { x = int.Parse(Console.ReadLine()); }
      catch { if (args.Length == 0) continue;
            else if(Console.In.Peek() == -1) break;
                  else continue;          }
      sum += x;
      if(x == 0) break;
   }  // while
   Console.WriteLine("sum = "+sum);
}
```

Результат выполнения

- **Запуск №1**

Данные прочитаны из файла in.txt

sum = 18

- **Запуск №2**

Вводите числа с клавиатуры:

3

5 6

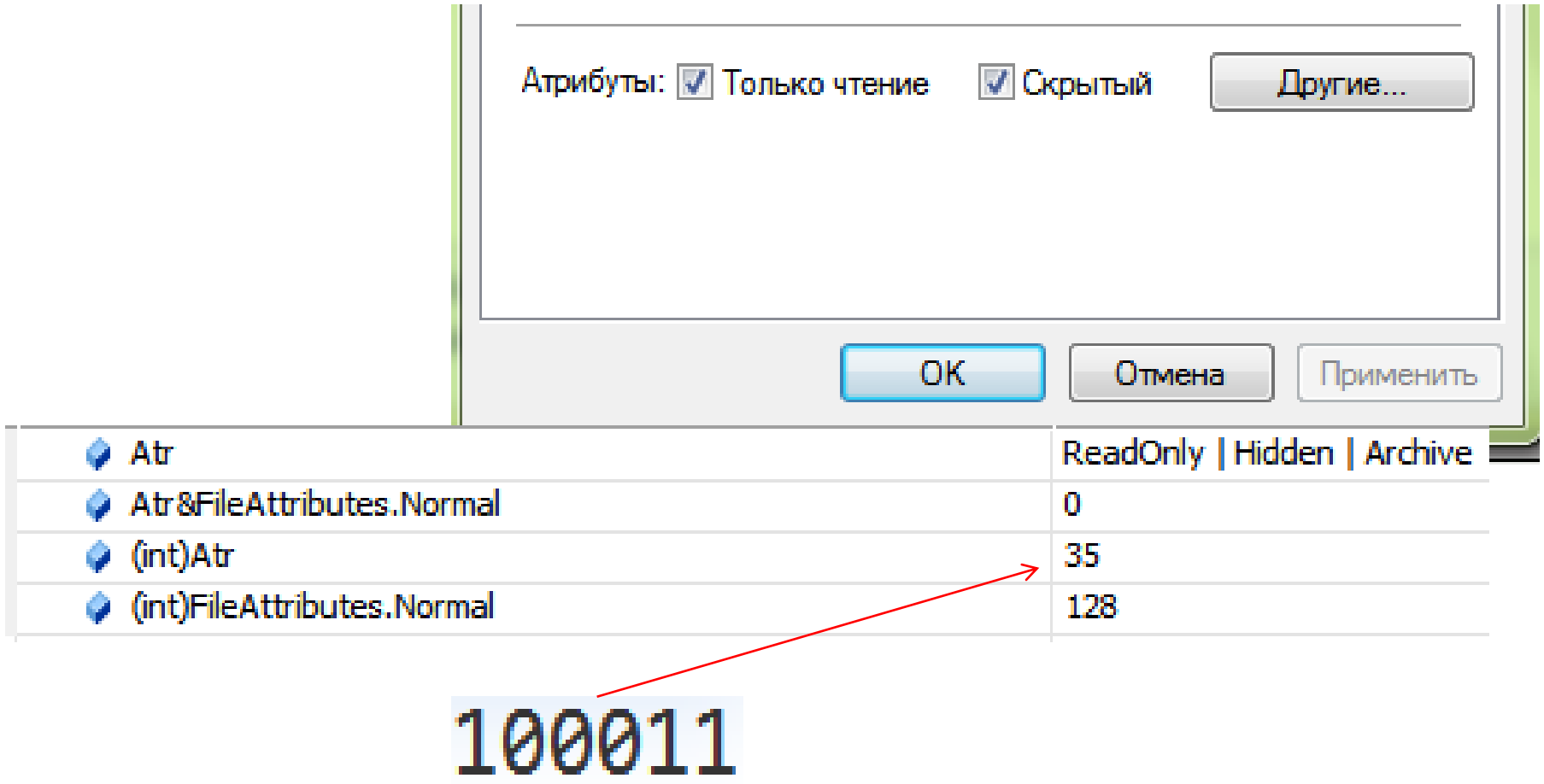
7

0

sum = 10

Значения перечисления System.IO.FileAttributes

```
FileInfo FP1 = new FileInfo(@"file.txt");
FileAttributes Atr = FP1.Attributes;
```



100011

Работа с буфером (пример)

```
static void Main(string[] args)
{
    StreamWriter FP = new StreamWriter("File.txt");
    int j = 0;
    do
    {

        for(int i = 0; i <100; j++, i++, FP.WriteLine(" " + j))
            FP.Write("s");
    } while (Console.ReadKey().Key == ConsoleKey.Enter);
    FP.Close();
}
```