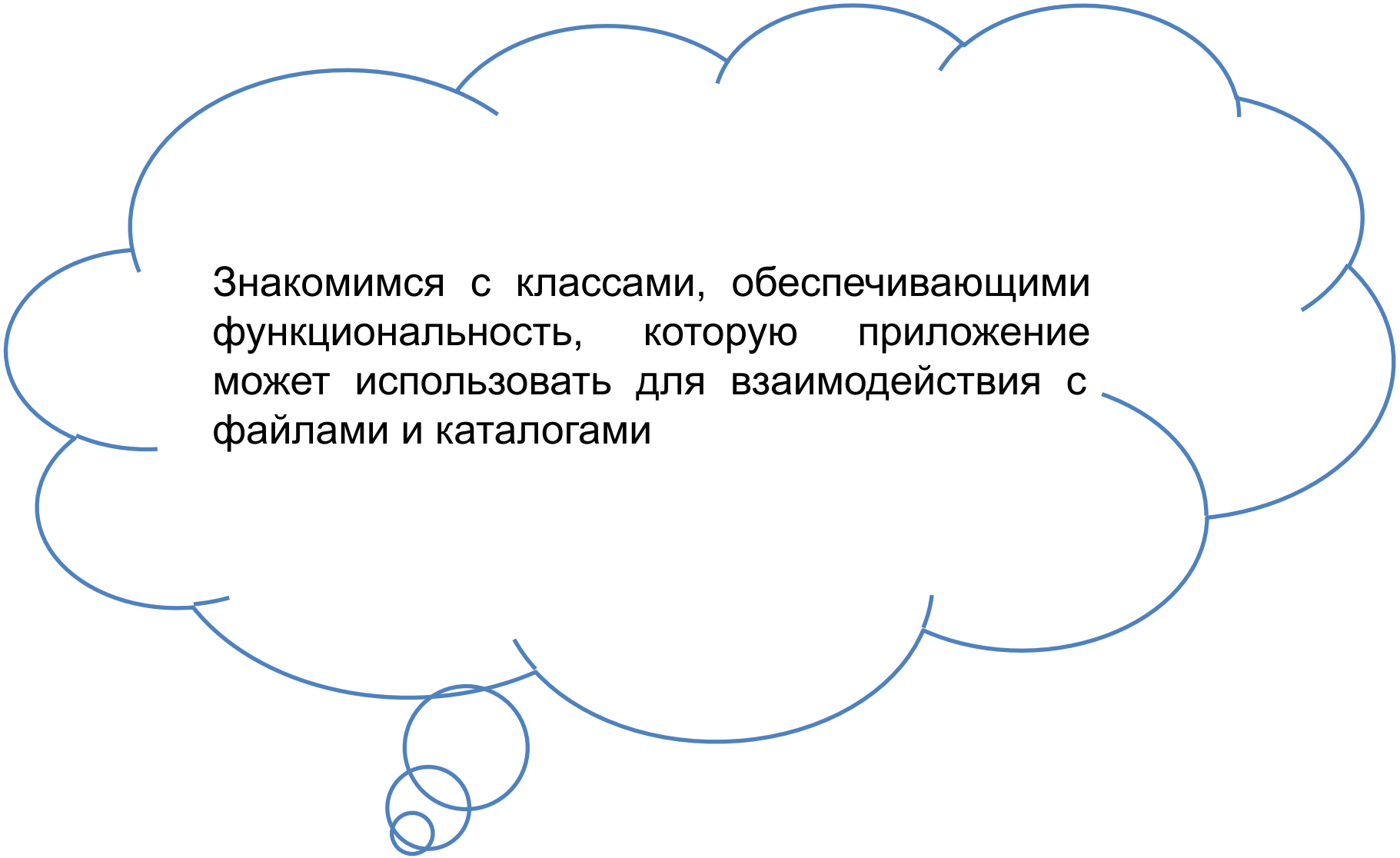
NET.C#.09 Работа с файловой системой

Resource Development Dep.
Author:

Доступ к файловой системе

Чтение и запись файлов с помощью потоков

Доступ к файловой системе



Управление файлами

Для многих приложений общим требованием является способность взаимодействовать с файлами файловой системы

- создание новых файлов
- копирование файлов
- удаление файлов
- перемещение файлов из одной директории в другую

.NET Framework предоставляет несколько классов из пространства имен System.IO. к которым относятся классы File и FileInfo

Класс File

Copy()	**Exists()**	
Create()	**Move()**	**. . .**
Delete()	**Copy()**	

Управление файлами

Класс FileInfo

CopyTo()	Exists	. . .
Open()	Length	
Delete	Name	

```
string sourceFile = "...";
string destFile = "...";
bool overwrite = false;
File.Copy(sourceFile, destFile, overwrite);
```

```
string filePath = "...";
FileInfo file = new FileInfo(filePath);
string destPath = "...";
file.CopyTo(destPath);
```

Чтение из файлов и запись в файлы

Чтение данных из файлов

```
string filePath = "myFile.txt";
byte[] data = File.ReadAllBytes(filePath);
string[] lines = File.ReadAllLines(filePath);
string data = File.ReadAllText(filePath);
```

Запись данных в файлы

```
string[] fileLines = { "Line 1", "Line 2", "Line 3" };
File.AppendAllLines(filePath, fileLines);
string fileContents = "I am writing this text to a file called
myFile.txt";
File.AppendAllText(filePath, fileContents);
byte[] fileBytes = { 12, 134, 12, 8, 32 };
File.WriteAllBytes(filePath, fileBytes);
string[] fileLines = { "Line 1", "Line 2", "Line 3" };
File.WriteAllLines(filePath, fileLines);
string fileContents = "I am writing this text to a file called
myFile.txt";
File.WriteAllText(filePath, fileContents);
```

Управление директориями

.NET Framework предлагает классы Directory и DirectoryInfo пространства имен System.IO, позволяющие производить поиск и управлять каталогами

Класс Directory

```
string dirPath = "...";
Directory.CreateDirectory(dirPath);
string[] files = Directory.GetFiles(dirPath);
bool dirExists = Directory.Exists(dirPath);
string sourcePath = "...";
string destPath = "...";
Directory.Move(sourcePath, destPath);
```

Класс DirectoryInfo

```
string dirPath = ". . .";
DirectoryInfo dir = new DirectoryInfo (dirPath);
dir.Create();
dir.Delete();
bool exists = dir.Exists;
DirectoryInfo[] dirs = dir.GetDirectories();
```

Управление путями

Пространство имен System.IO содержит класс Path, который может использовать для анализа и построения файлов и имен папок для указанной файловой системы

- GetDirectoryName()
- GetFileNameWithoutExtension
- GetExtension()
- GetRandomFileName
- GetFileName()
- GetTempFileName

```
string path = " . . . ";
string dirs = Path.GetDirectoryName(path);
string ext = Path.GetExtension(path);
string fileName = Path.GetFileName(path);
string fileName = Path.GetFileNameWithoutExtension(path);
```

Использование общих диалоговых окон файловой системы

Пространство имен Microsoft.Win32 содержит классы OpenFileDialog, иобеспечивающие функциональность, позволяющую пользователю просматривать файл или указывать имя файла, а также создавать требуемые папки

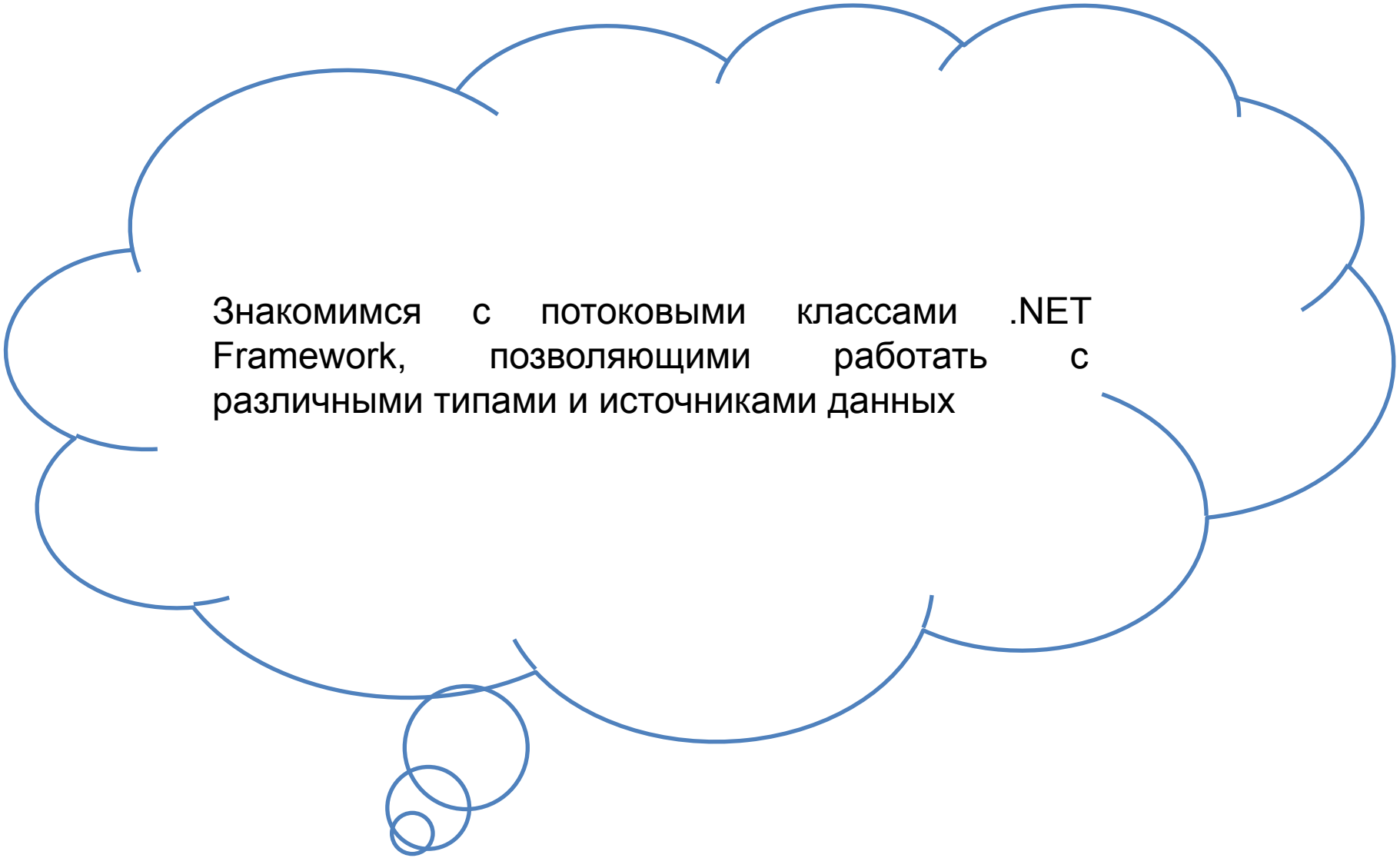
Класс OpenFileDialog

```
OpenFileDialog openDlg = new OpenFileDialog();
...
openDlg.Title = "Browse for a file to open";
openDlg.Multiselect = false;
openDlg.InitialDirectory = ". . . ";
openDlg.Filter = "Word (*.doc) | *.doc;";
```

Класс SaveFileDialog

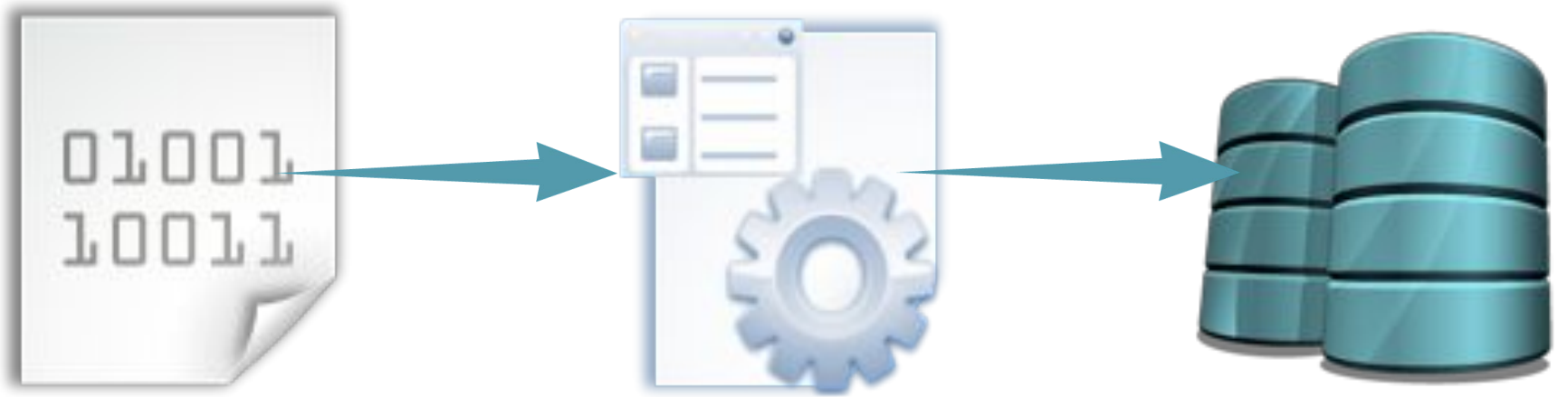
```
SaveFileDialog saveDlg = new SaveFileDialog();
...
saveDlg.Title = "Browse for a save location";
saveDlg.DefaultExt = "doc";
saveDlg.AddExtension = true;
saveDlg.InitialDirectory = @"C:\Users\Student\Documents";
saveDlg.OverwritePrompt = true;
```

Чтение и запись файлов с помощью потоков



Что такое потоки?

Поток представляет собой последовательность байтов, которые могут поступать из файла файловой системы, сети связи или памяти и позволяют считывать или записывать данные в источник данных посредством небольших управляемых пакетов данных



Что такое потоки?

Потоки обеспечивают следующие операции

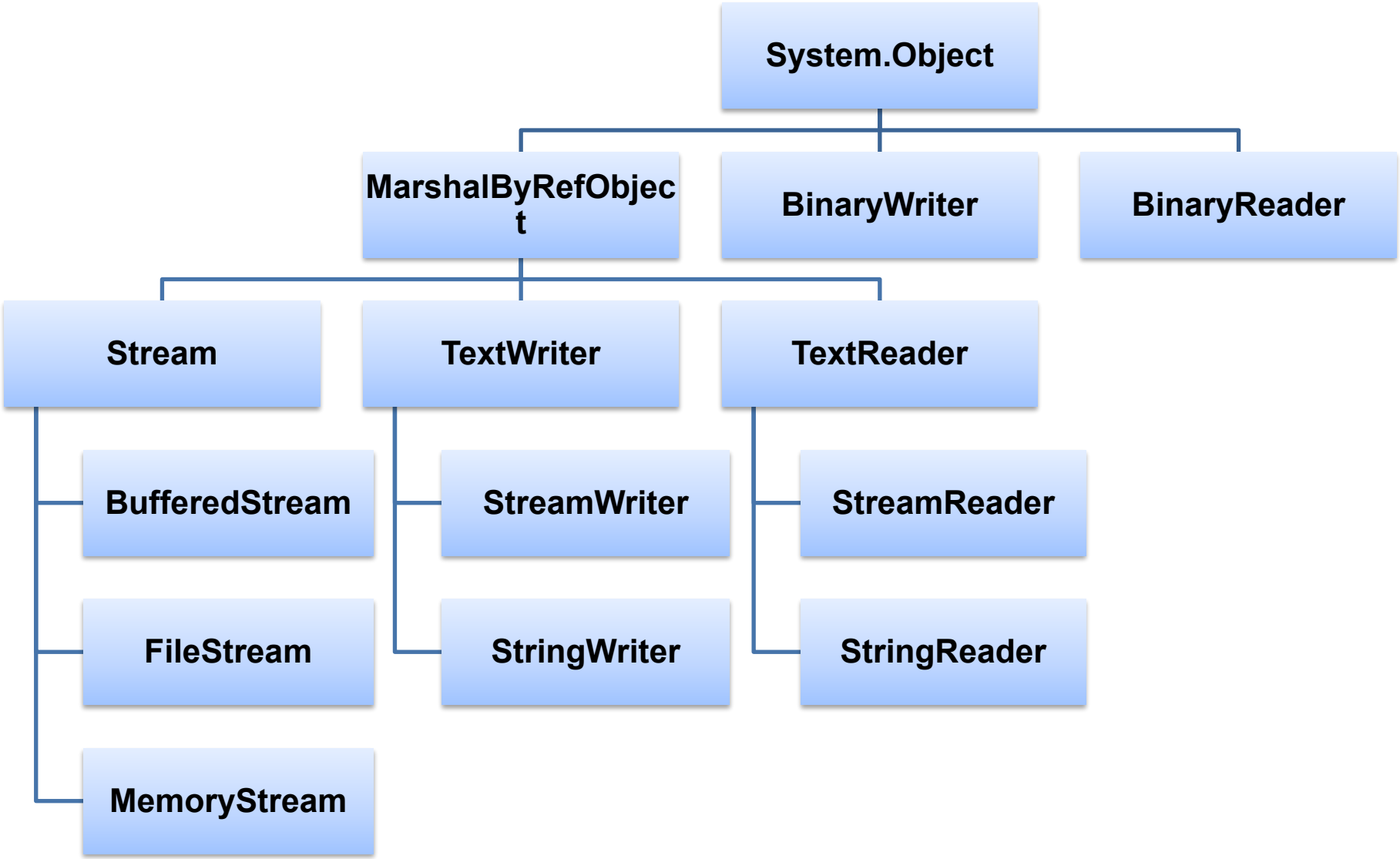
- чтение из потока
- запись в поток
- поиск в потоке (зависит от вида резервного хранилища потока)

NET Framework предоставляет несколько потоковых классов, позволяющих работать с различными данными и источниками данных, для выбора какого-либо из них, необходимо учитывать следующее:

- Какой тип данных читается или записывается, например, двоичный или буквенно-цифровой
- Где хранятся данные, например, в локальной файловой системе, в памяти или на веб-сервере в сети

Библиотека классов .NET Framework предоставляет несколько классов из пространства имен System.IO, которые можно использовать для чтения и записи файлов с помощью потоков

Что такое потоки?



Что такое потоки?

Класс Stream определяет общие функциональные возможности, которые обеспечивают все потоки; он представляет универсальное представление последовательности байтов вместе с операциями и свойствами, которые обеспечивают все потоки

Класс Stream и его производные классы предоставляют универсальное представление различных типов ввода и вывода, изолируя программиста от отдельных сведений операционной системы и базовых устройств

Класс Stream нельзя использовать напрямую, нужно реализовать специализации этого класса, оптимизированные для выполнения потокового ввода/вывода для конкретных типов источников данных

Чтение и запись двоичных данных

При построении объектам BinaryReader и BinaryWriter предоставляется поток, подключенный к источнику данных, из которого нужно читать или в который необходимо писать

```
string filePath = "...";
FileStream file = new FileStream(filePath);
...
BinaryReader reader = new BinaryReader(file);
...
BinaryWriter writer = new BinaryWriter(file);
```

- Close()
- ReadBytes()
- Read()
- BaseStream
- ReadByte()

- Close()
- Write()
- Flush()
- BaseStream
- Seek()

Когда использование объектов BinaryReader или BinaryWriter завершено, необходимо вызвать метод Close, чтобы флешировать поток и освободить любые ресурсы, связанные с потоком

Необходимо также закрыть объект FileStream, предоставляющий данные для объектов BinaryReader и BinaryWriter.

Чтение и запись двоичных данных

```
string destinationFilePath = @"C:\. . .\BinaryDataFile.bin";
byte[] dataCollection = { 1, 4, 6, 7, 12, 33, 26, 98, 82, 101
};
FileStream destFile = new FileStream(
                      destinationFile...th,
                      FileMode.Create
                      FileAccess.Write);
BinaryWriter writer = new BinaryWriter(destFile)
foreach (byte data in dataCollecti...
{
    writer.Write(data);
}
writer.Close();
destFile.Close();
```

Коллекция байт

Создание объекта FileStream для взаимодействия с файловой системой

Создание объекта BinaryWriter, передавая объект FileStream как параметр

Запись каждого байта в поток

Закрыть оба потока для сброса данных в файл

```
00000000  01 04 06 07 0C 21 1A 62  52 65
```

Чтение и запись двоичных данных

```
string sourceFilePath = @"C:\. . .\BinaryDataFile.bin";
FileStream sourceFile = new FileStream(
                        sourceFilePath,
                        FileMode.Open
                        FileAccess.Read);
BinaryReader reader = new BinaryReader(sourceFile);
int position = 0;
int length = (int)reader.BaseStream.Length;
byte[] dataCollection = new byte[length];
int returnedByte;
while ((returnedByte = reader.Read()) != -1)
{
    dataCollection[position] = (byte)returnedByte;
    position += sizeof(byte);
}
reader.Close();
sourceFile.Close();
```

Создание объекта FileStream для взаимодействия с файловой системой

Создание объекта BinaryWriter, передавая объект FileStream как параметр

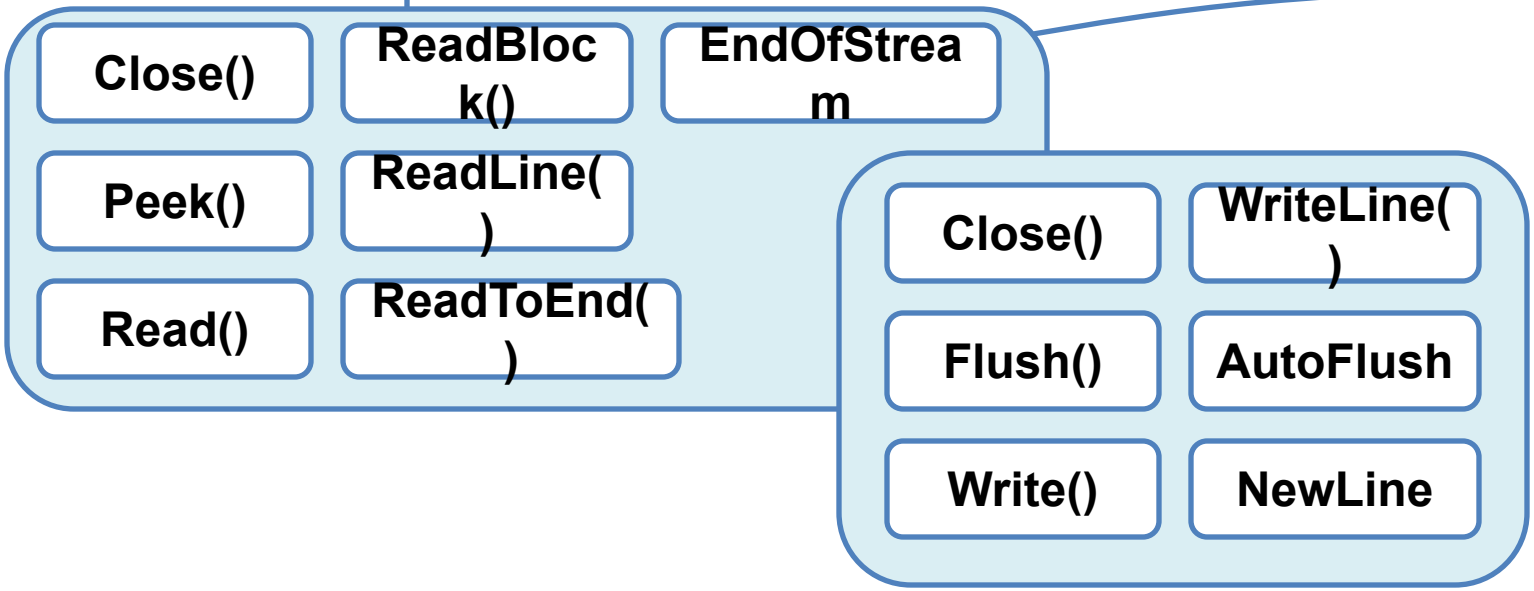
Закрытие потоков для освобождения всех дескрипторов файлов

Если операции чтения из файла или записи в файл генерируют исключение, следует убедиться, что потоки и дескрипторы файлов освобождены

Чтение и запись текста

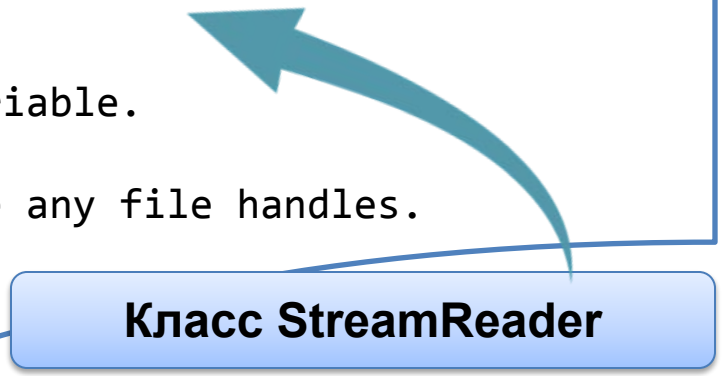
Для чтения и записи текстовых данных в файл используются классы StreamReader и StreamWriter

```
string filePath = "...";
FileStream file = new FileStream(filePath );
...
StreamReader reader = new StreamReader(file);
...
StreamWriter writer = new StreamWriter(file);
```



Чтение и запись текста

```
string sourceFilePath = @"C:\. . .\TextDataFile.txt";
// Create a FileStream object so that you can interact with the file system
FileStream sourceFile = new FileStream(
                        sourceFilePath, // Pass in the source file path.
                        FileMode.Open,   // Open an existing file.
                        FileAccess.Read);// Read an existing file.
StreamReader reader = new StreamReader(sourceFile);
StringBuilder fileContents = new StringBuilder();
// Check to see if the end of the file has been reached.
while (reader.Peek() != -1)
{
    // Read the next character.
    fileContents.Append((char)reader.Read());
}
// Store the file contents in a new string variable.
string data = fileContents.ToString();
// Always close the underlying streams release any file handles.
reader.Close();
sourceFile.Close();
```



Чтение и запись текста

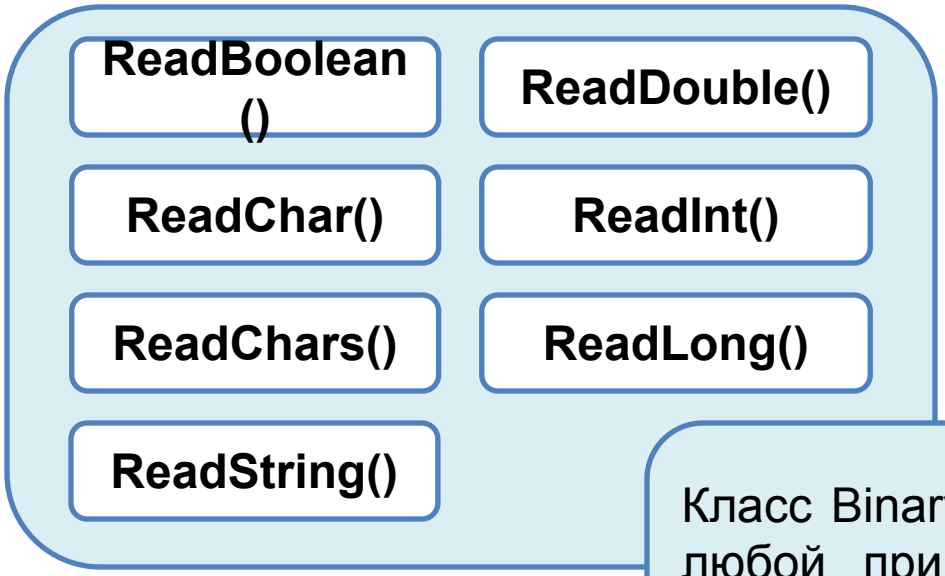
```
string sourceFilePath = @"C:\. . .\TextDataFile.txt";
string data;
// Create a FileStream object so that you can
// interact with the file system.
FileStream sourceFile = new FileStream(
                            sourceFilePath,  // Pass in the source file path.
                            FileMode.Open,   // Open an existing file.
                            FileAccess.Read);// Read an existing file.
StreamReader reader = new StreamReader(sourceFile);
// Read the entire file into a single string variable.
data = reader.ReadToEnd();
// Always close the underlying streams release any file handles.
reader.Close();
sourceFile.Close();
```

Чтение и запись текста

```
string destinationFilePath = @"C:\. . .\TextDataFile.txt";
string data = "Hello, this will be written in plain text";
// Create a FileStream object so that you can interact with the file
// system.
FileStream destFile = new FileStream(
                      destinationFilePath, // Pass in the destination path.
                      FileMode.Create,     // Always create new file.
                      FileAccess.Write);   // Only perform writing.
// Create a new StreamWriter object.
StreamWriter writer = new StreamWriter(destFile);
// Write the string to the file.
writer.WriteLine(data);
// Always close the underlying streams to flush the data to the file
// and release any file handles.
writer.Close();
destFile.Close();
```

Чтение и запись примитивных типов данных

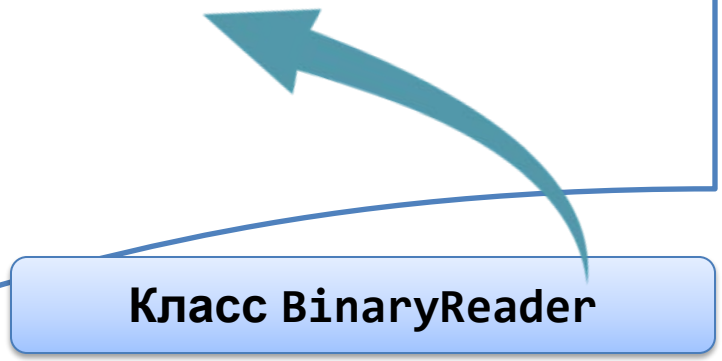
Классы BinaryReader и BinaryWriter предоставляют методы, позволяющие читать и записывать любые данные в любой примитивный тип данных, который включает в себя целые, вещественные, булевские данные и строки



Класс BinaryReader позволяет читать любой примитивный тип данных с помощью 16 конкретных методов чтения

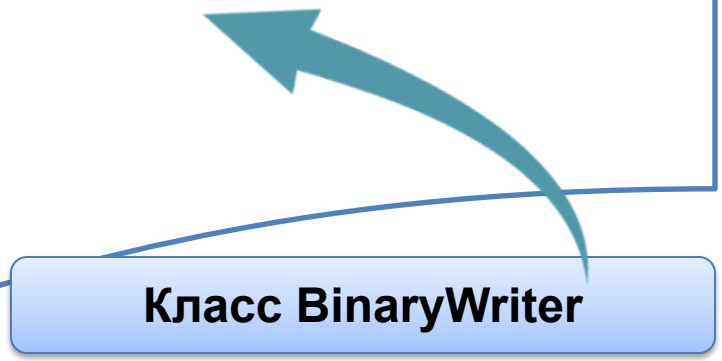
Чтение и запись примитивных типов данных

```
string sourceFilePath = @"C:\. . .\PrimitiveDataTypeFile.txt";
FileStream sourceFile = new FileStream(
                            sourceFilePath,  // Pass in the source file path.
                            FileMode.Open,   // Open an existing file.
                            FileAccess.Read);// Read an existing file.
BinaryReader reader = new BinaryReader(sourceFile);
bool boolValue = reader.ReadBoolean();
byte byteValue = reader.ReadByte();
byte[] byteArrayValue = reader.ReadBytes(4);
char charValue = reader.ReadChar();
char[] charArrayValue = reader.ReadChars(4);
decimal decimalValue = reader.ReadDecimal();
double doubleValue = reader.ReadDouble();
float floatValue = reader.ReadSingle();
int intValue = reader.ReadInt32();
long longValue = reader.ReadInt64();
sbyte sbyteValue = reader.ReadSByte();
short shortValue = reader.ReadInt16();
string stringValue = reader.ReadString();
uint unintValue = reader.ReadUInt32();
ulong ulongValue = reader.ReadUInt64();
ushort ushortValue = reader.ReadUInt16();
reader.Close();
sourceFile.Close();
```



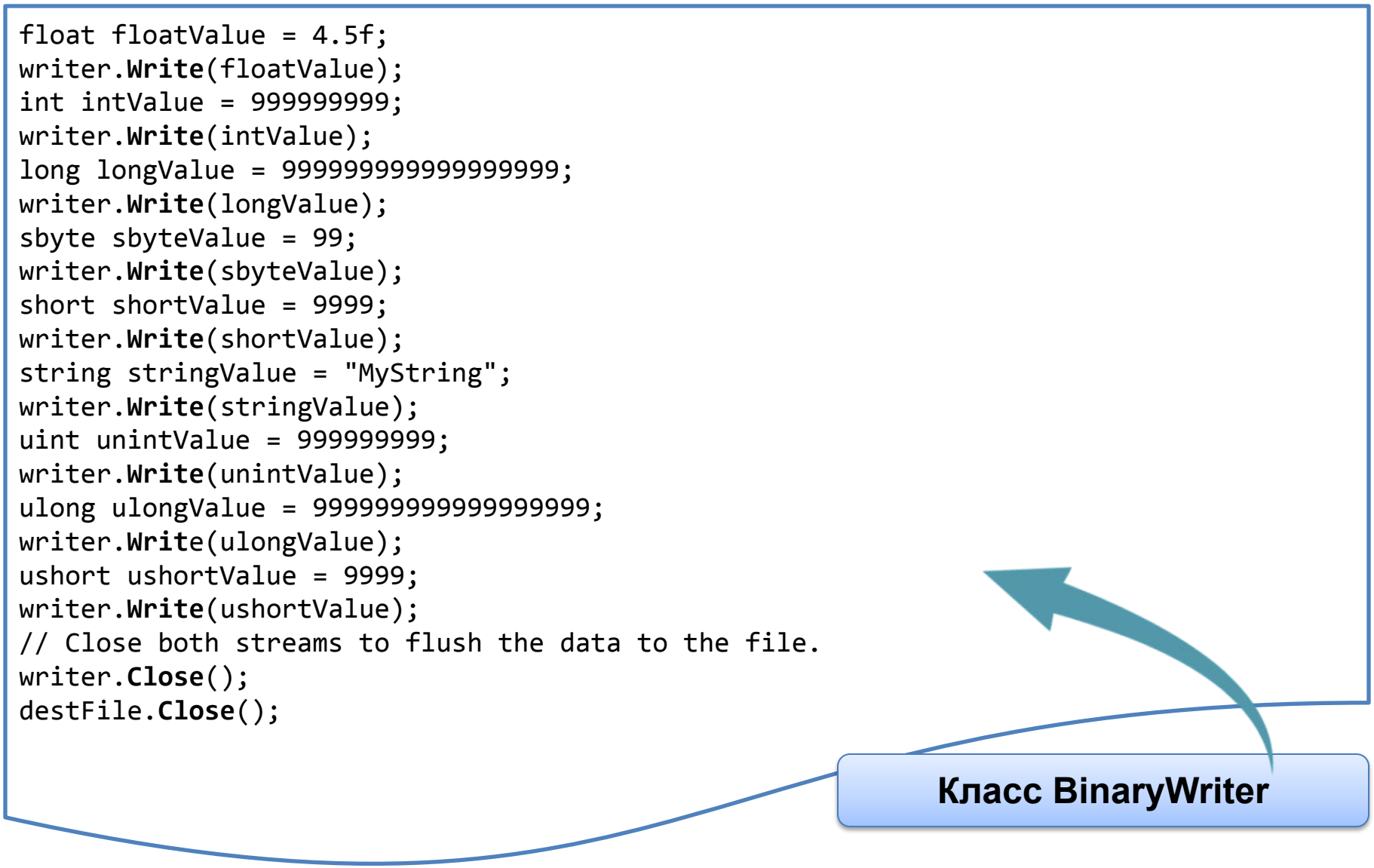
Чтение и запись примитивных типов данных

```
string destinationFilePath = @"C:\. . .\PrimitiveDataTypeFile.txt";
FileStream destFile = new FileStream(
                          destinationFilePath, // Pass in the destination path.
                          FileMode.Create,     // Always create new file.
                          FileAccess.Write);   // Only perform writing.
BinaryWriter writer = new BinaryWriter(destFile);
bool boolValue = true;
writer.Write(boolValue);
byte byteValue = 1;
writer.Write(byteValue);
byte[] byteArrayValue = { 1, 4, 6, 8 };
writer.Write(byteArrayValue);
char charValue = 'a';
writer.Write(charValue);
char[] charArrayValue = { 'a', 'b', 'c', 'd' };
writer.Write(charArrayValue);
decimal decimalValue = 1.00m;
writer.Write(decimalValue);
double doubleValue = 2.5;
writer.Write(doubleValue);
```



Чтение и запись примитивных типов данных

```
float floatValue = 4.5f;
writer.Write(floatValue);
int intValue = 999999999;
writer.Write(intValue);
long longValue = 999999999999999999;
writer.Write(longValue);
sbyte sbyteValue = 99;
writer.Write(sbyteValue);
short shortValue = 9999;
writer.Write(shortValue);
string stringValue = "MyString";
writer.Write(stringValue);
uint unintValue = 999999999;
writer.Write(unintValue);
ulong ulongValue = 999999999999999999;
writer.Write(ulongValue);
ushort ushortValue = 9999;
writer.Write(ushortValue);
// Close both streams to flush the data to the file.
writer.Close();
destFile.Close();
```



Демонстрация: Чтение и запись в файлы

Solution: NET.CSharp.09

Project: FileSystemApplication

Thanks
for Your Attention

EPAM Systems — NET.C#.09 Работа с файловой системой

By
FirstName LastName
Title

EPAM Systems
41 University Drive, Suite 202 | Newtown, PA 18940
p: +1 267 759 9000 | **f:** +1 +1 267 759 8989 | **e:** info@epam.com | **w:** www.epam.com