Streams, Files and Directories

File Types, Using Streams and Manipulating Files

SoftUni Team

Technical Trainers

Software University

http://softuni.bg

Table of Contents

SoftUni Foundation

Have a Question?

SoftUni Foundation

sli.do

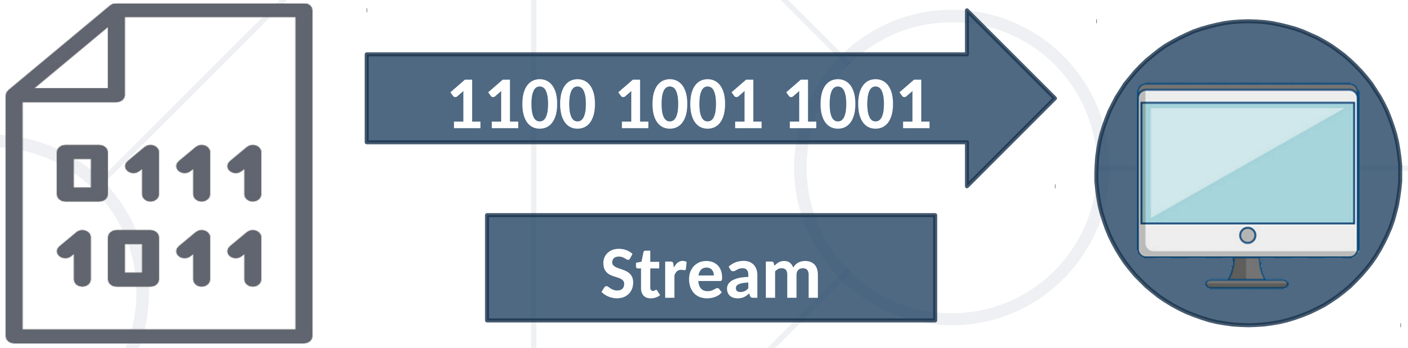
#csharp-advanced

What Is a Stream?

Basic Concepts

What is a Stream?

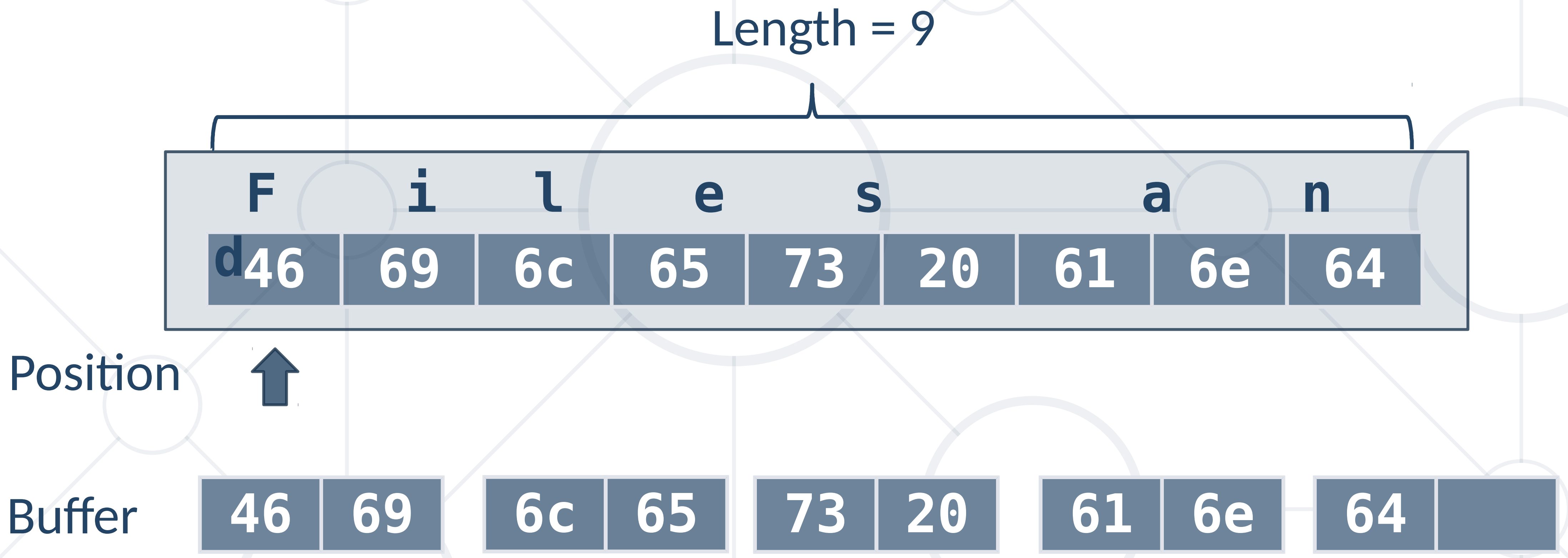
- Streams are used to **transfer data**
- We open a stream to:
  - **Read** data
  - **Write** data



Stream Basics

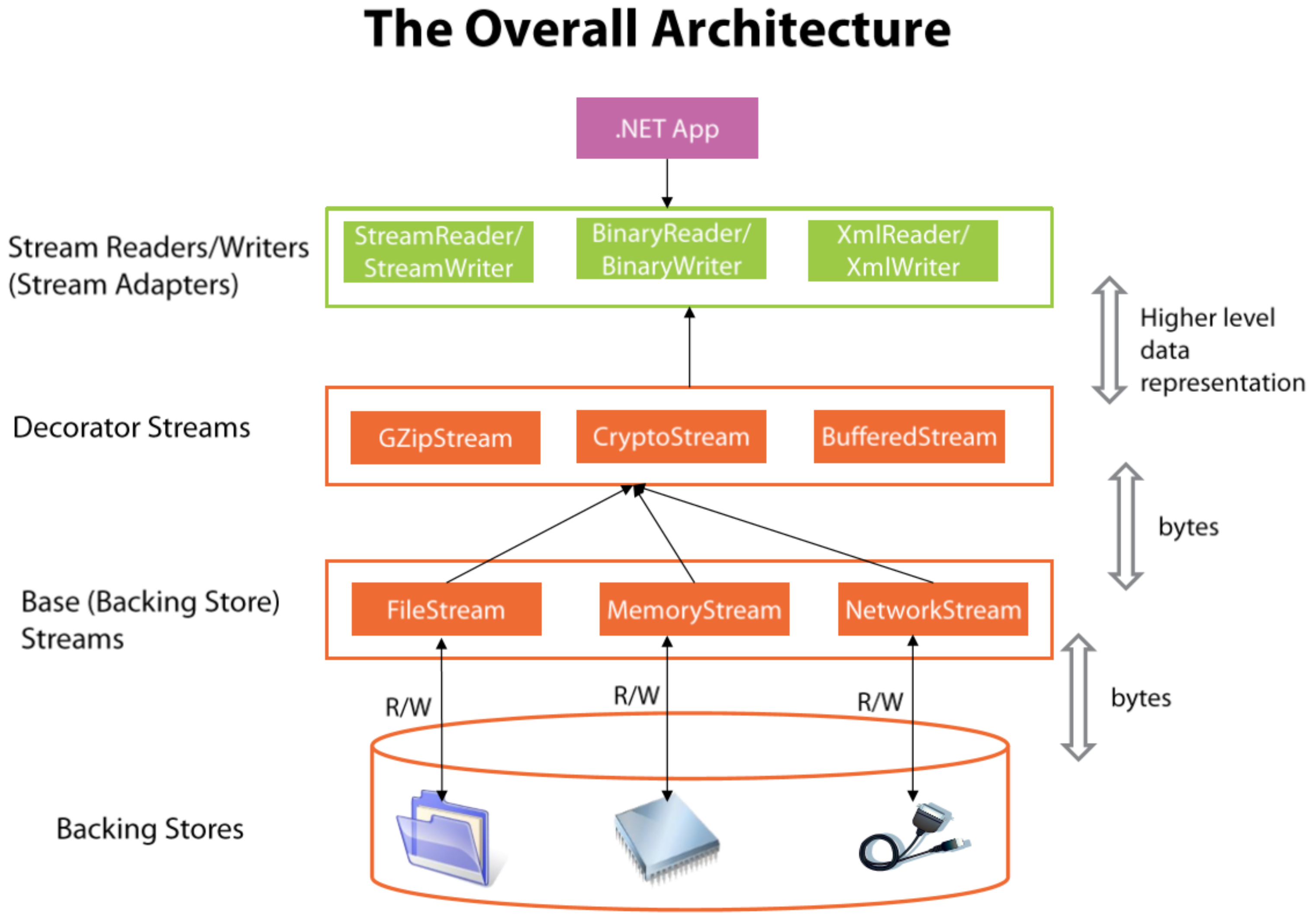
- Streams are means for **transferring** (reading and writing) **data**
- Streams are ordered **sequences of bytes**
  - Provide **sequential** access to its elements
- Different types of streams are available to access different data sources:
  - **File** access, **network** access, **memory** streams and others
- Streams are opened **before** using them and closed **after** that

Stream – Example



- Position is the current position in the stream
- Buffer keeps **n** bytes of the stream from the current position

Stream Types in .NET



Readers and Writers

Text and Binary Readers/Writers

Readers and Writers

SoftUni Foundation

- Readers and writers are **classes**, which facilitate the work with streams
- Two types of streams
  - **Text** readers/writers – `StreamReader` / `StreamWriter`
    - Provide methods `ReadLine()`, `WriteLine()` (similar to working with the system class `Console`)
  - **Binary** readers/writers – `BinaryReader` / `BinaryWriter`
    - Provide methods for working with primitive types
      - `ReadInt32()`, `ReadBoolean()`, `WriteChar()`, etc.

Using statement

- Streams, readers, files, etc. use certain resources
- **Using** statement closes them and releases their resources

```
var reader = new
StreamReader(fileName);
using (reader)
{
  // Use the reader here
}
```

Problem: Odd Lines

- Read the content from your Input.txt file
- Print the odd lines on the console
- Counting starts from 0

```
Two households, both alike in dignity,
In fair Verona, where we lay our scene,
From ancient grudge break to new mutiny,
Where civil blood makes civil hands un-
clean.
```

```
In fair Verona, where we lay our scene,

Where civil blood makes civil hands un-
clean.
```

Solution: Odd Lines

```
var reader = new StreamReader("Input.txt");
using (reader){
int counter = 0;
string line = reader.ReadLine();
using (var writer = new StreamWriter("Output.txt")){
    while (line != null){
        if (counter % 2 == 1){
            writer.WriteLine(line);}
    counter++;
    line = reader.ReadLine();}}}
```

Problem: Line Numbers

- Read your Input.txt file
- Insert a number in front of each line of the file
- Save it in Output.txt

```
Two households, both alike in dignity,
In fair Verona, where we lay our scene,
From ancient grudge break to new mutiny,
Where civil blood makes civil hands un-
clean.
```

```
1. Two households, both alike in dignity,
2. In fair Verona, where we lay our scene,
3. From ancient grudge break to new mutiny,
4. Where civil blood makes civil hands un-
clean.
```

Solution: Line Numbers

```
using (var reader = new StreamReader("Input.txt"))
{
    string line = reader.ReadLine();
    int counter = 1;
    using (var writer = new
StreamWriter("Output.txt"))
    {
        while (line != null)
        {
            writer.WriteLine($"{counter}. {line}");
            line = reader.ReadLine();
            counter++;
        }
    }
}
```

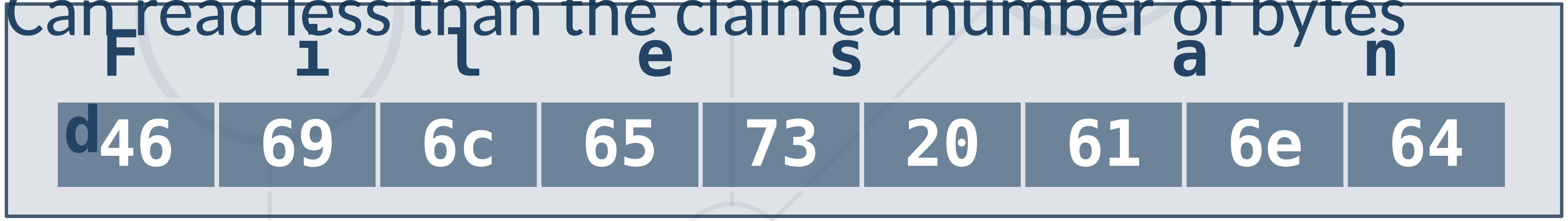
Base Streams

The System.IO.Stream Class

- The base class for all streams is `System.IO.Stream`
- There are defined methods for the main operations with streams in it
- Some streams do not support read, write or positioning operations
  - Properties `CanRead`, `CanWrite` and `CanSeek` are provided
  - Streams which support positioning have the properties `Position` and `Length`

Methods of System.IO.Stream Class

- `int Read(byte[] buffer, int offset, int count)`
  - Read as many as `count` bytes from input stream, starting from the given `offset` position
  - Returns the number of read bytes or 0 if end of stream is reached
  - Can freeze for undefined time while reading at least 1 byte
  - Can read less than the claimed number of bytes



F	i	l	e	s		a	n	d
46	69	6c	65	73	20	61	6e	64

Methods of System.IO.Stream Class (2)

- `Write(byte[] buffer, int offset, int count)`
  - Writes a sequence of `count` bytes to an output stream, starting from the given `offset` position
  - Can freeze for undefined time, until it sends all bytes to their destination
- `Flush()`
  - Sends the internal buffers data to its destination (data storage, I/O device, etc.)

Methods of System.IO.Stream Class (3)

- `Close()`
  - Calls `Flush()`
  - Closes the connection to the device (mechanism)
  - Releases the used resources
- `Seek(offset, SeekOrigin)`
  - Moves the position (if supported) with given offset towards the beginning, the end or the current position

File Stream

The FileStream Class

- Inherits the `Stream` class and supports all its methods and properties
  - Supports reading, writing, positioning operations, etc.
- The constructor contains parameters for:
  - File name
  - File open mode
  - File access mode
  - Concurrent users access mode

The FileStream Class (2)

```
var fs = new FileStream(string fileName, FileMode,
               [FileAccess], [FileShare]);
```

Optional parameters

- **FileMode** – opening file mode
  - **Open**, **Append**, **Create**, **CreateNew**, **OpenOrCreate**, **Truncate**
- **FileAccess** – operations mode for the file
  - **Read**, **Write**, **ReadWrite**
- **FileShare** – access rules for other users while file is opened
  - **None**, **Read**, **Write**, **ReadWrite**

Writing Text to File – Example

```
string text = "Кирилица";
var fileStream = new FileStream("../../log.txt",

FileMode.Create);
try {
  byte[] bytes = Encoding.UTF8.GetBytes(text);
  fileStream.Write(bytes, 0, bytes.Length);
}
finally
{ fileStream.Close(); }
```

try-finally guarantees the stream will always close

Encoding.UTF8.GetBytes() returns the underlying bytes of the character

Problem: Slice File

SoftUni Foundation

- Slice the file sliceMe.txt in 4 equal parts
- Name them
  - Part-1.txt
  - Part-2.txt
  - Part-3.txt
  - Part-4.txt

Solution: Slice File

```
string destinationDirectory = ""; // Add saving path
string sourceFile = ""; // Add file path
int parts = 4;
List<string> files = new List<string> { "Part-1.txt", "Part-
     2.txt ", "Part-3.txt ", "Part-4.txt" };
using (var streamReadFile = new FileStream(sourceFile,
FileMode.Open)){
    long pieceSize =
        (long)Math.Ceiling((double)streamReadFile.Length /

         parts);
    for (int i = 0; i < parts; i++){
        long currentPieceSize = 0;
        // Continues to next slide
    }}
```

Solution: Slice File(2)

```
using (var streamCreateFile = new
FileStream(files[i], FileMode.Create)){
    byte[] buffer = new byte[4096];
    while ((streamReadFile.Read(buffer, 0,
        buffer.Length)) == buffer.Length){
        currentPieceSize += buffer.Length;

        streamCreateFile.Write(buffer, 0,
            buffer.Length);
        if (currentPieceSize >= pieceSize){break;}
    }
}
```

File Class in .NET

.NET API for Easily Working with Files

Reading Text Files

- **File.ReadAllText()** **string** – reads a text file at once

```
using System.IO;
…
string text = File.ReadAllText("file.txt");
```

- **File.ReadAllLines()** **string[]** – reads a text file's lines

```
using System.IO;
…
string[] lines = File.ReadAllLines("file.txt");
```

Writing Text Files

- Writing a **string** to a text file:

```
File.WriteAllText("output.txt", "Files are fun :)");
```

- Writing a **sequence** of strings to a text file, at separate lines:

```
string[] names = { "peter", "irina", "george", "maria" };
File.WriteAllLines("output.txt", names);
```

- **Appending** additional text to an existing file:

```
File.AppendAllText("output.txt", "\nMore text\n");
```

Directory Class in .NET

.NET API for Working with Directories

Basic Directory Operations

- **Creating** a directory (with all its subdirectories at the specified path), unless they already exists:

```
Directory.CreateDirectory("TestFolder
");
```

- **Deleting** a directory (with its contents):

```
Directory.Delete("TestFolder", true);
```

- **Moving** a file or a directory to a new location:

```
Directory.Move("Test", "New Folder");
```

Listing Directory Contents

- **GetFiles()** – returns the names of the files (including their paths) in the specified directory

```
string[] filesInDir =
  Directory.GetFiles("TestFolder");
```

- **GetDirectories()** – returns the names of the subdirecto-ries
(including their paths) in the specified directory

```
string[] subDirs =

Directory.GetDirectories("TestFolder");
```

Problem: Calculate Folder Size

- You are given a folder named **TestFolder**
- Calculate the size of all files in the folder (without subfolders)
- Print the result in a file "output.txt" in Megabytes

```
output.txt

5.161738395690
92
```

Solution: Calculate Folder Size

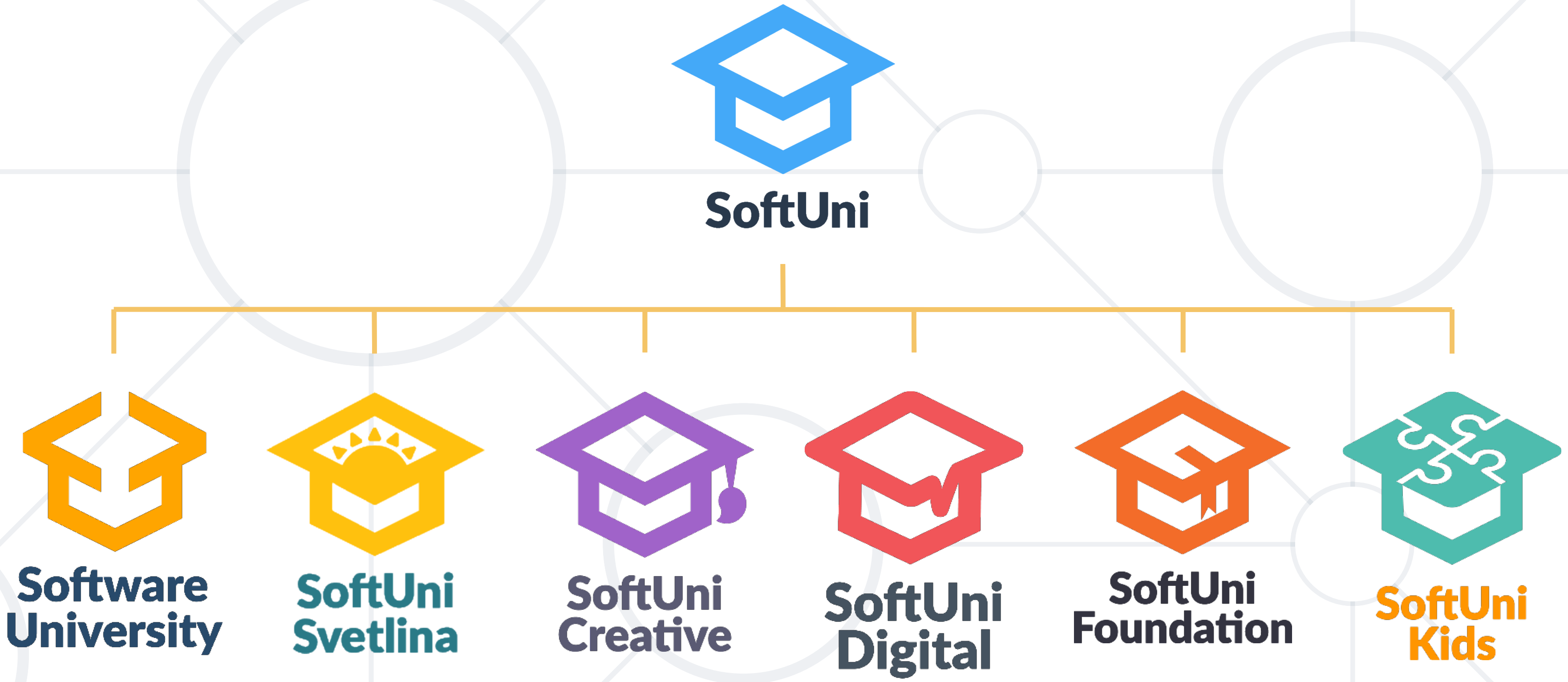
```
string[] files =
Directory.GetFiles("TestFolder");
double sum = 0;
foreach (string file in files)
{
  FileInfo fileInfo = new FileInfo(file);
  sum += fileInfo.Length;
}
sum = sum / 1024 / 1024;
File.WriteAllText("output.txt",
sum.ToString());
```

Live Exercises

Summary

- Streams are ordered sequences of bytes
  - Serve as I/O mechanisms
  - Can be **read** or **written** to (or both)
  - Always close streams by using `try-finally` or `using(…)`
- Use the **File** class to work with files
- Use the **Directory** class to work with directories

Questions?


SoftUni
Software University
SoftUni Svetlina
SoftUni Creative
SoftUni Digital
SoftUni Foundation
SoftUni Kids

SoftUni Diamond Partners

SoftUni Foundation

XSsoftware

SBTech

telenor

SoftwareGroup
doing it right

NETPEAK

SmartIT

Postbank
Решения за твоето утре

SUPER HOSTING .BG

INDEAVR
Serving the high achievers

INFRAGISTICS

STEMO
Computer Systems & Software

SoftUni Organizational Partners

SoftUni Foundation

IS ИНФОРМАЦИОННО ОБСЛУЖВАНЕ

OneBit SOFTWARE

Lukanet.com

WORLD OF MYTHS

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education and Employment Opportunities
  - softuni.bg
- Software University Foundation
  - http://softuni.foundation/
- Software University @ Facebook
  - facebook.com/SoftwareUniversity
- Software University Forums
  - forum.softuni.bg

License

SoftUni Foundation