|  |
| --- |
| SEW |
| Read & Write |
| StreamReader/W., FileStream, BinaryReader/W., BinaryFormatter |

|  |
| --- |
| Felix  27.11.2021  3h |

# Inhaltsverzeichnis

Inhalt

[Inhaltsverzeichnis 1](#_Toc89434382)

[Angabe 1 << StreamReader & Writer – Ten Numbers >> 1](#_Toc89434383)

[Theorie 1 << StreamReader & Writer – Ten Numbers >> 1](#_Toc89434384)

[Quellcode 1 << StreamReader & Writer – Ten Numbers >> 2](#_Toc89434385)

[Angabe 2 << FileStream - Simultanes Lesen & Schreiben >> 2](#_Toc89434386)

[Theorie 2 << FileStream - Simultanes Lesen & Schreiben >> 2](#_Toc89434387)

[Quellcode 2 << FileStream - Simultanes Lesen & Schreiben >> 3](#_Toc89434388)

[Angabe 3 << SR & SW with FileStream - Using FileStream >> 4](#_Toc89434389)

[Theorie 3 << SR & SW with FileStream - Using FileStream >> 4](#_Toc89434390)

[Quellcode 3 << SR & SW with FileStream - Using FileStream >> 4](#_Toc89434391)

[Angabe 4 << Using-Block with SR & SW - 1x1 Tabelle >> 5](#_Toc89434392)

[Theorie 4 << Using-Block with SR & SW - 1x1 Tabelle >> 5](#_Toc89434393)

[Quellcode 4 << Using-Block with SR & SW - 1x1 Tabelle >> 5](#_Toc89434394)

[Angabe 5 << BinaryReader & -Writer >> 6](#_Toc89434395)

[Theorie 5 << BinaryReader & -Writer >> 6](#_Toc89434396)

[Quellcode 5 << BinaryReader & -Writer >> 6](#_Toc89434397)

[Angabe 6 << BinaryFormatter >> 7](#_Toc89434398)

[Theorie 6 << BinaryFormatter >> 7](#_Toc89434399)

[Quellcode 6 << BinaryFormatter >> 7](#_Toc89434400)

# Angabe 1 << StreamReader & Writer – Ten Numbers >>

**READ & WRITE**

Löse die Aufgabenstellung vom Foliensatz Stream.pfd ab Folie 54. Erstelle eine Projektmappe mit mehreren Projekten darin. Löse je Projekt eine der unten angegebenen Aufgaben.

* StreamReader & Writer – Ten Numbers

Erstelle eine kurze Zusammenfassung wann macht welche Klasse Sinn.

Welche Bedeutung haben Streams und ganz konkret Filestreams.

Erstelle ein Resolutionfile und gib dieses sowie die gezippte Projektmappen im Teams ab.

# Theorie 1 << StreamReader & Writer – Ten Numbers >>

Mit dem StreamWriter und StreamReader kann man ganz leicht Daten in Dateien reinschreiben bzw. auslesen.

# Quellcode 1 << StreamReader & Writer – Ten Numbers >>

static void Write(string text, StreamWriter sw)

{

try

{

sw.WriteLine(text);

}

catch (Exception e)

{

Console.WriteLine(e.Message);

}

finally

{

sw.Close();

}

}

static string Read(StreamReader sr)

{

string text = "";

try

{

while (sr.Peek() != -1)

{

text += sr.ReadLine();

}

}

catch (Exception e)

{

Console.WriteLine(e.Message);

}

finally

{

sr.Close();

}

return text;

}

Write("1, 2, 3, 4, 5, 6, 7, 8, 9, 10", new StreamWriter("text.txt", false));

Console.WriteLine(Read(new StreamReader("text.txt")));

# Angabe 2 << FileStream - Simultanes Lesen & Schreiben >>

**READ & WRITE**

Löse die Aufgabenstellung vom Foliensatz Stream.pfd ab Folie 54. Erstelle eine Projektmappe mit mehreren Projekten darin. Löse je Projekt eine der unten angegebenen Aufgaben.

* FileStream - Simultanes Lesen & Schreiben

Erstelle eine kurze Zusammenfassung wann macht welche Klasse Sinn.

Welche Bedeutung haben Streams und ganz konkret Filestreams.

Erstelle ein Resolutionfile und gib dieses sowie die gezippte Projektmappen im Teams ab.

# Theorie 2 << FileStream - Simultanes Lesen & Schreiben >>

Wir haben zwei FileStreams, das eine liest, während das andere schreibt.

# Quellcode 2 << FileStream - Simultanes Lesen & Schreiben >>

FileStream fs\_w = new FileStream("file.txt", FileMode.Truncate, FileAccess.Write, FileShare.Read);

FileStream fs\_r = new FileStream("file.txt", FileMode.Open, FileAccess.Read, FileShare.Write);

StreamWriter sw = new StreamWriter(fs\_w);

StreamReader sr = new StreamReader(fs\_r);

// moin in file.txt schreiben und flushen

sw.WriteLine("moin");

sw.Flush();

// Inhalt auslesen

Console.WriteLine(sr.ReadToEnd());

// moin2 in file.txt schreiben und flushen

sw.WriteLine("zweites moin");

sw.Flush();

// Inhalt auslesen

Console.WriteLine(sr.ReadToEnd());

// alles Closen

sw.Close();

sw.Close();

# Angabe 3 << SR & SW with FileStream - Using FileStream >>

**READ & WRITE**

Löse die Aufgabenstellung vom Foliensatz Stream.pfd ab Folie 54. Erstelle eine Projektmappe mit mehreren Projekten darin. Löse je Projekt eine der unten angegebenen Aufgaben.

* SR & SW with FileStream - Using FileStream

Erstelle eine kurze Zusammenfassung wann macht welche Klasse Sinn.

Welche Bedeutung haben Streams und ganz konkret Filestreams.

Erstelle ein Resolutionfile und gib dieses sowie die gezippte Projektmappen im Teams ab.

# Theorie 3 << SR & SW with FileStream - Using FileStream >>

Wir haben einen FileStream, der zuerst schreibt und dann liest.

# Quellcode 3 << SR & SW with FileStream - Using FileStream >>

FileStream fs = new FileStream("file.txt", FileMode.OpenOrCreate, FileAccess.ReadWrite);

StreamWriter sw = new StreamWriter(fs);

StreamReader sr = new StreamReader(fs);

// moin in file.txt schreiben und flushen

sw.WriteLine("moin");

sw.Flush();

// Zeiger wieder auf Begin der Datei setzen

fs.Seek(0, SeekOrigin.Begin);

// Inhalt auslesen

Console.WriteLine(sr.ReadToEnd());

// FileStream schließen

fs.Close();

# Angabe 4 << Using-Block with SR & SW - 1x1 Tabelle >>

**READ & WRITE**

Löse die Aufgabenstellung vom Foliensatz Stream.pfd ab Folie 54. Erstelle eine Projektmappe mit mehreren Projekten darin. Löse je Projekt eine der unten angegebenen Aufgaben.

* Using-Block with SR & SW - 1x1 Tabelle

Erstelle eine kurze Zusammenfassung wann macht welche Klasse Sinn.

Welche Bedeutung haben Streams und ganz konkret Filestreams.

Erstelle ein Resolutionfile und gib dieses sowie die gezippte Projektmappen im Teams ab.

# Theorie 4 << Using-Block with SR & SW - 1x1 Tabelle >>

Wir verwenden einen Using-Block, damit wir kein sw.Close() machen müssen.

# Quellcode 4 << Using-Block with SR & SW - 1x1 Tabelle >>

static void WriteTabelle(string filename)

{

using (StreamWriter sw = new StreamWriter(filename))

{

for (int i = 1; i <= 10; i++)

{

for (int j = 1; j <= 10; j++)

{

sw.Write(i + "x");

sw.Write(j + "=");

sw.WriteLine(i \* j);

}

sw.WriteLine("=========");

}

}

}

static void ReadFile(string filename)

{

using(StreamReader sr = new StreamReader(filename))

{

while(sr.Peek() != -1)

{

Console.WriteLine(sr.ReadLine());

}

}

}

WriteTabelle("tabelle.txt");

ReadFile("tabelle.txt");

# Angabe 5 << BinaryReader & -Writer >>

**READ & WRITE**

Löse die Aufgabenstellung vom Foliensatz Stream.pfd ab Folie 54. Erstelle eine Projektmappe mit mehreren Projekten darin. Löse je Projekt eine der unten angegebenen Aufgaben.

* BinaryReader & -Writer

Erstelle eine kurze Zusammenfassung wann macht welche Klasse Sinn.

Welche Bedeutung haben Streams und ganz konkret Filestreams.

Erstelle ein Resolutionfile und gib dieses sowie die gezippte Projektmappen im Teams ab.

# Theorie 5 << BinaryReader & -Writer >>

Der BinaryReader bzw. BinaryWriter speichert Daten binär.

# Quellcode 5 << BinaryReader & -Writer >>

static void BinWrite(string filename)

{

FileStream fs = new FileStream(filename, FileMode.Create);

BinaryWriter bw = new BinaryWriter(fs);

bw.Write(187);

bw.Write(3.14159265358767);

bw.Write("Felix");

bw.Close();

}

static void BinRead(string filename)

{

FileStream fs = new FileStream(filename, FileMode.Open, FileAccess.Read);

BinaryReader br = new BinaryReader(fs);

Console.WriteLine(br.ReadInt32() + "\n" + br.ReadDouble() + "\n" + br.ReadString());

br.Close();

}

BinWrite("data.bin");

BinRead("data.bin");

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

# Angabe 6 << BinaryFormatter >>

**READ & WRITE**

Löse die Aufgabenstellung vom Foliensatz Stream.pfd ab Folie 54. Erstelle eine Projektmappe mit mehreren Projekten darin. Löse je Projekt eine der unten angegebenen Aufgaben.

* BinaryFormatter

Erstelle eine kurze Zusammenfassung wann macht welche Klasse Sinn.

Welche Bedeutung haben Streams und ganz konkret Filestreams.

Erstelle ein Resolutionfile und gib dieses sowie die gezippte Projektmappen im Teams ab.

# Theorie 6 << BinaryFormatter >>

Mit dem Binary Formatter kann Klassen super binär in Dateien speichern.

# Quellcode 6 << BinaryFormatter >>

using BinaryFormatter\_Person;

using System.Runtime.Serialization.Formatters.Binary;

static void BinWrite(string name, int age)

{

Person person = new Person();

person.Name = name;

person.Age = age;

BinaryFormatter formatter = new BinaryFormatter();

Stream stream = new FileStream("binaryformatter.txt", FileMode.Create, FileAccess.Write, FileShare.None);

formatter.Serialize(stream, person);

stream.Close();

}

static void BinRead()

{

BinaryFormatter formatter = new BinaryFormatter();

Stream stream = new FileStream("binaryformatter.txt", FileMode.Open, FileAccess.Read, FileShare.Read);

Person person = (Person)formatter.Deserialize(stream);

stream.Close();

}

BinWrite("Felix", 16);

BinRead();

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace BinaryFormatter\_Person

{

[Serializable]

public class Person

{

public string? Name { get; set; }

public int Age { get; set; }

}

}