Входно-изходни потоци

Учителски екип

Обучение за ИТ кариера

https://it-kariera.mon.bg/e-learning/

Съдържание

1. Какво са потоците?
2. Reader и Writer класове
3. Типове потоци
   - File, Memory, Network потоци
   - Crypto, Gzip потоци

Какво е поток?

- Потоците се използват за пренос на информация
- Ние отваряме поток за да:
  - Прочетем данни
  - Запишем данни

Основното за потоците

- Потоците са създадени за пренос (четене и запис) на данни
- Потоците са подредена последователност от байтове
  - Осигуряват последователен достъп до своите елементи
- Има различни типове потоци за разните типове данни:
  - Достъп до файлове и мрежа, потоци в паметта и други
- Потоците трябва да се отворят преди употреба и да се затворят накрая

Поток – пример

- Position е текущата позиция в потока
- Buffer пази **n** байта от потока от текущата позиция

Reader и Writer класове

- Reader и writer са класове, улесняващи работата с потоците
- Има два типа потоци:
  - Текстово четене/запис – **`StreamReader`** / **`StreamWriter`**
    - Имат методи **`.ReadLine()`**, **`.WriteLine()`** (подобно на работата с **`Console.*`**)
  - Двоично четене/запис – **`BinaryReader`** / **`BinaryWriter`**
    - Имат методи за работа с примитивни типове – **`.ReadInt32()`**, **`.ReadBoolean()`**, **`WriteChar()`** и т.н.

Задача: Четене на файл

- Прочетете цялото съдържание на Program.cs файла
- Отпечатайте го на конзолата с номера на редове

```
using System;
using System.IO;

class Program
{
```

```
Line 1: using System;
Line 2: using System.IO;
Line 3:
Line 4: class Program
Line 5: {
```

Решение: Четене на файл

```
StreamReader reader = new StreamReader("somefile.txt");
using (reader)
{
  int lineNumber = 0;
  string line = reader.ReadLine();
  while (line != null)
  {
    lineNumber++;
    Console.WriteLine("Line {0}: {1}", lineNumber, line);
    line = reader.ReadLine();
  }
}
```

Задача: Запис във файл

- Прочетете вашия **`Program.cs`** файл
- Обърнете на обратно всеки негов ред
- Запишете резултата в **`reversed.txt`**

```
using System.IO;

class Program
{
```

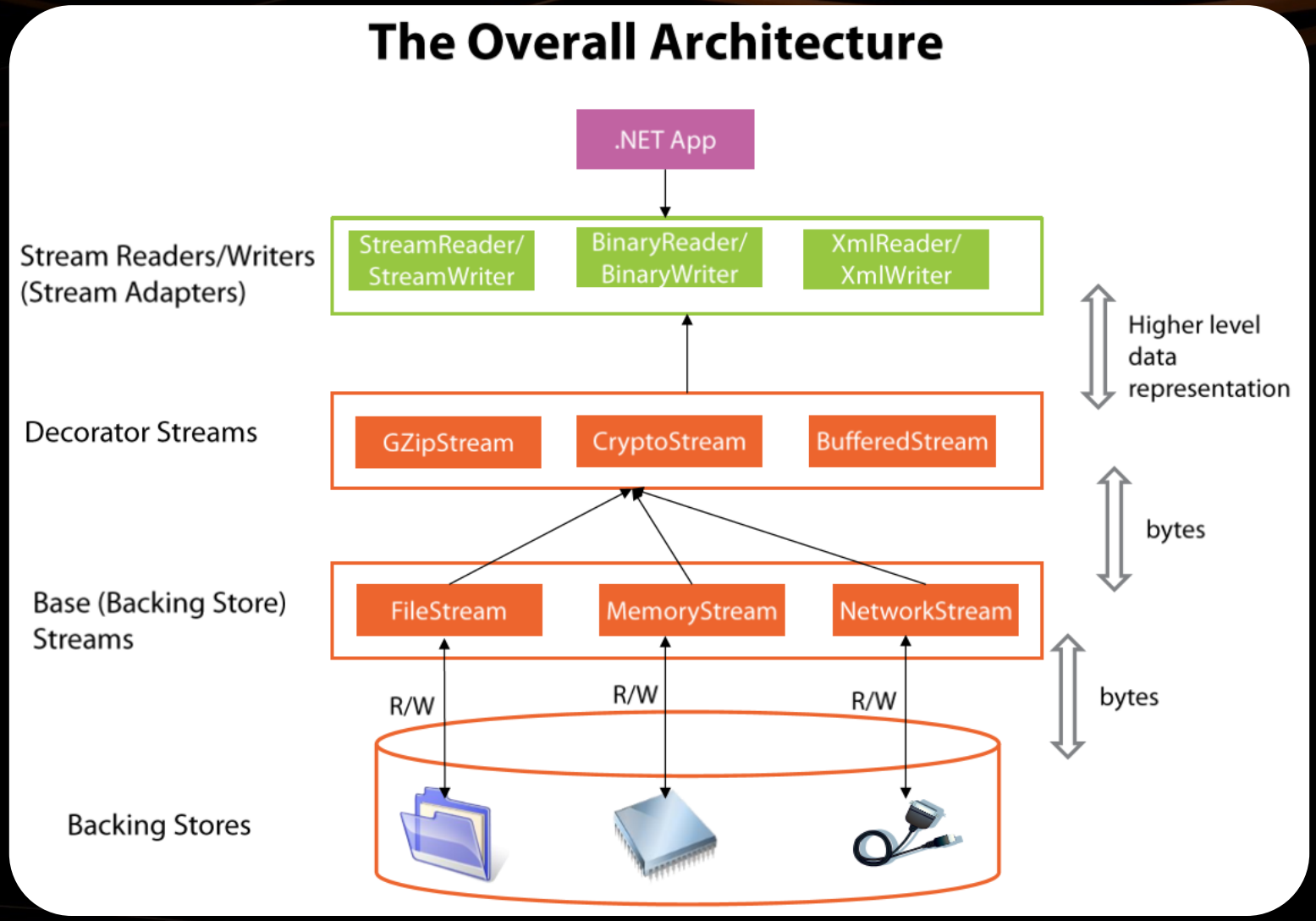
```
;OI.metsyS gnisu

margorP ssalc
{
```

Запис на текста на файл наобратно – пример

```
using (var reader = new StreamReader("../../Program.cs"))
{
  using (var writer = new StreamWriter("../../reversed.txt"))
  {
    string line = reader.ReadLine();
    while (line != null)
    {
      for (int i = line.Length - 1; i >= 0; i--)
      {
        writer.Write(line[i]);
      }
      writer.WriteLine();
      line = reader.ReadLine();
    }
  }
}
```

Типове потоци в .NET

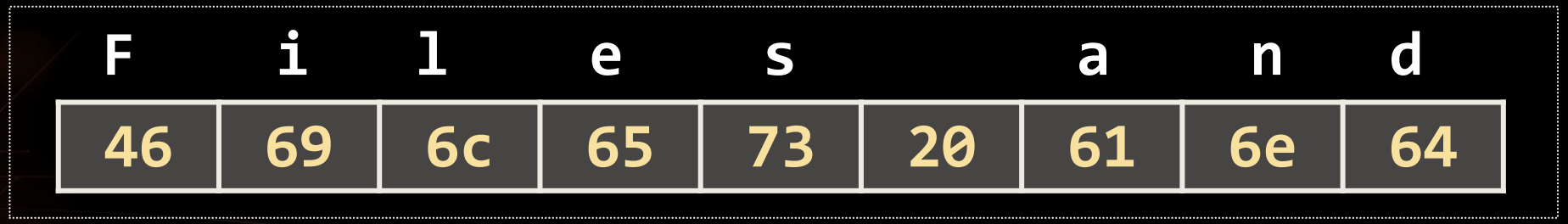


Класът System.IO.Stream

- Базовият клас за всички потоци е **System.IO.Stream**
- Той има дефинирани методи за основните операции с потоци
- Някои потоци не поддържат четене, запис или позициониране
  - Затова има свойства **CanRead**, **CanWrite** и **CanSeek**
  - Потоците, които поддържат позициониране имат свойства **Position** и **Length**

Методи на класа `System.IO.Stream`

- `int Read(byte[] buffer, int offset, int count)`
  - Чете `count` байта от входящия поток, започвайки от дадена `offset` позиция
  - Връща броя прочетени байтове или 0 ако е достигнат края
  - Може да замръзне за неопределено време докато прочете поне 1 байт
  - Може да прочете по-малко от обявения брой байтове



Методи на класа `System.IO.Stream` (2)

- `Write(byte[] buffer, int offset, int count)`
  - Записва поредица от `count` байта в изходящ поток, започвайки от дадена `offset` позиция
  - Може да замръзне за неопределено време, докато изпрати всички байтове по назначение
- `Flush()`
  - Изпраща вътрешно буферираните данни към тяхното назначение (устройство за съхранение на данни, за вход/изход или друго)

Методи на класа System.IO.Stream (3)

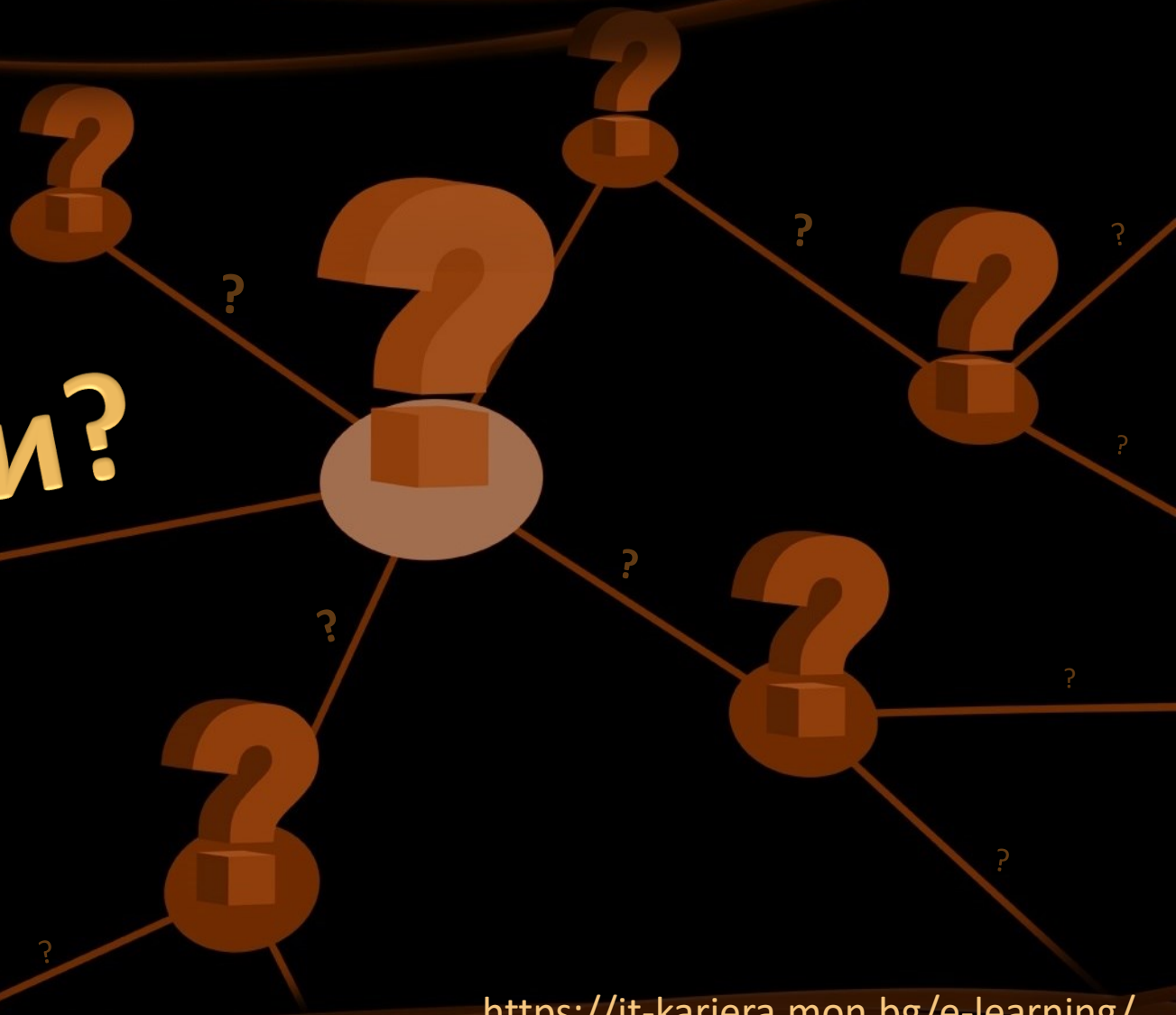
- `Close()`
  - Извиква `Flush()`
  - Прекъсва връзката към устройството
  - Освобождава заетите ресурси
- `Seek(offset, SeekOrigin)` – премества позицията (ако това се поддържа като операция) с определено отместване спрямо началото, края или текущата позиция

Обобщение

- Потоците са подредени поредици от байтове
  - Използват се за входно/изходни механизми
  - Могат да са за четене, за запис или и за двете
  - Може да са с различна природа – файл, мрежа, памет, устройство и т.н.
- Reader и writer класовете улесняват работата с потоците чрез предоставяне на допълнителна функционалност (например четене на цели редове)
- Винаги затваряйте потоците чрез `using(…)` или `try-finally`

Входно-изходни потоци

Въпроси?



Министерство на образованието и науката (МОН)

- Настоящият курс (презентации, примери, задачи, упражнения и др.) е разработен за нуждите на Национална програма "**Обучение за ИТ кариера**" на МОН за подготовка по професия "Приложен програмист"

- Курсът е базиран на учебно съдържание и методика, предоставени от **фондация "Софтуерен университет"** и се разпространява под свободен лиценз **CC-BY-NC-SA**