Exercise: Streams, Files and Directories

Problems for the "C# Advanced" course @ SoftUni
You can check your solutions in Judge

NOTE: For these problems follow the instructions for the required methods and classes. For each problem **submit zipped folder** of your project **without** the **"bin"** and **"obj"** folders in it.

1. Even Lines

Write a program that reads a **text** file (e. g. **text.txt**) and prints on the console its **even lines**. Line numbers start from 0. Use **StreamReader**. Before you print the result, replace **{'-', ', ', '. ', '! ', '? '}** with **'@'** and **reverse the order of the words**.

Note: use the following structure:

```
namespace EvenLines
{
    using System;

    public class EvenLines
    {
        static void Main()
        {
            string inputFilePath = @"..\..\..\text.txt";

            Console.WriteLine(ProcessLines(inputFilePath));
        }

        public static string ProcessLines(string inputFilePath)
        {
        }
    }
}
```

Examples

Input file: text.txt	Output (at the console)
-I was quick to judge him, but it wasn't his fault. -Is this some kind of joke?! Is it? -Quick, hide here. It is safer.	fault@ his wasn't it but him@ judge to quick was @I safer@ is It here@ hide @Quick@

2. Line Numbers

Write a program that **reads** a **text file** (e. g. **text.txt**) and inserts **line numbers** in front of **each** of its **lines and count all the letters and punctuation marks**. The result should be **written** to **another** text file (e. g. **output.txt**). Use the static class **File** to read and write all the lines of the input and output files.

Note: use the following structure:

```
public class LineNumbers
{
    public static void ProcessLines(string inputFilePath, string outputFilePath)
    {
    }
}
```

Examples

text.txt	output.txt
-I was quick to judge him, but it wasn't his fault. -Is this some kind of joke?! Is it? -Quick, hide here. It is safer.	Line 1: -I was quick to judge him, but it wasn't his fault. (37)(4) Line 2: -Is this some kind of joke?! Is it? (24)(4) Line 3: -Quick, hide here. It is safer. (22)(4)

3. Copy Binary File

Write a program that copies the contents of a binary file (e. g. **copyMe.png**) to another binary file (e. g. **copyMe-copy.png**) using **FileStream**. You are **not allowed** to use the **File** class or similar helper classes.

Note: use the following structure:

```
namespace CopyBinaryFile
{
    using System;

    public class CopyBinaryFile
    {
        static void Main()
        {
            string inputFilePath = @"..\..\..\copyMe.png";
            string outputFilePath = @"..\..\..\copyMe-copy.png";

            CopyFile(inputFilePath, outputFilePath);
        }

        public static void CopyFile(string inputFilePath, string outputFilePath)
        {
        }
    }
}
```

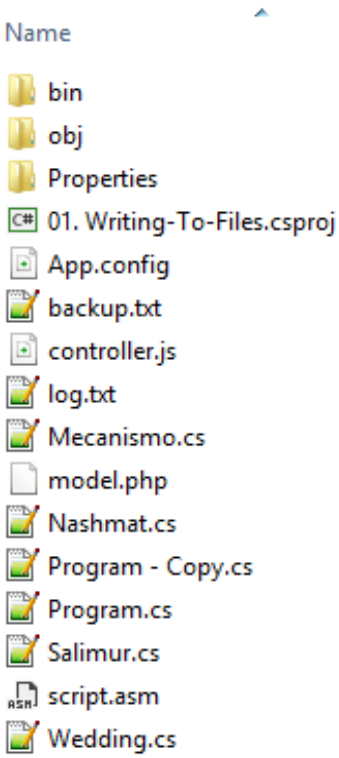
4. Directory Traversal

Write a program that traverses a given **directory** for **all files** with the given **extension**. Search through the **first level** of the **directory only.** Write information about each **found** file in a text file named **report.txt** and it should be saved on the **Desktop**. The files should be **grouped** by their **extension**. **Extensions** should be **ordered** by the **count** of their files **descending**, then by **name alphabetically**. **Files** under an extension should be **ordered** by their **size**. **report.txt** should be saved on the **Desktop**. Ensure the desktop path is always valid, regardless of the user.

Note: use the following structure:

```
namespace DirectoryTraversal
{
    using System;

    public class DirectoryTraversal
    {
        static void Main()
        {
            string path = Console.ReadLine();
            string reportFileName = @"\report.txt";

            string reportContent = TraverseDirectory(path);
            Console.WriteLine(reportContent);

            WriteReportToDesktop(reportContent, reportFileName);
        }

        public static string TraverseDirectory(string inputFolderPath)
        {
        }

        public static void WriteReportToDesktop(string textContent, string
reportFileName)
        {
        }
    }
}
```

Examples

Input	Directory View	report.txt
.		.cs --Mecanismo.cs - 0.994kb --Program.cs - 1.108kb --Nashmat.cs - 3.967kb --Wedding.cs - 23.787kb --Program - Copy.cs - 35.679kb --Salimur.cs - 588.657kb .txt --backup.txt - 0.028kb --log.txt - 6.72kb .asm --script.asm - 0.028kb .config --App.config - 0.187kb .csproj --01. Writing-To-Files.csproj - 2.57kb .js --controller.js - 1635.143kb .php --model.php - 0kb

5. Copy Directory

Write a method, which **copies a directory with files** (**without its subdirectories**) to another directory. The input folder and the output folder should be given as parameters from the console. If the output folder already exists, first delete it (together with all its content).

Note: use the following structure:

```
namespace CopyDirectory
{
    using System;

    public class CopyDirectory
    {
        static void Main()
        {
            string inputPath = @$"{Console.ReadLine()}";
            string outputPath = @$"{Console.ReadLine()}";

            CopyAllFiles(inputPath, outputPath);
        }

        public static void CopyAllFiles(string inputPath, string outputPath)
        {
        }
    }
}
```

6. *Zip and Extract

Write a program that **creates** a **ZIP** file (archive), holding a given **input file,** and **extracts** the ZIP-ed file from the archive into in separate **output file**.

- Use the **`copyMe.png`** file from your resources as input and **zip** it into a ZIP file of your choice, e. g. **`archive.zip`**.
- **Extract** the file from the archive into a new file of your choice, e. g. **`extracted.png`**.

If your code works correctly, the input and output files should be the same.

Note: use the following structure:

```
namespace ZipAndExtract
{
    using System;
    using System.IO;

    public class ZipAndExtract
    {
        static void Main()
        {
        }

        public static void ZipFileToArchive(string inputFilePath, string
zipArchiveFilePath)
        {
        }

        public static void ExtractFileFromArchive(string zipArchiveFilePath, string
fileName, string outputFilePath)
```

```
            {
            }
        }
}
```

Hints

- Use the **`ZipFile`** class.
- The **entry** in the ZIP file should hold the **file name only** without its path.