C# File Handeling

Exercise #1

```
using System.IO;
using System.Windows;

namespace FileHandling.Exe
{
    4 references
    public partial class Exe1 : Window
    {
        1 reference
        public Exe1()
        {
            InitializeComponent();

            string fPath = "example.txt";
            string content = "Hello, World!";

            File.WriteAllText(fPath, content);
        }
    }
}
```

Exercise 1: Creating and Writing to a Text File

1. **Objective**: Learn to create a file and write simple text to it.
2. **Steps**:
3. In a new console application, use the `System.IO` namespace.
4. Use `File.WriteAllText` to create a file named "example.txt" and write the text "Hello, World!" to it.
5. Run the program and check that "example.txt" contains the text "Hello, World!".
6. **Expected Outcome**: A file named "example.txt" is created with "Hello, World!" as its content.

example - פנקס רשימות

קובץ עריכה עיצוב תצוגה עזרה

Hello, World

fPath - מאפשר לה לאחסן את נתיב הקובץ.
אם לא ניתן נתיב תיקיה, הוא יבור או יבא לקובץ בספריית הבאגה הנוכחית של הישום.
content - תוכן המחרוזת
File.WriteAllText(fPath, content) - שימוש בספריה System.IO בכדי ליצר או להחליף את example.txt עם התוכן שנתן. אם הקובץ לא קיים הוא יוצר ואם קיים אז התוכן יוחלף בתוכן החדש.

Exercise #2

```
using System.IO;
using System.Runtime.InteropServices;
using System.Windows;


namespace FileHandling.Exe
{
    4 references
    public partial class Exe2 : Window
    {
        [DllImport("kernel32.dll")]
        1 reference
        public static extern bool AllocConsole();
        1 reference
        public Exe2()
        {
            InitializeComponent();
            AllocConsole();

            string fPath = "example.txt";
            try
            {
                string content = File.ReadAllText(fPath);
                Console.WriteLine("Content of example.txt:");
                Console.WriteLine(content);
            }
            catch (FileNotFoundException)
            {
                Console.WriteLine("The file example.txt not found!");
            }
        }
    }
}
```

ייבוא של ה-console

Exercise 2: Reading from a Text File

1. **Objective**: Read and display text from a file.
2. **Steps**:
3. Ensure "example.txt" exists with some content.
4. Use `File.ReadAllText` to read the contents of "example.txt".
5. Print the contents to the console.
6. **Expected Outcome**: The console displays the content of "example.txt".

ל-WPF אין console כברירת מחדל, לכן נשתמש ב-AllocConsole() כדי להקצות console שמו יוצג הפלט.

C:\Users\user\source\repos\FileHandling\FileHandling\bin\Debug\net8.0-windows\FileHandling.exe

```
Content of example.txt:
Hello, World!
```

קריאת הקובץ והצגתו ב-console.

Exercise #3

```
using System.IO;
using System.Runtime.InteropServices;
using System.Windows;


namespace FileHandling.Exe
{
    4 references
    public partial class Exe3 : Window
    {
        [DllImport("kernel32.dll")]
        1 reference
        public static extern bool AllocConsole();
        1 reference
        public Exe3()
        {
            InitializeComponent();
            AllocConsole();

            string[] lines = { "Line1", "Line2", "Line3", "Line4", "Line5" };

            string fPath = "multilines.txt";

            File.WriteAllLines(fPath, lines);

            Console.WriteLine("File multilines.txt created: ");

            foreach (string line in lines)
            {
                Console.WriteLine(line);
            }

            Console.ReadLine();
        }
    }
}
```

Exercise 3: Writing Multiple Lines to a File

1. **Objective**: Write multiple lines to a text file.
2. **Steps**:
3. Create a `string[]` array with several strings (e.g., "Line 1", "Line 2", "Line 3").
4. Use `File.WriteAllLines` to write these lines to a new file called "multilines.txt".
5. Verify the file to ensure each string appears on a new line.
6. **Expected Outcome**: A file named "multilines.txt" is created with each string on a separate line.

יצירת הקובץ multilines.txt
וכתיבה לתוכו מערך של מחרוזות

קריאה של כל שורה מהקובץ
והדפסתו ב- console.

```
C:\Users\user\source\repos\FileHandling\FileHandling\bin\Debug\net8.0-windows\File
File multilines.txt created:
Line1
Line2
Line3
Line4
Line5
```

Exercise #4

```
using System.Runtime.InteropServices;
using System.Windows;
using System.IO;
using System.Xml;


namespace FileHandling.Exe
{
    4 references
    public partial class Exe4 : Window
    {
        [DllImport("kernel32.dll")]
        1 reference
        public static extern bool AllocConsole();
        1 reference
        public Exe4()
        {
            InitializeComponent();
            AllocConsole();

            string fPath = "example.txt";
            string txtAppend = "Additional Line";

            File.AppendAllText(fPath, Environment.NewLine + txtAppend);

            string updatedContent = File.ReadAllText(fPath);
            Console.WriteLine("\nUpdated content of example.txt:");
            Console.WriteLine(updatedContent);

            Console.ReadLine();

        }
    }
}
```

Exercise 4: Appending Text to a File

1. **Objective**: Add text to an existing file without overwriting it.
2. **Steps**:
3. Use `File.AppendAllText` to add the line "Additional Line" to "example.txt".
4. Run the program and check "example.txt" to see the new line appended.
5. **Expected Outcome**: "example.txt" now contains the original text plus "Additional Line" at the end.

הוספה של שורה נוספת
לתוכן המקורי של הקובץ
ע"י File.AppendAllText

קריאת הקובץ המעודכן
והדפסת תוכן הקובץ ל-console

```
C:\Users\user\source\repos\FileHandling\FileHandling\bin\Debug\net8.0-window

Updated content of example.txt:
Hello, World!
Additional Line
```

Exercise #5

```
using System.Runtime.InteropServices;
using System.Windows;
using System.IO;


namespace FileHandling.Exe
{
    4 references
    public partial class Exe5 : Window
    {
        [DllImport("kernel32.dll")]
        1 reference
        public static extern bool AllocConsole();
        1 reference
        public Exe5()
        {
            InitializeComponent();
            AllocConsole();

            string fPath = "example.txt";

            if (File.Exists(fPath))
            {
                string content = File.ReadAllText(fPath);
                Console.WriteLine("Content of example.txt:");
                Console.WriteLine(content);
            }
            else
            {
                Console.WriteLine("File does not exist.");
            }
            Console.ReadLine();
        }
    }
}
```

Exercise 5: Checking if a File Exists

1. **Objective**: Check if a file exists before performing operations.
2. **Steps**:
3. Use `File.Exists` to check if "example.txt" exists.
4. If it does, read and print its content; otherwise, display a message like "File does not exist."
5. **Expected Outcome**: The program either reads and displays "example.txt" or prints a message saying it does not exist.

פעולה בסיסית בטיפול בקבצים כדי להבטיח שהתוכנה לא תנסה לקרוא קובץ לא קיים.

שימוש ב- File.Exists בודק את קיומו של הקובץ אם קיים נציג את התוכן ל- console.

C:\Users\user\source\repos\FileHandling\FileHandling\bin\D

```
Content of example.txt:
Hello, World!
Additional Line
```

Exercise #6

```
using System.Windows;
using System.IO;
using System.Runtime.InteropServices;

namespace FileHandling.Exe
{
    4 references
    public partial class Exe6 : Window
    {
        [DllImport("kernel32.dll")]
        1 reference
        public static extern bool AllocConsole();

        1 reference
        public Exe6()
        {
            InitializeComponent();
            AllocConsole();

            string fPath = "example.txt";

            if (File.Exists(fPath))
            {
                File.Delete(fPath);
                Console.WriteLine("File example.txt was Deleted");
            }
            else
            {
                Console.WriteLine("File does not exist.");
            }
            Console.ReadLine();
        }
    }
}
```

Exercise 6: Deleting a File

1. **Objective**: Safely delete a file.
2. **Steps**:
3. Use `File.Exists` to check if "example.txt" exists.
4. If it does, use `File.Delete` to delete it.
5. Run the program and check that "example.txt" no longer exists.
6. **Expected Outcome**: "example.txt" is deleted if it existed.

C:\Users\user\source\repos\FileHandling\FileHandling\

```
File example.txt was Deleted
```

net8.0-windows › Debug › bin › FileHandling › FileHandling › repos › source › user

שם	תאריך שינוי	סוג	גודל
FileHandling.deps	10/11/2024 19:05	JSON File	1 KB
FileHandling.dll	10/11/2024 22:39	הרחבת יישום	27 KB
FileHandling	10/11/2024 22:39	יישום	136 KB
FileHandling.pdb	10/11/2024 22:39	Program Debug D...	19 KB
FileHandling.runtimeconfig	10/11/2024 19:05	JSON File	1 KB
multilines	10/11/2024 21:45	מסמך טקסט	1 KB

נבדוק האם הקובץ קיים. במידה וקיים נמחק את הקובץ ע"י File.Delete ונציג ל- Console הודעה על אישור מחיקת הקובץ.

Exercise #7

```
using System.Windows;
using System.IO;
using System.Runtime.InteropServices;

namespace FileHandling.Exe
{
    4 references
    public partial class Exe7 : Window
    {
        [DllImport("kernel32.dll")]
        1 reference
        public static extern bool AllocConsole();
        1 reference
        public Exe7()
        {
            InitializeComponent();
            AllocConsole();

            string[] lines = { "Line1", "Line2", "Line3", "Line4", "Line5" };
            string fPath = "sample.txt";
            File.WriteAllLines(fPath, lines);

            if (File.Exists(fPath))
            {
                Console.WriteLine("Contents of sample.txt:");

                foreach (string line in File.ReadLines(fPath))
                {
                    Console.WriteLine(line);
                }
            }
            else
            {
                Console.WriteLine("File does not exist.");
            }
            Console.ReadLine();
        }
    }
}
```

Exercise 7: Reading a File Line by Line

1. **Objective**: Read each line in a file individually.
2. **Steps**:
3. Create a file named "sample.txt" with multiple lines of text.
4. Use File.ReadLines to read the file line by line.
5. Print each line to the console.
6. **Expected Outcome**: Each line from "sample.txt" is printed on a new line in the console.

C:\Users\user\source\repos\FileHandling\FileHandling

```
Contents of sample.txt:
Line1
Line2
Line3
Line4
Line5
```

נבדוק האם הקובץ קיים.
נקרא את הקובץ שהה אחר שורה
ונכתב את כל השורות לקובץ באמצעה
אחת.

Exercise #8

```
using System.Windows;
using System.IO;
using System.Runtime.InteropServices;


namespace FileHandling.Exe
{
    4 references
    public partial class Exe8 : Window
    {
        [DllImport("kernel32.dll")]
        1 reference
        public static extern bool AllocConsole();
        1 reference
        public Exe8()
        {
            InitializeComponent();
            AllocConsole();

            string fPath = "binaryfile.dat";

            using (FileStream fileStream = new FileStream(fPath, FileMode.Create, FileAccess.Write))
            {
                using (BinaryWriter bw = new BinaryWriter(fileStream))
                {
                    bw.Write(0);
                    bw.Write("Binary");
                    bw.Write(22);
                }
            }
            Console.WriteLine($"File '{fPath}' has been created.");
            Console.ReadLine();
        }
    }
}
```

Exercise 8: Using FileStream to Write Binary Data

1. **Objective**: Learn to work with binary files.
2. **Steps**:
3. Use FileStream to create and open a file named "binaryfile.dat" for writing.
4. Use BinaryWriter with the FileStream to write some integer and string data.
5. Close the file and verify its creation.
6. **Expected Outcome**: "binaryfile.dat" is created with binary data that can be opened later with a binary reader.

יצירת הקובץ הבינארי
(Create) וכתיבה
לתוכו (Write)

C:\Users\user\source\repos\FileHandling\FileHandling\bin\Debug\net8.

```
File 'binaryfile.dat' has been created.
```

binaryfile.dat

```
00000000  00 00 00 00 06 42 69 6E  61 72 79 16 00 00 00     .....Binary....
```

ניצור קובץ בינארי binaryfile.dat. בתוך הקובץ נשמר הייצוג
הבינארי של הנתונים שכתבנו לקובץ.

Exercise #9

```
using System.Windows;
using System.IO;
using System.Runtime.InteropServices;

namespace FileHandling.Exe
{
    4 references
    public partial class Exe9 : Window
    {
        [DllImport("kernel32.dll")]
        1 reference
        public static extern bool AllocConsole();
        1 reference
        public Exe9()
        {
            InitializeComponent();
            AllocConsole();

            string fPath = "binaryfile.dat";

            if (File.Exists(fPath))
            {
                using (FileStream fileStream = new FileStream(fPath, FileMode.Open, FileAccess.Read))
                {
                    using (BinaryReader br = new BinaryReader(fileStream))
                    {
                        int firstInt = br.ReadInt32();
                        string message = br.ReadString();
                        int secondInt = br.ReadInt32();

                        Console.WriteLine("Data read from binaryfile.dat:");
                        Console.WriteLine($"First Integer: {firstInt}");
                        Console.WriteLine($"String: {message}");
                        Console.WriteLine($"Second Integer: {secondInt}");
                    }
                }
            }
            else
            {
                Console.WriteLine("File 'binaryfile.dat' does not exist.");
            }
            Console.ReadLine();
        }
    }
}
```

Exercise 9: Reading Binary Data with FileStream

1. **Objective:** Read and interpret binary data from a file.
2. **Steps:**
3. Use FileStream and BinaryReader to open and read the data from "binaryfile.dat".
4. Display the read integer and string data on the console.
5. Verify the data matches what was written in the previous exercise.
6. **Expected Outcome:** The console displays the integer and string data stored in "binaryfile.dat".

נפתוק שהקובץ קיים במידה וקיים נרצה לפתוח (open) כ- ולקרוא ממנו (Read)

נקרוא את הנתונים מהקובץ הבינארי ואז נציג את התוכן של הקובץ ל-console.

```
C:\Users\user\source\repos\FileHandling\FileHandling\bin\Debug\net8.0-wir
Data read from binaryfile.dat:
First Integer: 0
String: Binary
Second Integer: 22
```

Exercise #10

```
using System.Windows;
using System.IO;
using System.Runtime.InteropServices;

namespace FileHandling.Exe
{
    4 references
    public partial class Exe10 : Window
    {
        [DllImport("kernel32.dll")]
        1 reference
        public static extern bool AllocConsole();
        1 reference
        public Exe10()
        {
            InitializeComponent();
            AllocConsole();

            string fPath = "original.txt";
            string content = "This is original file";
            string fCopy = "copy.txt";
            string fMove = "moved.txt";

            File.WriteAllText(fPath, content);

            if (File.Exists(fPath))
            {
                File.Copy(fPath, fCopy, true );
                if (File.Exists(fCopy))
                {
                    File.Move(fCopy, fMove);
                }
            }
            else
            {
                Console.WriteLine("File does not exist.");
            }
            Console.ReadLine();
        }
    }
}
```

Exercise 10: File Copy and Move Operations

1. **Objective:** Learn to copy and move files.
2. **Steps:**
3. Create a file named "original.txt" with some sample text.
4. Use File.Copy to make a copy named "copy.txt".
5. Use File.Move to move "copy.txt" to a new location, renaming it to "moved.txt".
6. Verify that "copy.txt" is moved and renamed as "moved.txt".
7. **Expected Outcome:** The file is successfully copied and moved to a new location with the new name.

ניצור קובץ original.txt ונכניס תוכן לתוכו. אם הקובץ הזה קיים ונרצה לבצע העתק של הקובץ לקובץ בשם copy.txt עי File.Copy

אם הקובץ copy.txt קיים אז נרצה להעביר את הקובץ ואת תוכנו לקובץ אחר בשם moved.txt עי File.Move.

בסוף נשאר עם 2 הקבצים original.txt ו- moved.txt.

original - פנקס רשימות

קובץ עריכה עיצוב תצוגה עזרה

This is original file

moved - פנקס רשימות

קובץ עריכה עיצוב תצוגה עזרה

This is original file