Malo Mouron B1

Gestion des fichiers

TP 11

R2.01

Table des matières

[1. Fournir le code nécessaire pour réaliser ce travail. 2](#_Toc162276321)

[2. Donnez le code nécessaire pour extraire les informations demandées d’un nom de fichier complet. 3](#_Toc162276322)

[3. Donnez le code nécessaire pour copier un fichier donné en fichier d’un autre nom dans le même répertoire (§ capture ci-dessous) 4](#_Toc162276323)

[4. Donnez le code nécessaire pour copier un fichier donné dans un répertoire donné, mais en conservant le même nom (§ capture ci-dessous) 5](#_Toc162276324)

[5. Donnez le code nécessaire pour ouvrir un fichier (choisir un petit fichier) et afficher sur la console son contenu, octet par octet, en code hexadécimal, comme sur la capture ci-dessous : 6](#_Toc162276325)

[6. Donnez le code nécessaire pour afficher sur la console le contenu d’un fichier texte dont le chemin est donné, en lisant le fichier en une fois (choisir un PETIT fichier) : 7](#_Toc162276326)

[7. Donnez le code nécessaire pour réaliser ce programme 8](#_Toc162276327)

[8. Donnez le code nécessaire pour afficher, sur la console, le contenu d’un fichier .zip dont le chemin est donné (nom, taille originale, taille compressée de chaque fichier), comme sur la capture ci-dessous : 9](#_Toc162276328)

## 1. Fournir le code nécessaire pour réaliser ce travail.

*/// <summary>  
/// Traite un répertoire et retourne une représentation en chaîne de chaque fichier dans le répertoire.  
/// </summary>  
/// <param name="cheminRepertoire">Le chemin du répertoire à traiter.</param>  
/// <returns>Une représentation en chaîne de chaque fichier dans le répertoire.</returns>  
public static string TraiterRepertoire*(*string* cheminRepertoire)  
{  
 DirectoryInfo infoRepertoire = *new* DirectoryInfo(cheminRepertoire);  
 FileInfo[] fichiers = infoRepertoire.GetFiles();  
 *string* resultat = "";  
  
 *foreach* (FileInfo fichier *in* fichiers)  
 {  
 FileInfoDetails detailsFichier = *new* FileInfoDetails(fichier);  
 resultat += detailsFichier.GetFileInfo() + "\n";  
 }  
  
 *return* resultat;  
}

Console.*WriteLine*(FileInfoDetails.*TraiterRepertoire*("C:/Users/malom/Desktop/mm076547/S2/R2\_07 - Graphes/TP/"));

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

*/// <summary>  
/// Initialise une nouvelle instance de la classe FileInfoDetails.  
/// </summary>  
/// <param name="file">L'objet FileInfo à partir duquel extraire les détails.</param>  
public* FileInfoDetails(FileInfo file)  
{  
 \_nomFichier = file.Name;  
 \_nomFichierSansExtension = file.Name.Substring(0, file.Name.Length - file.Extension.Length);  
 \_extension = file.Extension;  
 \_nomRepertoire = file.DirectoryName;  
 \_repertoireRacine = file.Directory.Root.ToString();  
 \_repertoireParent = file.Directory.Parent.ToString();  
 \_dateCreation = file.CreationTime;  
 \_derniereDateEcriture = file.LastWriteTime;  
 \_derniereDateAcces = file.LastAccessTime;  
 \_tailleFichier = file.Length;  
}

Ci-dessus la class que j’utilise pour initialiser toute les données du fichier charger en paramètre qui seront servit dans d’autres fonctions ci-dessous

## 2. Donnez le code nécessaire pour extraire les informations demandées d’un nom de fichier complet.

*/// <summary>  
/// Obtient des informations de base sur le fichier.  
/// </summary>  
/// <returns>  
/// Une chaîne de caractères contenant le répertoire , le nom du fichier et l'extention du fichier.  
/// </returns>  
public string* GetFileInfoRac()  
{  
 *return* "Base Folder :" + NomRepertoire + "\n" +  
 "File Name :" + NomFichier + "\n" +  
 "Extension :" + Extension;  
}

FileInfo file = *new* FileInfo("C:/Users/malom/Desktop/mm076547/S2/R2\_07 - Graphes/TP/4 - Iron Quest.pdf");

FileInfoDetails fileInfoDetails = *new* FileInfoDetails(file);

Console.*WriteLine*(fileInfoDetails.GetFileInfoRac());

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

## 3. Donnez le code nécessaire pour copier un fichier donné en fichier d’un autre nom dans le même répertoire (§ capture ci-dessous)



*/// <summary>  
/// Copies the current file to a new file with the same name, but with " (copy)" appended before the extension.  
/// The new file is created in the same directory as the current file.  
/// </summary>  
public void* CopyMyFile()  
{  
 *// Combine the directory path and the new file name to create the path for the new file  
 string* newFilePath = Path.*Combine*(\_nomRepertoire, \_nomFichierSansExtension + " (copy)" + \_extension);  
  
 *// Combine the directory path and the current file name to create the path for the current file  
 string* oldFilePath = Path.*Combine*(\_nomRepertoire, \_nomFichier);  
  
 *// Copy the current file to the new file* File.*Copy*(oldFilePath, newFilePath);  
}

FileInfo file = *new* FileInfo("C:/Users/malom/Desktop/mm076547/S2/R2\_07 - Graphes/TP/4 - Iron Quest.pdf");

FileInfoDetails fileInfoDetails = *new* FileInfoDetails(file);

fileInfoDetails.CopyMyFile();

Une image contenant capture d’écran, texte, Police

Description générée automatiquement

## 4. Donnez le code nécessaire pour copier un fichier donné dans un répertoire donné, mais en conservant le même nom (§ capture ci-dessous)

Une image contenant texte, Police, capture d’écran

Description générée automatiquement

*/// <summary>  
/// Copies the current file to a new file with the same name in a specified directory.  
/// </summary>  
/// <param name="newPath">The directory where the new file will be created.</param>  
public void* CopyMyFileIntoPath(*string* newPath)  
{  
 *// Combine the new directory path and the current file name to create the path for the new file  
 string* newFilePath = Path.*Combine*(newPath, \_nomFichier);  
  
 *// Combine the current directory path and the current file name to create the path for the current file  
 string* oldFilePath = Path.*Combine*(\_nomRepertoire, \_nomFichier);  
  
 *// Copy the current file to the new file* File.*Copy*(oldFilePath, newFilePath);  
}

FileInfo file = *new* FileInfo("C:/Users/malom/Desktop/mm076547/S2/R2\_07 - Graphes/TP/4 - Iron Quest.pdf");

FileInfoDetails fileInfoDetails = *new* FileInfoDetails(file);

fileInfoDetails.CopyMyFileIntoPath("C:/Users/malom/Desktop/mm076547/S2/R2\_07 - Graphes");

Une image contenant texte, capture d’écran, Police, logiciel

Description générée automatiquement

## 5. Donnez le code nécessaire pour ouvrir un fichier (choisir un petit fichier) et afficher sur la console son contenu, octet par octet, en code hexadécimal, comme sur la capture ci-dessous :

Une image contenant texte, Police, capture d’écran, nombre

Description générée automatiquement

*/// <summary>  
/// Affiche le contenue du fichier en hexadeciamle.  
/// </summary>  
public static void DisplayFileContentInHexadecimal*()  
{  
 Console.*Write*("Give me a file path : ");  
 *var* filePath = Console.*ReadLine*();  
 *using* (*var* fileStream = File.*OpenRead*(filePath))  
 {  
 *var* buffer = *new byte*[1];  
  
 *while* (fileStream.Read(buffer, 0, buffer.Length) > 0)  
 Console.*Write*(buffer[0].ToString("X2") + " ");  
 }  
 Console.*WriteLine*();  
}

FileInfoDetails.*DisplayFileContentInHexadecimal*();

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

## 6. Donnez le code nécessaire pour afficher sur la console le contenu d’un fichier texte dont le chemin est donné, en lisant le fichier en une fois (choisir un PETIT fichier) :

*/// <summary>  
/// afficher sur la console le contenu d’un fichier texte dont le chemin est donné.  
/// </summary>  
public static void DisplayFileContent*()  
{  
 *// Prompt the user for a file path* Console.*Write*("Give me a file path : ");  
 *var* filePath = Console.*ReadLine*();  
  
 *// Assert that the file path is not null* Debug.*Assert*(filePath != *null*, *nameof*(filePath) + " != null");  
  
 *// Open the file at the given path for reading  
 using var* reader = *new* StreamReader(filePath);  
  
 *// Read the entire content of the file  
 var* contenu = reader.ReadToEnd();  
  
 *// Print the content of the file to the console* Console.*WriteLine*(contenu);  
}

FileInfoDetails.*DisplayFileContent*();

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

## 7. Donnez le code nécessaire pour réaliser ce programme

*/// <summary>  
/// Writes user input to a file.  
/// </summary>  
public static void WriteUserInputToFile*()  
{  
 *// Prompt the user for a file path* Console.*Write*("Give me a file path: ");  
 *var* filePath = Console.*ReadLine*();  
  
 *// Check if the file path is writable  
 if* (!File.*Exists*(filePath) || !*new* FileInfo(filePath).IsReadOnly)  
 {  
 *// Initialize a boolean variable to control the loop  
 bool* continueWriting = *true*;  
  
 *// Enter a loop that continues until the user enters "fin"  
 while* (continueWriting)  
 {  
 *// Prompt the user for a text* Console.*Write*("Please enter a text (enter 'fin' to stop): ");  
 *var* text = Console.*ReadLine*();  
  
 *// If the text is "fin", set continueWriting to false  
 if* (text == "fin")  
 {  
 continueWriting = *false*;  
 }  
 *else* {  
 *// Otherwise, append the text as a new line in the file  
 using* (*var* streamWriter = File.*AppendText*(filePath))  
 {  
 streamWriter.WriteLine(text);  
 }  
 }  
 }  
 }  
 *else* {  
 *// If the file path is not writable, inform the user* Console.*WriteLine*("The file path is not writable.");  
 }  
}

FileInfoDetails.*WriteUserInputToFile*();

Une image contenant texte, capture d’écran, Police

Description générée automatiquementUne image contenant texte, capture d’écran, Police

Description générée automatiquement

## 8. Donnez le code nécessaire pour afficher, sur la console, le contenu d’un fichier .zip dont le chemin est donné (nom, taille originale, taille compressée de chaque fichier), comme sur la capture ci-dessous :

Une image contenant texte, Police, capture d’écran, noir

Description générée automatiquement

*/// <summary>  
/// Displays the details of a zip file.  
/// </summary>  
public static void DisplayZipFile*()  
{  
 *// Prompt the user for a zip file path* Console.*Write*("Give me a zip-file path : ");  
 *string* zipFilePath = Console.*ReadLine*();  
  
 *// Open the zip file for reading  
 using* (ZipArchive archive = ZipFile.*OpenRead*(zipFilePath))  
 {  
 *// Iterate over each entry in the zip file  
 foreach* (ZipArchiveEntry entry *in* archive.Entries)  
 {  
 *// Print the name, original size, and compressed size of each entry* Console.*Write*($"Nom : {entry.FullName} ; ");  
 Console.*Write*($"Taille originale : {entry.Length} bytes ; ");  
 Console.*WriteLine*($"Taille compressée : {entry.CompressedLength} bytes");  
 }  
 *// Print a separator line* Console.*WriteLine*("--------------------------------------------------");  
 }  
}

FileInfoDetails.*DisplayZipFile*();

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement