TP Manipulation de fichiers en C#.

I Création du projet

- Créer un nouveau projet nommé Manipulation de Fichiers.
- Renommer Form1.cs en FormFichiers.
- Ajouter au formulaire un contrôle TextBox avec les propriétés suivantes:

Name	textBoxSource
Location	95;8
Size	184;20

II) Travail sur l'interface.

Le contrôle OpenFileDialog

- Ajouter un contrôle OpenFileDialog.
- Ajouter au formulaire un bouton avec les propriétés suivantes:

Name	buttonOuvertureFichier
Location	8;8
Size	80;23
Text	Source:

- Créer son événement Click et compléter le code:

```
private void buttonOuvertureFichier_Click(object sender, EventArgs e)
{
    openFileDialog1.InitialDirectory = @"C:\";
    openFileDialog1.Title = "Sélectionner un fichier";
    openFileDialog1.FileName = "";
}
```

La 1° ligne indique le répertoire à afficher à l'apparition de la boite de dialogue; @ permet de travailler avec une chaîne Unicode, donc un seul \ est nécessaire.
La propriété Title définit le texte à afficher dans la barre de titre de la boite de dialogue.
La propriété FileName définit le nom du fichier choisi.

La propriété Filter permet de spécifier les types de fichiers qui s'afficheront dans la boite de dialogue.
La syntaxe est: Description | *.extension

Exemples: openFileDialog1.Filter = "Fichiers Text (*.txt) | *.txt"
openFileDialog1.Filter = "Bitmaps Windows (*.bmp) | *.bmp | Fichiers JPEG (*.jpg) | *.jpg";

Il est possible de spécifier le filtre par défaut par la propriété FilterIndex.

Exemple: openFileDialog1.FilterIndex = 1; //1 est le premier filtre déclaré.

- Compléter le gestionnaire par le code ci-dessous:

```
//Ouverture de la boite de dialogue et test du bouton cliqué dans cette boite.
if (openFileDialog1.ShowDialog() != DialogResult.Cancel)
    //Nom du fichier sélectionné dans le champ texte du TextBox.
    textBoxSource.Text = openFileDialog1.FileName;
else
    textBoxSource.Text = "";
```

- Compiler et tester le projet.

Le contrôle SaveFileDialog

- Créer une nouvelle TextBox avec les propriétés suivantes:

Name	textBoxDestination
Location	95;40
Size	184;20

- Créer un nouveau bouton avec les propriétés suivantes:

Name	buttonEnregistrerFichier
Location	8;40
Size	80;23
Text	Destination

- Ajouter un nouveau contrôle SaveFileDialog
- Créer le gestionnaire d'événement Click du bouton et compléter son code:

```
private void buttonEnregistrerFichier_Click(object sender, EventArgs e)
{
    //nom affiché dans la barre de titre de la boite de dialogue
    saveFileDialog1.Title = "Specifier la destination du nom de fichier";
    //Filtre sur les noms de fichiers à enregistrer
    saveFileDialog1.Filter = "Fichiers texte (*.txt) | *.txt";
    //filtre sélctionné par défaut est le 1°
    saveFileDialog1.FilterIndex = 1;
    //Ouvre une boite de dialogue si le fichier existe déjà
    saveFileDialog1.OverwritePrompt = true;
    //affichage de la boite et test du bouton cliqué
    if (saveFileDialog1.ShowDialog() != DialogResult.Cancel)
        //le nom de fichier choisi s'affiche dans la textBox
        textBoxDestination.Text = saveFileDialog1.FileName;

}
```

III) Travail avec le système de fichiers.

Pour manipuler les dossiers et les fichiers, il faut utiliser l'espace de noms **System.IO, et plus particulièrement System.IO.File, et System.IO.Directory.**

Test de l'existence d'un fichier avec la méthode Exists()

- **Ajouter cette méthode à la classe du formulaire.**

```
bool ExistenceFichier()
{
    //test si le nom de fichier présent dans la textBox existe
    if (!System.IO.File.Exists(textBoxSource.Text))
    {
        MessageBox.Show("Le fichier source n'existe pas");
        return false;
    }
    else
        return true;
}
```

Copie de fichier avec la méthode Copy()

- Placer sur le formulaire un nouveau bouton avec les propriétés suivantes:

Name	buttonCopieFichier
Location	96;80
Size	75;23
Text	Copier

- Créer le gestionnaire d'événement Click de ce bouton.

Le code de ce gestionnaire va copier le fichier dont le nom est dans la TextBox source, l'enregistrer avec le nom spécifié dans la TextBox destination.

```
private void buttonCopieFichier_Click(object sender, EventArgs e)
{
    //Appel de la méthode qui teste l'existence du fichier
    if (ExistenceFichier() == false)
        return;
    System.IO.File.Copy(textBoxSource.Text, textBoxDestination.Text);
    MessageBox.Show("Copie de fichier réussie");

}
```

- Compiler et tester le projet de la façon suivante:

- Sélectionner un fichier (txt) à l'aide du bouton source.
- Cliquer sur le bouton destination, et donner un nouveau nom de fichier (resultat.xt), cliquer sur enregistrer.
- Cliquer sur le bouton Copier, et contrôler le résultat.

Déplacement de fichier avec la méthode Move()

- Placer sur le formulaire un nouveau bouton avec les propriétés suivantes:

Name	buttonDeplacer
Location	96;112
Size	75;23
Text	Deplacer

- Créer le gestionnaire d'événement Click de ce bouton.

Le code de ce gestionnaire va déplacer dans un nouvel emplacement le fichier dont le nom est dans la TextBox source. (Le fichier sera enregistré avec le nom spécifié dans la TextBox destination).

```
private void buttonDeplacer_Click(object sender, EventArgs e)
{
    //Appel de la méthode qui teste l'existence du fichier
    if (ExistenceFichier() == false)
        return;
    System.IO.File.Move(textBoxSource.Text, textBoxDestination.Text);
    MessageBox.Show("Déplacement de fichier réussie");

}
```

- Compiler le projet et tester le déplacement de fichier. Pour le fichier de destination, faire un essai avec le même nom que le fichier source, et avec un nom différent.

Remarque:

Si le chemin source et destination est le même, le fichier est simplement renommé dans le même dossier.

Suppression de fichier avec la méthode Delete()

ATTENTION: cette méthode supprime définitivement le fichier (pas dans la corbeille)

- Placer sur le formulaire un nouveau bouton avec les propriétés suivantes:

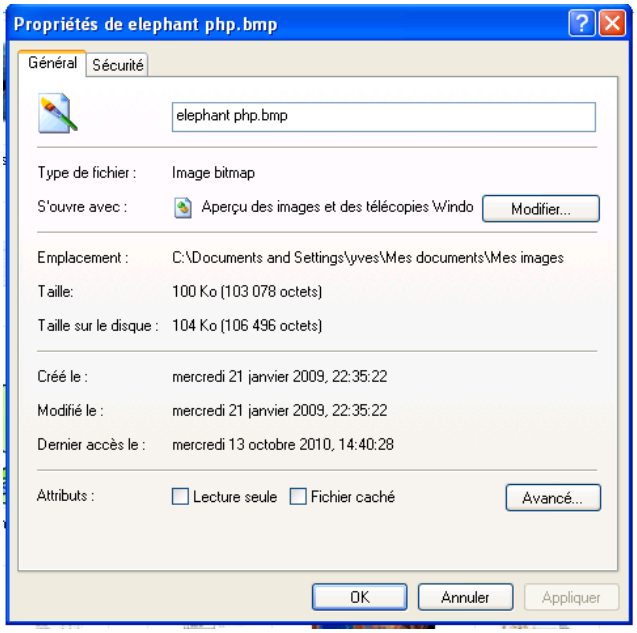
Name	buttonSupprimer
Location	96;144
Size	75;23
Text	Supprimer

- Créer le gestionnaire d'événement Click de ce bouton.

Le code de ce gestionnaire va supprimer le fichier dont le nom est placé dans la TextBox source.

```
private void buttonSupprimer_Click(object sender, EventArgs e)
{
    //Appel de la méthode qui teste l'existence du fichier
    if (ExistenceFichier() == false)
        return;
    if (MessageBox.Show("Etes-vous sûr de vouloir supprimer le fichier source?",
        "Vérification de la suppression",MessageBoxButtons.YesNo,MessageBoxIcon.Question)
        == DialogResult.Yes)
    {
        //suppression du fichier dont le nom est dans la textBox source
        System.IO.File.Delete(textBoxSource.Text);
        MessageBox.Show("Fichier supprimé");
    }
}
```

Informations et attributs de fichiers



L'objet System.IO.File permet de récupérer les données de l'onglet Général de la boîte de dialogue Propriétés d'un fichier.

Pour cela il faut utiliser:

⇒ L'objet **File** pour les informations de date et heure.
⇒ L'objet **FileAttributes** pour les attributs de fichier.

Informations de date et heure:

Méthodes de l'objet File pour récupérer les informations de date et heure (type DateTime)

Propriété	*Description*
GetCreationTime	Retourne la date et l'heure de création du fichier
GetLastAccessTime	Retourne la date et l'heure du dernier accès au fichier
GetLastWriteTime	Retourne la date et l'heure de la dernière modification du fichier

Attributs d'un fichier

La méthode **GetAttributes()** de l'objet File retourne une énumération **FilesAttributes** qui contient entre autres les membres suivants:

Attribut	*Signification*
Archive	Etat d'archivage du fichier. Les applications exploitent cet attribut pour marquer les fichiers à sauvegarder et à supprimer.
Répertoire	Ce fichier est un répertoire.
Fichier caché	Le fichier est masqué et n'apparaît donc pas dans le listing des répertoires classique.
Normal	Ce fichier est normal et aucun autre attribut n'est défini.
Lecture seule	Ce fichier est en lecture seule.
Système	Ce fichier appartient au système d'exploitation ou est utilisé exclusivement par ce dernier.
Temporaire	Ce fichier est un fichier temporaire.

Principe:

⇒ Créer une variable objet de type FileAttributes.
⇒ Stocker dans cette variable objet la valeur de retour de GetAttributes()
⇒ Tester les membres de cet objet (voir tableau ci-dessus) pour connaître l'état des attributs du fichier.

- Placer sur le formulaire un nouveau bouton avec les propriétés suivantes:

Name	buttonAttributs
Location	204;201
Size	75;23
Text	Attributs

- Créer le gestionnaire d'événement Click de ce bouton et compléter le avec le code suivant:

```
private void buttonAttributs_Click(object sender, EventArgs e)
{
    String LectureSeule;
    String DateCreation;

    //test si un fichier est ouvert
    if (openFileDialog1.FileName == "")
    {
        MessageBox.Show("Pas de fichier ouvert");
        return;
    }
    //Récupération de la date de création
    DateCreation = System.IO.File.GetCreationTime(textBoxSource.Text).ToString();
    MessageBox.Show("la date de création du fichier " + textBoxSource.Text + " est: " + DateCreation);

    //création de l'objet de type FileAttributes
    System.IO.FileAttributes ObjetAttributs;
    //Récupération des attributs du fichier dont le nom est dans la textBox source
    //Copie des attributs dans l'objet ObjetAttributs (via GetAttributs)
    ObjetAttributs = System.IO.File.GetAttributes(textBoxSource.Text);
    //test d'un attribut (lecture seule)
    //masque logique pour connaitre uniquement l'attribut Lecture seule
    //le resultat (0 ou 1) ne dépend uniquement que de l'attribut Lecture seule
    int resultat = (int)(ObjetAttributs & System.IO.FileAttributes.ReadOnly);
    if (resultat == 0)
        LectureSeule = "Ce fichier n'est pas en lecture seule";
    else
        LectureSeule = "Ce fichier est en lecture seule";

    MessageBox.Show(LectureSeule);



}
```

- Compiler et tester le projet.

Travail avec les répertoires

Il faut utiliser System.IO.Directory. Le principe est le même qu'avec les fichiers:

Quelques méthodes et exemples:

Méthode	Exemple
CreateDirectory()	System.IO.Directory.CreateDirectory("c:\MonDossier");
Exists()	System.IO.Directory.Exists("c:\test"),
Move()	System.IO.Directory.Move("c:\RepEnCours", "RepNouveau");
Delete()	System.IO.Directory.Delete"c:\temp",true);

IV) TP

- Compléter le projet "visionneuse d'images" afin d'afficher les informations et les attributs du fichier image sélectionné dans la PictureBox.
  L'affichage se fera sur le formulaire, dans un contrôle Label par exemple.

IV) Lecture et Ecriture dans un fichier texte.

Ecriture dans un fichier texte:

Pour cela on utilise la classe **StreamWriter**. Celle-ci possède plusieurs constructeurs surchargés (voir feuilles annexes) pour plus de précisions.

Exemple: **System.IO.StreamWriter objFichier = new System.IO.StreamWriter("c:\test.txt",true);**

Le fichier est ouvert en écriture en mode Mise à jour grâce à true. C'est-à-dire que si le fichier existe déjà les nouvelles données seront rajoutées à la fin du fichier.
Sans true, ou avec false, le fichier est remplacé par un nouveau fichier du même nom.

Méthodes: voir feuille annexe.

- Créer un nouveau projet.
- Placer un bouton appelé Ecriture.
- Compléter son événement click avec le code suivant:

```
private void buttonEcriture_Click(object sender, EventArgs e)
{
    System.IO.StreamWriter ObjetEcriture = new System.IO.StreamWriter("c:\\test\\test.txt");
    ObjetEcriture.WriteLine("salut");
    ObjetEcriture.WriteLine("bonjour");
    ObjetEcriture.WriteLine("au revoir");
    ObjetEcriture.Close();  //fermeture du fichier
    ObjetEcriture.Dispose(); //Libération des ressources utilisées par l'objet.

}
```

- Compiler, tester le projet, observer le résultat dans le fichier test.txt

- Modifier le gestionnaire d'événement ci-dessus de la façon suivante:

```
private void buttonEcriture_Click(object sender, EventArgs e)
{
    System.IO.StreamWriter ObjetEcriture = new System.IO.StreamWriter("c:\\test\\test.txt");
    ObjetEcriture.Write("salut");
    ObjetEcriture.Write("bonjour");
    ObjetEcriture.Write("au revoir");
    ObjetEcriture.Close();  //fermeture du fichier
    ObjetEcriture.Dispose(); //Libération des ressources utilisées par l'objet.

}
```

- Compiler, tester le projet, observer le résultat dans le fichier test.txt

Lecture dans un fichier texte.

Pour cela on utilise la classe **StreamReader**. Celle-ci possède plusieurs constructeurs surchargés (voir feuilles annexes) pour plus de précisions.

Exemple:

```
System.IO.StreamReader ObjetLecture = new System.IO.StreamReader("c:\\test\\test.txt");
```

Méthodes: Voir feuilles annexe.

- Placer un bouton appelé Lecture.
- Compléter son événement click avec le code suivant:

```
private void buttonLecture_Click(object sender, EventArgs e)
{
    System.IO.StreamReader ObjetLecture = new System.IO.StreamReader("c:\\test\\test.txt");
    string Contenu;
    Contenu = ObjetLecture.ReadToEnd();
    ObjetLecture.Close();
    ObjetLecture.Dispose();
    MessageBox.Show(Contenu);

}
```

Travaux pratiques:

- Modifier le projet "liste" afin d'enregistrer dans un fichier le contenu de la liste, lors de l'appui sur un bouton.

Rappel: on peut accéder à un élément de la liste ListBoxAnimaux avec le code suivant:

listBoxAnimaux.Items[index]

le 1° élément de la liste a l'index 0.

- Modifier le projet liste afin de recharger la liste avec le contenu d'un fichier enregistré auparavant, lors de l'appui sur un bouton.

Remarque: voir la méthode ReadLine().