

SEQUENCE 1 – PERSISTANCE DE DONNEES

SEANCE 1 - FORMAT TXT, CSV, JSON, XML

OBJECTIFS

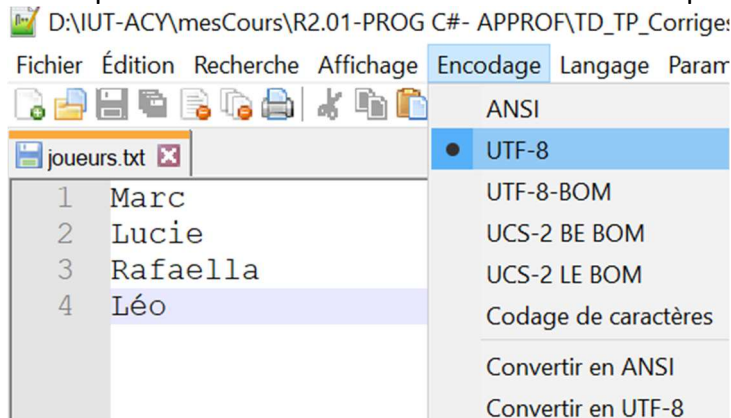
- Découvrir la persistance : écriture, lecture de données simples, la sérialisation , la désérialisation d'objets.
- Etudier différents formats de stockage de données
- Utiliser une liste
- Revoir les notions de base d'une classe

CONSIGNES

Créez un nouveau répertoire « **R2_01_Approfondissement_C#** », un sous répertoire « **Sequence_1_PersistenceCollection** »

EXERCICE 1 : FORMAT TXT

1. Créez un projet « **Exo1_FormatTxt** » en mode console dans une solution nommée « **Seance1_Decouverte** ».
2. Au sein de votre projet, créez un répertoire « data » dans lequel vous créerez un simple fichier texte « Joueurs.txt » . A l'aide d'un éditeur comme notepad++, vous mettrez quelques prénoms (dont au moins un avec un accent), un par ligne. Attention à l'encodage du texte. Par défaut, les méthodes utilisées s'appuient sur UTF8. Pensez donc à vérifier que le texte est en UTF8. Faites attention à ne pas créer de ligne vide à la fin.



3. Depuis l'explorateur de solution, indiquez au sein de ses propriétés du fichier, « copier si plus récent » dans le répertoire de sortie. Rappel : c'est pour que le fichier soit recopié dans le répertoire BIN à côté de l'exé.

4. Chargez les prénoms dans une liste de String, nommée lesJoueurs à l'aide du code ci-dessous :

```
String pathData = @"data\joueurs.txt";
// ou String pathData = "data\\joueurs.txt";

// liste pour charger les données du fichier en mémoire
List<String> lesJoueurs = new List<string>();

// Chargement des données
try
{
    StreamReader reader = new StreamReader(pathData);
    // ou StreamReader reader = new StreamReader(pathData, System.Text.Encoding.UTF8);

    while (!reader.EndOfStream) // reader.EndOfStream == false
    {
        String? unJoueur = reader.ReadLine();
        if ( !String.IsNullOrEmpty(unJoueur))
            lesJoueurs.Add(unJoueur);
    }
    reader.Close();
}
catch (Exception e) {
    Console.WriteLine("Probleme avec le fichier \n" + e);
    Environment.Exit(2); }
```

5. Affichez les joueurs à l'aide de la boucle foreach : Ici unJoueur est une variable de type string qui est écrasée automatiquement à chaque tour de boucle par la valeur suivante dans la liste.

```
foreach (String unJoueur in lesJoueurs)
{
    Console.WriteLine(unJoueur);
}
```

6. Permettez à l'utilisateur d'ajouter à la **liste (et non au fichier !)** autant de nouveaux joueurs qu'il le désire. Attention vérifiez que le prénom saisi par l'utilisateur ne soit pas déjà dans la liste à l'aide de la méthode Contains de la classe List

```
public bool Contains (T item); // T indique que cela fonctionne quelque
soit le type des éléments contenus dans la liste
```

<pre>Léo Nouveau prenom ?s pour stopper Nina</pre>	<pre>Rafaella Léo Nouveau prenom ?s pour stopper Léo Attention, ce prénom existe déjà.</pre>
--	--

7. Puis triez la liste à l'aide de la méthode Sort de la classe List. Réaffichez la liste pour voir si cela a fonctionné.
- ```
public void Sort ();
```

**REMARQUE : Contains s'appuie sur la méthode Equals de la Classe String et Sort s'appuie sur la méthode CopmpareTo de la classe String.**

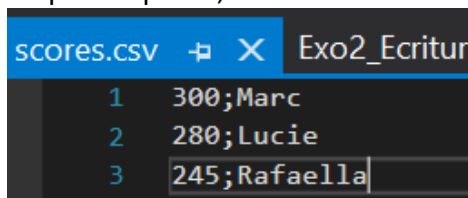
8. Sauvegardez la liste dans le fichier. Ici, on réécrit tout.

```
// Enregistrement des données en mémoire dans le fichier
try
{
 StreamWriter writer = new StreamWriter(pathData);
 foreach (String unJoueur in lesJoueurs)
 {
 writer.WriteLine(unJoueur);
 }
 writer.Close();
}
catch (Exception e) { Console.WriteLine("Probleme avec le fichier \n" + e);
Environment.Exit(2);}
}
```

9. Relancez votre programme pour vous assurer de la bonne sauvegarde. Rem : le fichier modifié est celui présent dans le répertoire BIN à côté de l'exé. (et non celui de départ placé à la racine du répertoire de projet)

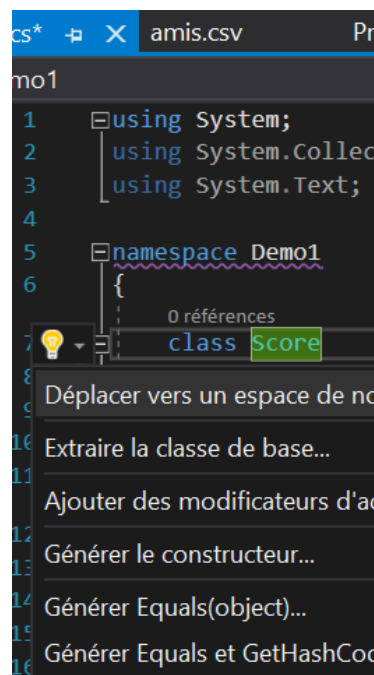
## EXERCICE 2 : FORMAT CSV

1. Ajoutez un projet « **Exo2\_FormatCSV** » à votre solution. Dans un répertoire Data, créez un fichier « Scores.csv » dans lequel vous mettrez quelques scores : nombre de points suivis d'un prénom, un score par ligne, attention à l'encodage. Les informations doivent être séparées par « ; ». Rem : stockez les scores en ordre décroissant.



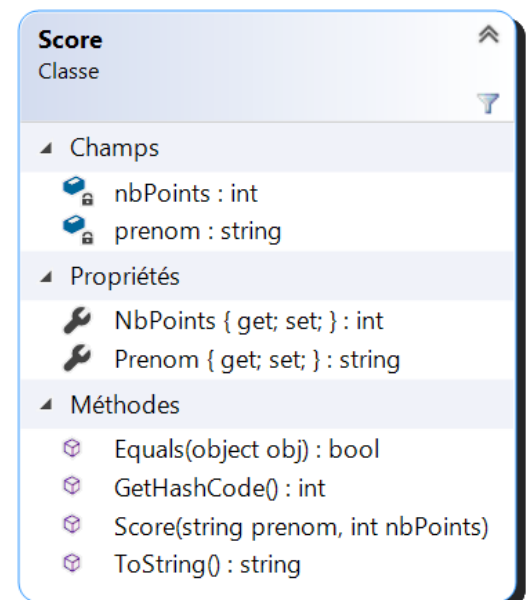
```
scores.csv X Exo2_Ecriture
1 300;Marc
2 280;Lucie
3 245;Rafaella
```

2. Chargez les prénoms et les points dans une liste de Scores. Pour cela :
- ajoutez à votre projet une classe Score. Champs privés : nbPoints et prenom. Encapsulez-les, générez constructeur, Equals et GetHashCode à partir des propriétés. Puis générez et modifiez la substitution de ToString. Rappel :



```
using System;
using System.Collections;
using System.Text;

namespace Demo1
{
 class Score
 {
 }
}
```



**Score**  
Classe

- Champs
  - nbPoints : int
  - prenom : string
- Propriétés
  - NbPoints { get; set; } : int
  - Prenom { get; set; } : string
- Méthodes
  - Equals(object obj) : bool
  - GetHashCode() : int
  - Score(string prenom, int nbPoints)
  - ToString() : string

- Reprenez le code de l'exo1. Et modifiez le pour avoir une liste de Score et non plus une liste de String. Utilisez la méthode Split pour extraire le prénom et le score de la ligne afin de les réunir au sein d'un objet Score.

Exemple pour utiliser Split : `String exemple = "Léo;Fasser;18";`  
`String[] infos = exemple.Split(";");`  
`Console.WriteLine(infos[0] + " " + infos[1] + " a " + infos[2] + " ans");`  
**Léo Fasser a 18 ans**

- Affichez la liste des scores. Maintenant ajoutez une ligne vide en fin de fichier ou au milieu et testez. Cela provoque une erreur, faites ce qu'il faut dans votre code pour éviter cette erreur.

```

Marc : 300 points
Lucie : 280 points
Rafaella : 245 points

```

- Puis faites en sorte que l'utilisateur puisse en ajouter autant de scores que nécessaire. A la fin, sauvegardez la nouvelle liste dans le fichier. Testez.

```

Entrez un nouveau score ? sinon S pour stopper
200
Entrez le prénom du joueur ?
Léo

Marc : 300 points
Lucie : 280 points
Rafaella : 245 points
Léo : 200 points

Entrez un nouveau score ? sinon S pour stopper
```

- Attention : il faut garder les éléments en ordre décroissant ! Pour cela, vous utiliserez la méthode : **Sort()** : pour trier la liste. Mais attention, elle va s'appuyer sur la méthode CompareTo de la classe Score, vous devez donc la définir dans Score. Pour cela :

- Dites que votre classe est Comparable :  
`internal class Score : IComparable<Score>`
- Puis aidez-vous de l'assistant pour générer CompareTo

```
class Score: IComparable<Score>
{
 private string prenom;
 private int nbPoints;

 public Score(string prenom, int nbPoints)
 {
 this.Prenom = prenom;
 this.NbPoints = nbPoints;
 }

 public int CompareTo(Score? other)
 {
 throw new NotImplementedException();
 }
}
```

- Complétez la méthode

```
public int CompareTo(Score? other)
{
 if (other == null)
 throw new ArgumentNullException("score ne doit pas être null");
 return this.NbPoints.CompareTo(other.NbPoints);
}
```

6. Utilisez Sort() puis Reverse() : pour inverser la liste car on veut un tri décroissant.

### EXERCICE 3 : FORMAT JSON

1. Ajoutez un projet « **Exo3\_FormatJSON** » à la solution .
2. Avec un simple éditeur, ajoutez à la racine du répertoire de votre projet le fichier « Scores.json » contenant les lignes suivantes :  
Rem : On a ici 2 scores  

|                                                                       |
|-----------------------------------------------------------------------|
| [{"NbPoints":300,"Prenom":"Marc"}, {"NbPoints":280,"Prenom":"Lucie"}] |
|-----------------------------------------------------------------------|
3. Copiez/Collez la classe Score du projet précédent. Pensez à modifier son namespace !
4. Désérialisez le fichier json afin d'obtenir une liste de Score puis affichez la liste (cf vignettes de cours. Pensez à importer le package NewtownSoft.Json
5. Faites en sorte que l'utilisateur puisse en ajouter autant qu'il le désire. A la fin, sérialisez la liste pour la sauvegarder dans le fichier json. Testez.

**EXERCICE 4 : FORMAT XML**

1. Ajoutez un projet « **Exo4\_FormatXML** » à la solution « TD1 ».
2. Avec un simple éditeur, ajoutez à la racine du répertoire de votre projet le fichier « Scores.xml » contenant les lignes suivantes :  
Rem : On a ici 2 scores

```
<?xml version="1.0" encoding="utf-8"?>
<ArrayOfScore xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <Score>
 <NbPoints>300</NbPoints>
 <Prenom>Marc</Prenom>
 </Score>
 <Score>
 <NbPoints>280</NbPoints>
 <Prenom>Lucie</Prenom>
 </Score>
</ArrayOfScore>
```

3. Pensez à l'option « Copier si plus récent »
4. Copiez/Collez la classe Score du projet précédent. Pensez à modifier son namespace !
5. Désérialisez le fichier xml afin d'obtenir une liste de Score puis affichez la liste.  
Rappel : Pour sérialiser des objets, il faut que leur classe : ici Score :
  - Soit public
  - Ait un constructeur par défaut ( constructeur sans paramètre )
6. Faites en sorte que l'utilisateur puisse en ajouter autant qu'il le désire. A la fin, sérialisez la liste pour la sauvegarder dans le fichier xml. Testez.

**EXERCICE POUR LES PLUS RAPIDES**

Les exercices 2,3 et 4 sont identiques. Seul le format des fichiers changent. Fusionnez pour ne faire plus qu'un seul projet . Ce dernier demandera à l'utilisateur le nom du fichier sur lequel il faut travailler. Ainsi en fonction, de l'extension, vous déclencherez les traitements adaptés que vous aurez pris soin de ranger dans une classe « PersistenceScores » par exemple....