Tutoriel phase 2

But

- Faire une application console de A-Z
- Lire un fichier et exploiter ces données

Initialisation du projet

Avant d'utiliser votre IDE, ouvrez votre terminal et tapez cette commande :
Dotnet new console --name PenduProject
Elle va créer :

- Un PenduProject.csproj. Ce type de fichier stocke la liste des fichiers inclus dans le projet, les assemblies du projet, le GUID du projet et d'autres informations comme la version du projet.
- Un Program.cs le point d'entrée de votre programme.
- Un dossier obj utilisé pour stocker des fichiers de manière temporaires durant le processus de compilation.

Le but de ce projet va d'être de créer un pendu. Pour plus de facilité, on travaillera avec la langue anglaise car l'encodage des accents n'aura pas à être gérer.

Pour vous aider à lire ce fichier, je vous conseille d'utiliser la classe **StreamReader** (https://docs.microsoft.com/fr-fr/dotnet/api/system.io.streamreader?view=netframework-4.7.2) elle permet d'utiliser des stream de lectures pour lire les flux ici un fichier texte.

Pour vous aider je vous conseille de faire en deux étapes :

- Choisir un mot du fichier et le mettre en mémoire
- Faire la partie où vous devinez le mot

Pour pouvoir gérer l'intéraction entre le joueur et le programme il faut utiliser les méthodes de la classe static Console :

- WriteLine pour écrire un texte
- ReadLine pour récupérer la réponse du joueur

Voici les règles du jeu :

- Dans la console le jeu doit choisir un mot
- Seuls les mots ayant 6 lettres minimums peuvent être choisis
- Au début de chaque tour la console affiche le mot composé de * pour les lettres non devinées et de lettres pour le reste
- Vous avez 11 chances pour deviner le mot avant de perdre

Pour initialiser le projet :

- Faire un git clone de ce projet github :
  - git clone https://github.com/plouiserre/DevoxxPenduDotnetCore.git

- L'initialisation du projet se trouve sur la branche master
- Il y a un fichier dico.txt listant des mots de la langue anglaise.

Pour votre exercice vous allez devoir utiliser la classe Random du framework .NET. Pour cela voici la documentation correspondante :

- https://docs.microsoft.com/fr-fr/dotnet/api/system.random?view=netframework-4.7.2
- https://stackoverflow.com/questions/2706500/how-do-i-generate-a-random-int-number

Vous avez 30 minutes pour réaliser ce projet en solo, vous pouvez le tenter seul ou suivre la solution détaillée juste après.

Correction

Phase 1 : choisir le mot dans le dictionnaire

Récupérer directement la solution

Si vous voulez rapidement la solution il suffit de :

- Faire un git clone de ce projet github
  - git clone https://github.com/plouiserre/DevoxxPenduDotnetCore.git
- Récupérer en local la branche « ChooseWords », en utilisant ces commandes
  - git checkout ChooseWords
- Vous avez le code gérant cette partie

Développement de cette partie

A la racine du projet créez la classe **DictonaryData**. Son rôle est de choisir le mot à deviner parmi le dictionnaire.

Tout d'abord dans le constructeur on va initialiser trois variables :

- Le nom du fichier et le nombre de lettres minimales qui vont être transmis par le constructeur.
- La liste de String « Words »

```
public DictonaryData(string fileName, int minimalLetter)
{
    FileName = fileName;
    Words = new List<string>();
    _minimalLetter = minimalLetter;
}
```

Ensuite une méthode private aura pour rôle de récupérer les mots fichier supérieurs à la variable _minimalLetter et de compter les mots choisis. Les mots illégibles seront stockés dans la liste Words.
Le fichier sera parcouru ligne par ligne en vérifiant que le nombre de lettre minimal est respectée et le nombre de ligne du fichier sera stocker dans la variable _totalLines.

```
private void CountLines(){
    using(StreamReader file = new StreamReader(FileName))
    {
        string line = string.Empty;
        while((line = file.ReadLine())!= null){
            if(line.Length >= _minimalLetter){
            _totalLines += 1;
            Words.Add(line);
            }
        }
    }
}
```

La seconde méthode privée devra choisir de manière aléatoire un nombre de 0 à la valeur de la variable _totalLines qu'on stockera dans _lineChoose.

```
private void DecideLine(){
    Random random= new Random();
    _lineChoose = random.Next(_totalLines);
}
```

Pour finir la méthode publique « ChooseWords » sera le point d'entrée pour lancer l'algorithme et choisir le mot à deviner.

```
public void ChooseWords(){
    CountLines();
    DecideLine();
    WordChoose = Words[_lineChoose];
}
```

Pour finir voici la classe complète

```cs
using System;
using System.Collections.Generic;
using System.IO;

namespace PenduProject
{
    public class DictonaryData
    {
        private int _totalLines { get; set; }
        private int _minimalLetter { get; set; }
        private int _lineChoose { get; set; }
        private List<string> Words { get; set;}
        public string FileName { get; set; }
        public string WordChoose { get; set; }

        public DictonaryData(string fileName, int minimalLetter)
        {
            FileName = fileName;
            Words = new List<string>();
            _minimalLetter = minimalLetter;
        }

        public void ChooseWords(){
            CountLines();
            DecideLine();
            WordChoose = Words[_lineChoose];
        }

        private void CountLines(){
            using(StreamReader file = new StreamReader(FileName))
            {
                string line = string.Empty;
                while((line = file.ReadLine())!= null){
                    if(line.Length >= _minimalLetter){
                    _totalLines += 1;
                    Words.Add(line);
                    }
                }
            }
        }

        private void DecideLine(){
            Random random= new Random();
            _lineChoose = random.Next(_totalLines);
        }
    }
}
```

Maintenant il reste plus qu'à définir le point d'entrée du projet « Program.cs » qui consiste à :

- Initialisez la classe « DictonaryData ».
- Appelez la méthode « ChooseWords » pour lancer la recherche du mot à chercher
- Affichez le mot choisi.

```
using System;

namespace PenduProject
{
    class Program
    {
        static void Main(string[] args)
        {
            DictonaryData dico = new DictonaryData("data/dico.txt", 6);
            dico.ChooseWords();

            Console.WriteLine("Le mot choisi est {0} ", dico.WordChoose);

            Console.Read();
        }
    }
}
```

Phase 2 : jouez avec le joueur

Récupérer directement la solution

Si vous voulez rapidement la solution il suffit de :

- Récupérer en local la branche « finalGame », en utilisant ces commandes
  - git checkout finalGame

Développement de cette partie

La classe « PlayGame » à la base du projet sera la base de cette partie. Elle gère la communication et les interactions avec le joueur mais et détermine si vous avez gagné ou perdu.

Le constructeur va initialiser trois variables :

- Le mot à deviner et le nombre de vies du joueur sont transmis par le constructeur.
- La liste de char « AllCharsToGuess» qui comportera la liste des lettres à deviner.

La première méthode aura pour tâche de prendre le mot à deviner et de mettre les lettres qu'elle contient « AllCharsToGuess » en évitant les doublons. Pour être appelée de l'extérieur, elle doit être mise en « Public ».
La variable privé « wordToGuess » est le mot à deviner. Il provient du mot choisi de la classe « DictonaryData ».
La variable privé « totalLife » est le nombre de vie que possède au départ le jeu. A chaque fois qu'il se trompe il perd une vie. Quand elle devient égal à 0, le joueur a perdu.
Dans le constructeur on initialise « AllCharsToGuess » et on initialise « wordToGuess » et « totalLife ».

```
public void InitWord(){
    foreach(char letter in wordToGuess){
        if(!AllCharsToGuess.Contains(letter))
            AllCharsToGuess.Add(letter);
    }
}
```

La méthode privée « DisplayWordToGuess » aura pour rôle de gérer l'affichage du mot. Pour cela il retournera le string « wordToDisplay » composée soit de * pour les lettres non trouvées ou soit des lettres trouvées.

```
private string DisplayWordToGuess(){
    string wordToDisplay = string.Empty;
    foreach(char letter in wordToGuess){
        if(AllCharsToGuess.Contains(letter))
            wordToDisplay += "*";
        else
            wordToDisplay += letter;
    }
    return wordToDisplay;
}
```

PRESENTER isGameFinished en amont

Cette variable est à false tant que le jeu continue. Dès que le joueur gagne elle passe à true et on quitte la boucle while.

La méthode public Play est le point central de la classe. Il se compose d'une boucle while qui s'arrêtera lorsque le joueur aura deviné le mot ou perdu en ayant plus aucune vie.

Dans cette boucle on affiche d'abord le mot en appelant la méthode « DisplayWordToGuess » puis on demande au joueur de choisir une lettre. Sa saisie doit être validée sans lui faire perdre de vie en cas de mauvaise saisie (un chiffre ou plusieurs lettres). Si c'est ok on vérifie si la lettre choisie compose bien le mot :

- Si oui on le félicite, on retire la lettre de la liste « AllCharsToGuess ». Puis on vérifie si le joueur a tout deviné. Si oui on passe le boolean « isGameFinished » à true pour sortir de la boucle while.
- Si ce n'est pas le cas, on lui enlève une vie. S'il n'en possède plus on lui indique qu'il a perdu sinon on lui affiche un message lui expliquant qu'il n'a pas trouvé la bonne lettre avec le bon nombre de vie qui lui reste.

```
public void Play(){
    while(!isGameFinished && totalLife > 0){
        string wordDisplay = DisplayWordToGuess();
        Console.WriteLine(wordDisplay);
        Console.WriteLine("Choisissez une lettre");
        string response = Console.ReadLine();
        if(response.Length != 1){
            Console.WriteLine("Il faut choisir une lettre!!! On recommence le tour.");
            continue;
        }
        char letterProposed = response[0];
        if(AllCharsToGuess.Contains(letterProposed)){
            Console.WriteLine("Bravo {0} etait bien une lettre dans le mot", letterProposed);
            AllCharsToGuess.Remove(letterProposed);
            if(AllCharsToGuess.Count == 0){
                Console.WriteLine("Vous avez gagné");
                isGameFinished = true;
            }
        }
        else{
            totalLife -= 1;
            if(totalLife == 0)
                Console.WriteLine("Vous avez perdu!");
            else
                Console.WriteLine("Pas la bonne lettre. Il vous reste {0}", totalLife);
        }
    }
}
```

Voici le code de la classe en entier

```
using System;
using System.Collections.Generic;

namespace PenduProject
{
    public class PlayGame
    {
        private bool isGameFinished {get; set;}
        private List<char> AllCharsToGuess {get; set;}
        private string wordToGuess { get; set; }
        private int totalLife {get; set;}

        public PlayGame(string word, int life)
        {
            wordToGuess = word;
            AllCharsToGuess = new List<char>();
            totalLife = life;
        }

        public void InitWord(){
            foreach(char letter in wordToGuess){
                if(!AllCharsToGuess.Contains(letter))
                    AllCharsToGuess.Add(letter);
            }
        }

        public void Play(){
            while(!isGameFinished && totalLife > 0){
                string wordDisplay = DisplayWordToGuess();
                Console.WriteLine(wordDisplay);
                Console.WriteLine("Choisissez une lettre");
                string response = Console.ReadLine();
                if(response.Length != 1){
                    Console.WriteLine("Il faut choisir une lettre!!! On recommence le tour.");
                    continue;
                }
                char letterProposed = response[0];
                if(AllCharsToGuess.Contains(letterProposed)){
                    Console.WriteLine("Bravo {0} etait bien une lettre dans le mot", letterProposed);
                    AllCharsToGuess.Remove(letterProposed);
                    if(AllCharsToGuess.Count == 0){
                        Console.WriteLine("Vous avez gagné");
                        isGameFinished = true;
                    }
                }
                else{
                    totalLife -= 1;
                    if(totalLife == 0)
                        Console.WriteLine("Vous avez perdu!");
                    else
                        Console.WriteLine("Pas la bonne lettre. Il vous reste {0}", totalLife);
                }
            }
        }

        private string DisplayWordToGuess(){
            string wordToDisplay = string.Empty;
            foreach(char letter in wordToGuess){
                if(AllCharsToGuess.Contains(letter))
                    wordToDisplay += "*";
                else
                    wordToDisplay += letter;
            }
            return wordToDisplay;
        }

    }
}
```

La classe « Program » devra en plus initialiser la classe PlayGame et appelez les méthode InitWord et Play.

```
using System;

namespace PenduProject
{
    class Program
    {
        static void Main(string[] args)
        {
            DictonaryData dico = new DictonaryData("data/dico.txt", 6);
            dico.ChooseWords();

            Console.WriteLine("Le mot choisi est {0} ", dico.WordChoose);

            PlayGame game = new PlayGame(dico.WordChoose, 11);
            game.InitWord();
            game.Play();

            Console.Read();
        }
    }
}
```